修　士　論　文

小規模メッシュベースのライブメディアストリーミングオーバレイにおけるオンライン時間を考慮したQoS改善手法の提案

QoS Improvement for Small Mesh-based Live Media Streaming Overlays Using Online Duration Awareness

指導教官　　江崎 浩 教授

東京大学大学院情報理工学系研究科
電子情報学専攻

氏　名　　48-076416 **姜　鵬**

提　出　日　　**平成** 21 **年** 02 **月** 04 **日**

Abstract

Broadband Internet environment has rapidly introduced the live media streaming service over the globe. Peer-to-peer live media streaming has emerged as one of promising solutions, which has been shown to be cost effective and easy to deploy over the existing Internet. There are basically two approaches for peer-to-peer live media streaming: tree-based and mesh-based. Mesh-based approach employs peers into a mesh configuration to exchange the data, which greatly enhances churn resiliency. CoolStreaming [7] is one of the milestones in mesh-based approach research and development. PPLive [3], a derivative, has achieved great success and is now streaming to hundreds of thousands of users simultaneously. Mesh-based overlay systems can provide better QoS for large number of participants with sufficient available bandwidth. However, with small number of participants and poor available bandwidth, the QoS is degraded. To be specific, bad service continuity, long start-up time and long lags. In such a case, the service observed by the receiver peers is sometimes abandoned. This paper focuses on improving the QoS of mesh-based live media streaming overlay, especially the QoS of CoolStreaming when the swarm size is small and aggregated upload bandwidth is not sufficient. Considering our purpose, it is natural to first modify the legacy mesh-based approach to fit the target scenario better, so as to have a better QoS. After that, further QoS improvement would not be easy challenge. The distribution of peers' upload ability will not change and, in our opinion, the only possible solution would be to encourage peers to stay online longer so as to have a larger and much stable swarm. The idea is that, firstly, we guarantee the QoS of the peers with long online time and relatively good communication ability so as to guarantee at least there are a few peers who can receive tolerable QoS, which at the same time serves as an incentive to encourage peers to stay longer online.

In this paper, we discuss the reason of poor performance in the small mesh-based live media streaming overlay. Then, we propose the design changes and parameter adjustments to the legacy mesh-based live media streaming scheme, which enables the overlay to be more robust to extreme heavy churn and bandwidth insufficiency and enables the overlay to have a better performance even the size of the swarm is not so large. Also, by the application of rank-based incentive policy, a novel peering scheme which partially alleviates the bad influence the heavy churn and extreme low upload ability peers brings to the stable ones is proposed. The whole design is referred to as SMODA which means *QoS improvement for Small Mesh-based Overlays using online Duration Awareness*. The key idea is to divide mesh-based overlay

peers into groups with high rank and low rank based on the online duration and upload ability. And, we encourage dense partnership formations among same rank peers and a few partnership formations among peers with different ranks. The pair-wised incentive method is not applied here because it increases the average hops that a chunk reaches the destination, which has very bad effect on the overall performance in a heavy churn environment. We confirmed that this proposal can at least guarantee tolerable QoS of the resource-abundant long online time peers, by the computer simulation.

An event-driven simulator is written in Java to evaluate the proposed architecture framework. Simulation results show that the proposed design changes and parameter adjustments can improve the small overlay live streaming system, so as to achieve much stable and better QoS and the proposed rank-based incentive method can greatly improve the QoS for the stable and bandwidth abundant peers.

Keywords: Mesh-based, Live Media Streaming, Event-driven, Rank-based

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

With the rapid deployment of broadband Internet access, we can recognize that the multimedia contents delivery and exchange emerge viable and popular. Rather than downloading contents from a server which will probably cause a bottleneck at the server, peer-to-peer approach offers a low cost and scalable solution, while increasing a significant amount of Internet traffic. When we use peer-to-peer file sharing applications (e.g., BitTorrent [34]) to download multimedia files, peers are not synchronized. After the completion of the downloading, the user then can start to watch or listen the multimedia contents. Peer-to-peer VoD service offers another solution which also creates an unsynchronized overlay, but supports quick start of playback. On the contrary, peer-to-peer live media streaming applications create semi-synchronized overlays to stream live media contents to peers. Semi-synchronized here means the playback time at a peer is just a few seconds after the origin of the contents referred to as origin peer.

Initially, as IP multicast, tree-based live media streaming approach has been researched. While tree-based method works well for IP multicast, it often is vulnerable to peer churn when constructing an application peer-to-peer overlay. This is because an application level overlay peer can easily crash or leave at will. Giving the demand to simultaneously support a large number of participants and cope with high bandwidth requirements, it becomes much challenging.

Later, mesh-based approach such as DONet/CoolStreaming [7] has been proposed, which contrast to tree-based design in that it does not construct and maintain an explicit structure for chunk exchange. The idea is to use the availability of data to guide the data flow rather than constantly repair a tree-based structure in a highly dynamic peer-to-peer environment. This technique brings a number of unique advantages such as scalability, resilience and effectiveness in coping with dynamics and heterogeneity. Mesh-based approach achieves great success. Many derivatives like PPLive [3], PPStream [4], and UUSee [5] have received millions of dollars of venture investments and attract millions of users. According to PPLive, which de-

livers about 1000 channels to audiences, the number of simultaneous users has reached as large as 450,000 by Feb. 2007, which need an aggregated downloading speed of 215 gigabits/sec given a streaming rate of 500 Kbps.

Mesh-based overlay systems can provide better QoS for large number of participants with sufficient available bandwidth. However, with small number of participants and poor available bandwidth, the QoS is degraded. To be specific, bad continuity, long start-up time and long lags. The meaning of these three QoS terms will be explained in section 2.3.2. In such a case, the service observed by the receiver peers is sometimes abandoned. This paper tackles with this problem and focuses on improving the QoS of the mesh-based live media streaming overlay, especially the QoS of CoolStreaming when the swarm size is small and aggregated upload bandwidth is not sufficient. Considering our purpose, it is natural to first modify the legacy mesh-based approach to fit the target scenario better, so as to have a better QoS. After that, to further improve the QoS would be a tough task when restraining from using outside help. The distribution of peers' upload ability will not change and, in our opinion, the only way is to encourage peers to stay online longer so as to have a larger and more stable swarm. The idea is to first guarantee the QoS of the peers with long online time and relatively good communication ability so as to guarantee at least there are a few peers which can receive tolerable QoS, which at the same time serves as an incentive to encourage peers to stay longer online.

In this study, firstly, the causes of this bad performance are investigated by surveying related works. Then, we propose parameter adjustments and design changes so as to match the character of the small overlays which have high ratio of unstable and upload-poor peers. To build a system that is robust to extreme unstable environment when the swarm size is small, we design SMODA. In order to achieve high performance, a set of practical challenges have to be addressed including how to disperse chunks more quickly; how to avoid bottleneck at the start; how to add new partners as early as possible; how to encourage cooperation among peers when their lags are quite different and how to improve the QoS of the stable peers with not so poor upload ability. We modify various aspects of legacy mesh-based approach, to be specific, mainly CoolStreaming approach to cope with these practical challenges.

Besides, based on the online duration model in [19], which shows that peers that have stayed for a longer time will statistically stay for a longer remaining time, a rank-based incentive method is introduced, which struggles to alleviate the bad effect that the high ratio of unstable and poor upload ability peers bring to the stable ones. The key idea is to divide mesh overlay peers into groups with high rank and low rank based on the online duration and upload ability. And encourage dense partnership formations among same rank peers and a few partnership formations among peers with different ranks. The pair-wised incentive method is not used here because this increases the average hops that a chunk reaches the destination, which has very bad effect to the overall performance in the heavy churn environment.

We have built an event-driven simulator in Java to evaluate our proposals. The topology file is generated using the Boston University Representative Internet Topology Generator (BRITE) [13]. Our experiments involve verifying all the parameter adjustments and design changes one by one and verifying the effect of rank-based incentive method. The results of experiments are discussed thoroughly and demonstrate that SMODA achieves smooth QoS even in very unstable scenarios and affirm that the rank-based incentive proposal is a promising practical solution to guarantee the resource-abundant long online time peers to have relatively better QoS. Finally, this work concludes with the conclusions and future work.

1.2 Research Purpose

The purpose of this research is to improve the QoS of mesh-base live media streaming overlay, especially the QoS of CoolStreaming when the swarm size is small and the aggregated upload bandwidth is insufficient. To achieve this, parameter adjustments and design changes are made to cope with the small overlay environment especially the extreme unstable and upload bandwidth insufficient environment. The further purpose is to guarantee at least a few long online time and upload abundant peers can have tolerable QoS and thus encourage peers to stay longer online so as to have a more stable and larger swarm. And we hope this will further improve the quality of the streaming.

1.3 Composition of This Paper

The rest of this paper is organized as follows.
Chapter 2 discusses the background and the related researches.
Chapter 3 presents the proposed architecture.
Chapter 4 describes the simulator design and simulation setup.
Chapter 5 analysis the simulation results.
Chapter 6 discusses the future work and concludes this work.

Chapter 2

Background and Related Work

2.1 Internet Live Media Streaming

With the rapid deployment of high bandwidth Internet, a large number of new services have emerged including live media streaming service. A media stream can be live or on demand. On demand videos or audio streams are stored in servers for a relatively long period of time, and are available to be transmitted at a user's request. In contrast, live media streams are only available at one particular time, as the live video stream of the Olympic game. Live media streaming technology can be used in various areas, including broadcast TV programs on the Internet to thousands of users simultaneously, deliver video to large number of online games users simultaneously and assist distant education. It is a very promising and important technology.

2.2 Architectures for Internet Live Media Streaming

Before explain the architectures of Internet live media streaming, first, the definition of multicast [6] should be explained. Figure 2.1 is from the paper [11] written by Y. Chu et al., which illustrates naive unicast, IP multicast, and peer-to-peer multicast. Figure (b) is unicast, every packet can only get to one destination host. Figure (c) illustrates the IP multicast, the routers duplicate the packets and forward packets to a group of end hosts. For example, in this figure router R2 duplicates packets and forward them to C and D simultaneously. The Figure (d) illustrate P2P multicast, not the routers but the end users duplicate packets and forward it to other end host. In figure (d), C duplicates packets and forward them for D.

As shown in Figure 2.2, according to the paper [14] written by J. Liu et al., the architectures of live media streaming can be divided into basically three kinds of technologies. The first one is IP multicast [6] and it needs the support of routers, the second one is CDN which means Content Delivery Network and the third one is peer-to-peer multicast. This paper focuses on peer-to-peer multicast, in other words, peer-to-peer live media streaming.

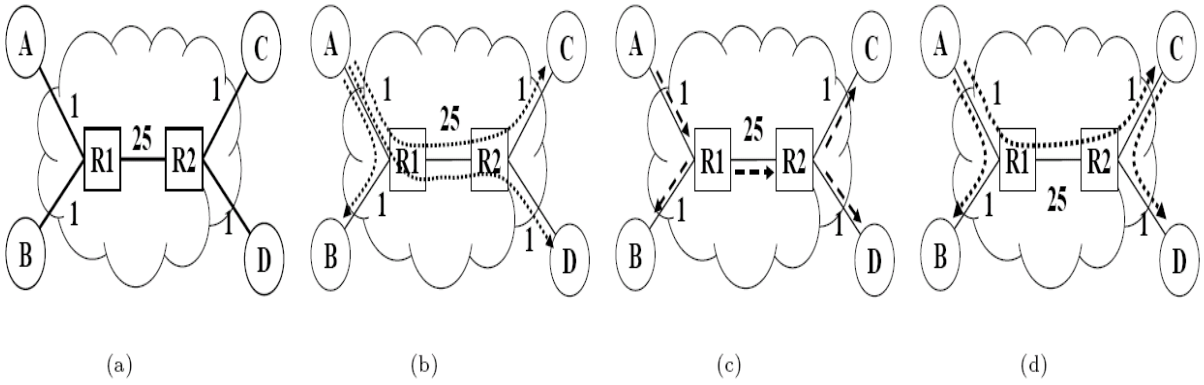

Fig. 2.1 An illustration of Unicast, IP Multicast and P2P Multicast in [11].

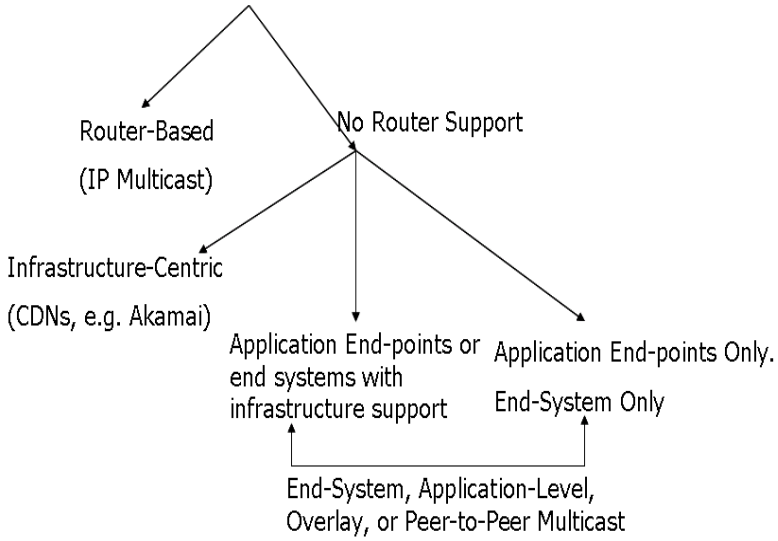

Fig. 2.2 An illustration of architecture choices in [6].

2.2.1 IP Multicast

For router based architecture, IP multicast requires routers to maintain per-group state. It is actually researched intensively a few years ago but is not widely deployed largely due to issues regarding scalability and deployment. As this technology needs all the participating routers to support multicast, it is not scalable and deploying it will need a lot money to

change all the routers to the new routers that support multicast. As a result, it is sparsely deployed.

2.2.2 CDN

There are basically 2 kinds of non-router based architectures to support live media streaming. The first kind is an infrastructure-centric architecture, which is also commonly referred to as Content Delivery Networks (CDNs), where the company deploys proxies at crucial locations on the Internet to provide performance improving services to end users transparently. End users request the data to nearby proxy points, and receive data more quickly. Such an approach has been deployed by companies such as Akamai [1], one of the largest commercial CDN service providers, which claims of handling 20% of total Internet traffic today, reports a peak aggregate capacity of 200 Gbps. This method is also money consuming.

2.2.3 Peer-to-Peer Live Media Streaming

The other non-router based architecture is a purely application layer peer-to-peer architecture, where unicast tunnels are formed by the users actually participating in the multicast group to deliver data among each other. Functionalities of such a peer-to-peer system such as system administration and maintenance are distributed among the peers, instead of being handled by a central server. There are a lot of advantages of this architecture, first, such approach does not require support from Internet routers and network infrastructure, and consequently is extremely cost-effective and easy to deploy. Second, a peer that participating in a steaming channel is not only downloading a video stream, but also uploading it to other peers in the same channel. Consequently, such an approach has the potential to scale with group size, as large number of participants also generates more potential supplies during the streaming session.

2.3 Peer-to-Peer Live Media Streaming

Although peer-to-peer live media streaming can be used in many areas, basically this paper emphasize on P2PTV. There are many kinds of P2PTVs that have already been deployed. In Japan, there is Yahoo BB-TV! and in China there are PPLive, PPStream, UUSee, etc.

2.3.1 Contrast to other Peer-to-Peer Applications

There are several distinguishing characteristics of peer-to-peer live media streaming system. First, Large scale, potential peers watching a popular channel can reach as many as tens of thousands. Second, high bandwidth requirements, as media streamings usually have a streaming rate of several hundred kilo bits per second. The third is time stringency, requiring

Category	Bandwidth-sensitive	Delay-sensitive	Scale
File download	X	X	Large
On-demand streaming	O	O	Large
Audio/video conferencing	O/X	O	Small
Simultaneous broadcast	O	O	Large

Table. 2.1 A taxonomy of typical Peer-to-Peer applications in [14].

timely streaming delivery.

Table 2.1 is from the paper [14] written by J. Liu et al., which illustrates the differences between peer-to-peer live media streaming (in this table, it uses another word simultaneous broadcast) and other peer-to-peer applications. On-demand streaming (e.g., [33], [35], [36]) and audio/video conferencing are also time stringent and have high bandwidth requirements. But, in on-demand streaming, the peers in the overlay are asynchronous, a big overlay is constructed for multiple streaming contents, and the challenges are different from live media streaming. Audio/video conferencing applications differ from live media streaming applications like P2PTV in that latency requirement is even more critical and such applications are typically of smaller scales, involving only a few to tens of participants.

2.3.2 Evaluation Parameters

There are several important evaluation parameters for peer-to-peer live media streaming. The first one is start-up delay. Start-up delay is the time interval from when one channel is selected until actual playback starts on the screen. The second one is playback lag, some peers watch frames in a channel minutes behind the server. That is called playback lag. Large playback lag is not acceptable because, for example, in a soccer game channel no one wants to watch a goal minutes after other people. The third one is streaming rate, high streaming rate means high quality but today the commercial peer-to-peer live media streaming system can only provide very low streaming rates. In the PPLive case, almost all the channels have a streaming rate less than 1000 Kbps. The fourth one is continuity index which is an index indicates the continuity of the streaming. It is the most important evaluation parameter, because we do not want our playback to stop. In the PPLive case, popular channels usually have continuity rate of more than 98 percents.

2.3.3 Architecture Choices

As mentioned in section 2.2, there are IP multicast, CDN and Peer-to-Peer live media streaming. And in the Peer-to-Peer live media streaming there are also basically 2 sub-kinds of approaches, namely, tree-based and mesh-based.

Tree-base Approach

In tree-based approach (e.g., [25], [26], [27], [28], [29]), nodes are deployed into multicast trees to deliver data, where each data packet is pushed using the same route. Peers in the tree structure have explicit parent and child relations. Such approaches are typically push-based, that is, when a peer receives a data packet, it also forwards copies of the packet to each of its children. Since all data packets follow this structure, it becomes critical to ensure the structure is optimized to offer good performance to all receivers. Tree-based approach is vulnerable to the failure of peers, particularly those higher in the tree may disrupt delivery of data to a large number of users, and potentially results in poor overall QoS. Further, a large number of peers are leaves in the tree, and their upload bandwidth is not being utilized.

Mesh-based Approach

Mesh-based designs (e.g., [30], [7], [32], [31]) are totally different in concept to the tree-based designs in that they do not construct and maintain an explicit structure for delivering data. Thesis like CoolStreaming [7] argues that, it is better to use the availability of data to guide the data flow in a high dynamic peer-to-peer environment than constantly repair a tree structure. In this approach, peers maintain a set of partners, and periodically exchange data availability information with the partners. A peer may retrieve unavailable data from one or more partners (i.e., failure of data transmission), or supply available data to partners (i.e., success of data transmission). There is no extra redundancy as peers pull data only if they do not already possess it. Furthermore, as there is no explicit parent and children relationship and peers can download unpossessed chunks from multiple partners if one of them is failed, the overlay is robust to failures. Partner leaving only means the number of its partners becomes small and it has to add a new one. But this will not affect the QoS as a peer can use other partners to receive data chunks. A random partner selection algorithm is used on the purpose of trying to fully use the potential bandwidth available between peers.

CoolStreaming [7] is one of the first mesh-based systems. As in Figure 2.3, a CoolStreaming peer, according to the paper [7] written by X. Zhang et al., consists of three key modules: (1) a membership manager, which helps the peer maintain a partial view of other overlay peers; (2) a partnership manager, which establishes and maintains partnership with other peers; (3) a scheduler, which schedules the transmission of video data.

According to the paper [14] written by J. Liu, the key aspect of the design where CoolStreaming differs from Tree-based approach is the lack of a formal structure for delivering data. More explicitly, a live video stream is divided into chunks which have an uniform length, and the availability of the chunks in the buffer of a peer is represented by a buffer map (BM). Each peer continuously exchange its BM with its partners (i.e. try to retrieve the BMs of its partners before schedule), and then determines which chunk is to be fetched from which partner accordingly. An example of the partnership in CoolStreaming is shown in Figure 2.4.

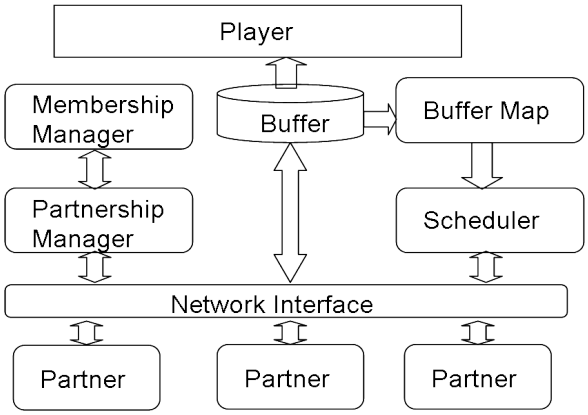

Fig. 2.3 An illustration of a CoolStreaming peer in [7].

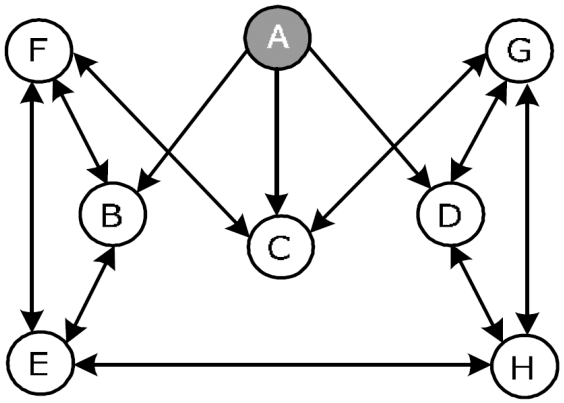

Fig. 2.4 An illustration of the partnership in CoolStreaming [7] (origin node: A).

Partnerships are constantly refined throughout a streaming session.

It is crucial to deliver chunks to peers timely and continuously for a mesh-based approach. In BitTorrent, peers are not synchronized, there is no order when downloading data segments. However, in CoolStreaming, peers are roughly synchronized, and any chunks downloaded after the playback time will be of no use. As shown in Figure 2.5, a trading window is used to represent the range of chunks that a peer is currently interested in. The range of buffered chunks is usually larger than trading window and is represented using a buffer map (BM). After got the BMs of its partners, a scheduling algorithm is executed to decide to fetch which chunk from which partners. It is also crucial to design a good scheduling algorithm for the dynamic and heterogeneous Internet. The scheduling algorithm needs to meet two requirements: the first is get chunks before the deadline of playback and the second is try to deliver good quality streaming to all the peers even in a high heterogeneous scenario. If not

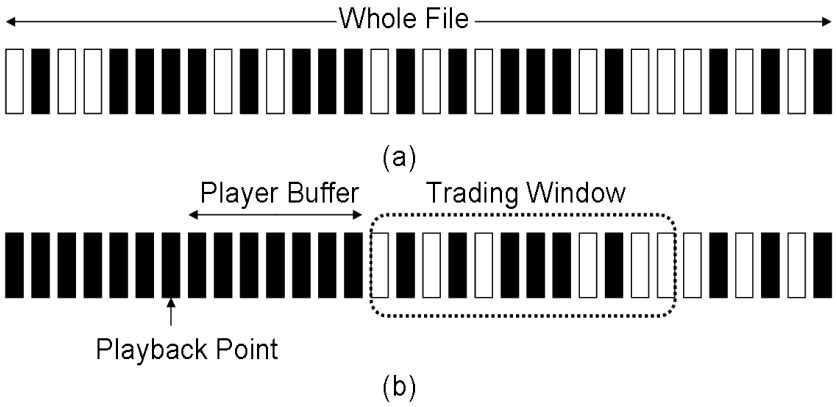

Fig. 2.5 An illustration of Buffer snapshots of BitTorrent and CoolStreaming

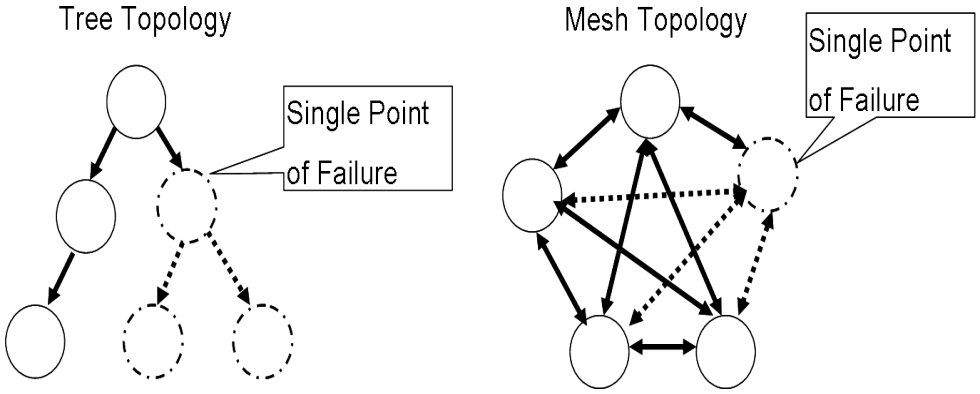

Fig. 2.6 Comparison of Tree-based and Mesh-based.

all of the chunks can be fetched on time, it is desirable to keep the missing ones to a minimum number so as not to jump too many frames and maintain a relatively continuous playback.

2.3.4 Comparison of Tree-based and Mesh-based

Both tree-based structured and mesh-based structureless overlays have been deployed in real Internet. Figure 2.6 is a comparison of tree-based approach and mesh-based approach in a case of single point failure. Tree-based ones have the vulnerability of not failure resilient. Nowadays, almost all the existing systems choose a data-driven approach. But mesh-based approach does not completely overcomes the challenges from the dynamic peer-to-peer environment. Mesh-based approach is simple, and easy to deploy, but at the same time suffers from a slow start-up and long lag[38]. A tree-based system does not suffer from slow start-up delay and long lag, but has to deal with the inherent instability, maintenance overhead, and

bandwidth under-utilization which are not easy to overcome.

2.4 Small Peer-to-Peer Live Media Streaming Overlay Issue

There are a lot of big challenges [14] in the field of peer-to-peer live media streaming. For example, how to stimulate peers to contribute more bandwidth, how to deal with flash crowds and how to deal with implementation issues like NAT and firewall. This thesis will only emphasize on the issue of that the small mesh-based overlay tends to cause QoS degradation. In commercial applications that support live media streaming, every channel forms an overlay. Popular channel forms big overlay and unpopular channel forms small overlay. Usually, unpopular channels have bad continuity, long start-up time and long playback lags [20].

2.4.1 PPlive Observation

PPLive (e.g., [3], [20], [21], [22]) is a derivative of mesh-based model like CoolStreaming and it is now the most popular peer-to-peer live media streaming application in the world. We observed PPLive carefully. There are about 1000 channels in PPlive. When starting to watch the popular channel, the downloading rate can grow to over 2000 Kbps and the continuity is very good. But when watching the unpopular channel, the downloading rate never exceeds 1000 Kbps and it is not stable, even the playback starts, it will stop very quickly and start to buffer again.

2.4.2 Evidences from BEAM and CoolStreaming

In order to support our observation, the experiment results from BEAM [17] and CoolStreaming [7] will be discussed. All suggest that small overlays usually have bad playback continuity. Figure 2.7 is from CoolStreaming, X axis denotes time and Y axis denotes the number of users and continuity index. The blue line denotes the number of users and the green dots denote the continuity index. We can infer from this figure that more users can lead to better continuity. Figure 2.8 is from a thesis of 07 Sigcomm [17], X axis denotes the number of peers and Y axis denotes the jitter rate. We can also infer from this graph that fewer peers lead to worse continuity. Here jitter rate is defined as follows.

$$Jitter\ Rate\ = \frac{(\sum_{i=0}^{N} J_i)}{N}\ where, \tag{2.1}$$

$$J_i = \frac{(\sum_{i=0}^{T} F_i)}{T} \tag{2.2}$$

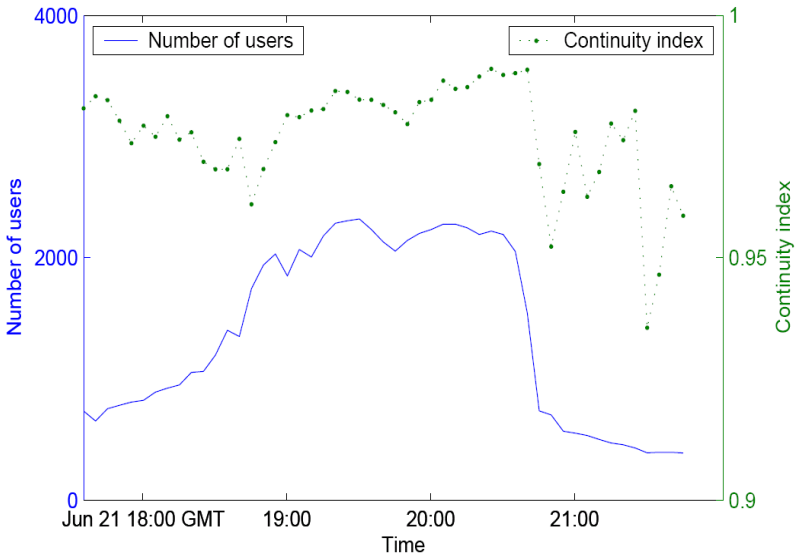

Fig. 2.7 Number of user and Continuity Index overtime.

$$F_i = \begin{cases} 0 & \text{if chunk arrived before media playback} \\ 1 & \text{if chunk not arrived before media playback} \end{cases} \tag{2.3}$$

Here, T and N denote the total chunks in the full streaming session and the total number of peers in the overlay respectively.

$$Continuity\ Index\ =\ 1\ -\ Jitter\ Rate \tag{2.4}$$

2.4.3 Small Live Media Streaming Overlay Resembles Random Graph

Based on the measurement study carried out by L.Vu et al [15]. While the file sharing peer-to-peer systems are known to exhibit Small World [12] behavior, media streaming peer-to-peer systems are less Small World, especially at small channel sizes (with as many as 500 peers) and thus have a worse QoS. Small World networks mean that kind of networks with dense local links and a few distant links as shown in Figure 2.9, which has been proved to be churn resilient and efficient. Small live media streaming overlays with peers less than 500 under current mesh-based scheme basically form random graphs [37], which bring worse QoS. Random graph topologies are neither efficient nor strong to network churns. Peer-to-peer file sharing overlays have small number of peers, usually far less than 500, but because the topology has enough time to evolve, peers are easier to establish a cooperation relation with

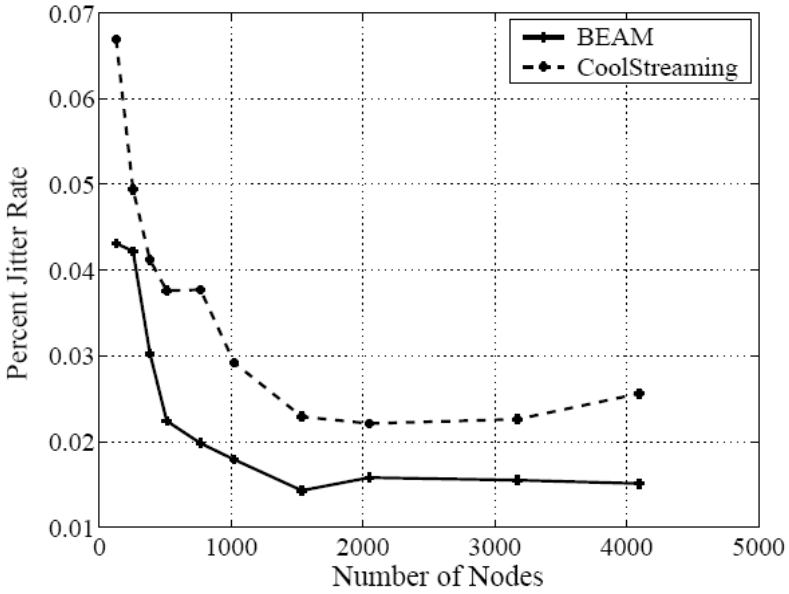

Fig. 2.8 Jitter Rates vs. number of Nodes.

each other. But for live media streaming, because it is time stringent, peers thus are forced to change partners frequently if the current one cannot provide necessary chunks in time. Thus, it is difficult for the small live media streaming overlay peers to establish stable cooperation relations with each other.

2.4.4 The Reasons

As stated in previous section, the first reason is that mesh-based live media streaming overlay with less than 500 peers forms a random graph which is less efficient and thus the upload bandwidth usage is insufficient. Another reason we conjecture is that because small overlays are usually formed by unpopular channels where the contents are not so interesting, peers tend to join the overlay and then leave very quickly which causes heavy churn and results in QoS deterioration.

2.5 Related Works

In SMODA system, a rank-based incentive method is used to improve the QoS of high rank peers. The idea of exploring the remaining online time of peers, and the rank-based incentive method are related to following works.

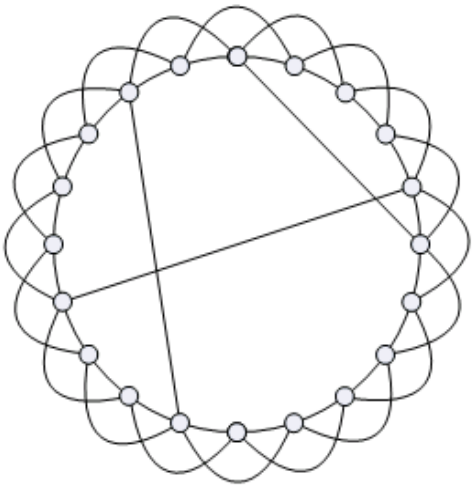

Fig. 2.9 An illustration of Small World Network.

2.5.1 User Model

In the former works such as [19], user model or in other word user character has been extracted from the service logs of actual commercial system and been studied to facilitate new protocol design. Their work specially studied the online duration evolution of end users. Figure 2.10 shows their study result, the X axis is the elapsed online time and the Y axis is the average expected remaining online time, which tells that a peer with longer online time is statistically has a great chance to have a long remaining online time. Please also refer to [39]. They think the main reason of this positive correlation may reside in the popularity-driven access pattern of live video programs. In their paper, this model is actually used to facilitate a scheme that takes advantage of the character that longer online peers are expected to have a longer remaining online time. But their system is based on an environment where there are parents and children relations. The method actually tries to employ an incentive mechanism by allowing children who have long online time to have priority when competing parents.

2.5.2 Pairwise Incentive

Pairwise incentive has been studied by previous works including [16] which deploys a resource and locality awareness pairwise incentive mechanism into the mesh-based live media streaming overlay. Please refer to Figure 2.11. Peers with similar lag and who are physically close to each other are more likely to become partners. Their design works well when resource is abundant and churn is not so heavy. But it is also very likely to result in a network which has a long data distribution route and less-meshed character, causing QoS deterioration in

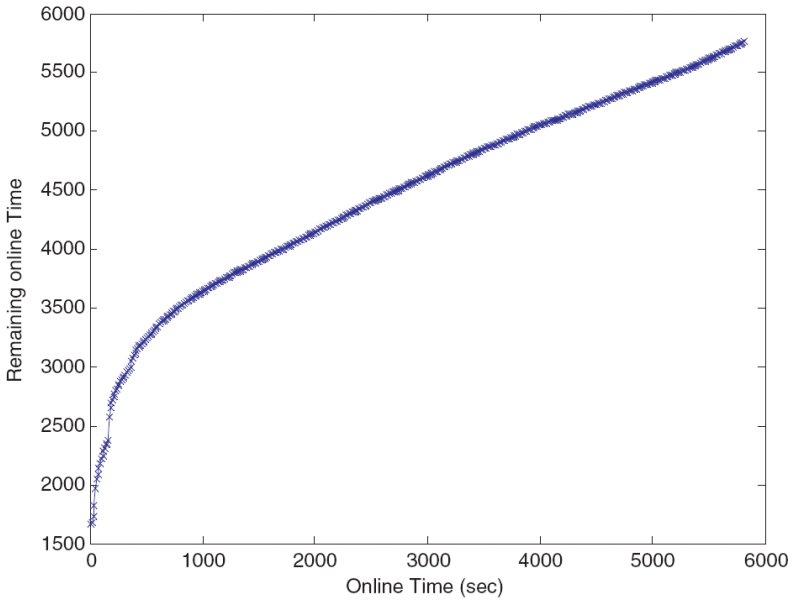

Fig. 2.10 An illustration of the relation between online time and the remaining online time.

heavy churn environment like the scenario that this research targets. However, our research benefits from their works not only because the pairwise method provides a comparison but also because their study provides detailed simulation setup information, from which we have learned a lot.

2.5.3 mTreeBone

mTreeBone [18] is a related work which organizes the stable peers into a tree bone, uses a push-based method for chunk delivery on the tree bone and organizes the unstable peers to meshes which attach to the tree bone. Please refer to Figure 2.12 which is from mTreeBone. The mesh that attached to the tree bone is however uses a pull-based method. We argue that the tree bone of this method may partially alleviates the bad effects that unstable peers brings to the stable ones but it still suffers from an inherent vulnerability of the tree architecture which means it may have deteriorated QoS due to tree bone peer failure. And this method requires switch between push-based method of chunk retrieval and pull-based ones which causes complexity. On the contrary, our goal is to design a protocol which is useful, simple

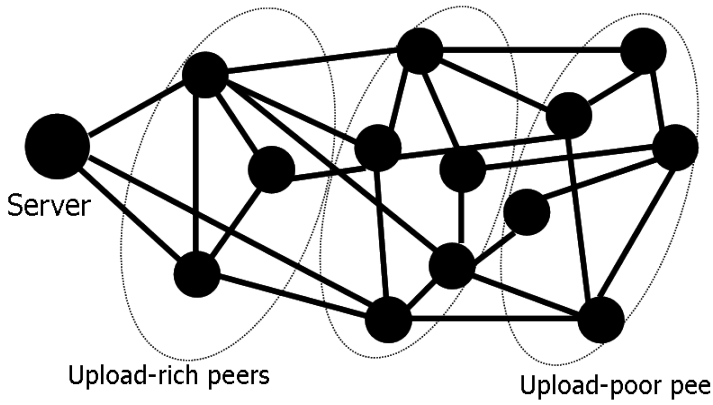

Fig. 2.11 An illustration of a mesh Live Media Streaming overlay using pairwise incentive.

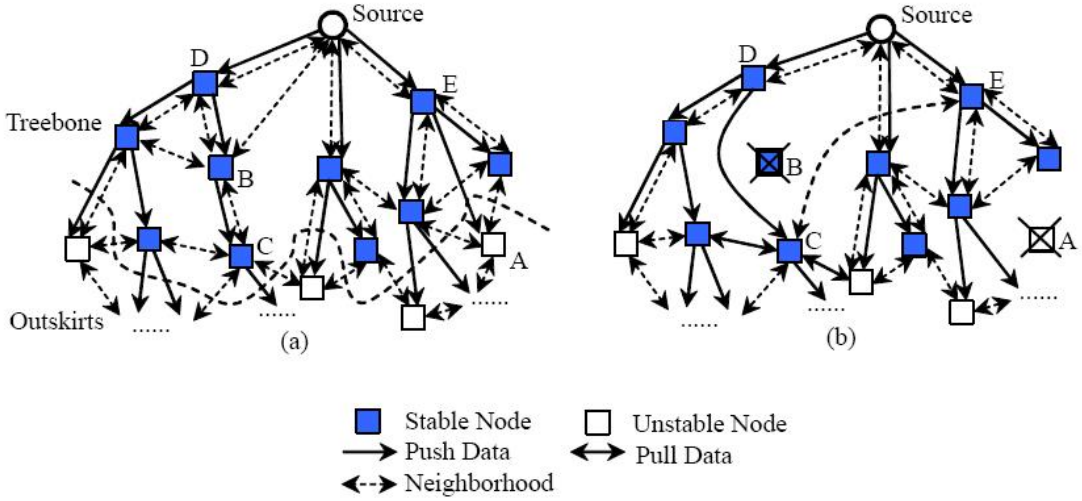

Fig. 2.12 mTreebone framework. (a) A hybrid overlay; (b) Handling node dynamics.

to deploy and robust, not having a lot of inherent vulnerabilities.

Chapter 3

Proposed Architecture

3.1 Overview

SMODA system consists of peers and a media streaming server. The media streaming server is the origin of the streaming content and it also serves as a tracker that assists the peers in the overlay to communicate with each other. SMODA system is designed specifically for the small heavy churn and upload bandwidth insufficient overlay with around 100 peers in it simultaneously.

When a new peer arrives, it contacts the server to submit its IP address and upload bandwidth. This is applicable because the server is alive all the time during the streaming process and serves as a rendezvous point. As the system is small, the server gives all the peers' information, which are now active to the coming peer. The new peer then begins to send partner request to random selected peers picked up from known online peers until it has NP partners. Here, NP is a system wide parameter denotes how many partners a peer should have. Partner formation is also designed specifically for the small and churn environment, which will be explained in detail in section 3.4.4. After added the initial partners, a peer refines its partners periodically, approximately every 30 seconds in SMODA system.

In SMODA system, the server is assumed to be a poor communication ability one which can only serve, for example, two times of the full streaming rate due to the bottleneck of its upload bandwidth. The server streams the contents to a selected number of peers, which are selected from the overlay using designed churn resilient peering scheme just as an ordinary peer does. Except initially, when the streaming starts, the partners of the server are selected from the peers with high upload bandwidth to avoid a bottleneck at the start. In SMODA, different chunks (sub-stream) are delivered to different partners of the server as CoolStreaming has mentioned, trying to disseminate chunks to the overlay as soon as possible, trying to inject different chunks to different peers to encourage them to cooperate together and trying to minimize the re-request of the missing chunks due to the heavy churn. Please refer to section 3.4.1.

Chunks available information is exchanged among partners periodically, approximately ev-

ery 1 second in SMODA system, using a buffer map (BM). Each peer continuously exchanges its BM with its partners and then schedules which chunk is to be fetched from which partner accordingly. A chunk is assumed to contain 0.1-second video data. A buffer map length of 120 seconds, 1200 chunks, can effectively represent the buffered data of a peer. And a trading window of 60 seconds is used to represent the chunks that a peer is interested in. When the data lag is larger than 60 seconds, it is reset to 1. Here, data lag means the time difference between the earliest generation time (second) of the data that a peer still haven't retrieved completely and the latest generation time of data that the server has already generated. The modified design regarding BM and lag reset will be presented in section 3.3 and section 3.4.3.

In order to cope with the assumed scenario, choices and changes have been made. The mainly parameter adjustments and design changes that SMODA system has made are listed below.

Crucial Parameter Adjustments and Design Changes

- Server delivers different substream of chunks which is fixed during a partnership span to different partners and encourages the cooperation among peers.
- Server defines the priority to select appropriate peers, which have sufficient available bandwidth.
- A peering policy that enables adding new partners as early as possible.
- A longer Trading Window, which enables peers' cooperation even when their lags are quite different, is defined

Besides, a method to avoid congestion and a method to deal with the defect of local rarest first scheduling algorithm have also been considered and is considered as future work.

Rank-based Incentive Proposal

A rank-based incentive method is also proposed, which struggles to alleviate the bad effect that the high ratio of unstable and poor upload ability peers bring to the stable ones. The key idea is to divide mesh overlay peers into groups with high rank and low rank based on online duration and upload ability. And encourage dense partnership formations among same rank peers and a few partnership formations among peers with different ranks. As an example, in SMODA, the criteria for rank 0 peers to become rank 1 peers are, be in the system for more than 10 minutes and have an upload bandwidth of more than 250 Kbps.

3.2 SMODA Protocol Parameters

In SMODA, all peers have identical implementation except the source (or server). The server generates data chunks at a constant streaming rate (SR) and sends them to the other peers. Peers then exchange chunks using a pull-based method in order to retrieve all the chunks

Parameter	Value	Description
TW	60	Trading window of a peer(s)
PB	5	Initial player buffer length(s)
SR	10	Server streaming rate(chunks)
NP	5	Number of partners for a peer
NPmax	Refer to section 3.4.4	Max number of partners for a peer
R_{TO}	1	Timeout of chunk request messages(s)
T_D	60	Min lag that triggers a buffer reset(s)
BW	120	Buffer map length(s)

Table. 3.1 SMODA system parameters.

and recover the original media. Every peer has a data buffer, where it stores the chunks it received. There are a lot of parameters for a peer. Table 3.1 lists important parameters used in SMODA system.

3.3 Lag Reset in Unstable Environment

Before go to the next section, it is better to explain the lag reset mechanism in unstable environment. In an extreme unstable and bandwidth insufficient environment, lags of peers become larger and larger constantly. If there is no reset mechanism, a peer's lag may become so large that there is no peer with similar lag, thus impossible for it to get any data from other peers. Please refer to figure 3.1 for an illustration of lag reset mechanism when using a trading window of 60 seconds. One block in this figure means a second of streaming data. Blue one means possession of that one second of data. Server generates data constantly. The data lag for peer P1 exceeds 60 and is reset to 1, 57 seconds of data retrieval is abandoned and the peer P1 focuses on pull newer data from partners. As in this figure, trading window length means the length of range of chunks (here 60 seconds of chunks) that a peer is interested in.

3.4 Design Changes and Parameter Adjustments

3.4.1 Parameter Adjustment for Server

It is crucial for the server to inject chunks into the overlay properly so as to disperse chunks as soon as possible in an extreme churn scenario. SMODA uses a similar method as CoolStreaming mentioned, which argues that in order to protect the server from overwhelmed by requests from its partners, if needed, the following method can be used. The CoolStreaming design assumes there are M partners. The server can set its BM advertising to the k-th partner

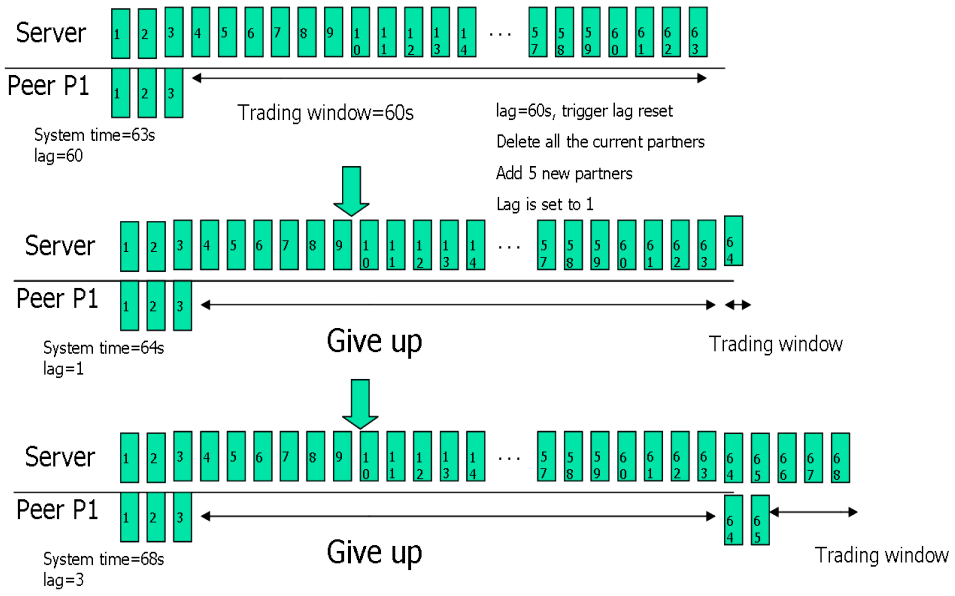

Fig. 3.1 An illustration of lag reset and Trading Window.

as

$$BM_{server}[i] = \begin{cases} 0 & \text{if i mod M !=k} \\ 1 & \text{if i mod M = k} \end{cases} \tag{3.1}$$

That is only the $(i \ mod \ M)^{th}$ partners will request chunk i from the server, and the remaining chunks will then be retrieved from other partners. It did not verify the effect of this proactive control of the server. SMODA uses a similar method which is when a peer becomes the partner of the server, the server issues that peer with a fixed k, the partner sequence number which is not used by the current partners of the server, for it to become the k-th partner. A fixed k means a fixed substream from the server. In a churn scenario, it is not preferable for a partner to receive substream that is changing time to time. Of course, the system is intelligent enough to select a k for the partner so as to receive a substream that is least received by other partners. For example, in SMODA simulation mentions in section 4.3, as the server can afford 2 times of the streaming rate and as the whole stream is divided into 5 substreams, please also refer to Figure 3.2 where one color chunks belongs to one substream, server has 10 partners and every substream of chunks is pulled 2 times equally by the partners most of the time. Using this method, different chunks are injected to different peers and almost all the upload ability of that peer is used to disperse this 1/5 of stream. This method does not only alleviate the burden of the server but at the same time makes chunk disperse more quickly in the overlay. CoolStreaming did not verify this design. And through simulation, the effect of this design is verified to show that in a high churn and upload bandwidth insufficient scenario, it

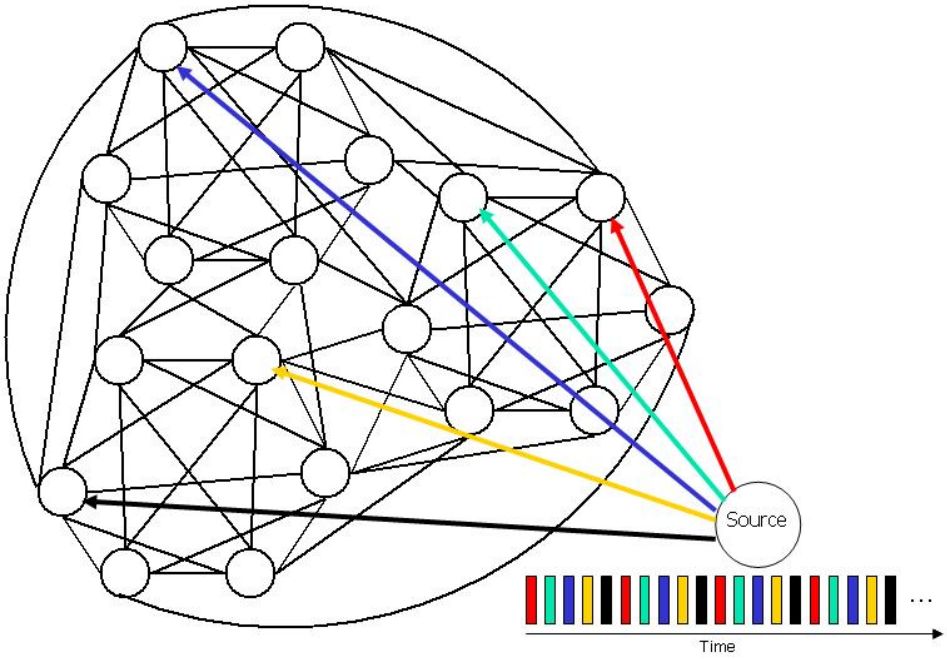

Fig. 3.2 An illustration of how server works in SMODA.

is better for the server to divide the whole stream into substreams so as to have more partners and so as to use the upload ability of those partners.

3.4.2 Avoid Bottleneck at The Start

As mentioned in section 3.4.1, the substream that a server's partner can retrieve from the server is fixed in SMODA so as to keep a relatively stable route for the chunks of that substream to disperse. Given an example of the scenario that the server can afford 2 times of the streaming rate, 2 partners receive the same substream. And if those 2 partners are peers with low communication capability, the whole system will have difficulty getting that particular substream. In order to solve this, it is preferable to have relatively good communication ability partners for the server. In SMODA system, the server (origin peer of the stream) has the priority to select good communication peers as partners, in simulation, to be specific, upload bandwidth abundant peers. The effect of this design is also verified through simulation.

3.4.3 A Longer Trading Window

It is also very important to design a good bit map (BM) exchange scheme, including when to reset a peer's lag. Here, the lag means data lag. The calculation of data lag is shown in section 4.5. When the lag is bigger than 60, it is reset to 1 in SMODA system. The length of

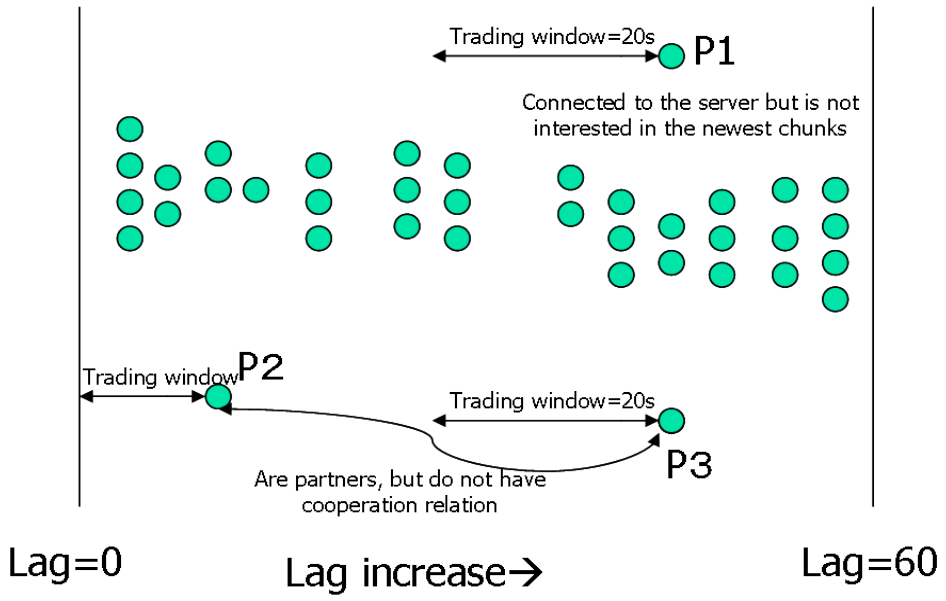

Fig. 3.3 An illustration of the relation between data lag and trading window size when trading window size is 10 seconds.

the BM and the size of the trading window are also important parameters. In SMODA, the BM length is designed to represent 120 seconds of streaming data. Because the lags among peers are smaller than 60 seconds and 120 seconds can represent the data that a receiver peer of a BM is interested in. The following Figure 3.4 depicts the relation between data lag and trading window size in SMODA system. A trading window of 60 seconds and lag reset point of 60 second means that peers are always interested in the latest chunks including the ones that is freshly generated by the server and peers can always have common interested chunks. When in a big and stable overlay, the trading window can overlap each other even using a small size, but it is desirable to use a big size to avoid the problem shown in Figure 3.3.

Figure 3.3 shows the scenario of using a trading window which is set to 20 seconds. As an example, PULSE [16] uses a trading window of 31.25 seconds which we think is not long enough. If peers are sparse and lags are much different, there is a chance that peers do not interested in the partners' data and cannot form a cooperation relationship. This is a less severe problem when peers' data lags can overlap each other in a big overlay because chunks can be relayed to the peers with long data lag. If a peer is a partner of the server and has a big data lag, it will not be interested in the freshly generated chunks of the server, which means it can not fulfill its obligation of delivering those freshly generated chunks to other peers. The trading window is enlarged to 60 seconds to solve those problems.

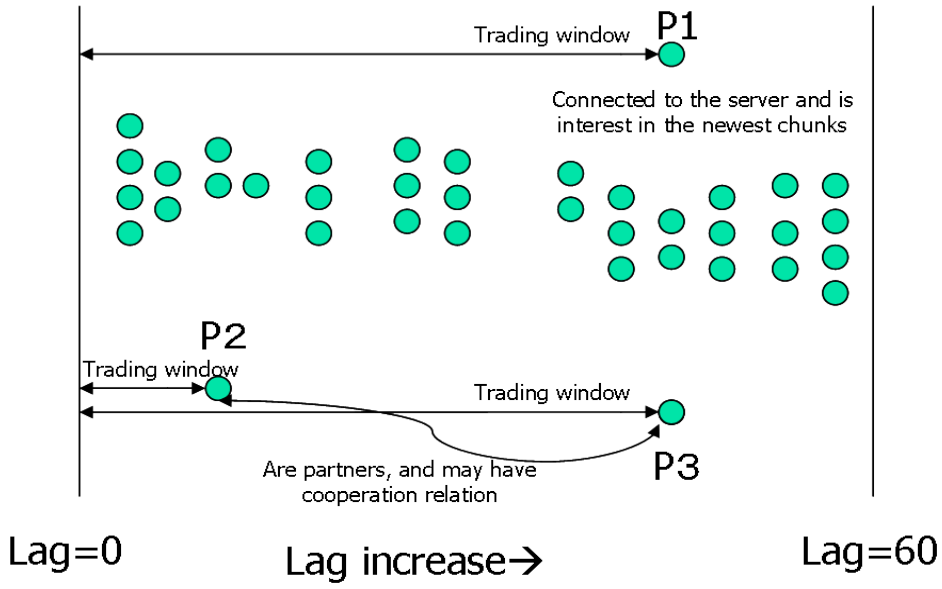

Fig. 3.4 An illustration of the relation between data lag and trading window size when trading window size is 60 seconds.

3.4.4 Churn Resilient Peering Scheme

In this section, a novel churn resilient peering scheme specific for small heavy churn overlay is presented, which uses a flexible policy to make partner formation easier and quicker. Because it is crucial to add a partner as soon as possible in heavy churn environment.

Peer Join

As CoolStreaming, when a peer joins the overlay, it contacts the server to submit its IP address and upload bandwidth. As the system is small, the server issues a list of all the peers' information, which are now online. The new peer then begins to send partner request to random selected peers picked up from the list until it has NP partners. A gossip-based protocol such as SCAMP [43] is assumed to be employed in the system to distribute the information such as heart beat information among peers. In a typical gossip algorithm, a peer sends a newly generated message to a set of randomly selected peers; these peers do similarly in the next round, and so do other peers until the message is spread to the whole overlay. There are several pioneering works that deploy gossip protocols into the live media streaming system, please refer to CoolStreaming [7] if you want to know the details of how gossip protocols works.

Select a partner condidate

Traditionally, when selecting a partner candidate, a few requirements need to be satisfied. A candidate is picked up randomly from the current known peers. The candidate should have following characters. First, the picked up peer should not be already a partner; second, the total number of partners should be less than NP; third, in SMODA simulation, the picked up peer should have a state of true.

Partner Request

When the candidate for partnership is selected, the peer who wants to add the selected candidate as a partner sends out a partner request message to that particular candidate. The candidate peer receives the request and sends out a partner acknowledge message if its total number of partners is less than NPmax and at the same time adds the sender as a new partner, regardless of if itself has send out a partner request message. NP which means number of partners is a system wide configuration parameter for all the peers denotes the desirable number of partners that a peer should have. And, NPmax is a new parameter which equals to NP in case of poor communication ability peers and equals to $uploadbandwidth/150$ if $uploadbandwidth/150$ is larger than NP in SMODA design. If the total number of partners at that particular time is NPmax, it then sends out a partner decline message to the sender of the request message. After received the partner acknowledge message, the sender adds the candidate peer as a new partner and if the returned message is a partner decline message, the sender peer tries to select another candidate again using the partner selection method stated in section 3.4.4 and then sends out another partner request message until it has tried for RETRY times. After a new partner is added, a TCP-friendly rate control protocol is assumed to be adopted to deliver chunks and exchange BMs and scheduling results. The BM information and scheduling results can be piggybacked to the data packets to achieve low-overhead effect.

Peer departure

In SMODA simulation, a peer departs from the system gracefully. Before its departure, it issues a departure message and this message is spread to the whole overlay using a gossip protocol. Also, it generates partner delete messages for its partners, which will be delivered to its partners and when a partner receives the message it deletes the departing peer from its partner list and triggers a new search for a possible partner to replace the departed one. Ungraceful departure is not considered in SMODA prototype and will be addressed when deriving a true application from this prototype.

Partner Refinement

In SMODA prototype, a peer refines its partners periodically, 30 seconds once. If the number of partners is less than NP, it does not delete a parner but only adds a new one using the partner selection procedure in section 3.4.4. And if the number of partners is more than NP, it only deletes the one of its partners who has the lowest score but does not try to add a new one. There is a case that the partner request process is activated less than 30 seconds ago due to lack of partners and a new partner is added, in such situation, refinement is not performed on the purpose of not causing extra churn to the system. There is a field for each partner which denotes the total chunks it has received from that partner, which will be set to 0 after the refinement completes. Note that, when added a new partner, the number of received chunk field attached with all its partners are reset to 0. The score for a peer j, in our prototype, is calculated using function $S_{i,j} + S_{j,i}$, where $S_{i,j}$ is the total number of chunks that peer i retrieved from peer j after the last time that field is set to 0. Intuitively, as a partner can be either a supplier or a receiver, it is preferable to delete a partner that does not contribute to this peer and at the same time does not rely on this peer. Here, both directions are taken into consideration.

Partner Consistency

In a heavy churn environment, there is a chance that partner information is not consistent. For example, peer P1 may think peer P2 is a partner but P2 does not have a partner of P1. In order to avoid this, when sending BM according to BM request, peers check if the receiver is a partner. If the requester of BM is not a partner of this peer, this peer sends a BM no message to indicate a partner information inconsistency. And after receives a BM no message, the receiver peer deletes the sender who is not a partner.

Issues Specific to Small Overlay

A flexible partner scheme including not lock up and more partners for powerful peers is used. Please refer to Figure 3.5 and Figure 3.6 for an example and more details. In the CoolSteaming design, the number of partners a peer can have is fixed to a system wide value NP. But in the system like PPLive and other commercial deployments, peers with very big upload ability can have much more partners than ordinary peers. There are measurement studies show that actually the PPLive system relies greatly on the peers with big upload ability. In SMODA system, in order to make adding a partner easier, a peer with high upload ability is also allowed to have more partners passively, which means it does not search for a partner when its number of partners is more than NP, but accepts the partner request if the total partner number is less than NPmax which is calculated as in section 3.4.4. And when it refines the partner relationship in a situation that total number of partner is more than NP and less than NPmax, it deletes one partner using the refinement algorithm but does not try

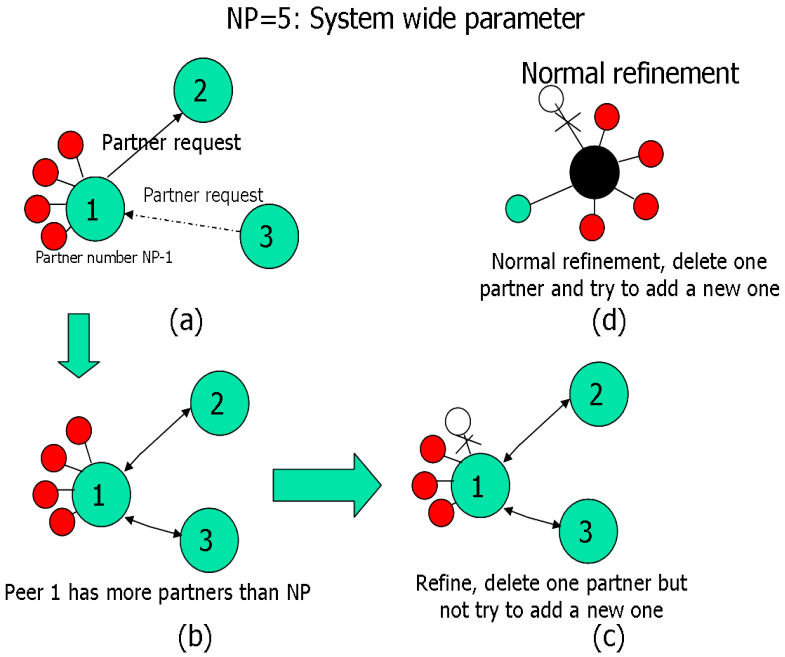

Fig. 3.5 An illustration of one of the modified point of the legacy peering scheme.

to add a new one. Figure 3.5 depicts this mechanism. The purpose of this design is to allow more peers in the system to have the ability to accept new partner requests immediately and thus reduces the average time to search for a new partner. Naturally, when a peer has sent out a partner request but still hasn't got any partner acknowledgement it answers no to the incoming partner reqests, but we argue that in a situation of high churn, this may increase the time of finding a partner for the peer who sent the partner request. This increase trend can become accelerated when more and more peers have sent out partner requests and do not accept new partner requests from other peers and eventually, the whole system will freeze. To avoid this kind of situation, a rule that temporarily accepts partner request even when itself has sent out a partner request is applied and because this can cause the problem of more than NP partners, when next refinement comes, as the situation of partner number is more than NP but less than NPmax, the peer deletes one partner but does not try to find a new one. Figure 3.6 depicts this mechanism in detail.

3.4.5 Other Design Changes and Expected Fture Improvements

It is really easy for a congestion situation to occur in the heavy churn environment. How to avoid congestion becomes an urgent problem. For example, a peer sends a chunk request for chunk C1 but still hasn't received the chunk even after 1 second, and the next time, it runs the scheduling algorithm and sends a chunk request for the same chunk C1 to another

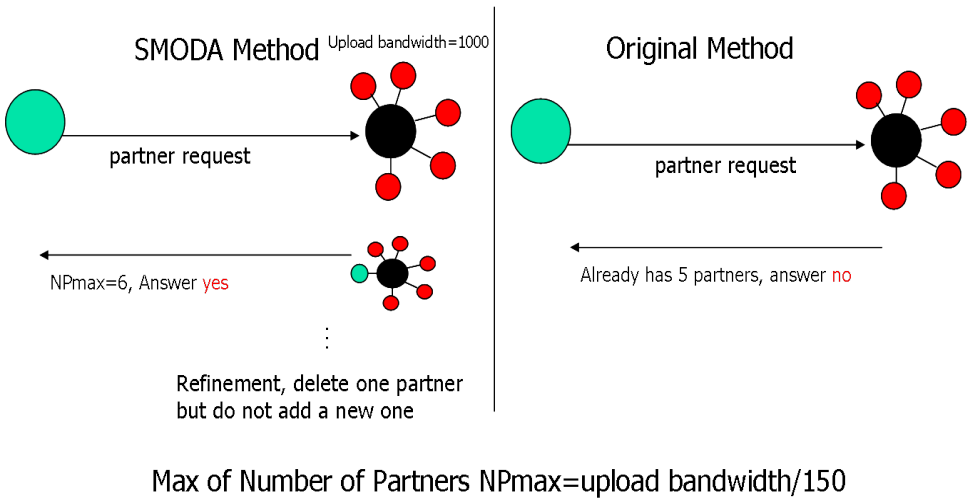

Fig. 3.6 An illustration of another modified point of the legacy peering scheme.

partner. Thus, afterwards, it may receive 2 copies of the same chunk C1, which is not only a waste of the bandwidth, but also makes the scheduling algorithm less useful. In SMODA, the timeout of chunk request messages(s) R_{TO} is set to 1 second which means that a chunk request for chunk delivery which is sent 1 second ago will be dropped to avoid congestion.

The local rarest first scheduling algorithm is an excellent scheduling algorithm adopted by CoolStreaming. It is one of biggest contribution that CoolStreaming made. However, in a high churn scenario, when a peer with a big data lag adds multiple partners with smaller data lag, there is a chance that there is difficulty for this peer to retrieve the oldest chunks as those chunks have already been retrieved by partners and are not the rarest chunks now. In such situation, the peer will download the rarest chunks and its data lag will become bigger and bigger as it still has not retrieved the older chunk that it needs. The overall QoS deteriorates as that peer cannot retieve the chunk that it needs. This will cause that peer to be lag reset and thus causing extra churn as it need to reset all the partners. Trying to solve this, I am thinking of picking up this situation only and trying to modify the rarest first scheduling algorithm to overcome this problem. Please refer to [42] for a research example of scheduling algorithm.

3.5 Rank-based Incentive

Until now, the goal is to improve the overall QoS of the small unstable and upload bandwidth insufficient overlay. When considering how to improve the QoS further, there is very little left that still can be done. For example, the upload ability of peers is fixed and cannot be changed. How to further improve the QoS? If peers with long online time are

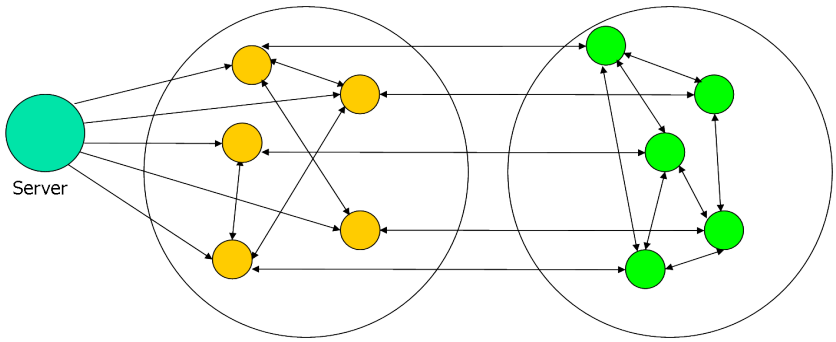

Fig. 3.7 An illustration of Rank-based incentive design.

provided with better QoS, peers will be incentivized to stay longer and this will make the swarm more stable and larger which we think maybe is related to QoS improvement. Besides, rather than no one could watch the stream comfortably, it is better for at least a few ones to be able to have comfortable quality of service. Based on this consideration, a rank-based incentive method that tries to provide long online time and not so poor communication ability peers with better QoS is proposed. Please refer to Figure 3.7. In this proposal, peers are divided into groups with high rank and low rank based on online time duration and upload ability. For example, in our simulation, peers, which have more than 10 minutes of online time and an upload bandwidth of more than 250 Kbps, have a high rank of 1. This proposal encourages dense partner formations among peers with same rank and a few partner formations among different ranks. Thus, the high rank stable and bandwidth abundant peers form a group with strong relations (dense links) and server delivers chunks to high rank group first. Thus the QoS of that group is improved.

Peer join and rank up

When a peer joins the overlay, it records its online time and sets its rank to 0. After staying for more than 10 minutes, a peer with upload bandwidth of larger than 250 Kbps reset its rank to 1, which means it is now more likely to find partners in the rank 1 group. The new incoming peer always first joins the lower rank group and after 10 minutes bandwidth-abundant ones join the higher rank group.

Partner Selection

As shown in Figure 3.8, the partner request algorithm when using the rank-based incentive is different from the partner request algorithm stated in section 3.4.4. First, a candidate is picked from known peers randomly as the legacy method and if the candidate is with the same rank of this peer, a partner request message is sent out to the candidate peer as the destination. However, if the ranks are different, a random integer between 0 to 99 is

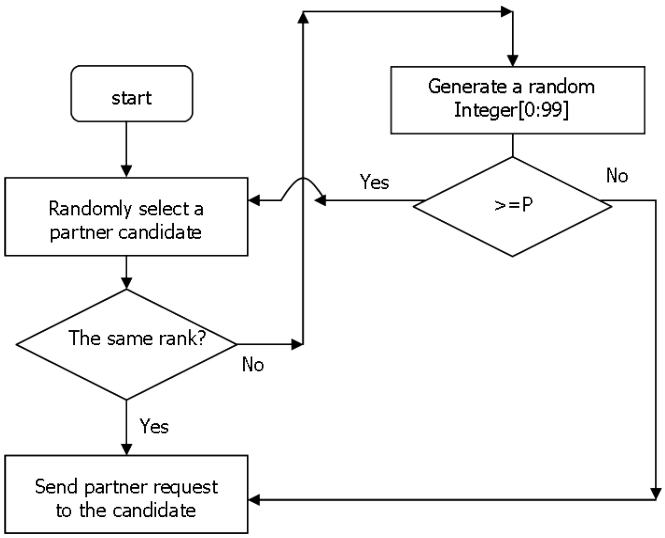

Fig. 3.8 An illustration of the new partner request algorithm.

then generated. When the generated integer is bigger than or equal to P, the peer selects another candidate peer again randomly, and if the integer is smaller than P, a partner request message is sent out regardless of the rank to the candidate peer as the destination. Aware that in a heavy churn environment, partners get updated frequently in the case of a partner leaving, the case of lag reset and the case of partner refinement. With this kind of partner selection method, peers with long online time can have a more strong relation between each other. P is an adjustable parameter of the system. When using a small P, the connections among same rank peers will be dense while the connections among different ranks will be sparse. When using a big P, the connections among same rank peers will be not so dense and the connections among different ranks will be a little denser and thus more resemble the traditional non-rank method. This parameter P is an important design parameter and its impact has been examined through experiments.

Consideration of This Design

When trying to design a peering scheme that can first guarantee the QoS of the high rank peers, a few aspects need to be taken into consideration.

(1) The first aspect is to guarantee the robustness of the designed peering scheme over a wide range of percentage of stable peers. For example, let's consider the situation that there are very few high rank peers (e.g., less than 5 peers). If we design a peering scheme that divides peers into groups clearly and does not allow chunks delivery from the low rank group peers to high rank group peers, the high rank group will become a bottleneck and the whole system service continuity will decrease. Based on this

consideration, when there is a partnership between a high rank peer and a low rank peer, SMODA system allows chunk retrieval from the low rank peer.

(2) The second aspect is to design a peering scheme that is robust to peer churn. The connections among same rank peers should be dense, if use a pairwise incentive method, the connections among peers will be less-mesh and have a hierarchy character, which means if some peers middle in the hierarchy leave, there could be difficulty for peers high in the hierarchy to connect to the peers low in the hierarchy. Which means when considering design an incentive method, we should not forget to keep the mesh character of the overlay as much as possible.

(3) The third aspect is to design a criterion to divide peers, here, 10 minutes of online time and bandwidth of more than 250 Kbps is used because through simulation we are confident that the high rank peers can really have good quality of streaming when using this criterion. Of course, this criterion should change according to streaming rate.

The peer-to-peer system is a self organized system in which server and peers cooperate together to serve a large number of peers by leveraging peers' upload ability. The protocol needs to be designed carefully and minor defect may cause severe service quality deterioration. Also, a peer-to-peer protocol design needs to be carefully evaluated by computer simulation. A design may seem applicable but result in severe service quality deterioration.

Chapter 4

Simulator Design and Simulation Setup

We developed an event-driven simulator using similar architecture as PeerSim ([23] and [40]) for evaluation. We did not use PlanetLab [2] for performance evaluation as we do not have access to the PlanetLab platform.

4.1 Cycle-based and Event-driven

Basically, there are two types of computer simulation methods to accomplish the simulation. The fist type is cycle-based and the second type is event-driven.

4.1.1 Cycle-based

The cycle-based model is simpler than the event-driven one. It lacks transport layer simulation and explicit time management. In other words, peers are called periodically, each cycle, in some sequential order to perform protocols. Please refer to Figure 4.1, where P1 indicates peer P1 is to be executed. Each cycle, peers are executed in different sequential order.

4.1.2 Event-driven

The way of peers are called in the event-driven method differs greatly from the cycle-based one. The events in this model need to be scheduled explicitly since there are no cycles. The event-driven one is more precise as it takes into consideration of the transport layer delay and manages the time in a smaller interval.

4.1.3 Comparison

Cycle-based is simpler and more light-weighted than the event-driven one and thus can support more peers. But as we are targeting the small overlay with less than 500 peers and as we are trying to simulate the transport layer, the simulator that the author wrote has been changed from the cycle-based model to the event-driven model which is more complicated,

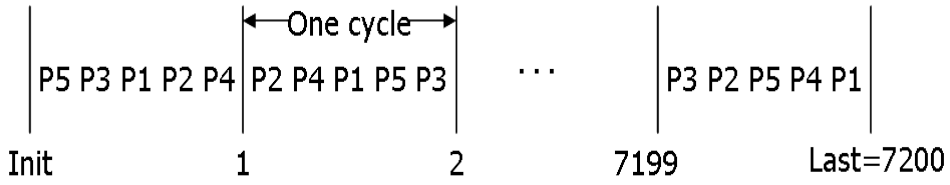

Fig. 4.1 An illustration of Cycle-based simulation.

can support transport layer simulation, and more flexible when employing the protocols.

4.2 Simulator Design

4.2.1 Overall View

Source Code of The Event-driven Simulator

- Engine.java
- Heap.java
- Import.java
- Route.java
- Client.java
- Server.java

The above is the overall view of the developed simulator. The Import.java is responsible for importing BRITE generated topology file into the simulation and selecting the possible overlay peers. After the importation, the Route.java takes care of calculating the delay between arbitrary pair of overlay peers using Dijkstra algorithm. The Client.java simulates an ordinary peer, it is the prototype of an ordinary peer and is in charge of dealing with the incoming events from the engine and generating new events which will be put into the event heap of the event-based engine. An event here is consists of the following fields. Please refer to figure4.3, An event has a time stamp field, the engine always takes out an event that has the smallest time stamp and delivers it to the relevant peer for execution. The Server.java is the implementation for server, actually, Server class is a subclass of Client class, but it also has its own special functions. Finally, the Engine.java is responsible for initializing all the peers at the start, then each time taking out one event from the event heap which has the smallest time stamp, calling the destination peer of that event and also takes care of observing all the active peers periodically.

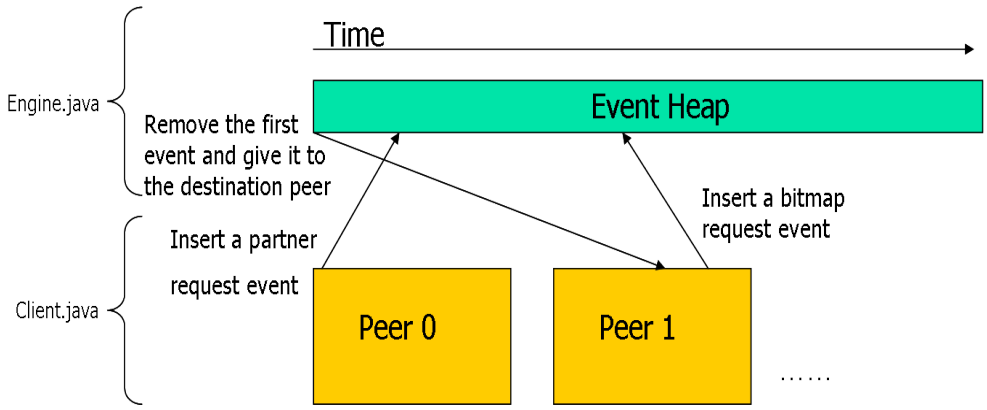

Fig. 4.2 An illustration of Event-driven simulation and event heap.

4.2.2 Event-based Engine and Event Heap

The Figure 4.2 explains how the event-driven engine and event heap work. Initially, peers are called by the event-driven engine. Let's use peer P1 as an example, after the first initialization, peer P1 sends out a partner request event, the next scheduling event with the destination of itself and the next refinement event also with the destination of itself to the event heap. All the peers do the similar function. Each time, the event-driven engine takes out an event with the smallest time stamp and calls the destination peer to process the message included in the event. In this way, time axis is managed and the simulation process is driven by events that peers generated. When an event is inserted into the event heap, the event heap actually looks at the Dst ID field and the Src ID field, modifies the time stamp by adding the delay between source node and destination node to the time stamp so as to simulate the Internet delay in this simulator. The event composition will be explained in detail later. The reason of using a heap for event is that we are trying to reduce the complexity of calculation. Neither the move out nor the insert execution has high calculation complexity. The move out calculation has an order of O(lgn) and the insert calculation has the order of O(lgn) where n is the total number of events in the event heap.

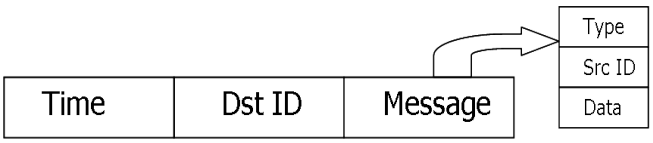

Fig. 4.3 An illustration of heap event fields.

4.2.3 Event Format and Message Types

Message Types

- INIT
- SCHEDULE
- REFINE
- PARTNERREQ
- PARTNERACK
- BITMAPREQ
- BITMAPACK
- JOIN
- LEAVE
- PARTNERRJT
- GET
- SEND

There are many types of messages, please refer to the above item box. In the SMODA prototype, a peer which is an entity of Client.java deals with all those incoming messages. An event is composed with following fields, please refer to Figure 4.3. Including, a Time field which is the time when the event-driven engine takes it out of the event heap, a Dst ID field which denotes the destination peer that the contained message should be delivered to and a Message field which contains a message type field, a Src ID field and a Data field.

4.2.4 Topology

The BRITE universal topology generator in the router mode is used to model the physical network topology. It uses Waxman model to generate the topology, which uses a random node placement and a growth type of incremental. In SMODA simulation, the delay information is calculated from the generated topology file.

Upload bandwidth	0 Kbps	250 Kbps	500 Kbps	750 Kbps	1000 Kbps	1500 Kbps
% of peers	0	40	20	20	20	0

Table. 4.1 A typical distribution of bandwidth setup.

Upload bandwidth	0 Kbps	250 Kbps	500 Kbps	750 Kbps	1000 Kbps	1500 Kbps
% of peers	0	20	40	20	20	0

Table. 4.2 Another distribution of bandwidth setup.

4.3 Simulation Setup

Simulations are done to simulate SMODA rank-based incentive method and compare with random partner selection algorithm used by DONet/CoolStreaming and a pairwise incentive algorithm which deploys a 1 to 1 incentive. As said in section 4.2.4, the BRITE universal topology generator is used to generate 2000 nodes and 200 peers are attached randomly to the generated nodes. The bottleneck is assumed to be at the last mile, not in the backbone. Actually, only the upload bandwidth ability is considered as the bottleneck in this system. The download speed is not considered as a bottleneck because 500 Kbps of streaming rate is not so high for the modern broadband access and peers in our system can seldom have a peak downloading rate higher than 1000 Kbps, which means that the downlink is seldom a problem. The delay between an arbitrary pair of nodes is typically tens of milliseconds. The uplink speed of the media server is 1000 Kbps which can support 2 full streams at the streaming rate of 500 Kbps. The peers in the small overlay which typically has around 100 participants are assumed to be of heterogeneous bandwidth classes. Please refer to Table 4.1 for a typical distribution of bandwidth setup. All the other experiments use a bandwidth setup shown in Table 4.1 except the first 3 proposals' evaluation(the first 5 experiments) which use a setup as shown in Table 4.2. We consider a live media streaming session duration of 120 minutes, which is a typical length of a movie and the full size in the streaming session is 500 Kbps*7200 seconds, approximately 450 MB. In order to simulate a scenario of bandwidth insufficiency, the bandwidth is set to be relatively low.

4.4 Churn Simulation and Initial Setup

Peers are assumed to join the overlay in the first second. After a peer has joined, it issues a schedule event and a refine event which will be forwarded to itself in the near future. When a schedule event has arrived, the peer generates chunk request events to corresponded peers, a new schedule event and if the number of partners it has is less than NP, a partner request

message.

Churn is modeled using Markov chain. When a peer is generated, two parameters are given. One is leave possibility parameter LEAVEP and the other is join possibility parameter JOINP. Engine checks the state of that peer every 1 second. If the state is false, the engine generates a random integer between 0 to 9999 and if the JOINP is bigger than the generated integer, the engine generates a JOIN event, delivers it to event heap with the Dst ID of that peer and changes the peer's state to true. On the contrary, if the state is true, the engine generates a random integer between 0 and 9999 and if the LEAVEP is bigger than the generated integer, the engine generates a LEAVE event, delivers it to the event heap with the Dst ID of that peer. In order to keep a relatively constant number of online peers, the JOINP and the LEAVEP is set to be the same, and peers can join and leave the system multiple times. After a peers leaves, all the data it has received and all the partners it has will be deleted. Initially, half of the 200 candidate peers are set to be online, which means the number of online peers is always around 100.

4.5 Evaluation Parameters' Calculation Method in this Simulator

Here, I would like to define the method to calculate the evaluation parameters used in this simulator. This simulator uses an initial player buffer of 5 seconds, which means if the continuous data the peer received after join the overlay is more than 5 seconds, it starts the playback. The continuity index at a certain time t is calculate using the following equations 4.1. N means the total number of peers now online and P_i means the total number of peers that is now playing the contents. Using a bigger initial player buffer leads to a better continuity index but at the same time causes a long start-up delay. Here, as this simulator uses an initial player buffer of 5 seconds, if the playback is once stopped, it will not resume until there is at least 5 seconds of continuous streaming data.

$$Continuity\ Index\ = \frac{(\sum_{i=0}^{N} P_i)}{N}\ where, \tag{4.1}$$

$$P_i = \begin{cases} 0 & \text{if the playback for peer i is not interrupted} \\ 1 & \text{if the playback for peer i is stopped} \end{cases} \tag{4.2}$$

The start-up delay for a particular peer is the time when it first starts to play subtract its online time, which can be calculated using the equation 4.3.

$$Start-up\ Delay\ = T_{play} - T_{online}\ where, \tag{4.3}$$

T_{play} means the system time when the peer started the playback and T_{online} means the system time when the peer joined the overlay. The average data lag lag_d of the system is the average of all the peers data lags. The playback lag lag_p equals to the data lag lag_d plus the seconds

of data that the peer's player buffers. Every 1 second, a peer's data lag lag_d is calculated using the following algorithm. If previously, the chunks of which the generation time is before and including a system time of C(second) have been completely retrieved. Please refer to the following item box.

Data Lag Calculation

lag_d=lag_d+1
while (the C+1 second chunks have been completely retrieved){
    lag_d=lag_d-1
    C=C+1
}

$$lag_p = lag_d + b_{player} \ where, \tag{4.4}$$

b_{player} is the continuous seconds of streaming data the player buffered at that time.

Chapter 5

Simulation Results and Discussion

5.1 Simulation Results for Design Changes and Parameter Adjustments

5.1.1 Parameter Adjustment for Server

In SMODA, chunks disperse method that the server delivers different chunks (sub-stream) to different peers is used. This method has been mentioned by CoolStreaming, please refer to section 3.4.1. This method can encourage cooperation among peers and partially avoid the waste of the server's upload bandwidth. Two experiments have been done using a bandwidth distribution shown in Table 4.2 to examine the effect of this proactive server control. LEAVEP and JOINP were set to 20 in these two experiments. The first experiment used a traditional method, the server had an upload ability of 2500 Kbps and it delivered chunks to 5 peers directly. In the second experiment, the server had an upload ability of 1000 Kbps and it delivered chunks to 10 peers directly. Using proactive server control, the server delivered 2 copies of the chunks to the 10 partners, every partner received 0.2 second of video data per second. Peers exchange chunks with each other to retrieve a full copy. Result for the first experiment is shown in Figure 5.2 and result for the second experiment is show in Figure 5.1. Comparison of those two figures shows that the method of not using proactive server control leads to unstable continuity index which degrades to less than 0.1 sometime while the method of using proactive server control results in stable continuity index all the time. The average continuity index for not using proactive server control is 0.723 and the average of using proactive server control is 0.829. We conjecture that it is because when using the proactive server control, server can have more partners and those partners can fully use their bandwidth to stream the substream (1/5 of the whole stream)to more other partners thus makes chunks spread more quickly and at the same time avoids the phenomenon of some chunks are fetched multiple times while others are fetched much fewer times or even not fetched. As discussed in section 3.4.1, experiment results confirm that the proactive server control leads to superior

performance.

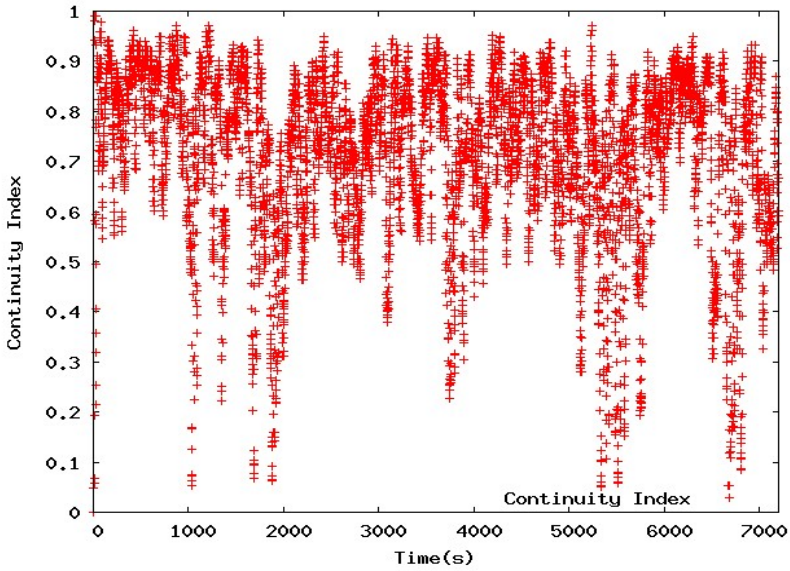

Fig. 5.1 Continuity Index of not using server proactive control.

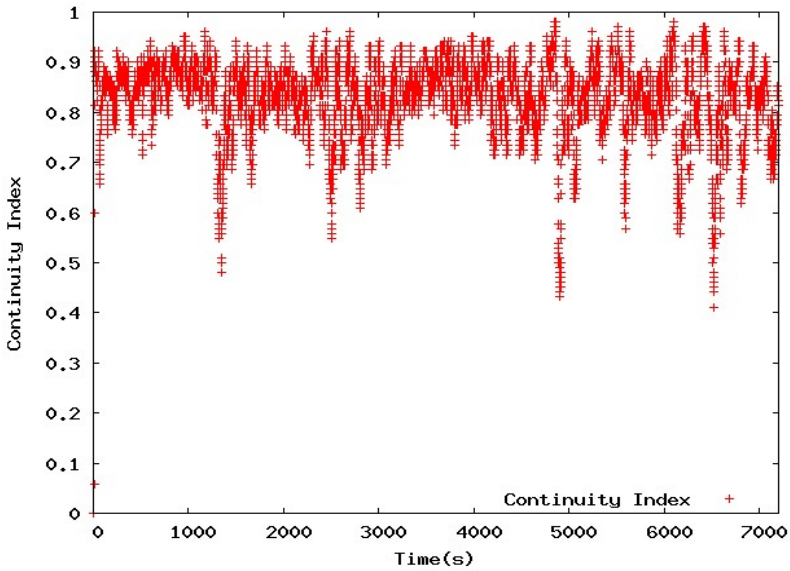

Fig. 5.2 Continuity Index of using server proactive control.

5.1.2 Avoid Bottleneck at The Start

As discussed in section 3.4.1, the quick disperse of chunks to peers from the server is essential for mesh-based system. SMODA system has adopted proactive server control method where server delivers different chunks (substream) to different partners. If all the receivers of a particular substream are poor upload bandwidth peers, all the peers will have problem retrieve that particular substream. To solve this, it is essential for the server to have bandwidth-abundant partners. As in other researches, peers connected to the server are required to have a minimum bandwidth of 500 Kbps in SMODA system. The Figure 5.3 shows an experiment result without this requirement. The experiment bandwidth distribution is shown in Table 4.2. LEAVEP and JOINP were set to 20 in this experiment. The continuity index deteriorated when the time was around 2300 second and 4500 second etc. And the average lag mounted at the same time to a very high number of nearly 50. We examined the logs of this experiment and the reason behind this performance deterioration is the bottleneck at the start. All the two receivers of a particular substream were peers with poor upload ability and all the system to be hungry with that substream, thus caused the deterioration of the overall continuity. All the peers could not get that substream, resulting lags to go up quickly to more than 40. After the program had been modified to avoid this bottleneck, another experiment was done and the results is show in Figure 5.4. LEAVEP and JOINP were set to 20 in this experiment. The continuity index is relatively smoother and the lags do not fluctuate so much and is below 20 most of the time. There are still small fluctuation in this figure around 2300 second and 6000 second. Investigation of the logs shows that the fluctuation is either caused by the local rarest first chunk retrieve policy or caused by poor partners of the peer which connect to the server. This is a future work and will be solved when there is enough time.

5.1.3 A Longer Trading Window

In a high churn and bandwidth insufficient scenario, when the lags are quite different with each other and the trading window size is small, peers usually cannot cooperate with each other. Another problem is that when a peer is connected to the server while having big lag and small trading window, it will not have interest to the newest data that the server generated and thus can not fulfill the obligation of disperse particular chunks generated by the server as soon as possible. The Figure 5.5 shows the experiment result when the trading window size is 20 seconds. As the previous experiments, the bandwidth distribution shown in Table 4.2 is used. LEAVEP and JOINP were also set to 20 in this experiment. The continuity index became 0 around 4500 second and never came back. Peers do not cooperate with each other when their data lags are quite different and this noncooperation caused the overall deterioration. In SMODA system, the trading window size is set to 60, which is the also the

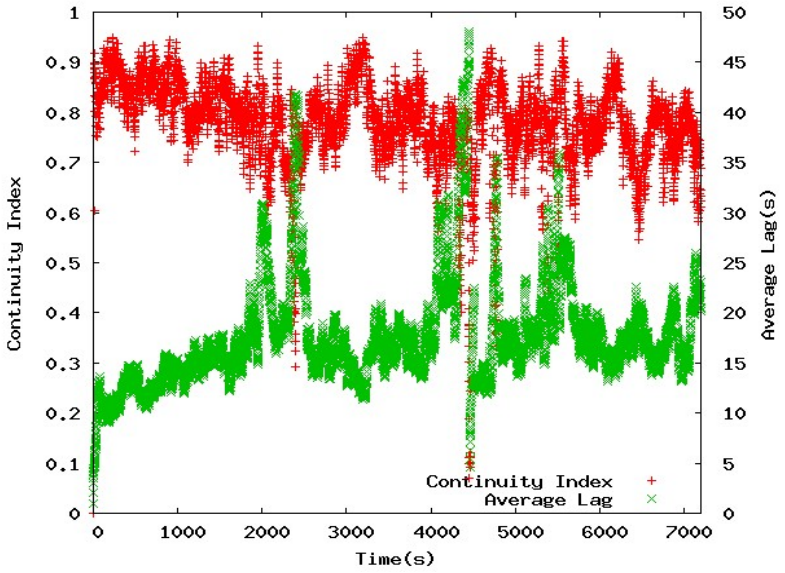

Fig. 5.3 Experiment result when there is a bottleneck at the start.

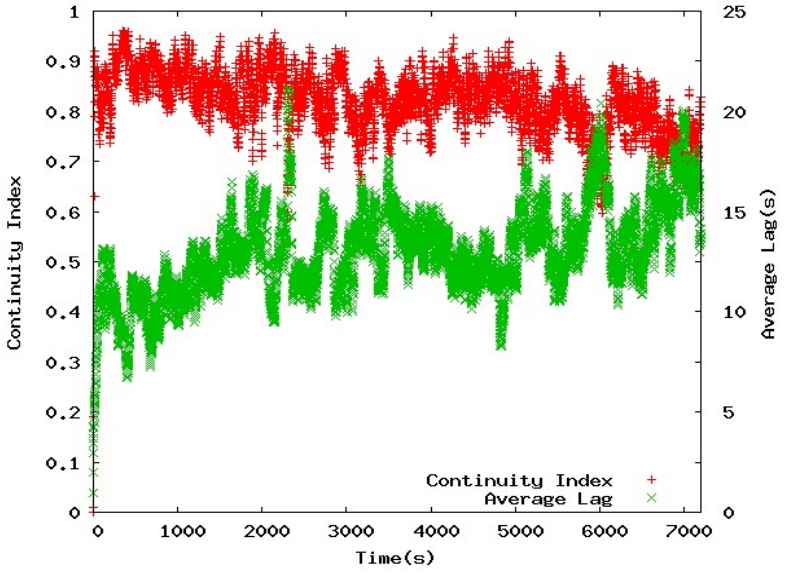

Fig. 5.4 Experiment result when server has the priority to select partners.

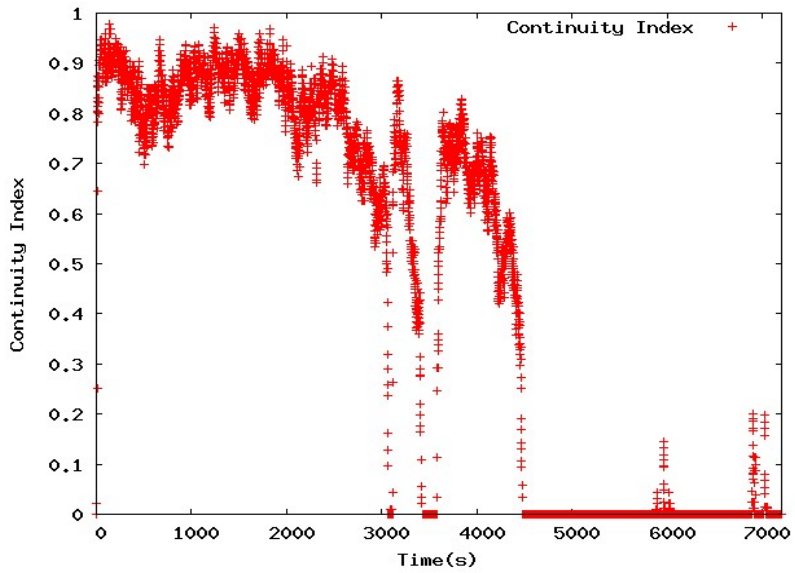

Fig. 5.5 Continuity Index when trading window size is 20 seconds.

maximum data lag that triggers lag reset, in order to make sure that even if a peer with the maximum lag of 60 seconds can cooperate with the peer with the minimum lag of 1 second. The continuity index becomes more smooth after this change. But in a stable scenario, it is better to set a small trading window, peers are more synchronized and the chunk exchange efficiency goes up when the trading window is small. Also, a small trading window leads to small overhead.

5.1.4 Churn Resilient Peering Scheme

As discussed in section 3.4.4, a peering scheme that makes adding a partner as soon as possible and makes full use of the bandwidth abundant peers is deployed in SMODA system. This is especially useful when the system is unstable and there are constantly a lot of peers which are reset. However, due to time limit, only 5 experiments have been done, the average continuity index of the CoolStreaming peering scheme is 0.658 and the average continuity index of the new peering scheme is 0.654. The bandwidth distribution used in these experiments is shown in Table 4.1. LEAVEP and JOINP were also set to 20 in these experiments. There is not much difference on the continuity index. We conjecture that it is because the additional partner is deleted when the peer refines due to the reason that the number of partner is over NP and this neutralizes the effect of adding a partner quickly. But on the other hand, the start-up delay data of this 5 experiments as in Table 5.1 shows that our method can shorten the start-up delay time by about 9.5% from an average value of 37058 milliseconds to an average value of 33530 milliseconds. We also argue here that, this method should be useful if

there are more bandwidth abundant peers and if there are a lot of peers been reset constantly. Due to time limit, here, more investigation to the effect of this method is assigned as a future work and personally the author believes that this peering scheme is important to an extreme unstable and heterogeneous system.

Table. 5.1 Start-up Delay(ms) improvement after new peering policy is applied.

Experiment Sequences	1	2	3	4	5
CoolStreaming Start-up Delay Avg.	38999	35330	38076	34492	38392
Proposed Peering Method Start-up Delay Avg.	34350	30865	34912	32284	35242

5.2 Results for Proposed Rank-based Incentive Architecture

We compare our rank-based incentive proposal with the random partner selection method which is used by CoolStreaming. The bandwidth distribution used here is shown in Table 4.1. LEAVEP and JOINP were set to 100 for 70% of the total peers and to be 0 for the other 30% of peers. Figure 5.6 depicts the results of using the rank-based incentive and Figure 5.7 depicts the results of using the original CoolStreaming random partner selection. In Figure 5.6 the green line is the continuity index of the rank 1 peers and the average value here is 0.964 and the red lines is the continuity index of all the peers with an average value of 0.479. In Figure 5.7, the green line also depicts the continuity index of the rank 1 peers which have an average continuity index of 0.854 and overall continuity index of 0.574. These two figures are shown to give readers a intuitive understanding of the proposed rank-based incentive method and how it affects the continuity index.

Four series, each series 7 experiments have been performed to compare the rank-based incentive method with the original CoolStreaming random partner selection method. The bandwidth distribution used here is shown in Table 4.1. LEAVEP and JOINP were set to 100 for 70% of the total peers and to be 0 for the other 30% of peers. Figure 5.8 depicts the continuity index of all the experiments. It shows that, the continuity index goes up for the rank 1 peer on the sacrifices of the overall continuity index. A simpler figure is Figure 5.9, it shows clearly that it is a tradeoff between the overall continuity index and the rank 1 peers' continuity index. When P=20, the rank 1 peers' continuity index is around 0.970, a relatively acceptable number, and the overall continuity does not deteriorate so much. We think P=20 is a good choice here. We argue here that, rather than no one can watch the live media comfortablely, it is better for, at least the most faithful and upload bandwidth abundant ones, to have good QoS. Actually, at the start, the purpose is to give good QoS to the rank 1 peers and because this good QoS, rank 1 peers can deliver better quality streaming to rank 0 peers and thus improve the overall continuity index, but as show in this figure, the

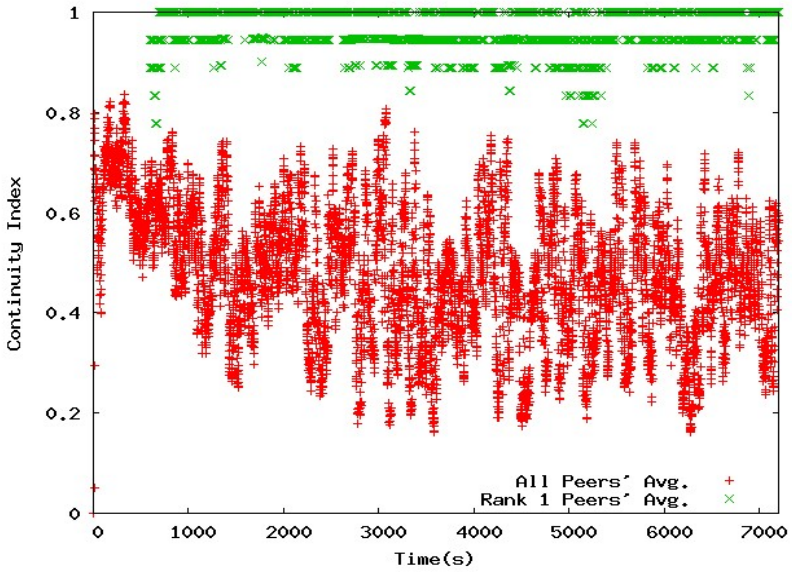

Fig. 5.6 Experiment result of the proposed Rank-based incentive.

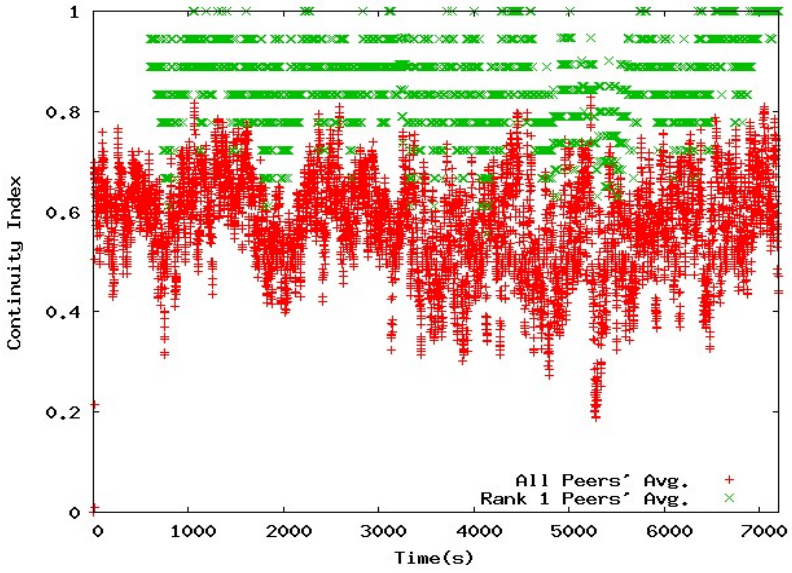

Fig. 5.7 Experiment result of the original CoolStreaming random partner selection.

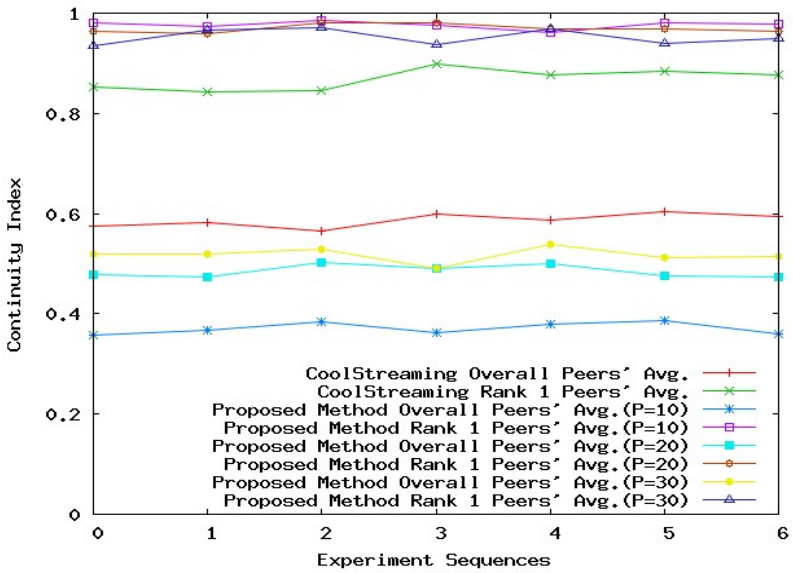

Fig. 5.8 Continuity Index using different parameter P and CoolStreaming's random partner selection policy (7 experiments each setting).

overall continuity index actually deteriorated.

The QoS evaluation parameter, data lag, has also been compared. The results are shown in Figure 5.10. It indicates that the data lags are shortened to half of the original length using the new rank-based method. When using the rank-based method, server (the origin peer) has a high rank and it tries to deliver chunks to high rank peers as high rank peers connect to each other with more relations. Due to this, chunks are first delivered to high rank peers and are transmitted between high rank peer members contributing to the improvement of the data lag which is cut to a half extent. This is encouraging and it shows that the proposed method really works.

Figure 5.11 illustrates the relation between the percentage of stable peers and the continuity index. The bandwidth distribution used here is shown in Table 4.1. LEAVEP and JOINP were set to 100 for stable peers and to be 0 for churn peers. The P parameter was set to 20 in these experiments. Using the proposed rank-based incentive method, the continuity index of the rank 1 peers improved in all the experiments. This means that the proposed rank-based incentive method is robust and effective to a wide variety of stable peers' percentage.

Figure 5.12 depicts the experiment results of changing the P parameter. In these experiments, the bandwidth distribution used here is shown in Table 4.1. LEAVEP and JOINP were set to 100 for 50% of the total peers and to be 5 for the other 50% of peers. Stable peers have a leave and join possibility of 0.0005/s which means around 30% of peers will leave the system in 10 minutes after they have joined. This figure also contains the experiment results

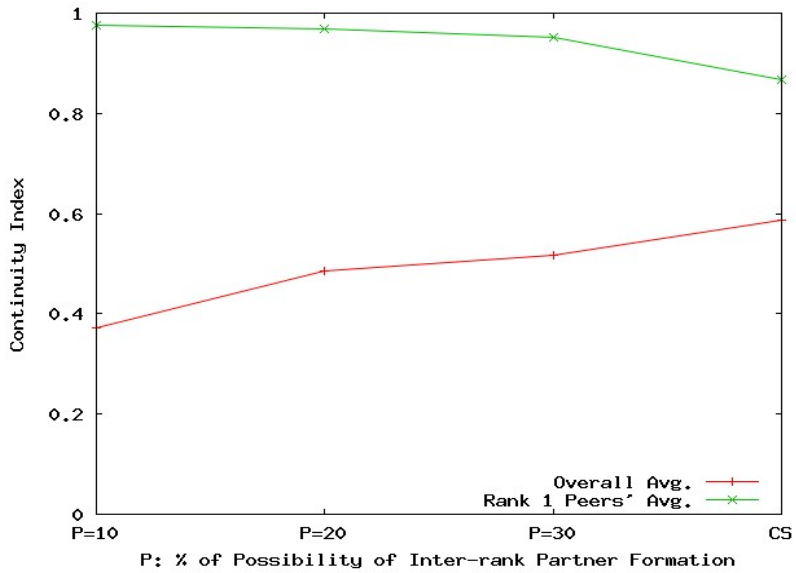

Fig. 5.9 Average Continuity Index using different parameter P and CoolStreaming's random partner selection policy (7 experiments' Avg.).

for CoolStreaming random partner selection algorithm and a parewise incentive algorithm. The information that can be inferred from this figure is that using the proposed rank-based algorithm, the continuity of high rank peers can be improved and it is better to use a P=20 when rank 1 peers' continuity index average is at the highest value. The parewise incentive algorithm experiment result in this figure suggests that our algorithm is superior when tackling with the assumed scenario.

All the above experiment results demonstrate the effectiveness and the robustness of the proposed rank-based inventive method. And we have also find out a practical parameter value P=20 through experiments.

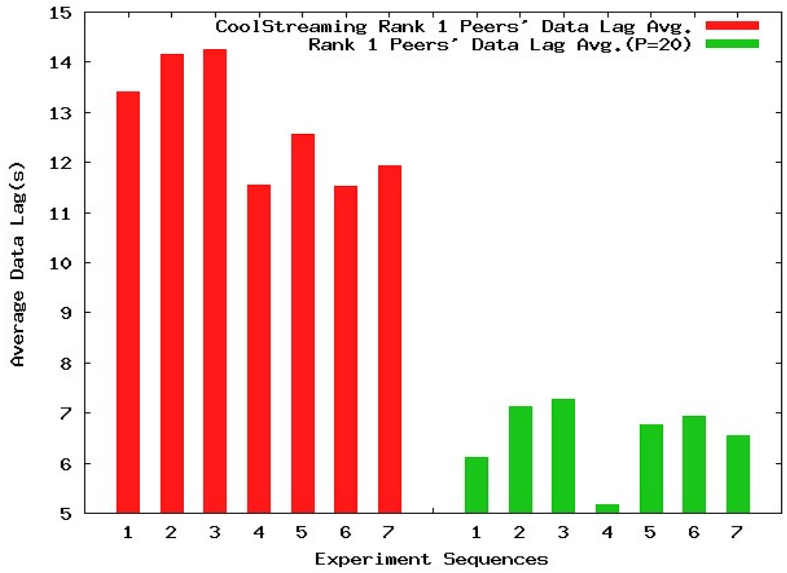

Fig. 5.10 Comparison of Data Lag between using Rank-based incentive and CoolStreaming's random partner selection for high rank peers.

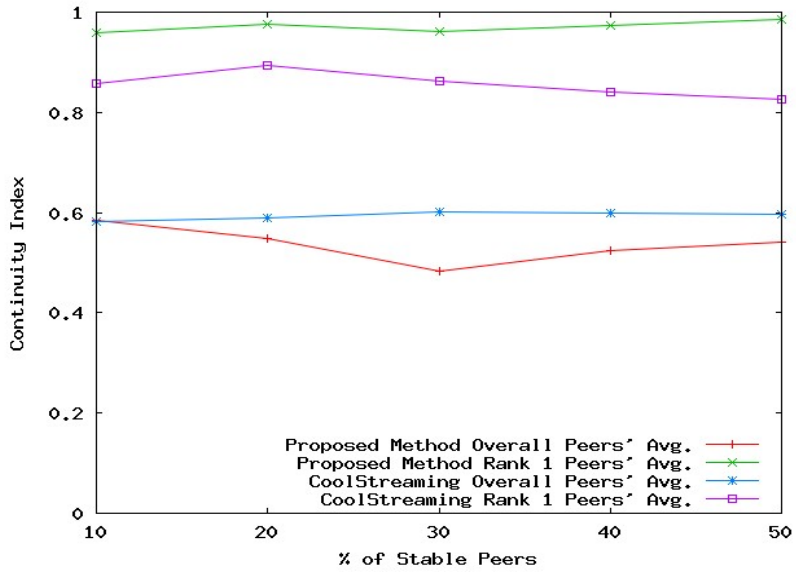

Fig. 5.11 Continuity Index as a function of the percentage of stable peers.

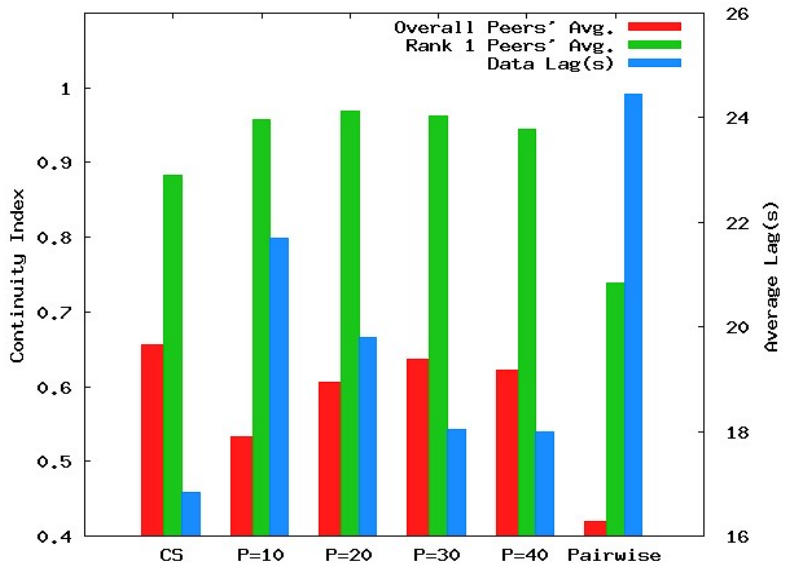

Fig. 5.12 Continuity Index and Data Lag comparison among Rank-based incentive, CoolStreaming's random partner selection and Pairwise incentive when the leave and join rate for a stable peer is 0.0005/s.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this study, we discuss the methodology to improve the QoS of the small mesh-based live media streaming overlay. Multiple design changes and parameter adjustments of the legacy method have been proposed so as to come up with the small heavy churn overlay environment. Also, we proposed a novel rank-based incentive method which divides mesh peers into groups with high rank and low rank based on online duration and upload ability. Dense partnership formations among the same rank peers are encouraged in our design. We extensively evaluated the performance of our proposed rank-based incentive method, the design changes and parameter adjustments using an event-driven simulator. Our experiments involve examining the effect of each design change and parameter adjustment one by one and evaluating the proposed rank-based incentive method under various scenarios. Simulation results demonstrate the effectiveness of parameter adjustments and design changes. Comparison with CoolSteaming random partner selection method and a pairwise incentive method shows that the proposed rank-based incentive method leads to superior performance in terms of continuity index and data lag of high rank peers, which demonstrate that the rank-based incentive method can successfully reduce the bad impact that the poor communication ability and unstable ones bring to the stable and upload bandwidth abundant ones.

6.2 Future Work

The future work would be;

(1) studying on more general methodology to improve the QoS for the live media overlay streaming system with wide variety of environment. Inter-overlay cooperation (e.g., [10], [41]) is a promising technique which we think maybe is the ultimate solution for the small overlay problem;

(2) deriving a true application out of SMODA prototype and evaluating SMODA design

in the real Internet environment;

(3) adding more amendments to the ongoing research. The expected amendments include minor improvement change to local rarest first scheduling algorithm described in section 3.4.5 and adopting MDC [24] coding into SMODA.

6.2.1 Adopt Coding Techniques into SMODA

Our research is also complementary to the coding researches such as multiple description codes (MDC), we are now thinking about applying MDC coding technology into SMODA by delivering more descriptions to the high rank group, which gives further incentive to the system.

6.2.2 Introduce Powerful Peers to The Small Overlay

When good QoS can not be guaranteed no matter which kind of inner changes and improvements we make because of extreme upload bandwidth insufficient and heavy churn, it is better to consider outside help. One possible solution would be to add dedicated servers to help the streaming, and another way would be to choose a few powerful peers from other resource-abundant overlays to help this small overlay. The commercial products usually use DHT such as chord to organize all the peers from various overlays and it is actually feasible to request powerful peer information from another overlay. AnySee [8] explores inter-overlay cooperation in tree-based environment and we are thinking about studying inter-overlay cooperation issue under mesh-based environment.

References

[1] Akamai Website: http://www.akamai.com

[2] PlanetLab Website: http://www.planet-lab.org

[3] PPLive Website: http://www.pplive.com

[4] PPStream Website: http://www.ppstream.com

[5] UUSee Website: http://www.uusee.com

[6] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," ACM Transaction on Computer Systems, May 1990.

[7] X. Zhang, J. Liu, B. Li, and TS. P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," in Proc. of INFOCOM, Mar. 2005.

[8] X.Liao, H. Jin, Y. Liu, L. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," in Proc. of INFOCOM, 2006.

[9] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into pplive: A measurement study of a large-scale p2p iptv system," in Proc. of IPTV Workshop, 2006.

[10] G. Tan and S. A. Jarvis, "Inter-Overlay Cooperation in High-Bandwidth Overlay Multicast," in Proc. of ICPP, Aug. 2006.

[11] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in Proc. of ACM SIGMETRICS, 2000.

[12] D.Watts and D. Strogats, "Collective dynamics os Small World Network," Nature, 440-442, 1998.

[13] A. Medina, I. Matta, and J. Byers, "BRITE: A flexible generator of Internet toplogies," Technical Report BU-CS-TR-2000-005, Boston University, 2000.

[14] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," in Proc. of the IEEE, 2006

[15] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays," Technical Report UIUCDCS-R-2006-275, University of Illinois at Urbana-Champaign, Aug. 2006.

[16] F. Pianese, J. Keller, and E. W. Biersack, "PULSE, a flexible P2P live streaming system," in Proc. of IEEE INFOCOM, Apr. 2006.

[17] D. Purandare, R. Guha, "An Alliance based Peering Scheme for Peer-to-Peer Live Media Streaming," in Proc. of SIGCOMM, 2007.

[18] F. Wang, Y. Xiong, and J. Liu, "TreeBone: A hybrid structure for efficient peer-to-peer

live streaming," Technical Report, Aug. 2006.

[19] Y. Tang, L. Sun, K. Zhang, S. Yang and Y. Zhong, "Longer, Better: On Extending User Online Duration to Improve Quality of Streaming Service in P2P Networks," in Proc. of IEEE ICME, Jul. 2007.

[20] X. Hei, C. Liang, J. Liang, Y. Liu, and K.W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," in Proc. of IPTV services over World Wide Web in conjunction with WWW2006, May 2006.

[21] F. Wang and J. Liu, "A Trace-based Analysis of Packet Flows in Data-driven Overlay Networks," Techinical Report, in preparation, 2006.

[22] T. Silverston and O. Fourmaux, "Measuring p2p iptv systems," in Proc. of NOSSDAV, 2007.

[23] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "PeerSim: A Peer-to-Peer simulator," http://peersim.sourceforge.net.

[24] V.K. Goyal, "Multiple description coding: Compression meets the net-work," IEEE Signal Processing Mag., vol. 18, pp. 74-93, Sept. 2001.

[25] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in Proc. of NOSSDAV May 2002.

[26] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over peer-to-peer network," Technical Report, Stanford University, 2001.

[27] D. Tran, K. Hua, and T. Do. "Zigzag: An efficient peer-to-peer scheme for media streaming," in Proc. of IEEE INFOCOM, Apr. 2003

[28] M. Castro, P. Druschel, AM. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in Proc. of ACM SOSP, Oct. 2003.

[29] R. Rejaie and A. Ortega, "PALS: peer to peer adaptive layered streaming," in Proc. of NOSSDAV, Jun. 2003.

[30] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in Proc. of IPTPS, Feb. 2005.

[31] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in Proc. of ACM SOSP, 2003.

[32] N. Magharei, A. Rasti, D. Stutzbach, and R. Rejaie, "Peer-to-Peer Receiver-driven Mesh-based Streaming," in Proc. of ACM SIGCOMM, Aug. 2005.

[33] C. Dana, D. Li, D. Harrison,and CN. Chuah, "BASS: BitTorrent Assisted Streaming System for Video-on-Demand," in Proc. of IEEE MMSP, Oct. 2005.

[34] B. Cohen, "Incentives build robustness in BitTorrent," P2P Economics Workshop, 2003.

[35] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for Supporting Streaming Applications," in Proc. of IEEE INFOCOM, Apr. 2006.

[36] S. Annapureddy, C. Gkantsidis, P. Rodriguez, and L. Massoulie, "Providing Video-on-Demand using Peer-to-Peer Networks," Microsoft Technical Report MSR-TR-2005-147, 2005.

[37] B. Bollobas, "Random Graphs," 2nd Edition, Cambridge University Press, 2001.

[38] M. Zhang, JG. Luo, L. Zhao, and SQ. Yang, "A Peer-to-Peer Network for Live Media Streaming Using a Push-Push Approach," in Proc. of ACM Multimedia, 2005.

[39] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments," in Proc. of IEEE INFOCOM, 2006.

[40] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers, "The state of peer-to-peer simulators and simulations," ACM SIGCOMM Computer Communication Review, 37(2):95-98, 2007.

[41] R. Keralapura, CN. Chuah, N. Taft, and G. Iannaccone, "Can Coexisting Overlays Inadvertently Step on Each Other," in Proc. of IEEE ICNP, 2005.

[42] M. Zhang, Y. Xiong, Q. Zhang, S. Yang, "On the optimal scheduling for media streaming in data-driven overlay networks," in Proc. of IEEE GLOBECOM, 2006.

[43] AJ. Ganesh, AM. Kermarrec, and L. Massoulie, "Peer-to-Peer Membership Management for Gossip-based protocols," IEEE Transactions on Computers, Feb. 2003.

Publication

[1] 姜 鵬, 江崎 浩, "QoS Improvement for Small Mesh-based Live Streaming Overlays Using On-line Duration Awareness," 電子情報通信学会 2009 年総合大会, B-16-1, 愛媛, 2009 年 3 月 17 日 (発表予定)

Acknowledgments

First of all I would like to take this opportunity to express my sincere gratitude to Dr. Hiroshi ESAKI, who has been my supervisor during my master course and research student course period. He provided me with many helpful suggestions, constructive advice and constant encouragement during the course of this work.

I also wish to express my cordial thanks to all the Esaki Lab. members. They have supported me from various aspects during everyday life. Someone helped me to find an apartment when I first came to Japan, someone gave me delicious bread when I was writing the thesis and did not have time to buy one, someone talked with me about my research and gave me valuable suggestion, etc. I shall miss the days we have spent together and hope you all have a nice life ahead.

My special appreciation goes to Panasonic Corporation. They have provided me with not only financial support but also daily life care during the passed 3 years. During the events and activities they organized I have expanded my view, made a lot of friends and spent much happy time.

Finally, I would like to express special thanks to family members who always kept me away from family responsibilities and encouraged me to concentrate on my study.

2009-02-03