LayerP2P: A New Data Scheduling Approach for Layered Streaming in Heterogeneous Networks

Xin XIAO, Yuanchun SHI, Yuan GAO
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China
Email: {xiaoxin00@mails., shiyc@}tsinghua.edu.cn,
gaoy03@gmail.com

Qian ZHANG
Department of Computer Science
Hong Kong University of Science and Technology
Email: qianzh@cs.ust.hk

***Abstract*—Although layered streaming in heterogeneous peer-to-peer networks has drawn great interest in recent years, there's still a lack of systematical studies on its data scheduling issue. In this paper, we propose a new scheduling approach for layered video streaming, called LayerP2P. The key idea and main contributions of LayerP2P come in two-fold: 1) According to the characteristics caused by layered coding, we propose four objectives that should be achieved by data scheduling: high throughput, high layer delivery ratio, low useless packets ratio, and low subscription jitter; 2) We design a 3-stage scheduling mechanism to request absent blocks, where the min-cost flow model, probability decision mechanism and multi-window remedy mechanism are employed in *Free Stage*, *Decision Stage* and *Remedy Stage*, respectively. Each stage has different scheduling objective while collaborates with each other, to achieve the above four objectives. Experimental results indicate that our approach outperforms other schemes in simulation environment. Besides, LayerP2P is implemented in the PDEPS Project in China, which is expected to be the first practical layered streaming system for education in peer-to-peer networks.**

I. Introduction

Peer-to-Peer streaming becomes a hot topic during recent years because on one hand it doesn't rely on the deployment of IP-multicast protocol, and on the other hand the data transmission among peers significantly reduce the bandwidth cost of the server, high scalability could then be achieved. Adding to the complexity of this problem is the heterogeneity of client networks, which leads to different requirements for streaming quality by different peers. To handle this, layered (*or* scalable) streaming in p2p networks is proposed and has drawn great interest [1], [4], [7], [8], [9].

The basic idea of layered encoding is that a raw video sequence is compressed into some non-overlapped streams, or layers. The *base layer* contains the basic data representing the most important features of the video. Additional layers, called *enhancement layers*, contain data that progressively refine the reconstructed video quality [19]. According to the available bandwidth, participating nodes subscribe a subset of the layers to reconstruct the video with certain quality degradation.

Although such encoding scheme brings more flexibility for participating peers to achieve adaptive video quality, it also causes unique challenge to the data scheduling (i.e. requesting and relaying data) for layered streaming: rather than merely maximizing the throughput and/or minimizing the packet delay, which is key optimization objective in the scheduling for non-layered streaming, the scheduling scheme for layered streaming should address more problems due to the distinct characteristics brought by layered coding. To name a few, the mismatch between throughput and layer delivery ratio, the useless packets in high layers due to the loss of packets in low layers, the over reaction to bandwidth variation when adjusting subscribing layers, and etc.

By now there have been several schemes proposed to address the scheduling issue in layered streaming (e.g. [1] employs Round-Robin method to request data, [8] aims to optimize the performance in mobile ad-hoc network, [9] utilizes alternative paths and network coding to improve throughput), however, by mainly focusing on throughput and/or packet delay, few of them have systematically studied the above characteristics caused by layered coding. In this paper, we propose the following four new metrics, which we believe are essential and critical to systematically study the data scheduling problem for layered streaming:

Throughput and Delay: maximizing the overlay throughput, as well as keeping low packet delay, is certainly the basic requirement;

Layer delivery ratio: in non-layered streaming, maximizing the node delivery ratio is almost equal to maximizing the throughput, which is not the case in layered streaming. In layered streaming, subscribing many layers but with low delivery ratio for each layer can also result in high throughput. However, the video quality cannot be high due to the layer dependency. Therefore, to ensure high delivery ratio for subscribed layers is also a key point;

Useless packets ratio: the decoding of upper layers depends on the availability of lower layers. If some lower layer packets are missed, the packets with the same sequence IDs in the upper layers cannot be correctly decoded and thus become useless. The useless packet ratio should be kept low;

Jitter prevention: bandwidth variation is common in today's Internet. Under such situation, if the node subscribes more layers immediately after bandwidth increased, it may have to drop the highest layers after a short while due to the bandwidth decreasing. This short-term subscribe-drop pair is called jitter, which not only brings fluctuation in node's quality of service (QoS) but also causes its buffer overflow or underflow. Hence, it should be prevented by the data scheduling scheme.

To achieve the above four objectives, in this paper, we present LayerP2P–our data scheduling model and the corresponding strategies to tackle the layered streaming scheduling problem in heterogeneous peer-to-peer networks. The origin of LayerP2P is based on the following consideration: to achieve high layer delivery ratio and low useless packets ratio, proper layers should be subscribed and the absent blocks in subscribed layers should be fetched in a reasonable way when the blocks are close to their playback time; and to maximize the throughput, the blocks far from their playback time need to be fetched as many as possible. In short, different scheduling goal should be accomplished based on the data blocks' playback deadline. Thus, in LayerP2P, we first model this scheduling problem in each node as a 3-stage problem according to the data blocks' playback time, where each stage has different scheduling objective. In the farthest "Free Stage", the min-cost network flow model is employed for data scheduling, to achieve high throughput. In the medium "Decision Stage", we use probability decision mechanism to decide how many layers should be subscribed for current window, so as to achieve high delivery ratio and low useless packet ratio. Jitter prevention is also taken into account when making decision. The "Remedy Stage" contains the blocks that have the most urgent playback time, in which the multi-window remedy mechanism is used to request the minor missed data blocks in the subscribed layers. These three stages work independently but collaborate with each other so as to achieve the above four objectives.

Extensive simulation experiments demonstrate that our approach outperforms other different existing methods in throughput and layer delivery ratio; also, it can effectively reduce the useless packet ratio and subscription jitter ratio. The average packet delay is kept within reasonable scope. Besides, LayerP2P is implemented in the Project of Digital Education for Public Service (PDEPS) in China, which is expected to be the first practical layered streaming system for education in peer-to-peer networks.

The remainder of this paper is organized as follows: Section II presents the related works. Our method for the scheduling problem in P2P layered streaming is described in section III. We conduct simulations to evaluate its performance and present its implementation in real network in section IV. Finally, Section V concludes this paper and gives future work.

II. Related Works

Layered streaming has emerged as a promising way to handle the heterogeneities of client networks [1], [4], [6]∼[13]. Due to the limited deployment of IP-multicast protocol, recently, the researches on it mainly focus on in application layer, especially in peer-to-peer environment. In the literature, the research works can be largely classified into two categories: 1) overlay construction (i.e. neighbor selection), and 2) data scheduling.

For overlay construction part, many related works have paid attention to building high performance p2p overlay for layered streaming, in order to deliver satisfying QoS. Representative works include [4], [5], [7], [12]. With these efforts, the overlay suited for layered p2p streaming could be built, which in turn provides underlying support for data scheduling. Some other works such as mOverlay [14], NICE [15], Zigzag [16], Narada [17] and Bullet [18] also take QoS-awareness into account when constructing overlay, in which NICE and Zigzag are tree-based, and mOverlay, Narada and Bullet are mesh-based. Although these works are not proposed specially for layered streaming, their key ideas are still useful to overlay construction for layered streaming.

For the data scheduling part, so far some valuable works have been proposed. PALS [1] adopted a diagonal buffer distribution and employed Round-Robin method to request data. [8] aimed to optimize the performance in mobile adhoc network. LION [9] utilized alternative paths and network coding to improve throughput. [10] studied the on-demand media distribution problem and proposed a peer-to-peer streaming solution in which how to optimally allocates the desired layers among multiple senders were addressed. [11] also considered the broadcast of on-demand video, in which the delivery of layered encoded file was part of their work, mainly investigating how the clients adjust subscription layers according to the bandwidth with the server, so as to achieve adaptation video quality. In short, these works make effort to maximize the system throughput or to deliver satisfying video layers according to the available bandwidth, whereas the distinct characteristics caused by layered encoding have not been systematically addressed. Therefore, we propose the four scheduling objectives and the LayerP2P mechanism in this paper to address the optimal data scheduling for layered streaming.

III. LayerP2P: Toward optimal scheduling for layered streaming

Before presenting the details of LayerP2P, we'd like to emphasize its design principle: as analyzed in Section I, the throughput should be guaranteed by fetching as many data blocks with latest playback time as possible; the integrity of data blocks should be ensured by subscribing appropriate layers when they're close to their playback time. Besides, sudden bandwidth drop may induce the absence of blocks in subscribed layers; to combat this, there should be a remedy mechanism to request the minor missed blocks with most urgent playback time. To address such unique requirement in layered streaming, we model the data scheduling scheme as a 3-stage problem according to the data blocks' playback time, and design corresponding strategies for each stage to achieve its scheduling goal. With the initial idea of LayerP2P presented in our preliminary work [13], in this paper, we put more efforts on refining the scheduling strategy in each stage; in addition, from which block to begin the data scheduling is also studied.

A. 3-Stage Model for Data Scheduling

With layered coding, the video stream is encoded into several layers; and each layer is partitioned into blocks. Thus each block has a layer ID (LID) and a block ID (BID). There are buffering windows on each node containing the blocks that the node is interested in (See Fig. 1). Every node periodically declares to its neighbors the blocks availability in its buffering windows, and requests the absent blocks from neighbors according to their declarations. The right bound of the buffering widows is the maximum block ID owned by the node. And the left-most part contains the blocks that have just been played or will be played soon, thus these blocks are read-only. Periodically, the buffering windows slide forward by a window's size.

The windows on the right side of the read-only buffer are divided into 3 stages, according to the playback time of blocks: the remedy stage contains the blocks with the most urgent playback time (*or* smallest BIDs); the free stage contains the blocks with the latest playback time (*or* largest BIDs); the medium stage is called decision stage. When the blocks enter the buffering windows, they firstly enter the free stage. With the periodical sliding forward of the buffering windows, they will arrive at the decision stage, and finally reach the remedy stage.

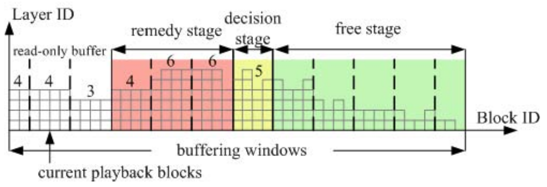


Fig. 1. The buffering windows and the 3-stage model. The numbers marked in the figure are the subscribed layers in each window.

B. The Remedy Stage

The blocks in the remedy stage are very close to the playback time, therefore, they have higher priorities than those in the decision and free stage, and the available bandwidth is firstly allocated to request the minor absent blocks in this stage. The remedy mechanism is very simple: suppose the remedy stage includes *k* windows, and the number of missed blocks within the subscribed layers in the newly entered window is *n*, thus the *n* blocks should be requested from neighbors during *k* windows' time, and on average just n/k blocks are requested in each window. However, how to choose *k*'s value is interesting: if it's too small, it's likely that there isn't enough time to request the missed blocks, especially when the blocks encounter bandwidth burst-and-drop in the decision stage; on the other hand, if it's too large, the startup waiting time will be very long, because after the blocks with earliest playback time leave the decision stage, they must stay in the remedy stage for *k* windows' time before being played.

We propose the following method to determine k's value: suppose the node encounters bandwidth burst-and-drop when making subscription decision, more specifically, assume the estimated bandwidth is B_h when subscription decision is made, and it drops to B_l soon. And assume one window's time is t_{wnd}, thus there will be $(B_h - B_l) * t_{wnd}$ data missed in the decision window. Therefore, when the window enters the remedy stage, $\frac{(B_h - B_l)*t_{wnd}}{k}$ data should be requested from neighbors in each remedy window. We use $(1 + \varepsilon)B_l$ to denote the bandwidth of the node after it enters the remedy stage (here, $0 \leq \varepsilon < 1$, which is a small number), for the bandwidth may have increased slightly from the lowest level B_l. To ensure the missed blocks to be completed fetched in the remedy stage, we have:

$$(1 + \varepsilon)B_l * t_{wnd} \geq \frac{(B_h - B_l) * t_{wnd}}{k} \tag{1}$$

Thus, we get

$$k \geq \frac{B_h - B_l}{(1 + \varepsilon)B_l} \tag{2}$$

In the subsequent experiments, the bandwidth variation is measured, thus k's value could be calculated according to (2). We also measure k's influence on the system performance, which will be shown in Section IV.

C. The Decision Stage

The decision stage contains only one window. The available bandwidth, not including the part allocated to remedy stage, is used to subscribe the layers in this stage. The scheduling objective in this stage is to decide how many layers should be subscribed for the blocks that just entered this stage from free stage, in order to achieve high delivery ratio for each layer and low useless packet ratio; meanwhile, it should also try to avoid subscription-jitter caused by bandwidth burst-and-drop. It works as follows: suppose when the window just enters the decision stage the missed blocks number in layer l is m_l, among which the neighbors can supply s_l (under bandwidth constraint). Thus, $m_l - s_l$ is the number of blocks that are still unavailable in layer l. We define delivery ratio of layer l as $DR_l = 1 - (m_l - s_l)/wnd\_blocks_l$, where wnd_blocks_l is the number of all blocks in layer l within one window (e.g. in Fig. 1, the wnd_blocks_l is 4). Then, the subscription probability of layer l, $SP(l)$, is:

$$SP(l) = \begin{cases} 1, if DR_l \geq ratio_{std} \\ DR_l / ratio_{std}, otherwise \end{cases} \tag{3}$$

Note that if layer l is unsubscribed, all the layers higher than it are unsubscribed. Here, $ratio_{std}$ is a constant threshold, whose value is important: on the one hand, if it's too large, condition (3) is difficult to meet, thus many layers might be unsubscribed, and correspondingly, the throughput of the system will decrease; on the other hand, if it's too small, each node is able to subscribe many layers, however, there'll be plenty of missed blocks in these layers and they should be requested in the remedy stage, which may be beyond the bandwidth capability, thus low delivery ratio and high useless packets ratio might be induced. The impact of $ratio_{std}$ under different value is studied in Section IV, where its most appropriate value will also be given due to the experiment results.

To prevent subscription jitter caused by bandwidth burst-and-drop, we define Jitter-Prevent-Factor (*or* JPF) for each layer l:

$$JPF_l = e^{\frac{HL_{last} - l}{K \cdot L}} \tag{4}$$

Here, HL_{last} is the highest subscribed layer in the last blocks window, and L is the total number of video layers, K is a coefficient which is set to 2 in the experiments. We further define the improved layer delivery ratio IDR_l as $IDR_l = DR_l * JPF_l$, and re-calculate the $SP(l)$ with IDR_l. We can see that if $l \gg HL_{last}$, JPF_l, as well as IDR_l, is very small, thus the probability of subscribing layer l is low; in contrast, if l is close to the bottom layers, JPF_l is very large, hence the probability of subscribing layer l is high.

Once the subscription decision is made, the absent blocks within the subscribed layers are requested. If they can't all be received in the decision stage, the absent ones will be re-fetched from neighbors when they enter the remedy stage.

D. The Free Stage

The remaining available bandwidth is allocated to free stage. The main objective of this stage is to freely requests the blocks that haven't entered decision stage, with the important blocks fetched in high priorities, so as to achieve high throughput.

In [2], Zhang et al. proposed optimal scheduling method for non-layered streaming, where the Min-Cost Flow Problem (MCFP) is employed to do scheduling. In this paper, we make use of the outcome of [2], and make some improvements: 1) the method is mainly used in free stage to achieve high throughput. Although we also use it to allocate to neighbors the requests for data within subscribed layers in decision and remedy stage, the major concern there is the layer delivery ratio, useless packets ratio and jitter prevention. Thus, we call the MCFP in [2] which treats the buffering windows as a whole as Pure-MCFP; 2) the method is modified to suit the scheduling of layered streaming, with the priorities of layers taken into account; 3) assume the subscribed highest layer in the current decision stage is *HL*, then the requests for absent blocks in free stage are limited within layers not higher than *HL*, rather than for all layers, so as to tackle possible bandwidth drop.

Without loss of generality, we consider node i, and similarly to [2], we firstly give some pre-definitions in TABLE I.

Then, the scheduling problem for free stage in node i could be expressed by the optimal problem below:

$$\begin{aligned}&\max \sum\nolimits_{k\in M_i^{HL}} \sum\nolimits_{j\in Nei(i)} IF_i^k h_j^k x_{i,j}^k\\ &s.t.\\ &(a) \sum\nolimits_{j\in Nei(i)} x_{i,j}^k \le 1, \forall i,k\\ &(b) \sum\nolimits_{k\in M_i^{HL}} size_k * x_{i,j}^k \le t_{wnd} * Bw_{j,i}, \forall i,j\\ &(c) \sum\nolimits_{k\in M_i^{HL}} \sum\nolimits_{j\in Nei(i)} size_k * x_{i,j}^k \le t_{wnd} * I_i, \forall i\end{aligned} \tag{5}$$

This problem can be translated into a min-cost flow problem, and the "CS2" library [21] can be employed to solve it. According to the solution to the problem, node i could optimally request missed blocks from neighbors to achieve the maximum throughput. In our implementation, the bandwidth $Bw_{j,i}$ is measured by TFRC [20]; and for the Importance Function, our definition is as follows:

$$IF_i^k = \alpha IF_N(\sum_{j\in Nei(i)} h_j^k) + \beta IF_L(L_k) + \gamma IF_T(BID_k - BID_{LB}) \tag{6}$$

where $\alpha+\beta+\gamma = 1$ and $\alpha,\beta,\gamma \ge 0$. IF_N denotes the "Number Importance", which is in negative correlation with the number of neighbors that hold block k; IF_L addresses the "Layer Importance", which is in negative correlation with the layer of block k; IF_T presents the "Time Importance", which is in negative correlation with k's playback time.

TABLE I
DEFINITIONS USED IN THE FREE STAGE SCHEDULING MODEL

Definition	Explanation
M_i^{HL}	node i's subscription scope, that is, the missed blocks in layers less equal to HL in free stage
$Nei(i)$	Neighbors of node i
LID_k	the layer block k is in
BID_k	block ID of block k
BID_{LB}	block ID of the left-most block in the buffering windows
$size_k$	size of block k
I_i	the incoming bandwidth capacity of node i
$Bw_{j,i}$	the end-to-end bandwidth from node j to i, minus the part allocated to the remedy and decision stage
t_{wnd}	one window's time
$h_j^k \in \{0,1\}$	"$h_j^k = 1$" denotes node j holds block k; "$h_j^k = 0$", otherwise
$x_{i,j}^k \in \{0,1\}$	"$x_{i,j}^k = 1$" means node i requests block k from node j; "$x_{i,j}^k = 0$", otherwise
IF_i^k	different block has different significance, so Importance Function IF_i^k of block k for node i is used to denote this

In a short summary, in above subsections, based on the distinct characteristics caused by layered coding, we model the data scheduling for layered streaming as a 3-stage problem according to the data blocks' playback time, with min-cost network flow model, probability decision mechanism and multi-window remedy mechanism integrated within it, to collaboratively achieve the four scheduling goals.

E. Where To Begin Data Request

When the newly joined node, say i, finishes the neighbor selection and enters the data scheduling phase, from which block to begin the first data request is a non-trivial problem, because its choice is directly related to the playback delay and the subsequent data request process. For ease of presentation, we refer to the node's left-most block in the decision window as *dec_blk*; furthermore, the node's first *dec_blk* is called *dec_blk*^{0}. Thus, the above problem equals how to choose node i's *dec_blk*^{0} (*or dec_blk*$_i^0$), with its neighbors' current *dec_blk* all known.

We propose the following mechanism to decide *dec_blk*$_i^0$, aiming to achieve both high data continuity and reasonable playback delay for the new node i: in i's first period of data scheduling phase, each neighbor sends the BID of its current dec_blk, as well as the data blocks around dec_blk to node i, without i's explicit requests. Meanwhile, the available bandwidths between node i and the neighbors are measured by TFRC. Then, at the end of this period, the following method is carried out to decide on $dec\_blk_i^0$.

For node $j \in Nei(i)$, assume its current dec_blk is dec_blk_j, and the bandwidth between node i and j is $Bw_{i,j}$. Besides, we define $f_j(i)$, denoting the benefit the new node i can get from neighbor j; and $g_j(i)$, denoting the benefit the new node i can provide for neighbor j, respectively. We firstly give the definition of $f_j(i)$:

$$f_j(i) = \begin{cases} 0, if dec\_blk_i^0 > dec\_blk_j \\ \dfrac{Bw_{j,i}}{E(Bw_{n,i}|n \in Nei(i))} * D_j(i), otherwise \end{cases} \tag{7}$$

Here, $E(Bw_{n,i}|n \in Nei(i))$ denotes the average bandwidth between the neighbors and node i. And the $\frac{Bw_{j,i}}{E(Bw_{n,i}|n \in Nei(i))}$ part in the equation reflects the idea that if the bandwidth between node j and i is better than the average bandwidth between the neighbors and node i, node j is a good neighbor so that the possible benefit achieved from it would be high. And the $D_j(i)$ part reflects the influence of the distance between $dec\_blk_i^0$ and dec_blk_j on the benefit that node i can get from j. If dec_blk_j is just a little ahead of $dec\_blk_i^0$, $D_j(i)$ should be small, because when node i requests data, j might not have the desired blocks; if the distance is larger, the probability that j can provide the desired blocks become higher, hence $D_j(i)$ should be larger; however, if $dec\_blk_j \gg dec\_blk_i^0$, the probability will converge to some value. Thus, we define $D_j(i)$ as follows:

$$D_j(i) = 1 - a^{-(dec\_blk_j - dec\_blk_i^0)} \tag{8}$$

Here, a is a constant, and the choice for its value will be presented in Section IV.

Similarly, $g_j(i)$ is defined as follows:

$$g_j(i) = \begin{cases} 0, if dec\_blk_i^0 \le dec\_blk_j \\ \dfrac{Bw_{i,j}}{E(Bw_{n,j}|n \in Nei(j))} * \tilde{D}_j(i), otherwise \end{cases} \tag{9}$$

The definition of $\tilde{D}_j(i)$ is similar to $D_j(i)$:

$$\tilde{D}_j(i) = 1 - a^{-(dec\_blk_i^0 - dec\_blk_j)} \tag{10}$$

Therefore, we can model the decision for $dec\_blk_i^0$ as the following optimal problem:

$$\begin{aligned} &\max \sum\nolimits_{j \in Nei(i)} (f_j(i) + g_j(i)) \\ &s.t. \\ &\sum \{Bw_{j,i}|dec\_blk_j \ge dec\_blk_i^0\} \\ \ge &\sum \{Bw_{i,j}|dec\_blk_j < dec\_blk_i^0\} \end{aligned} \tag{11}$$

Where the constraint is based on such consideration: the incoming bandwidth capability should be higher than the outgoing bandwidth capability, because only if the node can get sufficient incoming video layers, it's able to serve the partners better.

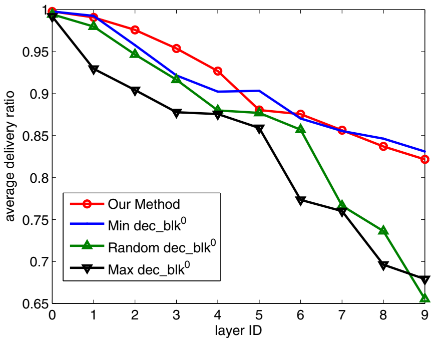


Fig. 2. The average layer delivery ratio when dec_blk^0 is decided by different methods.

TABLE II
THE AVERAGE THROUGHPUT AND PACKET DELAY WHEN dec_blk^0 IS DECIDED BY DIFFERENT METHODS

Method	**Average Throughput(Mbps)**	**Average Packet Delay(s)**
Our Method	1.8226	7.9786
Min dec_blk^0	1.9079	11.9438
Random dec_blk^0	1.1364	7.5074
Max dec_blk^0	0.6209	2.9136

IV. Implementation and Evaluation

In this section, firstly we conduct extensive simulation experiments to study the impacts of the data scheduling approaches under two categories of heterogeneous networks (i.e. heterogeneity caused by time-varying available end-to-end bandwidth, and heterogeneity due to the different bandwidth of the access network); then implementation and deployment of LayerP2P in real network is presented.

The experiments configurations are as follows: during the experiments, the participating nodes number varies from 100 to 500. The underlying *link-layer* topology is generated by GT-ITM [22], where the well-known transit-stub model is used. On top of the *link-layer* topology, the OCals approach, which is proposed in our previous work [7], is used to build the overlay, where only the nodes in stub-domains participate in the layered streaming, while the transit-domains act as the routers. The streaming server, which is located at node 0, provides layered streaming of totally 10 layers with each layer 250Kbps. Each node slides the buffering windows and schedule requests for all the stages once per second. During the experiment, the participating nodes leave the overlay randomly, at the rate 1 node/sec. Once the neighbor leaves or crashes, the failure recovery mechanism, which is proposed in our previous work [7], is employed to select a new neighbor.

A. Simulation under networks with time-varying available end-to-end bandwidth

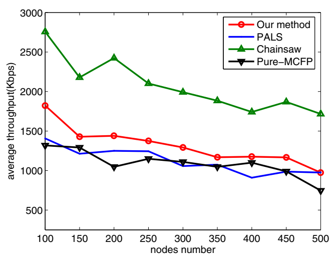
In the first set of experiments, the intra-transit bandwidths are set to 12Mbps, the transit-stub bandwidths are set to 6Mbps, and the intra-stub bandwidths are set to 10Mbps.

Therefore, the network heterogeneity is mainly caused by the time-varying end-to-end available bandwidth.

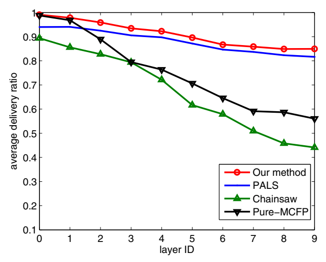
1) On the impact of dec_blk^0: Firstly, we study the system performance when $dec\_blk_i^0$ is determined by different methods: our method proposed in Section III.E; the "Min dec_blk^0" method, which makes $dec\_blk_i^0 = \min(dec\_blk)$ of the neighbors; the "Max dec_blk^0" method, which lets $dec\_blk_i^0 = \max(dec\_blk)$ of the neighbors; and the "Random dec_blk^0" method, in which $dec\_blk_i^0$ is randomly chosen from $[\min(dec\_blk) - 2 * wnd\_blks_l, \max(dec\_blk)]$ (See Section III.C for the definition of wnd_blks_l). In our method, the value of constant a in Equation (8) is determined like this: based on the analysis to $D_j(i)$ (see Section III.E), we let $D_j(i) = 0.99$ when $dec\_blk_j - dec\_blk_i^0 = 2 * wnd\_blks_l$, i.e. when dec_blk_j is beyond $dec\_blk_i^0$ by two windows, the benefit node i could get from j would be very high. Thus, we can get $a = 100^{\frac{1}{2*wnd\_blks_l}}$. To get the optimal solution of our method, we also limit $dec\_blk_i^0 \in [\min(dec\_blk) - 2 * wnd\_blks_l, \max(dec\_blk)]$, which greatly reduce the scope of $dec\_blk_i^0$, then the enumeration method could be adopted.

The number of nodes in the experiment is 100. We measure the delivery ratio of each layer, the throughput and packet delay averaged on every node under the above four methods. It's shown in Fig.2 and Table II that our method and the "Min dec_blk^0" have the highest layer delivery ratio, and the latter has just a little higher throughput than our method. However, our method reduces the packet delay by about 33.2%. This is because in our method the optimal value for $dec\_blk_i^0$ is calculated, which results in maximum benefits defined in Equation (11); in contrast, although the "Min dec_blk^0" method can guarantee the data continuity of the newly joined node, it also induces large playback delay. The throughputs in "Random dec_blk^0" and "Max dec_blk^0" methods are much lower, although the packet delay in "Max dec_blk^0" is the lowest. Therefore, in subsequent experiments, we adopt our method to determine $dec\_blk_i^0$ for the newly joined node i.

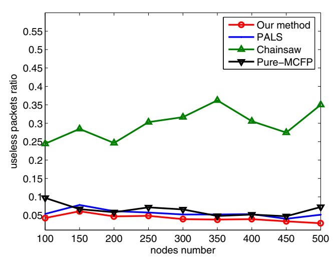
2) System performance under different scheduling approaches: In the following experiment, the influence of the data scheduling approach on the throughput of the layered streaming is measured. Compared with our approach, there're three other methods used: the PALS [1] approach, which uses a Round-Robin method to schedule requests; the Chainsaw [3] approach, which utilizes a purely random strategy to decide which block should be requested from which neighbor; and the Pure-MCFP [2] mechanism. In the experiments, the remedy window number $k = 3$, the $ratio_{std} = 0.6$. Fig. 3(a) depicts the average throughput of the nodes. Our method outperforms PALS and Pure-MCFP by 17.1% and 20.9%, respectively. Due to the Round-Robin data requesting mechanism, PALS can not reach the optimal scheduling. In contrast, Pure-MCFP does data scheduling based on the min-cost flow problem for the whole buffering windows, without consideration of determining the subscribed layers when the blocks are close to their playback time, thus the layer delivery ratio can not be guaranteed, which may cause the down-flow nodes not to find their needed blocks in the up-flow neighbors.



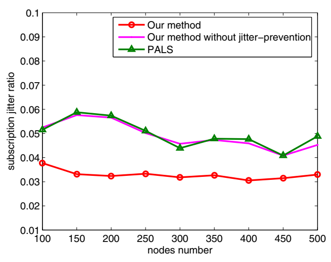
(a)



(b)



(c)



(d)

Fig. 3. Performance comparison for different scheduling methods. In Fig. (a)∼(d), throughput, layer delivery ratio, useless packets ratio and jitter ratio are compared, where remedy window number $k = 3$ and $ratio_{std} = 0.6$.

Therefore, the throughput achieved by our method is higher than in PALS and Pure-MCFP. Chainsaw approach achieves the highest throughput, because it utilizes a purely random data scheduling, no matter the blocks are necessary or not. But such scheduling can induce low layer delivery ratio, which will be shown in Fig. 3(b).

In Fig. 3(b), the layer delivery ratio averaged on each node is compared for the above four approaches. The number of participating nodes varies from 100 to 500, and the final result is averaged on the nodes number. The purely random scheduling used by Chainsaw achieves the lowest layer delivery ratio, because it ignores the different significance of different blocks. The Pure-MCFP is better than Chainsaw. Our method and PALS achieves the first and second highest delivery ratio, for they take the data integrity into account. Furthermore, since our method employs MCFP to optimally schedule the requests and proposes more reasonable decision mechanism, it performs better than PALS. In a short summary, our method outperforms PALS, Chainsaw and Pure-MCFP by 3.4%, 36.0% and 21.5%.

The data scheduling method should be both beneficial to the overlay performance and with low overhead. The useless packet ratio is such a parameter to present overhead. Thus, in Fig. 3(c), we test the useless packet ratio for the above four scheduling approaches. The experiments configurations are the same with 3(a). The useless packets ratio in our method is lower than that in PALS, Chainsaw and Pure-MCFP by 24.3%, 85.9% and 34.6%, respectively.

Fig. 3(d) shows the result of experiment on jitter. Assume the highest subscribed layer for three consecutive time window is $s1$, $s2$, $s3$, we say the subscription jitter happens if one of

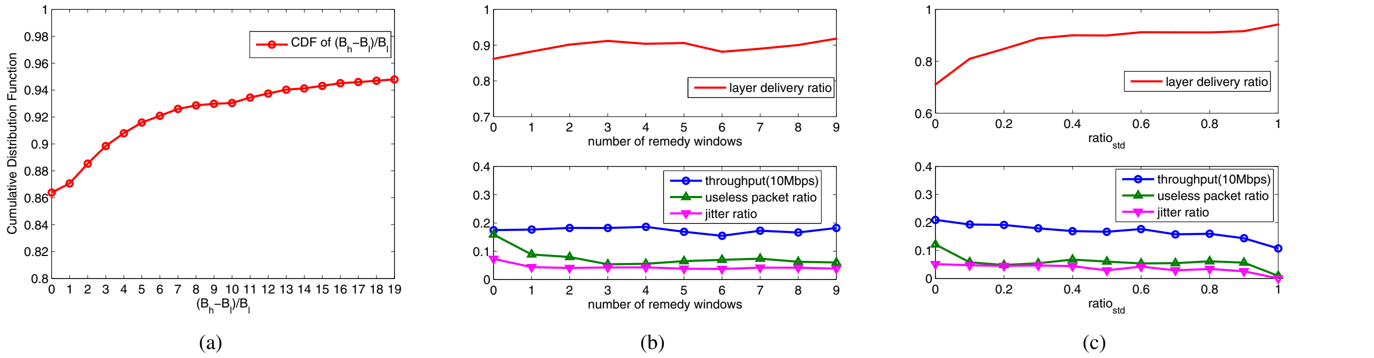


Fig. 4. In (a), the cumulative distribution function of the bandwidth drop (i.e.$(B_h - B_l)/B_l$) in the decision window is illustrated; In (b) and (c), performance comparison under different value of k and $ratio_{std}$.

the following conditions is met: 1) $s2 > s1$ and $s2 > s3$; 2) $s2 < s1$ and $s2 < s3$. We call 1) as "burst jitter" and 2) as "drop jitter". Both kinds of subscription jitters should be prevented to keep the video quality in a relatively stable state. We define jitter ratio as follows:

$$\eta_{jitter} = \begin{cases} \dfrac{|s2 - s1| + |s3 - s2|}{2 * L}, if\ 1)\ or\ 2)\ is\ met. \\ 0, otherwise \end{cases} \quad (12)$$

In the experiment, η_{jitter} is recorded once the jitter occurs within each node. The result shown in Fig.3(d) is the η_{jitter} averaged on every node. Because there isn't the concept of subscription layer in Chainsaw and Pure-MCFP, we only compare our method with PALS. Meanwhile, we also measure the jitter ratio in our method without the jitter prevention mechanism. It's shown in Fig.3(d) that the jitter ratio in our method is lower than those in others by 33.0% and 33.9%.

3) On the impact of system parameters k and $ratio_{std}$: In the next experiment, we study the influence of system parameters k and $ratio_{std}$. The participating nodes are 100 and other configurations are the same with Fig. 3(a).

As presented in Section III.B, the reasonable value of k can be decided by Equation (2), i.e. $k \geq \frac{B_h - B_l}{(1+\varepsilon)B_l}$. In the experiment, we measure the bandwidth variation during each decision window, and draw the CDF (Cumulative Distribution Function) of $\frac{B_h - B_l}{B_l}$ in Fig. 4(a). Note that if there's no bandwidth drop (i.e. $B_l \geq B_h$), we let $\frac{B_h - B_l}{B_l} = 0$. As illustrated in Fig. 4(a), the CDF increases very fast (from 86.4% to 89.8%) when $\frac{B_h - B_l}{B_l}$ varies from 0 to 3; when $\frac{B_h - B_l}{B_l}$ varies from 3 to 7, the CDF increases with a lower speed; and from 7 to 19, the increasing speed of CDF becomes even low, and it only increases from 92.6% to 94.8%. Meanwhile, Fig. 4(b) depicts the influence of k on the system performance. To make the figure clear, the unit of throughput is set to 10Mbps. We can see the value of k has little impact on throughput and jitter ratio. For the other two metrics, when $k \leq 3$, useless packets ratio/layer delivery ratio decreases/increase; when $k > 3$, they both don't change much. Based on the above analysis and experiments, we can see three remedy windows are enough to request the missed blocks within the subscribed layers; besides, such value of k will not induce large playback delay. Therefore, in the other experiments, we set $k = 3$.

Fig. 4(c) depicts the influence of parameter $ratio_{std}$ on the system. It still has little impact on jitter ratio. The layer delivery ratio increases with it, which is in accord with the analysis. And the throughput decreases slowly with it, because the larger it is, the lower the probability of subscribing higher layers is; as a result, higher layers are unsubscribed by most nodes, so the throughput decreases. Due to such tradeoff between layer delivery ratio and throughput, in the system implementation the $ratio_{std}$ is set to 0.6.

B. Simulation under networks with heterogeneous access bandwidth

In the second set of experiments, some changes are made to the network configurations. The intra-transit and transit-stub bandwidths are still set to 12M and 6Mbps, respectively; whereas the access bandwidths of the stub nodes are different from previous ones: there are five types of bandwidths: 10M, 5M, 2.5M, 1M and 500Kbps, and each type is owned by 20% of the nodes. Therefore, the network heterogeneity is mainly due to the average available bandwidth of the nodes.

1) System performance under different scheduling approaches: Under the above network configuration, the throughput, layer delivery ratio, useless packets ratio and jitter ratio are measured again and compared for the above four scheduling schemes, which is depicted in Fig.5(a)∼5(d).

Compared with the one in the first set of experiments, the average throughput of the system decreases significantly here (See Fig. 5(a)), for there're many participating nodes with low access bandwidth. Still, our method outperforms PALS and Pure-MCFP by 17.5% and 10.3%(The reason is the same with that presented in the first set of experiments).

The layer delivery ratio is also lower than that in the first set of experiments, as Fig. 5(b) depicts. This is because in heterogeneous networks, especially there're plenty of low-bandwidth nodes, the blocks needed by one node may be absent in their neighbors with high probability. Nevertheless, the layer delivery ratio achieved by our method is still satisfying,

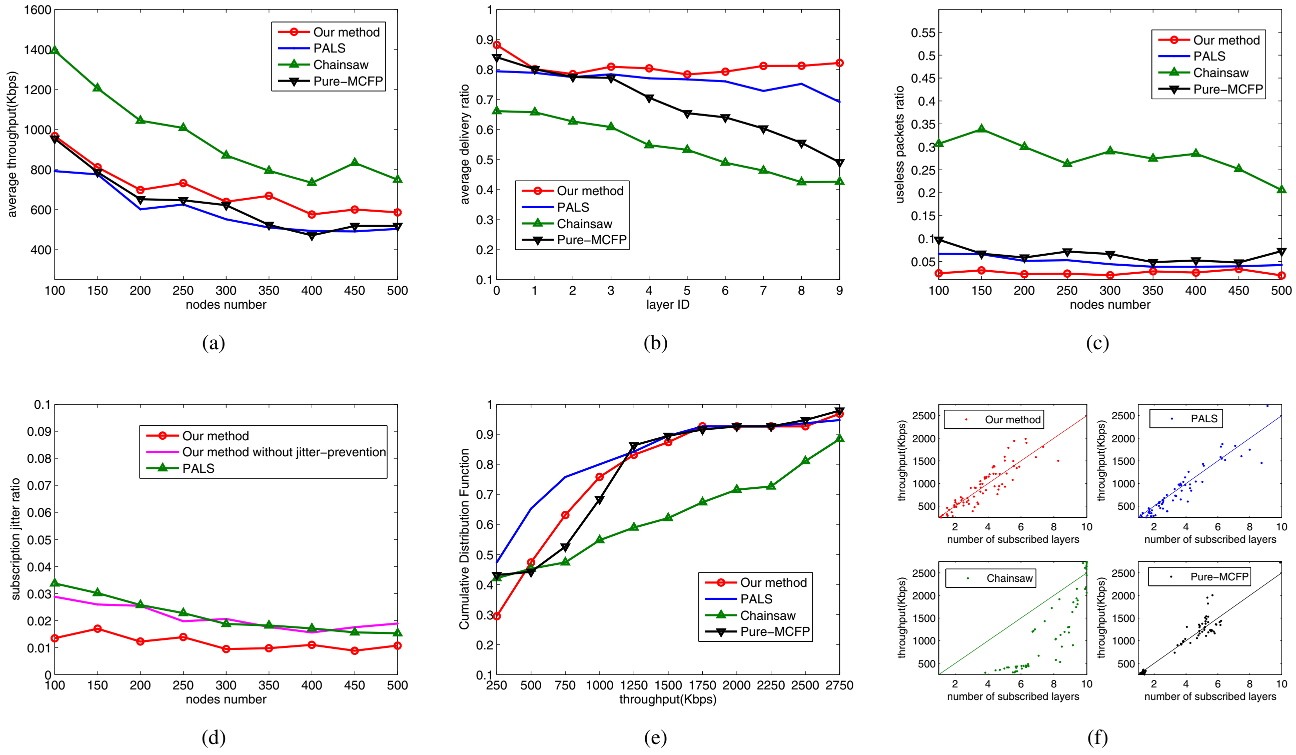


Fig. 5. Performance comparison for different scheduling methods. In Fig. (a)∼(d), throughput, layer delivery ratio, useless packets ratio and jitter ratio are compared, where the participating nodes have heterogeneous networks. In (e), the CDF of node's throughput is illustrated. And (f) depicts the distribution of the nodes on the (subscribed layers, throughput) plane.

which is 0.81 on average. This outperforms PALS, Chainsaw and Pure-MCFP by 6.4%, 49.0% and 18.4% gains.

Fig. 5(c) shows the useless packets ratio achieved by these methods. We can see the random data request way by Chainsaw results in the highest useless packets ratio, 0.2793 on average. The next is Pure-MCFP, 0.0644; the third is PALS, 0.0486; and our method achieves 0.0251. It's also shown that, the useless packets ratio doesn't increase much compared with the one in the first set of experiments. The reason is that although the layer delivery ratio decreases, the total number of layers subscribed by the node also decreases, so that the number of useless packets will not increase much.

As Fig. 5(d) illustrates, the jitter ratio is even lower than that in the first set of experiments. This is easy to understand: since on average the number of subscribed layers decreases, the gap between the subscribed layers number in two consecutive windows will also become short in high probability. The figure shows that our method achieves lower jitter ratio than other two methods (i.e. our method without the jitter prevention mechanism, and PALS) by 44.1% and 46.1%.

2) Nodes distributions under different scheduling approaches: In addition, the CDF (Cumulative Distribution Function) of the nodes' throughputs is presented in Fig. 5(e), to investigate the throughput distribution in the heterogeneous environment. The experiment result is interesting: in the Chainsaw method, the CDF is almost linear, because with the random data request mechanism, each node will maximize its available bandwidth. For other three methods, about 90% of the nodes are with throughput lower than 1.75Mbps, more than 80% are lower than 1.25Mbps, due to both the bandwidths heterogeneities and the data blocks the neighbors own. Most nodes are distributed in [250Kbps, 1Mbps], where the CDF of our method is the most close to linear, which matches the network configurations best.

The distribution of the nodes on the (subscribed layers, throughput) plane is illustrated in Fig. 5(f). Since there isn't the concept of subscription/un-subscription in Chainsaw and Pure-MCFP, the "subscribed layer" in them is referred to the highest layer in which data blocks are received. The figure shows there's significant mismatch between throughput and subscribed layers in Chainsaw. Because due to its random request mechanism, there might be some minor blocks in the higher layers but the throughput isn't so high as the layers. The distributions in the three other methods are approximately around the slope. The statistics from the log files show that the ratio of actual throughput and the corresponding theoretical throughput of the subscribed layers is 0.9479 in our method, which outperforms PALS, Chainsaw and Pure-MCFP by 14.8%, 50.1% and 1.6%, respectively.

C. Implementation In Real Network

In addition to the above simulation-based experiments, we have implemented the LayerP2P approach in the layered streaming in real network.

The core part of the system is implemented with ACE (Adaptive Communication Environment) [23], which is a cross-platform SDK, supporting the development under Windows, Linux and WinCE. As to the layered coder, we adopt the PFGS coding [24] among the well known layered coding technologies (e.g. FGS [25], PFGS [24] and 3D-Wavelet [26]) because it achieves fine grained scalability and channel adaptation, and it tries to use as many as predictions from the same layer as in the traditional SNR scalability schemes. In our implementation, the raw video is encoded into six layers.

To support the access of devices with heterogeneous networks, the peers are developed under both Windows and WinCE, thus Windows-installed PC and WinCE PDA (*or* SmartPhone) could participate and get adaptive video quality.

With the confidence brought by the simulation results and the pass in inner test, by now, the LayerP2P layered streaming has been deployed in the Project of Digital Education for Public Service (PDEPS) in China, aiming to provide adaptive video quality for heterogeneous users in the live teaching broadcast. As far as we know, it's the first practical layered streaming system for education in p2p networks. This project is in cooperation with China Central Radio and TV University, which has 1352 education centers over the country and 200,000 registered users of distant education. Currently, we're getting ready to provide the service of LayerP2P for the users by this October, and trace files of the user accesses will be collected to analyze their behaviors in layered streaming, which will be very helpful to the improvement of our scheduling approach.

V. Conclusion and Future Work

In this paper, we propose LayerP2P to investigate the data scheduling problem for layered video streaming in P2P networks. The contributions of this paper come in two folds: 1) rather than only focusing on throughput, we propose the four objectives that should be achieved by data scheduling for layered streaming, according to the characteristics caused by layered coding. 2) a 3-stage scheduling approach is given in this paper, with min-cost network flow model, probability decision mechanism and multi-window remedy mechanism integrated within it, to collaboratively achieve the scheduling goals. Its effectiveness and efficiency have been demonstrated by extensive simulations. Besides, it has been implemented in the PDEPS project for adaptive live teaching broadcast in distant education.

Future work will be carried out in two aspects: 1) In layered streaming, it's almost impossible to achieve 100% delivery ratio for each layer, thus the unavailability of these blocks might influence the data request of the down-flow peers. What is even worse is the cumulativeness of the absent blocks from up-flows to down-flows in the overlay. Hence, we plan to systematically study such phenomenon and propose effective method to solve it. 2) Based on the trace files to be collected in the PDEPS project, we'll analyze the user behaviors in the layered streaming so as to further improve the scheduling performance.

Acknowledgment

This work was supported by Project of Digital Education for Public Service in China under Grant No. 2006BAH02A24-4, China NSF under Grant No.60773139, in part by grant from HKUST under the contract RPC06/07.EG05 and BOEING002-CSE.07/08, by grant from National Basic Research Program of China (973 Program) under the contract No. 2006CB303100, the NSFC Oversea Young Investigator grant under Grant No. 60629203 and National 863 Program of China under Grant No. 2006AA01Z228.

References

[1] R. Rejaie, A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming", In *Proc. ACM NOSSDAV*, 2003.

[2] M. ZHANG, Y.Q. XIONG, Q. ZHANG, and S.Q. YANG, "On the Optimal Scheduling for Media Streaming in Data-driven Overlay Networks", In *Proc. IEEE GLOBECOM* 2006.

[3] V. Pai, K. Kumar, et al., "Chainsaw: Eliminating trees from overlay multicast", in *Proc. IEEE INFOCOM* 2005.

[4] L. Dai, Y. Cui, and Y. Xue, "Maximizing Throughput in Layered Peer-to-peer Streaming", in *Proc. IEEE ICC*, 2007.

[5] J. Liang, K. Nahrstedt, "RandPeer: Membership Management for QoS Sensitive Peer-to-Peer Applications", in *Proc. IEEE Infocom*, 2006.

[6] Q. Zhang, W. Zhu, and Y. Zhang, "Resource Allocation for Multimedia Streaming Over the Internet", in *IEEE Transactions on Multimedia*, VOL. 3, NO. 3, SEPTEMBER 2001, pp. 339 - 355.

[7] X. Xiao, Y.C. Shi, B.P. Zhang and Y. Gao, "OCals: A Novel Overlay Construction Approach for Layered Streaming", in *Proc. IEEE ICC*, 2008.

[8] M. Qin, R. Zimmermann, "Improving Mobile Adhoc Streaming Performance through Adaptive Layer Selection With Scalable Video Coding", in *Proc. ACM Multimedia*, 2007.

[9] J. Zhao, F. Yang, et al, "On Improving the Throughput of Media Delivery Applications in Heterogenous Overlay Network", in *Proc. IEEE Globecom*, 2006.

[10] Y. Cui and K. Nahrstedt, "Layered Peer-to-Peer Streaming", in *Proc. ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2003.

[11] L.Q. Shi, P. Sessini, et al., "Scalable Streaming for Heterogeneous Clients", In *ACM MM*06, October 23-27.

[12] Y. Okada, M. Oguro, et al., "A New Approach for the Construction of ALM Trees using Layered Video Coding", in *Proc. P2PMMS*, 2005.

[13] X. Xiao, Y.C. Shi and Y. Gao, "On Optimal Scheduling for Layered Video Streaming in Heterogeneous Peer-to-Peer Networks", in *Proc. ACM Multimeida*'08 as a short paper, pp.785-788.

[14] X. Zhang, Q. Zhang, Z. Zhang, G. Song and W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance", in *IEEE Journal on Selected Areas in Communications*, Jan. 2004, pp. 18- 28.

[15] S. Banerjee, B. Bhattacharjee, et al., "Scalable application layer multicast", in *Proc. ACM SIGCOMM*, 2002.

[16] D. A. Tran, K. A. Hua, et al., "Zigzag: An efficient peer-to-peer scheme for media streaming", in *Proc. IEEE INFOCOM*, 2003.

[17] Y. H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast", in *Proc. SIGMETRICS*, 2000.

[18] D. Kostic, A. Rodriguez, et al., "Bullet: high bandwidth data dissemination using an overlay mesh", in *Proc. ACM SOSP*, 2003.

[19] J. C. Liu, B. Li, Y. Q. Zhang, "Adaptive video multicast over the Internet", in *IEEE Multimedia*'03, vol.10, issue 1, pp.22-33.

[20] S. Floyd, et al., "Equation-based Congestion Control for Unicast Applications", in *Proc. ACM SIGCOMM*, 2000.

[21] A. Goldberg, "Andrew goldberg's network optimization library", [Online]. Available: http://www.avglab.com/andrew/soft.html

[22] Gt-itm's web site: http://www.cc.gatech.edu/projects/gtitm/

[23] ACE's web site: http://www.cse.wustl.edu/ schmidt/ACE.html

[24] S. P. Li, F. Wu, Y. Q. Zhang, "Study of a new approach to improve FGS video coding efficiency", MPEG99/m5583.

[25] W. P. Li, "Fine Granularity Scalability Using Bit-Plane Coding of DCT Coefficients", MPEG98/m4204.

[26] S. J. Choi and J. W. Woods, "Motion-Compensated 3-D Subband Coding of Video", in *IEEE Trans. on Image Processing*,1999,8(2):155-167.