June, 2006

Sequence of Linear Programming for Transmission of Fine-Scalable Coded Content in Bandwidth-Limited Environments

Xiao Su, *San Jose State University*
Tao Wang, *Synopsys Inc.*

Sequence of Linear Programming for Transmission of Fine-Scalable Coded Content in Bandwidth-Limited Environments

Xiao Su
Computer Engineering Department
San Jose State University
San Jose, CA 95192, USA

Tao Wang
Synopsys Inc.
700 E. Middlefield Road
Mountain View, CA 94043, USA

Abstract

In this paper, we propose an optimal peer assignment algorithm on peer-to-peer networks. This algorithm is designed to maximize the quality of transmitting fine-scalable coded content by exploiting the embedding property of scalable coding. To be more realistic, we assume that the requesting peer has a delay constraint to display the content within a certain delay bound, and it also has limited incoming bandwidth. We first use a simple example to illustrate the peer assignment problem, and then formulate this problem as a linear programming problem, followed by a nonlinear programming problem. To efficiently solve the second nonlinear problem, we transform it into a sequence of linear programming problems. Finally, we apply our proposed algorithm to both image and video transmissions in bandwidth-limited environments. Extensive experiments have been carried out to evaluate the complexity and performance of our approach by comparing it with both nonlinear formulation and two heuristic schemes. The results have verified the superior performance of our proposed algorithm.

1 Introduction

Peer-to-peer (P2P) networks are very promising for applications such as image and video transmissions, since a requesting peer may obtain content from a set of less congested or geographically closer supplying peers. This makes these applications less susceptible to bandwidth shortage and network congestion [21]. The delivery of images and videos depends largely on how they are coded before transmission, as coding algorithms define the property of coded bit streams.

Coding algorithms can be coarsely classified into two categories: scalable coding that embeds lower bit-rate bit streams into higher bit-rate bit streams, and non-scalable coding that does not have this embedding property [25]. Let C_1 and C_2 be the two bit streams generated by coding an image or a video in bit rate r_1 and r_2, respectively, where $r_1 < r_2$. Scalable coding generates C_1 as a prefix part of C_2, while non-scalable coding generates C_1 and C_2 as two entirely different bit streams.

Scalable coding algorithms can be further divided into two categories: fine-scalable and coarse-scalable coding [25]. Fine-scalable coding generates a fully embedded bit stream, whose quality increases with every additional bit. Coding algorithms, such as "Set Partitioning in Hierarchical Trees"(SPIHT) [20] and 3D-SPIHT [15], can generate such fine-scalable coded images and videos, respectively. Coarse-scalable coding generates a bit stream that consists of several layers, a base layer followed by one or more enhancement layers. The quality of a coarse-scalable coded bit stream only increases with an additional block of bits. The four scalability modes in MPEG-2 generate such coarse-scalable bit streams. Here, we focus on transmission of fine-scalable coded images and videos, generated by SPIHT and 3D-SPIHT.

If a peer requests an image or a video that is non-scalable coded in d bits/pixel, then another peer holding the *same* content coded in d_n bits/pixel can serve as a supplying peer, only if $d_n = d$. On the other hand, if a peer requests an image or a video that is scalable coded in d bits/pixel, then another peer holding the *same* content coded in d_s bits/pixel can serve as a supplying peer, regardless of the value of d_s. Therefore, utilizing the embedding property of scalable coding on P2P networks will involve more supplying peers, which have the requested content coded in different bit rates. However, to our best knowledge, none of the existing work seriously considered this embedding property, while designing its transmission strategies. In this paper, we will show how to use this embedding property to maximize the quality of transmitting fine-scalable coded content on P2P networks.

To be more realistic, we also consider the following two constraints. First, applications such as video transmission, are sometimes delay sensitive with some delay bound. For example, when a user downloads a video from the Internet, he may want to receive the best possible video preview within a couple of minutes and decide whether to continue downloading the video. Second, the requesting peer may have limited incoming bandwidth in some bandwidth-limited environments, for example, wireless personal area networks (WPAN) and cellular networks.

In summary, we study in this paper image and video transmissions in bandwidth-limited environments. The objective of our work is to investigate how to divide and assign the transmission task to a set of supplying peers, in order to maximize the quality of the downloaded content on P2P networks, under the constraints that a requesting peer has a delay bound and limited incoming bandwidth.

The paper is organized as follows. We first discuss related work in Section 2, and then use a simple example to illustrate the peer assignment problem in Section 3. Next we formally formulate and solve the problem in Section 4, and evaluate the proposed algorithm in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

Early commercial peer-to-peer file sharing systems, such as Napster [6] and Gnutella [3], normally identify a single supplying peer by its directory lookup algorithm, and download files from this single peer. More recent systems, such as KaZaA [4], eDonkey [2] and BitTorrent [1], adopt a more general data sharing model of downloading media files from multiple sources. However, these systems treat the media files as regular data files, and they do not explore the properties and structures of coded bit streams. In this case, when a peer requests for an image or a video, these systems will only treat the same content coded in the same bit rate as identical to the requested object. The files with the same content coded in different bit rates will be regarded as different objects. Hence, even for scalable coded media files, these systems may miss a lot of eligible supplying peers that have the same content coded in different bit rates. Our work exploits the embedding property of scalable coding to involve these missing supplying peers in transmission, and thus results in better quality of downloaded content.

In research community, there have been a lot of efforts developing efficient algorithms for live media streaming on P2P networks. Such efforts can be classified into two categories.

Research in the first category addresses how to construct and maintain an efficient overlay topology for media distribution. In CoopNet [17, 18], a video source collects information from other nodes to construct and maintain a distribution tree. This centralized model is efficient but presents a scalability problem. This motivated the proposal of distributed algorithms like SpreadIt [13], NICE [8], and ZIGZAG [23, 24]. These algorithms use hierarchical clustering to minimize transmission delay, limit work load, and distribute the tasks of tree construction and maintenance to a set of cluster heads. While such tree-based distribution structure is intuitive, it is also susceptible to unbalanced load. Solutions addressing this problem include building mesh-based trees in Narada [10], maintaining multiple distribution trees in SplitStream [9], adapting tree structures to reduce streaming latency for active users in ACTIVE [28], and relying on data-driven design in DONet [27].

Research in the second category studies how to deliver multimedia content to improve quality of service (QoS) from either network or end users' perspective. Xu [26] proposed an optimal media assignment algorithm (OTS_{p2p}) to minimize the initial buffering delay, and also studied how to amplify the overall system capacity for media streaming. However, other QoS parameters, such as latency, loss, and path diversity, were not considered. Cui [11, 12] exploited the buffer capacity at peer nodes to reduce the load on streaming servers when user requests are asynchronous and peer bandwidths are heterogeneous. In PALS [7, 19], receivers adaptively decided on the number of media layers and used a packet assignment algorithm to allocate the subset of packets in each layer to different senders. PALS adopted layered-coded streams to address the heterogeneity in peer bandwidths, but it did not dig deeper into the internal coding algorithms and bit streams to study how these can help improve end-to-end QoS.

The proposed work in this paper belongs to the second category. It differs from the existing work, in such a way that we have exploited the embedding property of scalable coding when designing our transmission schemes. Our previous work [22] along this line studied how to minimize the downloading time of scalable coded images on P2P networks, but it does not take into account the delay and incoming bandwidth constraints.

3 Illustrative Example

In this section, we walk through a simple example of image transmission in a low-bandwidth environment. It helps us gain a better understanding of the peer assignment problem, when a requesting peer has a delay bound and limited incoming bandwidth.

Suppose in a P2P system, there are four peers holding fine-scalable coded bit streams of a standard *Barbara* image of dimension 512×512. p_1 has the image coded in 0.25 bits/pixel, *i.e.* 0.25 bpp, resulting in the coded bit stream of size 64 kbits. p_2 has the image coded in 0.5 bpp (*i.e.*, coded bit stream of size 128 kbits). p_3 and p_4 have the image coded in 1 bpp (*i.e.*, coded bit stream of size 256 kbits). Furthermore, p_1, p_2, p_3, and p_4 have different outgoing bandwidths of 50 kbps, 20 kbps, 40 kbps, and 20 kbps, respectively. Such a setting models image transmission in a bandwidth-limited environment.

Now consider the scenario, where a requesting peer tries to download the *Barbara* image coded in 1 bpp within 2 seconds from these four supplying peers. Its incoming bandwidth is only 100 kbps, which is smaller than the total sum (130 kbps) of the outgoing bandwidths provided from the supplying peers.

There are two heuristic approaches to assign the transmission task to the supplying peers. The first heuristic method, called LongAssign, models the existing P2P file-sharing systems, such as eDonkey and BitTorrent. It only involves the supplying peers with the image coded in the same bit rate, and transmits the image based on their outgoing bandwidths. The supplying peer with a larger bandwidth gets to transmit a larger portion of the coded image.

Because the requested image is coded in 1 bpp, only p_3 and p_4 are eligible to work as the supplying peers. The sum of the outgoing bandwidths (60 kbps) from these two supplying peers is smaller than the incoming bandwidth of the requesting peer (100 kbps). In this case, the best image transmission scheme is to assign the transmission task to p_3 and p_4 based on their outgoing bandwidths. p_3 transmits 80 kbits of the coded bit stream, say between [0, 80) kbits, and p_4 transmits 40 kbits, say between [80, 120) kbits. At the delay bound of 2 seconds, the requesting peer is only able to decode from the 120 kbits of coded bit stream that has been received.

The second heuristic method, called GreedyAssign, tries to involve any possible peer with either a subset or a superset of the requested image at any time until the peer finishes transmitting its image. Since the peers hold the *Barbara* image of different sizes, they may finish their transmissions at different times. The peers need to dynamically adjust their sending rates based on the current number of the supplying peers and their supplying bandwidths. GreedyAssign maximally uses the bandwidths from the supplying peers at any given time, while satisfying the limit of incoming bandwidth of the requesting peer.

This heuristic utilizes the embedding property of scalable coding, and models a greedy implementation of parallel downloading. In GreedyAssign, all the four peers are eligible to work as supplying peers. First, all the four peers contribute to the transmission of the coded bit stream between $[0, 64)$ kbits. Note that during this period, the total bandwidth from the four supplying peers (130 kbps) is larger than the incoming bandwidth of the requesting peer (100 kbps). The supplying peers will cooperate to send at an aggregate bandwidth of 100 kbps. At $t = 0.64$ $(= 64/100)$ second, p_1 leaves because it does not have bit stream beyond 64 kbits. Then p_2, p_3, and p_4 work together to supply the coded bit stream between $[64, 128)$ kbits. At $t = 1.44(\approx 0.64 + 64/(20 + 40 + 20))$ second, p_2 leaves because it has no bit stream beyond 128 kbits. After that, both p_3 and p_4 continue their transmission until the delay bound ($t = 2$) is met. Using this greedy method, the requesting peer receives a bit stream of 162 $(\approx 128 + (2 - 1.44) \times (40 + 20))$ kbits.

Table 1: optimal peer assignment

peer identity	p_1	p_2	p_3	p_4
image segment (kbits)	$[0, 40)$	$[40, 80)$	$[80, 160)$	$[160, 200)$
transmission bandwidth (kbps)	20	20	40	20
time (second)	2	2	2	2

This greedy approach, however, does not result in the decoded image of best possible quality. An optimal algorithm, called SLPAssign (derived in Section 4), allocates the image transmission task, shown in Table 1. p_1 transmits the bit stream between $[0, 40)$ kbits at

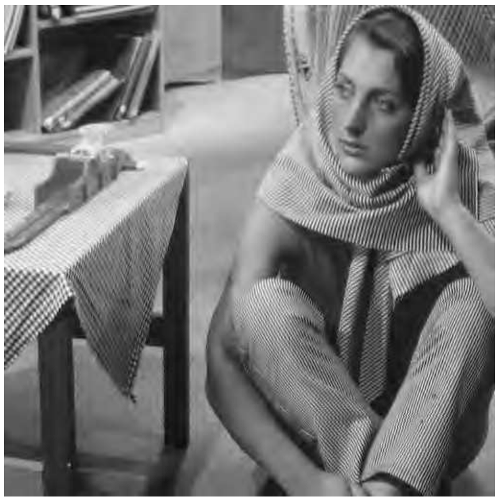
20 kbps; p_2 transmits between $[40, 80)$ kbits at 20 kbps; p_3 transmits between $[80, 160)$ kbits at 40 kbps; and p_4 transmits between $[160, 200)$ kbits at 20 kbps. All the peers start transmission at time 0 and finish at 2 second. This algorithm results in the transmission of 200 kbits in total. As the decoding quality is proportional to the size of the received bit stream, SLPAssign results in the best quality among the three transmission schemes.

To illustrate the difference in visual quality of the above three schemes, we use SPIHT [20] to generate fine-scalable coded *Barbara* images. Figure 1a), Figure 1b), and Figure 1c) show the decoded *Barbara* images, when 120 kbits, 162 kbits, and 200 kbits of the SPIHT-coded bit stream are received by applying LongAssign, GreedyAssign, and SLPAssign, respectively. The corresponding peak signal-to-noise ratios (PSNRs) are 30.90 dB, 32.96 dB, and 34.48 dB. Among the three algorithms, SLPAssign achieves the best objective PSNR as well as visual quality.

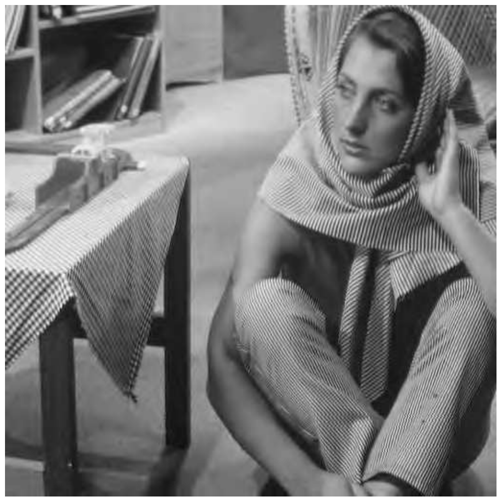
From this simple example, we can reach two conclusions. First, there exist more supplying peers such as p_1 and p_2, if the embedding property of scalable coding is exploited in transmission of scalable-coded content. Both GreedyAssign and SLPAssign possess this feature, resulting in better image quality than LongAssign. Second, in addition to utilizing this embedding property, it is very important to develop a systematic approach to find the optimal transmission scheme that can achieve the best decoding quality, under the constraints that the requesting peer has a delay bound and limited bandwidth.

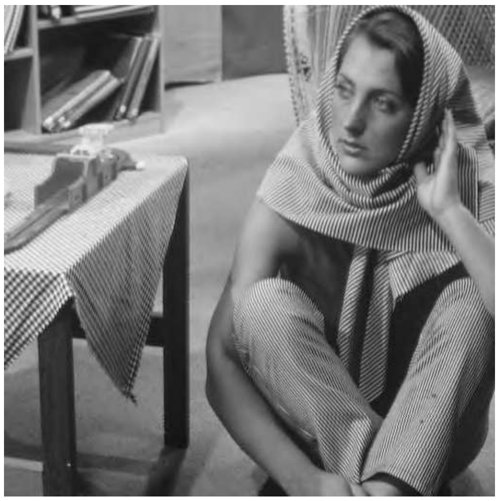
4 Problem Statement and Solution

The transmission of fine-scalable coded content on P2P networks can be done in four steps. First, a requesting peer prompts a user to specify his delay requirement (*e.g.,* the delay bound), or the delay bound is determined by the type of applications. Second, the requesting peer employs a directory lookup algorithm to locate a potential set of supplying peers for a given request. Third, the requesting peer applies the proposed peer assignment



a) LongAssign





b) GreedyAssign c) SLPAssign

Figure 1: Decoded *Barbara* images when the delay bound is 2 seconds and the incoming bandwidth of the requesting peer is 100 kbps, using a) LongAssign (PSNR = 30.90 dB), b) GreedyAssign (PSNR = 32.96 dB), and c) SLPAssign (PSNR = 34.48 dB).

algorithm to allocate the transmission task to the supplying peers, with the objective to maximize the quality of delivered content within the delay bound. Fourth, the supplying peers are informed about their own allocations by the requesting peer, and then start transmission.

Table 2: Notations used in the paper

Notation	Definition
n	the number of supplying peers
s_i	the size of the coded bit stream in peer i, $i = 1, 2, \ldots, n$
r_i	the coded bit rate of the content in peer i, $i = 1, 2, \ldots, n$
B_i	the outgoing bandwidth of peer i, $i = 1, 2, \ldots, n$
B^I	the incoming bandwidth of the requesting peer
B^O	the sum of the outgoing bandwidth of n supplying peers
T_u	user-defined delay bound

In this section, we will develop an optimal peer assignment algorithm applied in the third step. To facilitate further discussion, we first define some notations. For a given requesting peer, there are n supplying peers with coded bit streams of sizes, s_i $(i = 1, 2, \ldots, n)$, coded in different bit rates, r_i $(i = 1, 2, \ldots, n)$, and with different outgoing bandwidths, B_i $(i = 1, 2, \ldots, n)$. Without loss of generality, we assume $s_1 \leq s_2 \leq \ldots \leq s_n$ (that is, $r_1 \leq r_2 \leq \ldots \leq r_n$), otherwise we can always re-number the peers to follow this order.

The requesting peer tries to download an image or a video within the delay bound of T_u seconds, and its incoming bandwidth is B^I. The total bandwidth from the supplying peers is $B^O = \sum_{i=1}^{n} B_i$, and B^I can be either larger or smaller than B^O. For easy reference, we summarize the notations in Table 2.

Given the above notations, let us define concept of a *peer allocation vector*,

DEFINITION 1 *A peer allocation vector, $\mathcal{P} = \{(\Delta_i, b_i),\ i = 1, 2, \ldots, n\}$, is the vector in which the i^{th} element, (Δ_i, b_i), defines the size of segment Δ_i assigned to peer i for transmission and its transmission rate b_i.*

The objective here is to derive an optimal peer allocation vector, $\mathcal{P}^* = \{(\Delta_i^*, b_i^*),\ i = 1, 2, \ldots, n\}$, such that the quality of the content downloaded within delay bound T_u is optimal. Because the content is fine-scalable coded, its quality is proportional to the amount of the bit stream received by the requesting peer. Hence, to find optimal $\mathcal{P}^*$, we need to maximize the quantity of $\Delta_1 + \Delta_2 + \ldots + \Delta_n$, under the following constraints. First, each peer finishes transmitting its assigned segment Δ_i within delay bound T_u. Second, segment Δ_i assigned to each peer is within its bit stream boundary $[0, s_i]$. Third, transmission rate b_i assigned to each peer is no larger than its outgoing bandwidth B_i. Fourth, the total transmission rate of the supplying peers is within the incoming bandwidth of the requesting peer. In other words, this problem can be formulated as a constrained integer programming problem as follows,

$$\begin{aligned}
\max\nolimits_{\mathcal{P}} \quad & L(\mathcal{P}) = \Delta_1 + \Delta_2 + \ldots + \Delta_n \qquad (1)\\
\textit{subject to} \quad & \frac{\Delta_i}{b_i} \leq T_u,\\
& \sum_{k=1}^{i} \Delta_k \leq s_i,\\
& b_i \leq B_i,\\
& \sum_{k=1}^{n} b_k \leq B^I,\\
& \Delta_i, b_i \in \{0\} \bigcup Z^+, \text{ for } \; i = 1, 2, \ldots, n
\end{aligned}$$

which is a linear integer programming problem by transforming constraint $\Delta_i / b_i \leq T_u$ into $\Delta_i - b_i\, T_u \leq 0$.

To efficiently solve this integer programming problem, we first assume variables Δ_i and b_i take continuous values, and then employ a linear programming package, *lp_solve* [5], to find an optimal solution to the corresponding linear programming problem. After that, we derive the approximate integer solution to Eqn.(1) by rounding the continuous solution to its closest integer solution. In our previous work [22], we proved that the quality of this approximate solution is very close to the true optimal integer solution by establishing an upper bound between the two.

Let $\mathcal{P}_{T_u}$ be the solution obtained from Eqn.(1). If its objective value, $L(\mathcal{P}_{T_u})$, is equal to s_n, this means that delay bound T_u is large enough for the requesting peer to obtain the coded content of the best quality. In this case, the solution to Eqn.(1) may not be unique.

To illustrate this, let us visit a simple example. A peer requests from two supplying peers for an image within 5 (T_u) seconds. Suppose p_1 has its coded image of size 100 kbits (s_1) with the outgoing bandwidth of 50 kbps (B_1), and p_2 has its coded image of 120 kbits (s_2) with the outgoing bandwidth of 30 kbps (B_2). The requesting peer has its incoming bandwidth B^I larger than 80 (50 + 30) kbps. For this simple problem, we can easily find two optimal solutions: the first one is $\{(0, 50), (120, 30)\}$, and the second one is $\{(75, 50), (45, 30)\}$. In both cases, the requesting peer can receive the coded image of 120 kbits, which is the best-quality image that can be supplied by p_1 and p_2. However, the first solution results in much longer image transmission time ($\max\{0/50, 120/30\} = 4$ sec) than that of the second solution ($\max\{75/50, 45/30\} = 1.5$ sec). Obviously, the second solution is a preferable solution.

Therefore, we are motivated to find the solution that also minimizes the transmission time, in case that the objective value $L(\mathcal{P}_{T_u}) = s_n$. In other words, we need to solve an additional constrained optimization problem:

$$\min_{\mathcal{P}} \qquad T(\mathcal{P}) = \max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \ldots, \frac{\Delta_n}{b_n}\right\} \tag{2}$$

$$\begin{aligned}
\textit{subject to} \quad & \underset{k=1}{\overset{i}{}} \Delta_k \leq s_i, \text{ for } i = 1, 2, \ldots, n-1, \\
& \underset{k=1}{\overset{n}{}} \Delta_k = s_n, \\
& b_i \leq B_i, \\
& \underset{k=1}{\overset{n}{}} b_k \leq B^I, \\
& \Delta_i, b_i \in \{0\} \bigcup Z^+, \text{ for } i = 1, 2, \ldots, n
\end{aligned}$$

To solve this problem, we first introduce a new variable $y = \max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \ldots, \frac{\Delta_n}{b_n}\right\}$. Then we can transform the objective function into $\min_{\{\mathcal{P},y\}} T(\mathcal{P}, y) = y$, and also add some new constraints, $\Delta_i/b_i \leq y$ $(i = 1, 2, \ldots, n)$, into the above problem. This leads to the following equivalent optimization problem:

$$\begin{aligned}
\min_{\{\mathcal{P},y\}} \quad & T(\mathcal{P}, y) = y \qquad\qquad (3)\\
\textit{subject to} \quad & \mathop{}\limits_{k=1}^{i} \Delta_k \leq s_i, \text{ for } i = 1, 2, \ldots, n-1,\\
& \mathop{}\limits_{k=1}^{n} \Delta_k = s_n,\\
& b_i \leq B_i,\\
& \mathop{}\limits_{k=1}^{n} b_k \leq B^I,\\
& \Delta_i / b_i \leq y\\
& \Delta_i, b_i \in \{0\} \bigcup Z^+, \text{ for } i = 1, 2, \ldots, n
\end{aligned}$$

However, it is difficult to solve this problem, because constraints $\Delta_i/b_i \leq y$ are nonlinear (note that Δ_i, b_i, and y are all variables). In general, solving a nonlinear constrained optimization problem depends highly on its starting points. Without good starting points, it is easy to get stuck at local minima with poor solution qualities [16]. Sometimes, it is even difficult to find feasible solutions. In addition, it may take a long time to find the solutions.

Hence, instead of directly solving this nonlinear constrained problem, in Eqn.(3), we propose to find the optimal solution $\mathcal{P}^*$ by solving a sequence of the linear programming problems of Eqn.(1) with the delay bound gradually tightened.

As discussed above, the reason why we need to solve Eqn.(2) is that the delay bound T_u is too large. The purpose of solving Eqn.(2) is to find the minimum delay bound $T^* = T(\mathcal{P}^*)$, so that for any $T_u = T < T^*$, the solution $\mathcal{P}_T$ to Eqn.(1) satisfies $L(\mathcal{P}_T) < s_n$. Then the optimal solution $\mathcal{P}^*$ can be derived by solving Eqn.(1), given $T_u = T^*$.

In our approach, we aim to find this minimum delay bound T^* by bisection search. For efficient computation, it is important to find a good upper bound T^h and a good lower bound T^l. Obviously, T_u is a good choice for T^h. T^l can be set to zero, but we can find a better T^l by relaxing the bandwidth constraints in Eqn.(2). In other words, the lower bound T^l can be found by solving the following linear programming problem,

$$\min_{\vec{\Delta}} \quad T(\vec{\Delta}) = \max \quad \frac{\Delta_1}{B_1}, \frac{\Delta_2}{B_2}, \ldots, \frac{\Delta_n}{B_n} \tag{4}$$

$$\textit{subject to} \quad \overset{i}{\underset{k=1}{}} \Delta_k \leq s_i, \text{ for } i = 1, 2, \ldots, n-1,$$

$$\overset{n}{\underset{k=1}{}} \Delta_k = s_n,$$

$$\Delta_i \in \{0\} \bigcup Z^+, \text{ for } i = 1, 2, \ldots, n$$

which is derived from Eqn.(2) by removing the bandwidth constraints and the variables of transmission rates b_i. This formulation assumes that each supplying peer p_i can use its available bandwidth B_i, and the requesting peer has unlimited incoming bandwidth.

Let $\vec{\Delta}^o$ be the optimal solution to Eqn.(4). Then we can set the lower bound $T^l = T(\vec{\Delta}^o)$. To validate this lower bound, we need to prove $T(\vec{\Delta}^o) \leq T(\mathcal{P}^*)$.

Because $\mathcal{P}^* = \{(\Delta_i^*, b_i^*), i = 1, 2, \ldots, n\}$ is an optimal solution to Eqn.(2), it satisfies all the constraints of Eqn.(2). Note that the constraints of Eqn.(4) is a subset of Eqn.(2). Accordingly, if we substitute the corresponding segment vector $\vec{\Delta}^* = \{\Delta_i^*, i = 1, 2, \ldots, n\}$ into Eqn.(4), it will satisfy all the constraints of Eqn.(4). Thus, segment vector $\vec{\Delta}^*$ is a feasible solution to Eqn.(4). Because $\vec{\Delta}^o$ is the optimal solution to Eqn.(4), we can obtain $T(\vec{\Delta}^o) \leq T(\vec{\Delta}^*)$. In addition, due to the fact that $\frac{\Delta_i^*}{B_i} \leq \frac{\Delta_i^*}{b_i}$ for $i = 1, 2, \ldots, n$, we have $T(\vec{\Delta}^*) \leq T(\mathcal{P}^*)$. Therefore, we prove $T(\vec{\Delta}^o) \leq T(\mathcal{P}^*)$.

In summary, by solving Eqn.(4), we obtain a lower bound, $T^l = T(\vec{\Delta}^o)$, of the optimal solution to Eqn.(2), $T^*(= T(\mathcal{P}^*))$.

To solve Eqn.(4), we first introduce a new variable $y = \max\left\{\frac{\Delta_1}{B_1}, \frac{\Delta_2}{B_2}, \ldots, \frac{\Delta_n}{B_n}\right\}$ as before, and then we transform the objective function into $\min_{\{\Delta,y\}} T(\vec{\Delta}, y) = y$ by adding new constraints, $\Delta_i/B_i \leq y$ $(i = 1, 2, \ldots, n)$, into Eqn.(4). Since the new constraints are linear (note that Δ_i's and y are variables, and B_i's are not), this is also a linear programming problem, which can be efficiently solved by the *lp_solve* [5] package.

If the optimal solution $\vec{\Delta}^o$ to Eqn.(4) satisfies

$$\mathop{}_{k,\Delta_k^o \neq 0} B_k \leq B^I, \tag{5}$$

this means that the requesting peer has enough incoming bandwidth B^I to accommodate $\vec{\Delta}^o$. By setting

$$b_i = \begin{array}{ll} B_i & \text{if } \Delta_i^o \neq 0 \\ 0 & \text{otherwise} \end{array} \tag{6}$$

we obtain a peer allocation vector, $\mathcal{P}' = \{(\Delta_i^o, b_i), i = 1, 2, \ldots, n\}$.

Next we will prove that $\mathcal{P}'$ is an optimal solution to Eqn.(2). First, because $\vec{\Delta}^o$ is an optimal solution to Eqn.(4), it satisfies all the constraints of Eqn.(4). According to Eqn.(5) and Eqn.(6), we can obtain $b_i \leq B_i$ and $\sum_i b_i \leq B^I$, the bandwidth constraints of Eqn.(2). Hence, $\mathcal{P}'$ is a feasible solution to Eqn.(2), where all its constraints are satisfied. Second, from our earlier discussion, we know that $T(\vec{\Delta}^o) \leq T(\mathcal{P}^*) \leq T(\mathcal{P})$, for any feasible solution $\mathcal{P}$ to Eqn.(2). Since $T(\mathcal{P}') = T(\vec{\Delta}^o)$, we have $T(\mathcal{P}') \leq T(\mathcal{P})$ for any feasible solution $\mathcal{P}$. This means that $\mathcal{P}'$ is an optimal solution to Eqn.(2). In this case, we can stop here since we have already found the optimal solution $\mathcal{P}'$ to Eqn.(2).

```
0.  Given a user or application specified delay bound $T_u$
1.  Solve Eqn.(1) with delay bound $T_u$
2.  **If** $L(\mathcal{P}_{T_u}) < s_n$, **then** solution is found and stop
3.  Find the lower bound $T^l$ by solving Eqn.(4)
4.  **If** the solution satisfies Eqn.(5), **then** solution is found and stop
5.  Set the upper bound $T^h = T_u$
6.  **If** $T^l$ and $T^h$ are sufficiently close, **then** solution is found and stop
7.  Set $T = (T^l + T^h)/2$
8.  Solve Eqn.(1) with delay bound $T$
9.  **If** $L(\mathcal{P}_T) = s_n$, **then**
10.     set $T^h = T$
11. **Else**
12.     set $T^l = T$
13. **Endif**
14. **Goto** step 6
```

Figure 2: Sequence of linear programming solution.

On the other hand, if $\vec{\Delta}^o$ does not satisfy Eqn.(5), it implies that the bandwidth constraints of Eqn.(2) are violated. In this scenario, given T^h and T^l, we perform bisection search between T^l and T^h to find the minimum T^* by solving a sequence of linear programming problems of Eqn.(1), such that the optimal solution $L(\mathcal{P}_{T^*}) = s_n$. Figure 2 summarizes our peer assignment algorithm.

5 Experimental Results

In this section, we evaluate the computational complexity and performance of our proposed peer assignment algorithm (SLPAssign), shown in Figure 2.

In our SLPAssign algorithm, we set the terminating threshold of the bisection search (Step 6 in Fig.2) to be 0.001. The experiments were carried out in a Pentium-IV Linux PC with 1.8GHz CPU and 512MB memory.

5.1 Comparison with Nonlinear Solution

In this subsection, we will compare the computational complexity and performance of SLPAssign with the approach of directly solving the nonlinear formulation in Eqn.(2), when the delay bound T_u is larger than the minimum transmission time.

To solve this nonlinear formulation, we use the *fmincon()* function in Matlab. The function uses sequential quadratic programming [14] to find a constrained minimum, starting from an initial estimate. As is well known, the quality of the solution depends heavily on the choice of starting points. Two sets of starting points are evaluated in our experiments: 1) StartOne using the solution $\mathcal{P}$ derived from the first linear programming problem in Eqn.(1) as the starting point; and 2) StartTwo using an all-one vector as the starting point.

We compare the complexity and performance of SLPAssign with StartOne and StartTwo in the following two scenarios. In the first scenario, the incoming bandwidth B^I of the requesting peer is larger than the total outgoing bandwidth B^O of the supplying peers. Table 3 and Table 4 show the comparison results for 4 and 8 supplying peers, with respect to different delay bounds T_u (in seconds).

In the second scenario, we consider the case where the incoming bandwidth B^I is 50 percent of the total bandwidth B^O. The comparison results with respect to different delay bounds T_u (in seconds) for both 4 and 8 supplying peers are depicted in Table 5 and Table 6.

In these tables, the solution (in seconds) is referred to image transmission time $T(\mathcal{P})$ that measures the quality of peer assignment $\mathcal{P}$ obtained by using SLPAssign or nonlinear formulation. The CPU time in seconds measures time used to find the solution. The iteration number listed for SLPAssign shows the number of linear programming problems that SLPAssign needs to solve after Step 1 in Fig.2. We interpret the results as follows.

First, SLPAssign can always find the peer assignment with the minimum transmission

Table 3: Comparison results for 4 supplying peers among SLPAssign, StartOne, and StartTwo, when the incoming bandwidth of the requesting peer is larger than the total outgoing bandwidth of the supplying peers ($B^I \geq B^O$). The symbol n/s means that no solution is found within the maximum CPU time, 300 seconds.

delay bound	SLPAssign			StartOne		StartTwo	
	solution	CPU	iter	solution	CPU	solution	CPU
Tu=3	2.4	0.001	1	3.000	0.063	2.792	0.141
Tu=4	2.4	0.001	1	4.000	0.031	n/s	300.0
Tu=5	2.4	0.001	1	5.000	0.036	4.838	0.070
Tu=6	2.4	0.001	1	6.000	0.062	4.838	0.063
Tu=7	2.4	0.001	1	7.000	0.031	6.781	0.094
Tu=8	2.4	0.001	1	8.000	0.031	4.838	0.109
Tu=9	2.4	0.001	1	9.000	0.078	4.838	0.093
Tu=10	2.4	0.001	1	10.000	0.063	9.845	0.110
Tu=11	2.4	0.001	1	11.000	0.062	4.838	0.110

Table 4: Comparison results for 8 supplying peers among SLPAssign, StartOne, and StartTwo, when the incoming bandwidth of the requesting peer is larger than the total outgoing bandwidth of the supplying peers ($B^I \geq B^O$).

delay bound	SLPAssign			StartOne		StartTwo	
	solution	CPU	iter	solution	CPU	solution	CPU
Tu=4	3.236	0.001	1	4.000	0.078	3.254	0.406
Tu=5	3.236	0.001	1	3.347	0.417	3.272	28.719
Tu=6	3.236	0.001	1	4.185	0.11	3.261	6.000
Tu=7	3.236	0.001	1	4.185	0.188	3.266	0.406
Tu=8	3.236	0.001	1	4.187	0.25	3.253	0.500
Tu=9	3.236	0.001	1	4.185	0.14	3.303	0.609
Tu=10	3.236	0.001	1	4.180	0.219	3.294	0.297
Tu=11	3.236	0.001	1	4.185	0.125	3.281	39.172
Tu=12	3.236	0.001	1	4.185	0.172	3.373	0.375

time. However, the nonlinear solution does not guarantee to find a good peer assignment solution all the time. It is easy to get stuck at local minima. The solution quality of StartOne and StartTwo is poorer than that of SLPAssign.

Second, the nonlinear solution is highly dependent on the selected starting points. It is difficult to find a set of starting points that consistently work the best for all experiments.

Table 5: Comparison results for 4 supplying peers among SLPAssign, StartOne, and StartTwo, when the incoming bandwidth of the requesting peer is 50 percent of the total outgoing bandwidth of the supplying peers ($B^I = 50\% \times B^O$).

delay bound	SLPAssign			StartOne		StartTwo	
	solution	CPU	iter	solution	CPU	solution	CPU
Tu=4	3.939	0.004	13	4.000	0.031	4.000	0.085
Tu=5	3.939	0.004	15	5.000	0.031	5.000	0.094
Tu=6	3.939	0.004	14	6.000	0.031	3.939	0.093
Tu=7	3.939	0.005	17	7.000	0.032	4.075	0.109
Tu=8	3.939	0.004	14	8.000	0.031	8.000	0.093
Tu=9	3.939	0.004	16	9.000	0.032	9.000	0.094
Tu=10	3.939	0.004	14	10.000	0.032	10.000	0.094
Tu=11	3.939	0.004	15	11.000	0.032	11.000	0.088
Tu=12	3.939	0.005	17	4.457	0.141	12.000	0.078

Table 6: Comparison results for 8 supplying peers among SLPAssign, StartOne, and StartTwo, when the incoming bandwidth of the requesting peer is 50 percent of the total outgoing bandwidth of the supplying peers ($B^I = 50\% \times B^O$).

delay bound	SLPAssign			StartOne		StartTwo	
	solution	CPU	iter	solution	CPU	solution	CPU
Tu=5	4.461	0.006	12	4.457	0.203	5.000	0.203
Tu=6	4.457	0.007	13	4.457	0.103	6.000	0.141
Tu=7	4.458	0.007	13	4.457	0.11	5.253	0.391
Tu=8	4.457	0.007	14	4.460	0.36	8.000	0.156
Tu=9	4.457	0.007	14	4.457	0.36	6.285	0.359
Tu=10	4.457	0.007	14	4.457	0.266	4.602	1.172
Tu=11	4.457	0.008	16	4.457	0.125	5.276	5.985
Tu=12	4.457	0.008	16	4.457	0.109	5.778	121.812
Tu=13	4.457	0.009	18	4.457	0.141	13.000	0.141

StartTwo finds better solutions than those of StartOne in Table 4, whereas StartOne finds better solutions in Table 6.

Third, we measure the computational overhead as time used to find a solution divided by the image transmission time (that is, $\frac{\text{cpu}}{\text{solution}}$). The computational overhead of SLPAssign is negligible, but that of nonlinear solutions is much higher. In some cases, the overhead for solving nonlinear formulation can be very significant. Many problem instances in these four tables incur overhead larger than 100%.

In summary, our SLPAssign performs much better than the nonlinear solutions in both computational overhead and solution quality.

5.2 Comparison with Two Heuristics

To further evaluate the performance of SLPAssign, we compare it with two heuristic schemes, LongAssign and GreedyAssign, used in the example in Section 3.

We designed our experiments to simulate two types of applications: image transmission in low-bandwidth environments and video transmission in the Internet. To gain some insight how SLPAssign, LongAssign, and GreedyAssign respond with different delay bounds, we carried out the experiments to calculate the quality of downloaded content, as the delay bound T_u gradually increases from 0 to the time needed by GreedyAssign to finish downloading the content.

5.2.1 Image Transmission in Low Bandwidth Environment

In this subsection, we compare the performance of SLPAssign, LongAssign, and GreedyAssign for the application of image transmission in low bandwidth environments. The bandwidth in some wireless networks, such as bluetooth-based WPANs and cellular networks, is very limited. It is usually in the order of tens of kilobits per second. To model such an environment, we set the outgoing bandwidth B_i of supplying peers to be within 4 kbps and 32 kbps.

The image transmitted is the 512×512 *Barbara* image, fine-scalable coded by SPIHT [20]. The bit rates used by supplying peers to encode the *Barbara* image using SPIHT are randomly generated between 0.125 bpp and 1 bpp, resulting in the coded image of size s_i between 32 kbits and 256 kbits. The maximum size of the coded image is 256 kbits (*i.e.,* the *Barbara* image coded in 1 bpp).

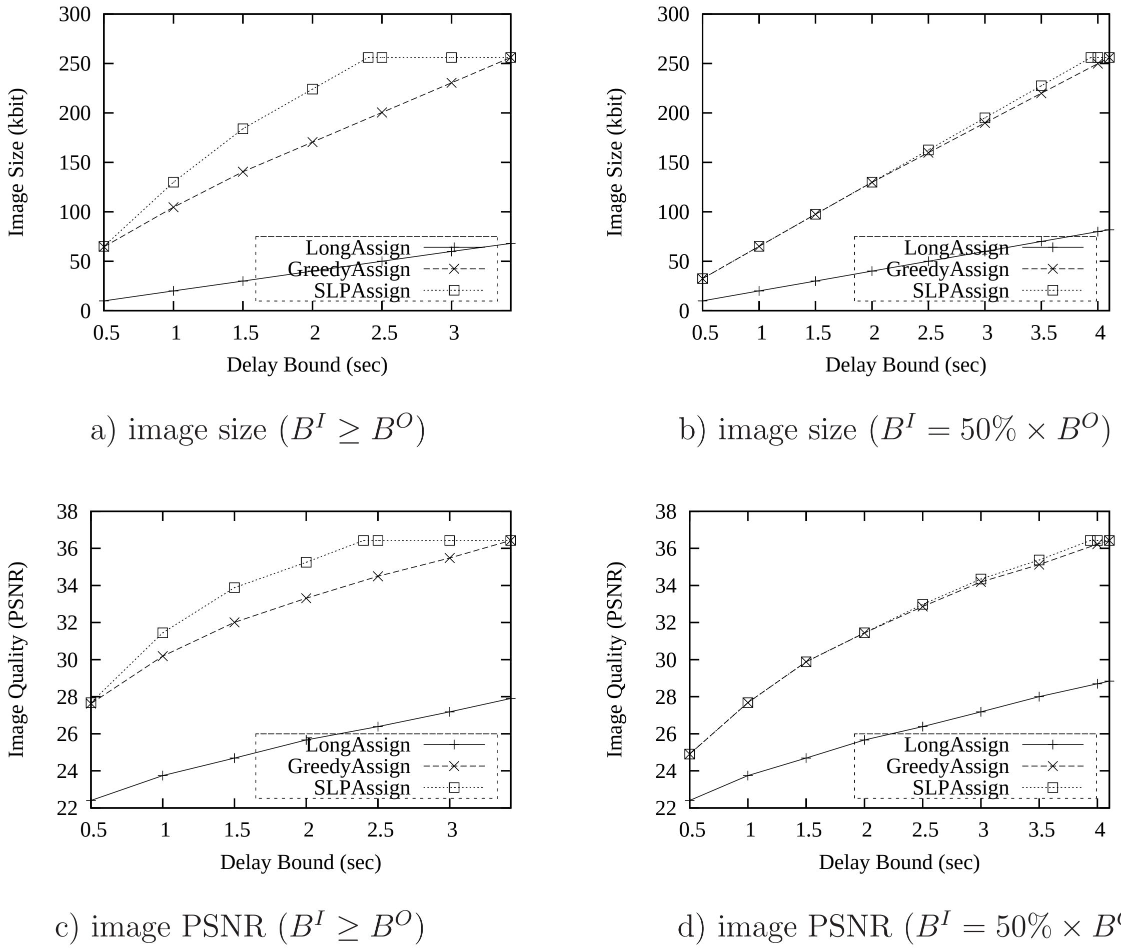
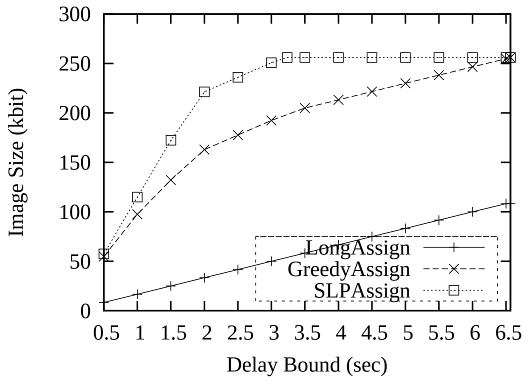


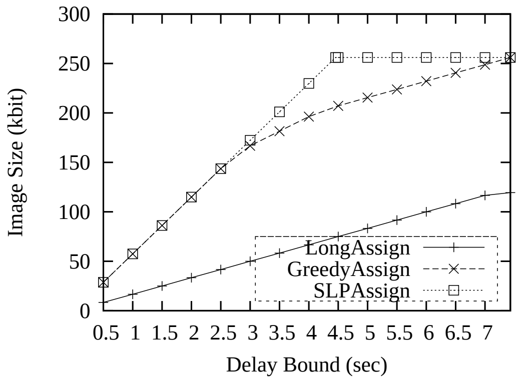
Figure 3: Comparison of downloaded image size (a and b) and quality (c and d) among SLPAssign, GreedyAssign, and LongAssign with respect to a range of delay bounds, in a P2P system with 4 supplying peers.

Due to the dynamic nature of wireless PANs or cellular networks, the number of the supplying peers is not likely to be very large. We set the number of the supplying peers to be less than 10, from which we choose two sets of representative comparison results and plot them in Figure 3 and Figure 4, respectively.

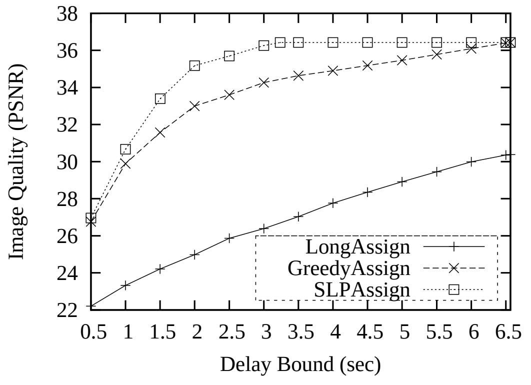
From these experimental results, we can make several observations. First, for any given delay bound T_u, LongAssign results in the worst image quality in terms of PSNR. This is not surprising, since LongAssign involves less number of supplying peers in image transmission. This shows the advantage of sharing scalable-coded content on P2P networks.



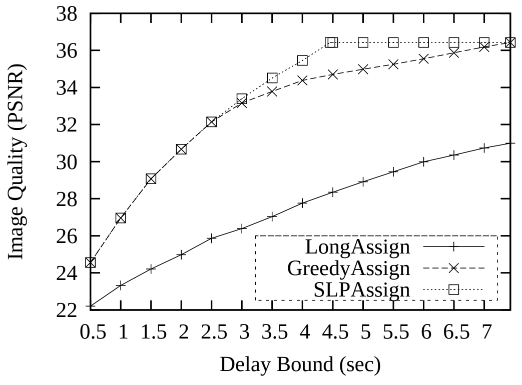
a) image size ($B^I \geq B^O$)



b) image size ($B^I = 50\% \times B^O$)



c) image PSNR ($B^I \geq B^O$)



d) image PSNR ($B^I = 50\% \times B^O$)

Figure 4: Comparison of downloaded image size (a and b) and quality (c and d) among SLPAssign, GreedyAssign, and LongAssign with respect to a range of delay bounds, in a P2P system with 8 supplying peers.

By exploiting the embedding property of scalable coding, we have more supplying peers available to contribute and share resources. In addition, SLPAssign always obtains the image of the best quality in terms of PSNR, for a given delay bound among the three approaches.

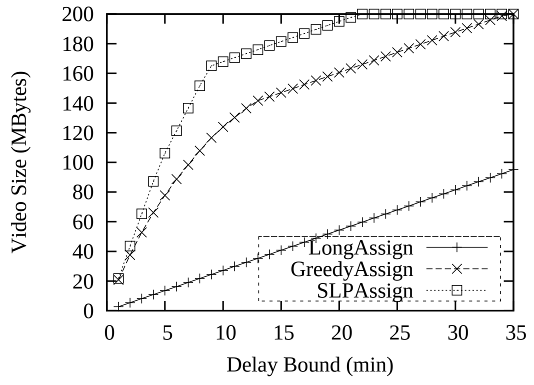
Second, SLPAssign results in much shorter image transmission time to obtain the best-quality image. For example, SLPAssign only takes half of the time used by GreedyAssign in Figure 4a) and 4c), and 60% of the time by GreedyAssign in Figure 4b) and 4d), respectively. Besides, when the delay bound is larger than the maximum downloading time of SLPAssign (*e.g.*, 2.4 seconds in Figure 3a) and 3c), 3.9 seconds in Figure 3b) and 3d), 3.2 seconds in Figure 4a) and 4c), 4.4 seconds in Figure 4b) and 4d)), SLPAssign obtains the best image in shorter than the delay bound, while GreedyAssign downloads an image with poorer quality at the delay bound.

Third, when the incoming bandwidth of the requesting peer is limited and the number of the supplying peers is small, the performance gap between SLPAssign and GreedyAssign becomes small. This is observed in Figure 3b) and 3d). Intuitively, we can understand this as follows. In Figure 3b) and 3d), there are 4 supplying peers, and the incoming bandwidth of the requesting peer is equal to 50% of the total outgoing bandwidth of the supplying peers. This bandwidth constraint can be considered as equivalent to reducing the number of supplying peers, and it is fair to say that we essentially have 2 supplying peers. The advantage due to the optimal scheduling of SLPAssign diminishes in such a small system, when the variance in the image sizes of the two supplying peers is small. In this scenario, GreedyAssign is able to achieve close-to-optimal peer assignment.

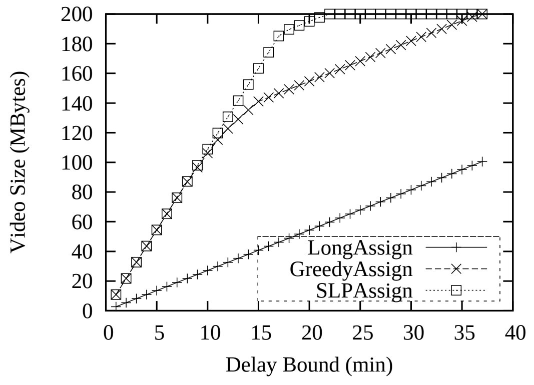
5.2.2 Video Transmission in the Internet

Video transmission is always a bandwidth-constrained application, even with the rapid adoption of broadband networks. Here we evaluate the performance of SLPAssign, LongAssign, and GreedyAssign for video transmission in the Internet. We set the outgoing bandwidth of supplying peers to be between 128 and 384 kbps, corresponding to the uplink bandwidths of DSLs and cable modems.

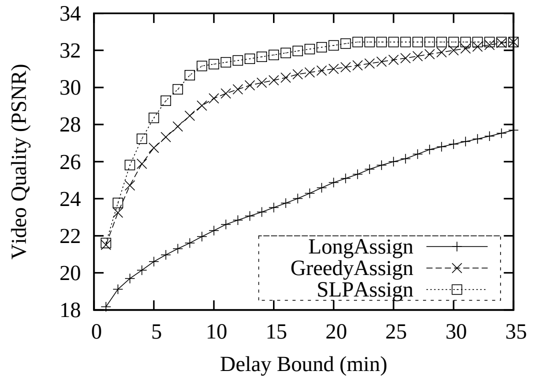
Since the standard public video sequences, such as MPEG-4 sequences, are relatively short, we used our own video sequence, which was captured and digitized from a boxing match in a TV program. This video sequence has 33100 frames in CIF format of dimension 352×288, representing about 18-minute video captured in 30 frames/sec. We used a fine-scalable video coding algorithm, 3D-SPIHT [15], to compress the video sequence into 200



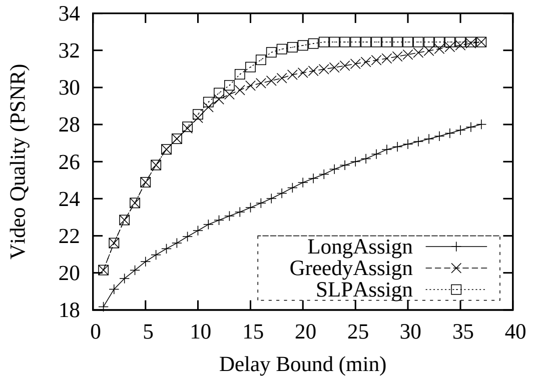
a) video size ($B^I \geq B^O$)



b) video size ($B^I = 50\% \times B^O$)



c) video PSNR ($B^I \geq B^O$)



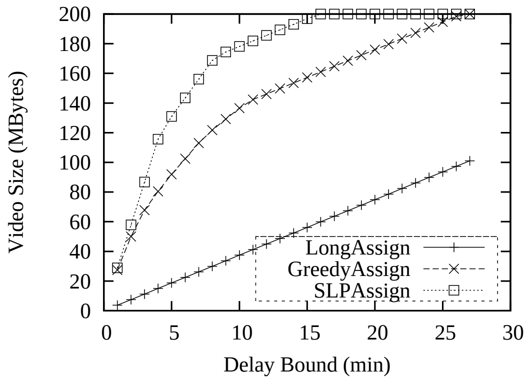
d) video PSNR ($B^I = 50\% \times B^O$)

Figure 5: Comparison of downloaded video size (a and b) and quality (c and d) among SLPAssign, GreedyAssign, and LongAssign with respect to a range of delay bounds, in a P2P system with 12 supplying peers

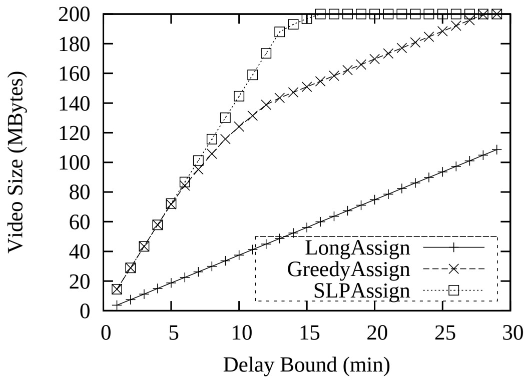
MBbytes.

We experimented with the P2P networks with the number of supplying peers ranging from 10 and 20. To save space, we only reported the results with 12 and 16 supplying peers. The other results are similar. Figure 5 and Figure 6 show the experimental results for 12 and 16 supplying peers, respectively.

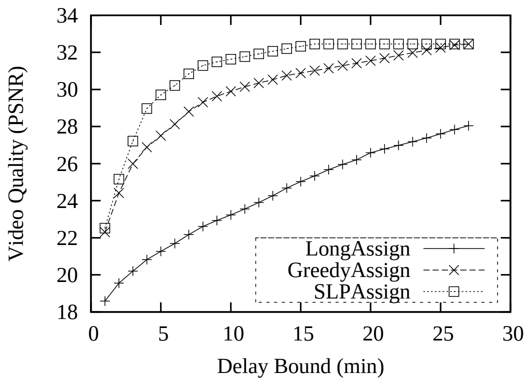
We can reach similar conclusions as in the image transmission experiments. LongAssign is still the worst among the three, and SLPAssign consistently outperforms GreedyAssign and LongAssign. Further, in our video transmission experiments, the performance gap



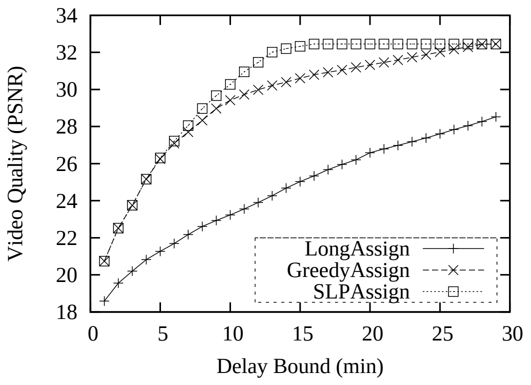
a) video size ($B^I \geq B^O$)



b) video size ($B^I = 50\% \times B^O$)



c) video PSNR ($B^I \geq B^O$)



d) video PSNR ($B^I = 50\% \times B^O$)

Figure 6: Comparison of downloaded video size (a and b) and quality (c and d) among SLPAssign, GreedyAssign, and LongAssign with respect to a range of delay bounds, in a P2P system with 16 supplying peers

between SLPAssign and GreedyAssign is much more significant. With the same delay bound, SLPAssign downloads up to 45 MBytes more video bit streams than GreedyAssign, and thus the video quality is up to 3dB better.

It is interesting to note that the performance advantage of SLPAssign becomes more obvious as the number of supplying peers increases. This is because when we have more supplying peers, it is more likely that the supplying peers have more chances to have the coded videos of different sizes. With more variances in the video sizes, GreedyAssign may result in poorer peer assignment in which the supplying peers leave the system at different time instances. However, SLPAssign can systematically maximize the utilization of the available bandwidth by scheduling the supplying peers to finish transmission at almost the same time.

The above analysis also explains why the performance gain of SLPAssign over GreedyAssign is larger when the incoming bandwidth is sufficient ($B_I \geq B_O$). When a requesting peer has more abundant incoming bandwidth, it is able to involve more supplying peers in transmission.

5.2.3 Result Summary

In summary, our experimental results have demonstrated that SLPAssign has negligible computational overhead and achieves excellent performance compared to the two heuristic approaches. The proposed SLPAssign is a general approach, and it can be used to download scalable coded images or videos in bandwidth-limited environments. Its performance gain becomes more significant when there are more supplying peers on P2P networks.

6 Conclusions and Future Work

In this paper, we studied how to maximize the quality of delivering fine-scalable coded content on P2P networks, under the constraints that the content has to be displayed within a delay bound and the incoming bandwidth of a requesting peer is limited. We formulated this problem as constrained optimization problems, and then efficiently solved them using a sequence of linear programming. The main contributions of this paper are three-fold. First, we have demonstrated how to exploit the embedding property of scalable coding to design optimal peer assignment strategies on P2P networks. To be more realistic, we have incorporated both delay and bandwidth constraints in our problem formulation. Second, we have proposed an efficient solution method to solve the nonlinear formulation by using a sequence of linear programming, and have proved its optimality. Third, we have applied our peer assignment algorithm to both image and video transmissions in bandwidth-limited environments. The experimental results have verified excellent performance of our proposed algorithm.

In the future, we plan to study transmission of coarse-scalable coded content, such as those generated by MPEG-2 scalability modes and JPEG2000.

References

[1] BitTorrent. http://bitconjurer.org/BitTorrent.

[2] eDonkey. http://www.edonkey2000.com.

[3] Gnutella. http://gnutella.wego.com.

[4] KaZaA. http://www.kazaa.com/us/index.htm.

[5] lp_solve 4.0. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.

[6] Napster. http://www.napster.com.

[7] V. Agarwal and R. Rejaie. Adaptive multi-source streaming in heterogeneous peer-to-peer networks. In *Proc. of SPIE/ACM Multimedia Computing and Networking*, San Jose, USA, Jan 2005.

[8] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. ACM SIGCOMM*, August 2002.

[9] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth content distribution in a cooperative environment. In *Proc. IPTPS'03*, February 2003.

[10] Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proc. ACM SIGMETRICS*, Jun 2000.

[11] Y. Cui, B. Li, and K. Nahrstedt. oStream: asynchronous streaming multicast in application-layer overlay networks. *IEEE journal on Selected Areas in Communications*, 22:91–106, January 2004.

[12] Y. Cui and K. Nahrstedt. layered peer-to-peer streaming. In *Proc. ACM/IEEE NOSSDAV*, 2003.

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

[14] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, 1981.

[15] Kim, Z. Xiong, and W. A. Pearlman. Low bit-rate scalable video coding with 3D set partitioning in the hierarchical trees (3D SPIHT). *IEEE Trans. on Circuits and Systems for Video Technology*, 10(8):1374–1387, December 2000.

[16] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag New York, Inc., 1999.

[17] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient peer-to-peer streaming. Technical Report MSR-TR-2003-11, Microsoft Research, March 2003.

[18] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proc. ACM/IEEE NOSSDAV*, Miami, FL, USA, May 2002.

[19] R. Rejaie and A. Ortega. PALS: Peer to peer adaptive layered streaming. In *Proc. ACM Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2003.

[20] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6:243-250, June 1996.

[21] D. Stolarz. Peer-to-peer streaming media delivery. In *Proc. of First Int'l Conf. on Peer-to-Peer Computing*, August 2001.

[22] X. Su, R. Fatoohi, and T. Wang. Optimizing transmission time of scalable coded images in peer-to-peer networks. *ACM Multimedia Systems Journal*, 2005.

[23] D. A. Tran, K. A. Hua, and T. T. Do. ZIGZAG: an efficient peer-to-peer scheme for media streaming. In *Proc. IEEE INFOCOM*, April 2003.

[24] D. A. Tran, K. A. Hua, and T. T. Do. A peer-to-peer architecture for media streaming. *IEEE journal on Selected Areas in Communications*, 22:121–133, January 2004.

[25] Y. Wang, J. Ostermann, and Y.-Q. Zhang. *Video Processing and Communications*. Prentice Hall, NJ, 1st edition, 2001.

[26] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. In *Proc. of SPIE/ACM Multimedia Computing and Networking*, San Jose, CA, January 2002.

[27] X. Zhang, J. Liuy, B. Liz, and T.-S. P. Yum. Data-driven overlay streaming: Design, implementation, and experience. In *Proc. IEEE INFOCOM*, Miami, Florida, Apr 2005.

[28] R. Zimmermann and L. S. Liu. Active: Adaptive low-latency peer-to-peer streaming. In *Proc. of SPIE/ACM Multimedia Computing and Networking*, San Jose, USA, Jan 2005.