January, 2005

Delay-Constrained Transmission of Fine Scalable Coded Content over P2P Networks

Xiao Su, *San Jose State University*
Yi Shang, *University of Missouri*
Tao Wang, *Synopsys, Inc.*
Yuqing Mai, *San Jose State University*

Delay-Constrained Transmission of Fine-Scalable Coded Content over P2P Networks

Xiao Su*, Yi Shang+, Tao Wang- and Yuqing Mai*

Computer Engineering Department*
San Jose State University
San Jose, CA 95192, USA

Department of Computer Science+
University of Missouri-Columbia
Columbia, MO 65211, USA

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

Abstract

Images or videos are generally coded before transmission, and they can be either non-scalable coded or scalable coded, which can be further classified into fine-scalable coded and coarse-scalable coded. In this paper, we focus on delivery of fine-scalable coded content. The objective of our work is to design algorithms to optimize the quality of fine-scalable coded images or videos on peer-to-peer networks when a requesting peer is delay-sensitive and has to display content within a certain delay bound.

Fine-scalable coding has two properties: (1) it embeds lower bit-rate bitstreams into higher bit-rate bitstreams; and (2) its coding quality increases with every additional bit in the coded bitstream. These imply that for a given requesting peer, there may exist a dynamic set of supplying peers that have the same content coded in different bit rates, have different outgoing bandwidth and may finish transmission at different times.

In this paper, we first illustrate the importance of peer assignment problem using an example. Then we formally define and formulate the problem as two integer programming problems, from which we derive an optimal peer assignment solution. Finally, we carry out extensive experiments to verify the excellent performance of the proposed peer assignment algorithms by comparing with other heuristic schemes.

1 Introduction

The peer-to-peer (P2P) architecture is especially appealing to content delivery applications, as 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 [18].

In this paper, we study how to maximize the quality of images and videos on peer-to-peer networks when a requesting peer is delay-constrained and has to display content within a delay bound. Here, the delay bound is usually determined by the types of applications. For example, when a user surfs the web, he may lose his patience if the transmission time exceeds several seconds. But if he downloads images or videos offline, this delay bound could be much longer. For the applications of video streaming, the delay bound should be smaller than the inter-frame interval to guarantee the smooth playback of video.

This problem largely depends on how the content is coded before transmission, since coding algorithms define the property of coded bit streams. For this purpose, we can coarsely classify image and video coding algorithms into two categories: scalable coding [16, 17, 22] that embeds lower bit-rate bitstreams into higher bit-rate bitstreams, and non-scalable coding that does not have this embedding property. To elaborate the difference between scalable and non-scalable coding, let us represent the coded bitstream as a string of k bits $C = c_1 c_2 \ldots c_k$, where k is a parameter depending on coded bit rates, and its value increases with bit rate. Let C_1 and C_2 be the bitstreams generated by coding an image in bit rate r_1 and r_2, respectively, and $r_1 < r_2$. 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 strings.

Scalable coding algorithms can be further classified as fine-scalable coding and coarse-scalable coding. Algorithms, such as "Set Partitioning in Hierarchical Trees" (SPIHT) [16], can generate fine-scalable coded content, whose quality increases with every additional bit

in the coded bitstream. Other scalable coding algorithms, such as JPEG2000 [13], may generate coarse-scalable coded content, whose quality only increases with a sequence of additional bits in the coded bitstream. In this paper, we focus on transmission of fine-scalable coded content, and will study transmission of coarse-scalable coded content in the future.

Next, let us analyze how coding algorithms affect content delivery in P2P networks. If a peer requests a multimedia clip that is non-scalable coded in d bit/pixel, then another peer holding the *same* clip coded in d_n bit/pixel can serve as a supplying peer, only if $d_n = d$. On the other hand, if a peer requests a clip that is scalable coded in d bit/pixel, then another peer holding the *same* clip coded in d_s bit/pixel can serve as a supplying peer, regardless of the value of d_s. Thus, requesting scalable coded content on peer-to-peer networks is likely to receive better Quality of Service (QoS) as more supplying peers are available.

In this paper we study the peer assignment problem in peer-to-peer networks with the objective to maximize delivery quality within a delay bound. In peer assignment, we divide scalable coded content into multiple parts, and then assign them to the supplying peers for transmission. Because the supplying peers have different outgoing bandwidth and have clips coded in different bit rates, different assignment solutions will result in significantly different quality of the content received by the requesting peer, especially when the number of the supplying peers is large. Therefore, it is very important to derive optimal peer assignment solutions to maximize the content quality.

We first illustrate the importance of the peer assignment problem through an example, and then formally define and formulate this problem as two integer programming problems, from which we derive an optimal peer assignment solution. Through extensive simulations, we show that our proposed optimal peer assignment algorithm is both efficient and effective, comparing with other heuristic algorithms.

The paper is organized as follows. We introduce the peer assignment problem using an example in Section 2, state our assumptions and define the problem in Section 3, and propose our solution in Section 4. To evaluate the performance of the proposed algorithm, we compare it with several heuristics in Section 5. After discussing related work in Section 6, we conclude the paper by identifying future research directions in Section 7.

2 Example

In this section, we walk through an image transmission example to gain a better understanding of issues underlying delay-constrained content delivery on P2P networks.

Suppose in a P2P system, there are four peers holding coded *lena* image (of dimension 512×512): p_1 has the image coded in 0.25 bit per pixel, *i.e.,* 0.25 bpp, (resulting in a coded bitstream of size 64 kbit), p_2 has the image coded in 0.5 bpp (*i.e.*, coded bitstream of size 128 kbit), p_3 and p_4 have the image coded in 1 bpp (*i.e.*, coded bitstream of size 256 kbit). Furthermore, p_1, p_2, p_3, and p_4 have different outgoing bandwidth of 50 kbps, 20 kbps, 40 kbps, and 20 kbps, respectively.

Now consider the scenario that a requesting peer tries to download *lena* image within 2 seconds from these four supplying peers, and its incoming bandwidth is greater than the sum of the supplying peers' outgoing bandwidth.

If the requested image is non-scalable coded in 1 bpp, then only p_3 and p_4 are eligible to work as supplying peers. In this case, the best image transmission scheme is illustrated in Figure 1, in which x-axis shows the image portions assigned to each peer for transmission during a time slot (*i.e.,* [0, 2] second), and y-axis shows this transmission slot. In this case, the coded image is assigned to p_3 and p_4 based on their outgoing bandwidth: p_3 transmits 80 kbit of the coded bitstream, between [0, 80) kbit, and p_4 transmits 40 kbit, between [80, 120) kbit. At $t = 2$ second, the requesting peer is only able to decode from the 120 kbit of

coded bitstream that has been received. As the requested image is non-scalable coded, the user will only get to see a portion of the requested image.

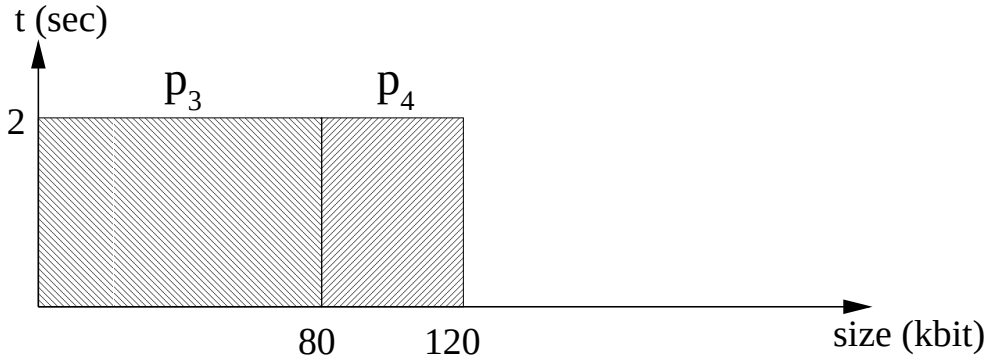


Figure 1: Transmission of a non-scalable coded image.

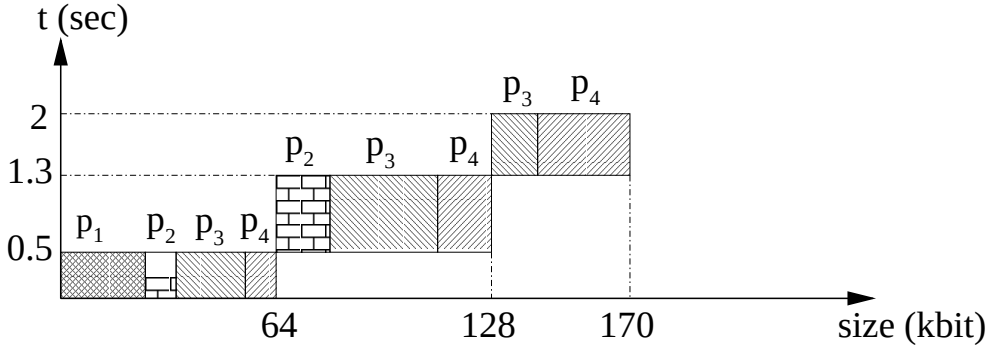


Figure 2: Transmission of a scalable coded image using a greedy approach.

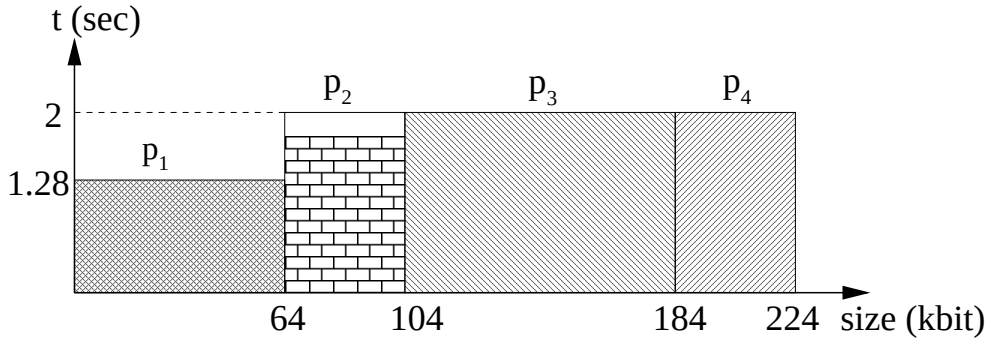


Figure 3: Transmission of a scalable coded image with an optimal assignment.

On the other hand, if the requested image is scalable coded in 1 bpp, then all the four peers are eligible to work as supplying peers. One possible way to allocate transmission

Figure 4: Non-scalable coded *lena* image (PSNR=16.90dB) when transmitted within 2 seconds.

a) greedy approach (PSNR=38.25dB) b) optimal approach (PSNR=39.60dB)

Figure 5: Scalable coded *lena* image when transmitted using a) a greedy approach and b) an optimal approach within 2 seconds.

tasks is to involve all the possible supplying peers in transmission whenever they have the image segments being transmitted. This greedy scheme is illustrated in Figure 2. First, all the four peers contribute to the transmission of coded bitstream Between $[0, 64)$ kbit. At $t = 0.5(\approx 64/(50 + 20 + 40 + 20))$ second, p_1 leaves because it does not have bitstream

beyond 64 kbit. Then, p_2, p_3, and p_4 work together to supply the coded bitstream between $[64, 128)$ kbit. At $t = 1.3(\approx 0.5 + 64/(20 + 40 + 20))$ second, p_2 leaves because it runs out of its bitstream. After that, p_3 and p_4 continue their transmission based on their outgoing bandwidth until the delay bound ($t = 2$) is met. Using this greedy method, the requesting peer receives a bitstream of 170 ($\approx 128 + (2 - 1.3) \times (40 + 20)$) kbit.

However, this method does not result in the decoded image of the best possible quality. An optimal method is shown in Figure 3. In this method, p_1 transmits the bitstream between $[0, 64)$ kbit, p_2 transmits between $[64, 104)$ kbit, p_3 transmits between $[104, 184)$ kbit, and p_4 transmits between $[184, 224]$ kbit. All the four peers start transmission at time 0, and p_1 finishes at 1.28 second, and the rest three peers finish at 2 second. This method results in a transmission of 224 kbit in total. As the decoding quality is proportional to the size of the received bitstream, this method results in the best possible quality among the three transmission schemes.

To illustrate the difference in visual quality of the three cases in the above example, we show the decoded *lena* images in Figure 4 and Figure 5. In all our experiments, we employ JPEG codec to generate non-scalable coded images and SPIHT to generate fine-scalable coded images. Figure 4 shows the *lena* image when only 120 kbit of the JPEG-coded bitstream is received. We can see that the decoded image is only a partial image with poor PSNR (16.90dB). Figure 5a) and 5b) show the decoded *lena* images when 170 kbit and 224 kbit of SPIHT-coded stream are received by using greedy assignment and optimal assignment approaches. We can see that the transmitted image using the optimal approach is visually sharper and has better objective quality (ΔPSNR=1.35dB).

Therefore, it is important to develop a systematic approach to find the transmission scheme that results in the best possible decoding quality with the delay constraint.

3 Assumptions and Problem Definition

In this section, we state our assumptions and formally define the problem of allocating transmission tasks to different supplying peers.

3.1 Assumptions

In our P2P system, delay-constrained transmission of fine-scalable coded content is done in four steps. First, a requesting peer prompts a user to specify his delay requirements, e.g., the delay bound for image transmission or the inter-frame interval for video streaming. Second, a requesting peer employs a certain directory lookup algorithm to locate a potential set of supplying peers for the request. Third, the requesting peer applies a peer assignment algorithm to divide transmission task to different supplying peers with the objective to maximize the delivery quality within the delay bound specified by the user. Fourth, the supplying peers are informed about their own allocations by the requesting peer and then start transmission. In this paper, we develop the peer assignment algorithm applied in the third step.

In this paper, we make the following assumptions on the peer-to-peer systems with regard to content delivery.

1. *Relationship in peers' bandwidth.* We consider a single requesting peer with incoming bandwidth B and multiple supplying peers with heterogeneous bandwidth $b_1, b_2, \ldots, b_n$, where n is the number of supplying peers. We assume that the sum of supplying bandwidth does not exceed the incoming bandwidth of the requesting peer, *i.e.,* $\sum_{i=1}^{n} b_i \leq B$. Considering the asymmetric property of current residential broadband access (such as cable modem and DSL), where the downlink bandwidth (*i.e.,* incoming bandwidth) is usually much larger than the uplink bandwidth (*i.e.,* outgoing bandwidth), this assumption is likely to be true for the scenarios when the

number of supplying peers is not huge.

2. *Reliable delivery.* We assume content is reliably transmitted, by using application-level or transport-level reliable delivery protocols. In this case, the bandwidth refers to effective bandwidth observed by peers using these protocols. Therefore, the algorithms developed in this paper also apply to the case of varying bandwidth, where b_i refers to effective (or expected) bandwidth. Then the objective of our algorithms is to maximize expected quality of fine-scalable coded content.

3. *Peer transience.* We also assume that all the participating peers are well-behaved, in other words, they will not leave the system before finishing their assigned transmission tasks.

In the rest of the paper, we will base our discussions mainly on image transmission, although the problem definitions and proposed algorithms can be easily extended and adapted to downloading fine-scalable coded videos, such as those coded by 3D-SPIHT [11].

3.2 Problem Definition

To facilitate further discussions, we first define some notations. For a given requesting peer, there are n supplying peers with image sizes, s_i $(i = 1, 2, \ldots, n)$, coded in different bit rates. Without loss of generality, we assume $s_1 \leq s_2 \leq \ldots \leq s_n$, otherwise we can re-number the peers to follow this order. These supplying peers have different outgoing bandwidth, b_i $(i = 1, 2, \ldots, n)$. The requesting peer asks to download an image within T_u seconds and its incoming bandwidth is no smaller than $\sum_{i=1}^{n} b_i$. Given the above notations, let us define the concept of *Peer Assignment Set*:

DEFINITION 1 *Peer Assignment Set of peer i (PAS_i) is defined as the set of small image segments:* $\{(x_{i,1}, x_{i,2}), (x_{i,3}, x_{i,4}), \ldots (x_{i,u_i}, x_{i,u_i+1}\}$, *where* $0 \leq x_{i,1} < x_{i,2} < x_{i,3} <$

$x_{i,4} < \ldots < x_{i,u_i} < x_{i,u_i+1} \leq s_i$, *and* $x_{i,j}$ *(*j *is odd) and* $x_{i,j+1}$ *(*$j+1$ *is even) are the lower and upper endpoints of the* $\left(\frac{j+1}{2}\right)^{th}$ *image segment,* $(x_{i,j}, x_{i,j+1})$*, which is assigned to peer* i *for transmission.*

Let us illustrate this concept using the above two examples. To find PAS_i $(i = 1, 2, 3, 4)$ for peer assignment solution derived from the greedy approach (Figure 2), we consider image segments transmitted during the three time slots: (1) between $[0, 0.5)$ second, p_1 transmits $(0, 25(\approx 50*64/(50+20+40+20)))$ kbit; p_2 transmits $(25, 34(\approx 25+20*64/(50+20+40+20)))$ kbit; p_3 transmits $(34, 54(\approx 34+40*64/(50+20+40+20)))$ kbit; and p_4 transmits $(54, 64)$ kbit; (2) between $[0.5, 1.3)$ second, p_2 transmits $(64, 80(= 64+20*64/(20+40+20)))$ kbit; p_3 transmits $(80, 112(= 80+40*64/(20+40+20)))$ kbit; and p_4 transmits $(112, 128)$ kbit; (3) between $[1.3, 2)$ second, p_3 transmits $(128, 156(\approx 128+40*43/(40+20)))$ kbit; and p_4 transmits $(156, 170)$ kbit. Therefore, PAS_1 is $\{(0, 25)\}$ with $u_1 = 1$; PAS_2 is $\{(25, 34), (64, 80)\}$ with $u_2 = 3$; PAS_3 is $\{(34, 54), (80, 112), (128, 156)\}$ with $u_3 = 5$; and PAS_4 is $\{(54, 64), (112, 128), (156, 170)\}$ with $u_4 = 5$. The unit of these numbers is kbit.

As a second example, in the peer assignment solution derived from the optimal approach in Figure 3, PAS_1 is $\{(0, 64)\}$; PAS_2 is $\{(64, 104)\}$; PAS_3 is $\{(104, 184)\}$; and PAS_4 is $\{(184, 224)\}$. In this case, the value of u_i $(i = 1, 2, 3, 4)$ is equal to 1, and the unit is again in kbit.

DEFINITION 2 *Peer Assignment Method (PAM) of an image transmission request consists of Peer Assignment Set for all the peers, i.e.,* PAS_1*,* PAS_2*, ...,* PAS_n *if they satisfy the following condition: image segments defined in the PAM combine into a large continuous image block, i.e., for any upper endpoint of an image segment in* PAS_i*,* $x_{i,j}$ *with even* j *(except the largest upper endpoint), there always exists an image segment, whose lower endpoint (*$x_{p,q}$ *with odd* q*) is equal to* $x_{i,j}$*.*

For example, PAM derived from the greedy approach refers to $\{(0, 25)\}$, $\{(25, 34), (64, 80)\}$, $\{(34, 54), (80, 112), (128, 156)\}$, and $\{(54, 64), (112, 128), (156, 170)\}$. This is because $(0, 25)$ (*i.e.,* $(x_{1,1}, x_{1,2})$, the first image segment in PAS_1) combines with $(25, 34)$ (*i.e.,* $(x_{2,1}, x_{2,2})$, the first image segment in PAS_2), which in turn combines with $(34, 54)$ (*i.e.,* $(x_{3,1}, x_{3,2})$, the first image segment in PAS_3), and if we continue this process, we find that the image segments defined in the above PAS_i combine into a continuous image block between $(0, 170)$ kbit.

As another example, PAM derived from the optimal approach refers to $\{(0, 64)\}$, $\{(64, 104)\}$, $\{(104, 184)\}$, and $\{(184, 224)\}$, and these segments also combine into a continuous image block in (0, 224) kbit.

DEFINITION 3 *A Peer Assignment Method is canonical if it satisfies the following two conditions: 1. the cardinality of PAS$_i$ ($i = 1, 2, \ldots, n$) is one, i.e., each PAS$_i$ has only one image segment $\{(x_{i,1}, x_{i,2})\}$, $i = 1, 2, \ldots, n$; 2. the end points that define the image segment in PAS$_i$, $i = 1, 2, \ldots, n$, follow an increasing order, i.e., $x_{1,1} < x_{1,2} = x_{2,1} < x_{2,2} = x_{3,1} < x_{3,2} \ldots x_{n-1,2} = x_{n,1} < x_{n,2}$.*

Intuitively, a canonical PAM sequentially assigns continuous image segments to peers based on their ordering. In the example we discussed above, the PAM derived from the optimal solution is canonical, while that of the greedy solution is not since it violates both cardinality and ordering conditions.

Given the above two definitions, we further define *Peer Allocation Length Vector*.

DEFINITION 4 *An Peer Allocation Length Vector (PALV) $\{\Delta_i, i = 1, 2, \ldots, n\}$ is the vector in which the i^{th} element, $\Delta_i = \sum_{k=1}^{u_i+1}(-1)^k x_{i,k} = x_{i,2} - x_{i,1} + x_{i,4} - x_{i,3} + \ldots + x_{i,u_i+1} - x_{i,u_i}$ defines the total size of the image segments in PAS$_i$ (PAS of peer i).*

PALV calculated from the greedy solution is $\{25, 25(=9+16), 80(=20+32+28), 40(=10+16+14)\}$, and PALV of the optimal solution is $\{64, 40, 80, 40\}$.

4 Proposed Canonical Peer Assignment Method

4.1 Optimal Canonical Peer Assignment Method

The objective of this work is to derive an optimal PAM so that the quality of the content delivered within a delay bound, T_u, is optimal. First, we prove that for any non-canonical PAM that results in optimal content quality, there exists an equivalent canonical PAM that also results in the optimal content quality. In other words, if we design our peer assignment solution following the canonical form, we are guaranteed to find an optimal solution.

THEOREM 1 *If there exists a non-canonical PAM (PAS_1, PAS_2, ..., PAS_n), which results in the optimal image quality, then a canonical PAM' (PAS'_1, PAS'_2, ..., PAS'_n) can always be found that also results in the optimal image quality.*

Its proof can be found in Appendix A. Basically, Theorem 1 has established the fact that if we sequentially assign continuous image segments to peers based on their ordering, we are guaranteed to find an optimal canonical peer assignment method. The key problem here is how to find the sizes of the image segments assigned to peers to transmit, *i.e.*, the PALV $\vec{\Delta} = \{\Delta_1, \Delta_2, \ldots, \Delta_n\}$. After deriving $\vec{\Delta}$, an optimal canonical peer assignment method can be found in the following way: $\text{PAS}_1 = \Delta_1$ and $\text{PAS}_i = \{(\sum_{k=1}^{i-1} \Delta_k, \sum_{k=1}^{i} \Delta_k)\}$, for $i = 2, \ldots, n$.

To find PALV with maximum image quality, we need to maximize the quantity of $\Delta_1 + \Delta_2 + \ldots + \Delta_n$, under the constraints that each peer finishes its assigned image segment within the delay bound T_u and that each peer has the assigned image segment within its boundary. In other words, this problem can be formulated as a constrained integer

programming problem in the following way:

$$\begin{aligned} \max_{\vec{\Delta}} \quad & L(\vec{\Delta}) = \Delta_1 + \Delta_2 + \ldots + \Delta_n \\ \textit{subject to} \quad & \Delta_i / b_i \leq T_u, \\ & \sum_{k=1}^{i} \Delta_k \leq s_i, \\ & \Delta_i \in \{0\} \bigcup Z^+, \quad i = 1, 2, \ldots, n \end{aligned} \tag{1}$$

If the optimal value, $L(\vec{\Delta})$, is equal to s_n $(= \max\{s_1, s_2, \ldots, s_n\})$, it means that the delay bound T_u is large enough for the requesting peer to obtain a coded image 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 can send its coded image of size 100 kbit (s_1) at the rate of 50 kbps (b_1), and p_2 can send its coded image of size 120 kbit (s_2) at the rate of 30 kbps (b_2). For this problem, we can easily find two optimal PALVs: PALV one is $\{0, 120\}$, and PALV two is $\{75, 45\}$. In both PALVs, the requesting peer has received an image coded in 120 kbit, which is the optimal quality image that can be supplied by p_1 and p_2. However, PALV one results in much longer image transmission time ($\max\{0/50, 120/30\} = 4$ sec) than PALV two does ($\max\{75/50, 45/30\} = 1.5$ sec). Obviously PALV two is a preferable solution.

Therefore, we need to find a PALV solution that also minimizes the image transmission time when $L(\vec{\Delta}) = s_n$. In other words, we need to solve an additional constrained

optimization problem:

$$\begin{aligned} \min_{\vec{\Delta}} \quad & \max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \ldots, \frac{\Delta_n}{b_n}\right\} \\ \textit{subject to} \quad & \sum_{k=1}^{i} \Delta_k \leq s_i, \text{ for } i = 1, 2, \ldots, n-1, \\ & \sum_{k=1}^{n} \Delta_k = s_n, \\ & \Delta_i \in \{0\} \bigcup Z^+, \text{ for } i = 1, 2, \ldots, n \end{aligned} \tag{2}$$

Note that the objective function in Eqn.(2) is nonlinear. To transform it into an equivalent linear problem, we introduce a new variable $y = \max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \ldots, \frac{\Delta_n}{b_n}\right\}$, and add the following set of constraints, $\frac{\Delta_i}{b_i} \leq y$ $(i = 1, 2, \ldots, n)$, into the above problem as follows.

$$\begin{aligned} \min_{\{\vec{\Delta}, y\}} \quad & y \\ \textit{subject to} \quad & \frac{\Delta_i}{b_i} \leq y, \text{ for } i = 1, 2, \ldots, n, \\ & \sum_{k=1}^{i} \Delta_k \leq s_i, \text{ for } i = 1, 2, \ldots, n-1, \\ & \sum_{k=1}^{n} \Delta_k = s_n, \\ & \Delta_i \in \{0\} \bigcup Z^+, \text{ for } i = 1, 2, \ldots, n \end{aligned} \tag{3}$$

To summarize, our algorithm to derive an optimal canonical PAM is outlined in Figure 6.

1. solve Eqn.(1)
2. **if** $L(\vec{\Delta}) = s_n$ **then**
3. solve Eqn.(3)
4. **end-if**
5. set $\text{PAS}_1 = \Delta_1$ and $\text{PAS}_i = \{(\sum_{k=1}^{i-1} \Delta_k, \sum_{k=1}^{i} \Delta_k)\}$, for $i = 2, \ldots, n$.

Figure 6: Optimal Peer Assignment Algorithm.

4.2 Distance Bounds for Sub-Optimal Peer Assignment Method

Figure 6 suggests that we may derive an optimal peer assignment method using one of the following two ways: a) by solving Eqn.(1); and b) by solving Eqn.(3), if $L(\vec{\Delta}) = s_n$.

Integer programming problems are normally expensive to solve, and this prevents it from being used in our applications. In our work, we employ a linear programming package, *lp_solve* [4], to solve both Eqn.(1) and Eqn.(3), and then use a rounding algorithm to find an integer solution. In this section, we define rounding algorithms for cases a) and b) and study their performance.

Let us denote $\vec{\hat{\Delta}} = \{\hat{\Delta}_1, \hat{\Delta}_2, \ldots, \hat{\Delta}_n\}$ as an optimal continuous PALV, and $\vec{\Delta} = \{\Delta_1, \Delta_2, \ldots, \Delta_n\}$ as an integer PALV obtained from $\vec{\hat{\Delta}}$.

In case a), we simply set $\Delta_i = \lfloor \hat{\Delta}_i \rfloor$, for $i = 1, 2, \ldots, n$. It's easy to see that the constraints in Eqn.(1) are still satisfied, as $\frac{\Delta_i}{b_i} \leq \frac{\hat{\Delta}_i}{b_i} \leq T_u$, and $\sum_{k=1}^{i} \Delta_i \leq \sum_{k=1}^{i} \hat{\Delta}_i \leq s_i$, for $i = 1, 2, \ldots, n$. The distance from the optimal integer PALV is less than the distance from the continuous PALV, which is upper bounded by $|\sum_{k=1}^{n} \hat{\Delta}_k - \sum_{k=1}^{n} \Delta_k| \leq n$. In practice, the number of supplying peers is not likely to be a large value (mostly less than 100), therefore, the rounded solution is within tens of bits from the optimal solution.

In case b), the delay bound T_u is large enough to receive the image of the best possible quality, and the objective is to minimize the transmission time to receive the best image. Let us first define vector $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\} = \{\hat{\Delta}_1, \hat{\Delta}_1 + \hat{\Delta}_2, \ldots, \sum_{k=1}^{n} \hat{\Delta}_k\}$. Observe that

1. Calculate $f_i = \hat{x}_i - \lfloor \hat{x}_i \rfloor$, for $i = 1, 2, \ldots, n-1$.
2. **foreach** i in $\{1, 2, \ldots, n-1\}$ **do**
3. **if** f_i is equal to zero **then** do nothing and skip
4. **if** $\frac{f_i}{b_{i+1}} < \frac{1-f_i}{b_i}$
5. **then** $x_i = \lfloor \hat{x}_i \rfloor$
6. **else** $x_i = \lceil \hat{x}_i \rceil$
7. **end-for**
8. $x_n = \hat{x}_n$

Figure 7: Rounding to integer solutions.

$\hat{x}_n$ is already an integer, as it satisfies the constraint, $\hat{x}_n = s_n$, in Eqn.(3), and s_n is an integer. For every $\hat{x}_i$ ($i = 1, 2, \ldots, n-1$), we can either round it to $\lfloor \hat{x}_i \rfloor$ or $\lceil \hat{x}_i \rceil$. The rounding criterion is to minimize the increase in the transmission time compared with the continuous solution. If we round $\hat{x}_i$ to $\lfloor \hat{x}_i \rfloor$, then the increase in transmission time is the time for peer $i+1$ to transmit the portion between $\lfloor \hat{x}_i \rfloor$ and $\hat{x}_i$. Similarly if we round $\hat{x}_i$ to $\lceil \hat{x}_i \rceil$, then the increase in transmission time is the time for peer i to transmit the portion between $\hat{x}_i$ to $\lceil \hat{x}_i \rceil$. Based on this analysis, we present our rounding algorithm in Figure 7. From the rounded vector, $(x_1, x_2, \ldots, x_n)$, we can derive peer assignment set PAS_i (Line 5 in Figure 6) as: $\text{PAS}_i = \{(x_{i-1}, x_i)\}$ for $i = 1, 2, \ldots, n$, where $x_0 = 0$.

To assess the quality of this rounded solution, Theorem 2 establishes an upper bound on the distance between this rounded solution and the optimal integer solution.

THEOREM 2 *The difference between the transmission time of the rounded solution obtained by the algorithm in Figure 7 and that of the optimal integer solution is upper bounded by* $\frac{1}{\min\{b_1, b_2, \ldots, b_n\}}$.

The proof of this theorem is included in Appendix B. As the bandwidth values, b_i ($i = 1, 2, \ldots, n$), are in the range of kbit per second, this upper bound will be a very small fractional value. Therefore, we can conclude that the quality of the rounded solution is very close to that of the optimal integer solution.

5 Experimental Results

To evaluate the performance of our proposed peer assignment algorithm (OptAssign) described in Section 4, we compare it with two other heuristic algorithms: a greedy assignment algorithm (GreedyAssign) in which a supplying peer always participates in transmission until it runs out of clip (details outlined in Figure 8) and a fastest single peer assignment algorithm (FastAssign) in which we choose the peer with the largest outgoing bandwidth to transmit. Among the three algorithms, OptAssign and GreedyAssign involve multiple peers in transmission, while FastAssign involves only one supplying peer, which is similar to the model of traditional P2P file-sharing systems [1].

1. set peer index (pi) to 0;
2. set transmitted image size (IS) to 0;
3. set overall transmission time (TT) to 0;
4. **while** $pi \leq n$ **do**
5. calculate the time (t_1) for $p_{pi}, p_{pi+1}, \ldots, p_n$ to transmit image between $[s_{pi-1}, s_{pi}]$.
6. add up t_1 to TT
7. **if** TT exceeds delay bound T_u **then**
8. calculate the size of image segment transmitted before T_u as is_{seg}
9. add is_{seg} to IS
10. **break**
11. **else**
12. add $s_{pi} - s_{pi-1}$ to IS
13. **end-if**
14. **end-while**
15. output transmission time (TT) and image size (IS)

Figure 8: Greedy Peer Assignment Algorithm.

In our experiments, we set the range of bandwidth to be between 512 bps (bit/sec) and 32 kbps, and consider the images of size 512×512 and 1024×1024 coded in either 0.5 bpp (bit per pixel) or 1 bpp. Therefore, for an image of 512×512, supplying peers may have coded images with sizes in $(0, 128)$ kbit (*i.e.,* (0, 0.5) bpp) or in $(0, 256)$ kbit (*i.e.,* (0, 1) bpp); similarly, for an image of 1024×1024, supplying peers may have coded images with sizes in $(0, 512)$ kbit (*i.e.,* (0, 0.5) bpp) or in $(0, 1024)$ kbit (*i.e.,* (0, 1) bpp).

Table 1: Comparison of three algorithms (OptAssign, GreedyAssign and FastAssign), when the requested coded image size is no greater than 128 kbit.

T_u (s)	#Peers	OptAssign			GreedyAssign			FastAssign		
		Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time
$T_u = 1$	2	31.8	30.90	1.0	31.8	30.90	1.0	21.4	29.30	1.0
	4	63.8	33.97	1.0	57.4	33.43	1.0	25.6	29.97	1.0
	8	114.6	36.56	1.0	92.5	35.66	1.0	28.1	30.35	1.0
	12	127.2	37.08	0.7	114.4	36.54	1.0	29.3	30.55	1.0
	16	127.8	37.11	0.6	122.5	36.88	1.0	30.1	30.67	1.0
	20	127.9	37.11	0.5	125.6	37.02	0.9	30.6	30.76	1.0
	24	128.0	37.11	0.4	127.1	37.08	0.9	30.8	30.80	1.0
$T_u = 2$	2	63.7	33.96	2.0	62.9	33.89	2.0	42.9	32.11	2.0
	4	109.2	36.35	1.9	94.3	35.74	2.0	47.2	32.62	2.0
	8	126.9	37.07	1.3	120.5	36.80	1.9	50.4	32.88	2.0
	12	128.0	37.11	0.8	127.5	37.10	1.4	51.9	33.00	2.0
	16	128.0	37.11	0.6	127.9	37.11	1.3	54.7	33.22	2.0
	20	128.0	37.11	0.5	127.9	37.11	1.1	55.1	33.25	2.0
	24	128.0	37.11	0.4	128.0	37.11	1.0	55.5	33.28	2.0
$T_u = 5$	2	114.0	36.53	4.1	109.0	36.34	4.5	89.9	35.48	5.0
	4	125.7	37.02	3.0	121.5	36.84	4.0	81.7	35.02	5.0
	8	128.0	37.11	1.4	127.8	37.10	2.5	81.3	35.00	5.0
	12	128.0	37.11	0.8	128.0	37.11	1.5	82.5	35.06	5.0
	16	128.0	37.11	0.6	128.0	37.11	1.3	82.5	35.06	5.0
	20	128.0	37.11	0.5	128.0	37.11	1.1	82.3	35.05	5.0
	24	128.0	37.11	0.4	128.0	37.11	1.0	81.4	35.00	5.0

For each of the image request, we perform the experiments for the peer-to-peer systems consisting of 2, 4, 8, 12, 16, 20 and 24 peers. The experiments are done on a Dell workstation with Pentium-III 2.4GHz CPU and 512M memory.

Tables 1, 2, 3 and 4 compare the performance of the three algorithms (OptAssign, GreedyAssign and FastAssign), when the requested image size is upper bounded by 128 kbit, 256 kbit, 512 kbit, and 1024 kbit, respectively. In each table, we show results for three different delay bounds in P2P systems consisting of various number of supplying peers. For each algorithm, we compute three performance metrics: 1) Size (in kbit) that refers to the image size transmitted within the delay bound; 2) PSNR that reports the quality of the received image; and 3) Transmission Time (in second) that shows the time to finish an image transmission request (note that in some cases the requested image has been received

Table 2: Comparison of three algorithms (OptAssign, GreedyAssign and FastAssign), when the requested coded image size is no greater than 256 kbit.

T_u (s)	#Peers	OptAssign			GreedyAssign			FastAssign		
		Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time
$T_u = 1$	2	32.1	30.93	1.0	32.1	30.93	1.0	21.9	29.39	1.0
	4	63.8	33.96	1.0	63.8	33.96	1.0	25.7	29.98	1.0
	8	125.1	36.99	1.0	97.7	35.88	1.0	28.3	30.39	1.0
	12	186.5	38.79	1.0	145.1	37.63	1.0	29.5	30.59	1.0
	16	223.9	39.59	1.0	173.5	38.33	1.0	30.1	30.68	1.0
	20	238.2	39.90	1.0	187.3	38.81	1.0	30.6	30.76	1.0
	24	246.7	40.09	0.9	207.8	39.25	1.0	30.8	30.79	1.0
$T_u = 2$	2	64.2	34.00	2.0	64.2	34.00	2.0	43.9	32.24	2.0
	4	127.5	37.09	2.0	125.6	37.02	2.0	51.4	32.96	2.0
	8	202.1	39.13	2.0	155.8	37.95	2.0	51.7	32.99	2.0
	12	231.3	39.75	2.0	195.3	38.98	2.0	54.9	33.24	2.0
	16	245.7	40.07	1.7	217.6	39.46	2.0	57.8	33.46	2.0
	20	251.0	40.18	1.6	230.0	39.72	2.0	57.7	33.46	2.0
	24	252.1	40.20	1.4	241.7	39.97	2.0	58.1	33.50	2.0
$T_u = 5$	2	159.6	38.04	5.0	148.3	37.73	5.0	109.6	36.36	5.0
	4	238.1	39.89	4.3	216.1	39.43	5.0	127.9	37.11	5.0
	8	244.8	40.05	3.9	225.4	39.62	4.9	99.8	35.97	5.0
	12	248.2	40.12	3.5	239.0	39.91	4.7	112.3	36.47	5.0
	16	253.9	40.24	2.5	250.3	40.17	4.0	123.3	36.91	5.0
	20	254.6	40.26	2.2	252.6	40.22	3.7	114.3	36.54	5.0
	24	254.6	40.26	2.1	253.7	40.24	3.3	116.7	36.64	5.0

before the delay bound). All the reported results are calculated as the average of successful runs of 100 randomly generated P2P systems. For example, the (Size, PSNR, Time) values report for 2-peer system with $T_u = 1$s are calculated as average (Size, PSNR, Time) values for 100 2-peer P2P systems, each of which has different coded image sizes and outgoing bandwidths.

We have the following observations on the comparison results.

- In all the experiments, FastAssign results in the worst image quality (in terms of PSNR) within the specified delay bounds. This is of course not surprising, since FastAssign only involves one supplying peer in transmission. On the other hand, this suggests the advantage of sharing scalable-coded contents in P2P networks, because scalable coding results in more peers contributing and sharing resources.

Table 3: Comparison of three algorithms (OptAssign, GreedyAssign and FastAssign), when the requested coded image size is no greater than 512 kbit.

T_u (s)	#Peers	OptAssign			GreedyAssign			FastAssign		
		Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time
$T_u = 1$	2	32.6	30.31	1.0	32.6	30.31	1.0	21.2	28.76	1.0
	4	67.1	32.73	1.0	67.1	32.73	1.0	25.9	29.44	1.0
	8	135.0	34.85	1.0	134.1	34.83	1.0	28.6	29.85	1.0
	12	199.0	35.96	1.0	194.4	35.88	1.0	29.5	29.98	1.0
	16	263.3	36.91	1.0	253.5	36.79	1.0	29.9	30.03	1.0
	20	330.7	37.75	1.0	306.0	37.42	1.0	30.3	30.07	1.0
	24	395.6	38.42	1.0	341.9	37.90	1.0	30.6	30.10	1.0
$T_u = 2$	2	65.1	32.66	2.0	65.1	32.66	2.0	42.5	31.29	2.0
	4	134.2	34.84	2.0	131.5	34.77	2.0	51.7	31.91	2.0
	8	270.0	36.98	2.0	254.4	36.80	2.0	57.2	32.27	2.0
	12	397.8	38.44	2.0	350.4	37.98	2.0	59.0	32.38	2.0
	16	484.1	39.15	1.9	430.2	38.73	2.0	59.8	32.43	2.0
	20	510.0	39.35	1.6	468.0	39.03	2.0	60.5	32.46	2.0
	24	511.4	39.36	1.4	481.8	39.13	2.0	61.1	32.49	2.0
$T_u = 5$	2	162.8	35.35	5.0	162.8	35.35	5.0	106.2	34.10	5.0
	4	333.6	37.80	5.0	287.3	37.19	5.0	127.8	34.66	5.0
	8	503.9	39.29	3.9	478.1	39.10	4.9	141.6	34.97	5.0
	12	511.8	39.36	2.8	502.9	39.28	4.5	146.1	35.05	5.0
	16	512.0	39.36	2.0	510.8	39.35	3.4	148.2	35.09	5.0
	20	512.0	39.36	1.7	511.6	39.36	3.1	150.2	35.12	5.0
	24	512.0	39.36	1.4	511.7	39.36	2.9	151.7	35.15	5.0

- In certain setups, images of the best possible quality can be transmitted within the delay bounds, illustrated by the actual Time values smaller than the specified delay bounds in the Tables. For each specified delay bound, OptAssign algorithm always results in the largest number of setups that have their actual transmission times smaller than the specified delay bound. For example, in Table 1, when $T_u = 1$s, OptAssign has 4 such setups, while GreedyAssign has only 2 such setups. Even in the experimental setups when both OptAssign and GreedyAssign can download the requested image before the delay bound, the actual time taken by OptAssign is much smaller than that taken by GreedyAssign. This can be attributed to the second integer programming formulation (Eqn.(3)) where we optimize transmission time when the delay bound is large enough to transmit the image of the best possible

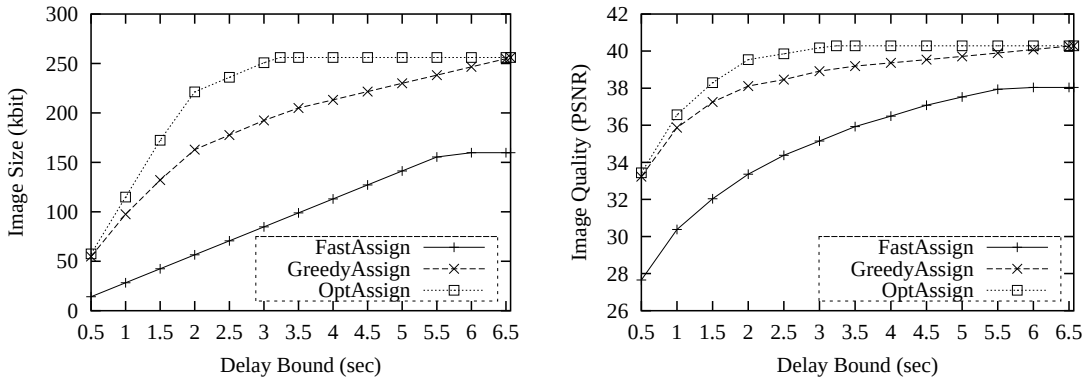
Table 4: Comparison of three algorithms (OptAssign, GreedyAssign and FastAssign), when the requested coded image size is no greater than 1024 kbit.

T_u (s)	#Peers	OptAssign			GreedyAssign			FastAssign		
		Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time	Size (kbit)	PSNR	Time
$T_u = 1$	2	31.8	30.21	1.0	31.8	30.21	1.0	20.8	28.64	1.0
	4	63.7	32.59	1.0	63.7	32.59	1.0	24.9	29.30	1.0
	8	131.9	34.78	1.0	131.1	34.76	1.0	28.4	29.81	1.0
	12	197.5	35.92	1.0	192.4	35.85	1.0	29.6	29.99	1.0
	16	262.5	36.89	1.0	251.7	36.77	1.0	30.3	30.07	1.0
	20	323.7	37.65	1.0	304.0	37.40	1.0	30.5	30.09	1.0
	24	387.4	38.34	1.0	351.0	37.98	1.0	30.8	30.12	1.0
$T_u = 2$	2	63.6	32.59	2.0	63.6	32.59	2.0	41.6	31.18	2.0
	4	127.3	34.65	2.0	125.1	34.60	2.0	49.8	31.79	2.0
	8	263.7	36.91	2.0	247.6	36.72	2.0	56.7	32.24	2.0
	12	395.0	38.42	2.0	347.9	37.96	2.0	59.2	32.40	2.0
	16	524.9	39.50	2.0	438.0	38.79	2.0	60.5	32.46	2.0
	20	645.6	40.41	2.0	497.4	39.23	2.0	61.0	32.48	2.0
	24	765.2	41.26	2.0	557.7	39.81	2.0	61.5	32.50	2.0
$T_u = 5$	2	157.8	35.27	5.0	141.0	34.96	5.0	102.9	34.01	5.0
	4	316.7	37.56	5.0	264.0	36.91	5.0	123.7	34.56	5.0
	8	598.3	40.08	5.0	468.8	39.03	5.0	141.2	34.96	5.0
	12	789.4	41.44	5.0	593.8	40.05	5.0	147.8	35.08	5.0
	16	975.6	42.65	4.7	752.3	41.17	5.0	151.0	35.13	5.0
	20	1014.4	42.87	4.1	862.8	41.94	5.0	152.2	35.16	5.0
	24	1021.3	42.91	3.5	927.9	42.36	5.0	153.6	35.19	5.0

quality.

When we observe smaller transmission time than delay bound, we might expect to see that the size of transmitted image is equal to the maximum size of available images. However, this is not the case with some experimental setups due to two reasons. First, there exist small distances from the optimal solutions when we apply rounding algorithms as described in Section 4.2. Second, reported values are in fact the average of 100 P2P setups. If only a subset of these 100 P2P setups finishes transmission before the delay bound while the rest of setups does not, we can expect to see that the average Time value is smaller than the delay bound and that the average Size value is also smaller than the maximum image size of supplying peers.

- In cases that both OptAssign and GreedyAssign can not finish downloading images



a) image size b) image quality (PSNR)

Figure 9: Comparison of a) transmitted image size and b) image quality (PSNR) for a range of delay bounds.

within specified delay bounds, the quality gap between OptAssign and GreedyAssign roughly increases with the number of peers in P2P systems. For example, in Table 4, for the experiments done with $T_u = 2$s, OptAssign and GreedyAssign result in the same quality in 2-peer P2P systems, and when the number of peers increases, the gain by OptAssign gradually increases until 1.55 dB in 24-peer P2P systems. This suggests that the gain by OptAssign is more evident in large P2P systems where peer assignment problem becomes more complicated.

Tables 1, 2, 3 and 4 showed comparison results when delay bound T_u is set to 1, 2, and 5 seconds. To gain some insight on how the three algorithms compare when the delay bound gradually increases, we carried out experiments to calculate the size and quality of downloaded images when delay bound (T_u) changes from 0 to the maximal downloading time with a step size of 0.5 second. Figure 9a) and 9b) illustrate comparison of image size and image quality, respectively. The results were obtained in a 8-peer P2P system when requesting 512 $\times$512 *lena* image and the maximal size of the coded bitstream held by all the supplying peers is 256 kbit (*i.e., lena* image coded in 1 bpp).

First, we see that FastAssign could not transmit the maximal size of the coded bit-

stream (256 kbit). This happened because the chosen supplying peer did not have the maximum-sized coded bitstream. Second, both FastAssign and GreedyAssign could transmit maximum-sized bitstreams, but as discussed before, OptAssign took only half of the time needed by GreedyAssign to fulfil the image transmission request (3.2 vs 6.6 second). Third, if we draw a vertical line corresponding to a specific delay bound in Figure 9b) (or 9a)), we can see that the image quality (or size) is always the best for OptAssign. When the delay bound is less than 3.2 second, the improvement in image quality (or size) due to OptAssign increases with delay bound T_u. To understand why, observe that when the delay bound is small, every supplying peer is able to contribute within this bound. In this case, the optimal peer assignment is indeed the greedy-based assignment in which each peer contributes based on its outgoing bandwidth. However, when the delay bound becomes larger, some peers may run out of image clips and have to leave before the delay bound is reached. In this case, the advantage of OptAssign becomes more obvious because the set of supplying peers is dynamic over time. After delay bound reaches 3.2 second, OptAssign's quality (or size) stays constant since it has reached its best possible value, while GreedyAssign gradually catches up in quality (or size).

To compare the subjective quality of images transmitted by three algorithms, Figure 10a), 10b) and 10c) plot the image transmitted by FastAssign, GreedyAssign, and OptAssign, respectively when T_u is equal to 2 seconds. We can see that the image transmitted by OptAssign is visually the best in terms of preserving high-frequency details.

6 Related Work

Early peer-to-peer file sharing systems, such as Napster [5], FastTrack [1] and Gnutella [2], normally identify a single supplying peer by a directory lookup algorithm. These systems differ in their choices of directory lookup algorithms [12]. A recent commercial system,

a) FastAssign (PSNR=33.36dB)

b) GreedyAssign (PSNR=38.12dB)

c) OptAssign (PSNR=39.54dB)

Figure 10: *lena* images downloaded using a) FastAssign b) GreedyAssign and c) OptAssign when $T_u = 2$ (s) in a 8-peer system.

KaZaA [3], extends their data sharing models from one supplying peer to a more general model of multiple supplying peers for each requesting peer. However, since KaZaA specializes in sharing music files that are non-scalable coded, it does not seriously explore the property of scalable coding and how scalable coding affects the set of supplying peers and the peer assignment problem. Our work demonstrates that exploring scalable coding

results in a larger set of supplying peers. By carefully designing peer assignment, we can improve content delivery in terms of both transmission time and content quality.

Multimedia streaming on P2P networks has been a relatively new topic, and most of the research efforts are focusing on live streaming. CoopNet [14, 15] proposed a framework to distribute live media content to a potentially large and highly dynamic population of hosts and to provide resilience to peer transience by redundancy in both network paths and data. To improve robustness in distributing live media on overlay networks, SplitStream [7] addressed how to evenly distribute load among participating peers. In terms of streaming architectures, NICE [6] and ZIGZAG [20, 21] proposed to organize peers into hierarchies when constructing application-level multicast for media streaming, and Jeon [10] defined cache lookup service, media scheduling and prefetching schemes for a streaming cache service. For on-demand streaming, Xu [23] addressed how to reduce startup latency and amplify system capacity under certain assumptions. Cui [8, 9] exploited the buffer capacity at peer nodes to reduce the load on streaming servers when the user requests are asynchronous and peers' bandwidth is heterogeneous. Our previous work [19] studies how to minimize transmission time of scalable coded images, but it does not consider delay constraints.

In summary, previous research work has focused on various other aspects of media streaming on P2P networks, and very few of them exploited the property of coded bitstreams.

7 Conclusions and Future Work

In this paper, we studied how to maximize the quality of delivering fine-scalable coded content on P2P networks. We formulated this problem as constrained optimization problems, from which we derived an optimal peer assignment algorithm. Although we have focused on image transmission here, the proposed algorithm can be easily adapted to streaming

motion SPIHT (MSPIHT) type of videos, where each frame is scalable coded independent of other frames. In this case, transmission of MSPIHT involves delivering a sequence of scalable images with similar sizes and delay bounds, so the proposed algorithms can be applied frame by frame. For SPIHT videos that are coded using motion estimation and motion compensation, we still need to study how to cope with different scalable structure in motion compensated frames.

In the future, we plan to extend our research in the following directions. First, we will design algorithms to incorporate the bandwidth constraint on the requesting peer. Second, we plan to study how peer transience affects peer assignment algorithms, and then develop some strategies to dynamically adapt to such network changes as peer departure (or crash), bandwidth fluctuation, and link failures. Third, we will develop algorithms for transmission of coarse-scalable coded content. And last, we will study how to extend the algorithms to motion compensated videos.

A Proof of Theorem on Canonical Peer Assignment

THEOREM 1 *If there exists a non-canonical PAM (PAS$_1$, PAS$_2$, ..., PAS$_n$), which results in the optimal image quality, then a canonical PAM' (PAS'$_1$, PAS'$_2$, ..., PAS'$_n$) can always be found that also results in the optimal image quality.*

PROOF: Let we represent PALV from the optimal non-canonical PAM (PAS$_1$, PAS$_2$, ..., PAS$_n$), as $\{\Delta_i, i = 1, 2, \ldots, n\}$, where $\Delta_i = \sum_{k=1}^{u_i+1}(-1)^k x_{i,k}$. We can construct a set of image segment sets (SS$_1$, SS$_2$, ..., and SS$_n$) from PALV in the following way: SS$_1$ = $\{(0, \Delta_1)\}$, SS$_2$ = $\{(\Delta_1, \Delta_1 + \Delta_2)\}$, ... SS$_i$ = $\{(\sum_{k=1}^{i-1}\Delta_k, \sum_{k=1}^{i}\Delta_k)\}$. To show that SS$_i$ is a valid Peer Assignment Set of peer i (we name it as PAS'$_i$), we need to verify that the image segment in SS$_i$ resides in the image boundary of peer i. By definition of PAS$_1$, we

have $x_{1,u_1+1} \leq s_1$, and also

$$\begin{aligned} x_{1,u_1+1} &= x_{1,1} + x_{1,2} - x_{1,1} + x_{1,3} - x_{1,2} + x_{1,4} - x_{1,3} + \ldots + x_{1,u_1+1} - x_{1,u_1} \\ &= (x_{1,2} - x_{1,1} + x_{1,4} - x_{1,3} + \ldots x_{1,u_1+1} - x_{1,u_1}) \\ &\quad +(x_{1,1} - 0) + (x_{1,3} - x_{1,2}) + (x_{1,5} - x_{1,4}) + \ldots (x_{1,u_1} - x_{1,u_1-1}) \\ &= \Delta_1 + \beta_1 + \beta_2 + \ldots + \beta_{\frac{u_1-1}{2}} \end{aligned}$$

where β_j ($\beta_j > 0$, for $j = 1, 2, \ldots, \frac{u_1-1}{2}$) denotes the length of segments assigned to peers $2, 3, \ldots, n$, and there are $\frac{u_1-1}{2}$ such segments. Therefore, we have $\Delta_1 < x_{1,u_1+1} \leq s_1$. Clearly, the image segment in SS_1 is within the image boundary of peer 1. Similarly, for peer i ($i > 1$) we have

$$x_{i,u_i+1} = \sum_{k=1}^{i} \Delta_k + \sum_{l=1}^{m} \alpha_l$$

where α_l (> 0) denotes the length of image segments assigned to peers k (with $k > i$), and m is the number of such continuous image segments. Again, we have $\sum_{k=1}^{i} \Delta_k < x_{i,u_i+1} \leq s_i$, so the image segment in SS_i is within the image boundary of peer i.

So far, we have proved that SS_i is a valid Peer Assignment Set of peer i (PAS'$_i$). It is also easy to see that PAS'$_1$, PAS'$_2$, ..., PAS'$_n$ constitute a canonical PAM', because it satisfies both cardinality and ordering conditions. This canonical PAM' results in the same quality as the non-canonical PAM, both transmitting an image of size $\sum_{i=1}^{n} \Delta_i$. Therefore, we have constructed a PAM' that also results in the optimal image quality. ■

B Proof of Theorem on Distance Bound

THEOREM 2 *The difference between the transmission time of the rounded solution derived by the algorithm in Figure 7 and that of the optimal integer solution is upper bounded*

by $\frac{1}{\min\{b_1,b_2,\ldots,b_n\}}$.

PROOF: Let t_{cont} and t_{int} be the minimum transmission time for continuous and integer versions of peer assignment, respectively. Because the search space for integer optimal peer assignment is a subspace of that of the continuous peer assignment, the optimal continuous solution will be at least as good as the optimal integer solution. Therefore, we have the following relationship: $t_{cont} \leq t_{int}$.

Denote t_{round} as the transmission time obtained by our rounding algorithm, then we have $t_{round} - t_{int} \leq t_{round} - t_{cont}$.

Let $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\}$ be an optimal continuous solution and $\{x_1 = \hat{x}_1 + \beta_1, x_2 = \hat{x}_2 + \beta_2, \ldots, x_n = \hat{x}_n + \beta_n\}$ be its rounded integer solution obtained by the algorithm in Figure 7. We first derive the bounds for β_i $(i = 1, 2, \ldots, n)$. From Line 4 in Figure 7, we know that if $f_i < \frac{b_{i+1}}{b_i+b_{i+1}}$, x_i is equal to $\lfloor \hat{x}_i \rfloor$, resulting in $\beta_i = -f_i$, therefore, $\beta_i > -\frac{b_{i+1}}{b_i+b_{i+1}}$. Similarly, if $f_i \geq \frac{b_{i+1}}{b_i+b_{i+1}}$, x_i is equal to $\lceil \hat{x}_i \rceil$, resulting in $\beta_i = 1 - f_i$, therefore, $\beta_i \leq \frac{b_i}{b_i+b_{i+1}}$. So we have $\beta_i \in (-\frac{b_{i+1}}{b_i+b_{i+1}}, \frac{b_i}{b_i+b_{i+1}}]$ for $i = 1, 2, \ldots, n-1$. From Line 9 in Figure 7, we have $x_n = \hat{x}_n$, so $\beta_n = 0$. On the boundary, we also set $x_0 = \hat{x}_0 = \beta_0 = 0$.

Having obtained the range of β_i, we can derive the distance between the rounded solution

and the optimal solution as follows.

$$
\begin{aligned}
& t_{round} - t_{int} \leq t_{round} - t_{cont} \\
\leq \quad & \max \quad \frac{(\hat{x}_i + \beta_i) - (\hat{x}_{i-1} + \beta_{i-1})}{b_i} - \frac{\hat{x}_i - \hat{x}_{i-1}}{b_i}, i = 1, 2, \ldots, n \\
= \quad & \max \quad \frac{\beta_i - \beta_{i-1}}{b_i}, i = 1, 2, \ldots, n \\
\leq \quad & \max \quad \left(\frac{b_i}{b_i + b_{i+1}} + \frac{b_i}{b_{i-1} + b_i} \right) \frac{1}{b_i}, i = 1, 2, \ldots, n \\
\leq \quad & \frac{1}{\min\{b_1, b_2, \ldots, b_n\}}
\end{aligned}
$$

■

References

[1] FastTrack. http://www.fasttrack.nu.

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

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

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

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

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

[7] 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.

[8] Y. Cui, B. Li, and K. Nahrstedt. oStream: asynchronous streaming multicast in application-layer overlay networks. *IEEE journal on Selected Areas in Communications*, January 2004 (accepted to appear).

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

[10] W. J. Jeon and K. Nahrstedt. Peer-to-peer multimedia streaming and caching service. In *Proc. of IEEE International Conference on Multimedia and Expo*, August 2002.

[11] 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.

[12] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, 2nd edition, 2003.

[13] International Standards Organization. JPEG 2000 image coding system. Final Committee Draft of ISO International Standard 15444 Part 1.

[14] 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.

[15] 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.

[16] 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.

[17] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445-3462, December 1993.

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

[19] X. Su and R. Fatoohi. Scalable coded image transmissions over peer-to-peer networks. In *Proc. Int'l Conf. on Multimedia and Expo*, pages 493–496, July 2003.

[20] 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.

[21] 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*, 2003 (accepted to appear).

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

[23] 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.