Efficient Peer Assignment for Low-Latency Transmission of Scalable Coded Images

Xiao Su and Tao Wang

Abstract: **In this paper, we propose efficient peer assignment algorithms for low-latency transmission of scalable coded images in peer-to-peer networks, in which peers may dynamically join and leave the networks. The objective of our algorithm is to minimize the transmission time of a requested image that is scalable coded. When an image is scalable coded in different bit rates, the bit stream encoded in a lower bit rate is a prefix subset of the one encoded in a higher bit rate. Therefore, a peer with the same requested image coded in *any* bit rate, even when it is different from the requested rate, may work as a supplying peer. As a result, when a scalable coded image is requested, more supplying peers can be found in peer-to-peer networks to help with the transfer. However, the set of supplying peers is not static during transmission, as the peers in this set may leave the network or finish their transmission at different times. The proposed peer assignment algorithms have taken into account the above constraints.**

In this paper, we first prove the existence of an optimal peer assignment solution for a simple identity permutation function, and then formulate peer assignment with this identity permutation as a mixed-integer programming problem. Next, we discuss how to address the problem of dynamic peer departures during image transmission. Finally, we carry out experiments to evaluate the performance of proposed peer assignment algorithms.

Index Terms: **Image transmission, peer-to-peer networks, scalable coding.**

I. INTRODUCTION

The peer-to-peer 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 [1]. The delivery of multimedia content, such as audio, speech, image, and video, is largely dependent 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 coding algorithms into two categories: Scalable coding [2], [3], [4] 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 $r_1 < r_2$. When an image is coded using a non-scalable coding scheme using two different bit rates, r_1 and r_2, the coding generates C_1 and C_2 as two entirely different strings. Traditional coding algorithms, such as JPEG [5] and H.263 [6] generate non-scalable coded bit streams. On the other hand, if an image is scalable coded in the same two different bit rates, r_1 and r_2, the generated bit stream C_1 is a prefix subset of C_2. Example coding algorithms that generate scalable coded bit streams include MPEG-2 [7] that produces layered bit streams normally consisting of a base layer and one or more enhancement layers and SPIHT [2] that generates fine-scalable coded bit streams whose quality increases with every additional bit. As fine-scalable coded bit streams offer better quality adaptation compared to layered bit streams, in this paper we will focus on transmission of fine-scalable coded bit streams generated by SPIHT.

Let us analyze how scalable coding impacts image transmission in peer-to-peer networks. When a peer requests for an image that is non-scalable coded in bit rate r_1, then only the peers that hold the same image in the same bit rate r_1 may work as supplying peers. Peers that hold the same image coded in different bit rates from r_1 cannot supply the image because their coded bit streams are completely different from C_1. In contrast, if the requested image is scalable coded in bit rate r_1, then the peers who have the *same* image scalable coded in any bit rate, r_2, regardless if r_2 is equal to r_1, can supply the image, as the bit stream will be either a subset (when $r_2 < r_1$) or a superset (when $r_2 > r_1$) of the requested image. Therefore, scalable coding makes more peers available to supply the requested image, and may potentially reduce the latency of image transmission.

To efficiently allocate and schedule image transmission from a set of supplying peers, we need to take into account the dynamic nature of peers in the peer-to-peer networks. First, as these peers may hold coded bit streams that are different in size, they may finish their transmission in different times. Second, peers may connect to the Internet through different access technologies, so they may have different outgoing bandwidth. Last, the peers may dynamically leave the network without finishing their transmission. Given this dynamic set of supplying peers, it is very important to investigate how to divide a scalable coded image into image segments and how to assign these segments to the supplying peers in order to optimize the overall quality of service (QoS) for a requesting peer.

The set of QoS parameters to be optimized depends on the types of applications. For *play-after-downloading* type of applications, file downloading time is the most important QoS parameter. For *play-while-downloading* type of applications, startup latency, playback jitter, and playback quality constitute the set

Manuscript received January 21, 2005; approved for publication by Song Chong, Division III Editor, January 17, 2008.

X. Su is with Computer Engineering Department, San José State University, San José, USA, email: xsu@email.sjsu.edu.

T. Wang is with Synopsys Inc. Mountain View, USA, email: tao.wang@synopsys.com.

of important QoS parameters. Here we focus on *play-after-downloading* type of applications.

To summarize, in this paper we study the peer assignment problem in peer-to-peer networks with the objective to minimize image transmission time. We first prove the existence of an optimal solution for a simple identity permutation, and then formulate the peer assignment problem as a mixed-integer programming problem. Because directly finding an optimal solution to such a problem incurs high computational cost, we develop an approximate rounding algorithm to find a sub-optimal solution. We then discuss how to address the dynamic departure of peers in the peer assignment.

The paper is organized as follows. We discuss related work in Section II, and present our peer-to-peer network model in Section III. We then define the peer assignment problem in Section IV. After proving the existence of an optimal solution for a simple identity permutation function in Section V, we formulate the peer assignment problem as a mixed-integer programming problem in Section VI. Finally we evaluate the proposed algorithms in Section VII. Section VIII concludes the paper by identifying future research directions.

II. RELATED WORK

Popular peer-to-peer file sharing systems, such as Napster [8], Gnutella [9], KaZaA [10], eDonkey [11], and BitTorrent [12], treat media files as regular data files, and they do not exploit the property of coded bitstreams. Peers are qualified to supply their images only when they have the exactly same coded bitstream as requested. This limits the set of supplying peers to be small and slows down image transmission.

There have been some research efforts to address various issues for live streaming on peer-to-peer networks. For example, CoopNet [13], [14] and SplitStream [15] proposed system design and framework to distribute media content on peer-to-peer networks. NICE [16] and ZIGZAG [17], [18] proposed algorithms and protocols to construct scalable application-level multicast for media streaming.

There have been related projects for on-demand streaming in peer-to-peer networks as well, but they have been studied under different assumptions. Optimal media assignment algorithm (OTS_{p2p}) [19] is designed to minimize the initial buffering delay for *play-while-downloading* applications, such as video on demand. It divides media files into *equal-sized* segments and assumes constant-bit-rate coding and transmission. Our proposed peer assignment schemes complement the above work in two ways. First, we target a different type of application with a different design objective. We focus on *play-after-downloading* applications with the objective to minimize the latency of image transmission, while OTS_{p2p} is designed for *play-while-downloading* applications with the objective to reduce the initial buffering delay for video streaming. Second, our algorithms exploit the property of the scalable coding algorithms and the structure of coded bit streams, while OTS_{p2p} does not exploit the property of coded bit streams. Another work, PALS [20], studied buffer management and receiver controlled adaptation to bandwidth availability from multiple peers. Our work differs from PALS in two aspects. First, the set of supplying peers in PALS is limited to the ones who have the same video layers as requested, therefore, it does not fully exploit the bandwidth resource from the peers that have different representations of the same file. Second, our work optimizes image transmission time while PALS studied heuristic algorithms for media streaming. Cui [30], [21] exploited the buffer capacity at peer nodes to reduce the load on streaming servers when the user requests are asynchronous and the peers' bandwidths are heterogeneous. In our previous work [23], we proposed a peer assignment problem and solved it using a heuristic algorithm. In this paper, we will systematically design an optimal algorithm for image transmission on peer-to-peer networks.

III. PEER-TO-PEER NETWORK MODEL

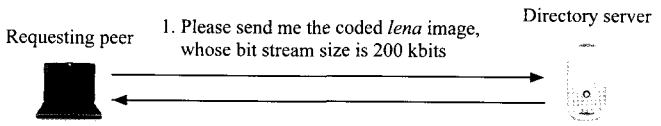
In our peer-to-peer system, transmission of scalable coded images is done in three steps. First, a requesting peer employs a certain directory lookup algorithm to locate a potential set of supplying peers for a requested image. Second, the requesting peer applies a peer assignment algorithm to allocate image segments to different supplying peers with the objective to minimize the overall image transmission time. Third, the supplying peers are informed about their own allocations by the requesting peer and then start transmission. Fourth, when some peers leave the network before finishing their transmission, the requesting peer re-allocates the unfinished image segments to the remaining peers.

In this paper, we make the following assumptions on the peer-to-peer systems with regard to scalable image transmission.

1. *Directory lookup.* There is a directory lookup server (centralized or distributed) to return a complete list of potential supplying peers. For each peer, it maintains the size of the image bit stream (*i.e.*, the coding rate) that a peer has and the peer's contribution of bandwidth.
2. *Supplying bandwidth.* Every peer in the system has a fair estimation of its outgoing bandwidth and timely updates the bandwidth with the directory lookup server.
3. *Relationship in peers' bandwidth.* We consider a single requesting peer with incoming bandwidth B and multiple supplying peers with heterogeneous bandwidth $b_1, b_2, \cdots, b_n$, 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.*, $\sum_{i=1}^{n} b_i \leq B$. This can be easily achieved by dropping some supplying peers if the above condition is violated.
4. *Reliable delivery.* We assume images are transmitted using reliable transport protocols, such as TCP. In this case, the bandwidth refers to effective bandwidth observed by peers using these protocols.
5. *Peer transience.* Peers may leave peer-to-peer networks at any time, even before they finish their assigned transmission tasks.

IV. 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 $(i = 1, 2, \cdots, n)$, coded in different bit rates



	Bit stream size (kbits)	Bandwidth (kbps)
Peer 1	60	20
Peer 2	120	40
Peer 3	200	10
Peer 4	210	30

Fig. 1. Configuration of an example peer-to-peer system.

and with outgoing bandwidth, b_i $(i = 1, 2, \cdots, n)$. Without loss of generality, we assume $s_1 \leq s_2 \leq \cdots \leq s_n$, otherwise we can re-number the peers to follow this order. The requesting peer asks for an image of size M, which is less than or equal to the maximum image size, $\max\{s_1, s_2, \cdots, s_n\}$, otherwise the request cannot be satisfied. Given the above notations, let us define the following two concepts.

Definition 1 *Image allocation vector is defined as a partition of the requested image:* $\{x_0, x_1, x_2, \cdots, x_n\}$ *with* $x_0 = 0$ *and* $x_n = M$, *so that the portion between* $(x_{j_i-1}, x_{j_i}]$ *is assigned to peer* i, *where* $j_i \in \{1, 2, \cdots, n\}$.

Here the mapping from the index of peer i to the index of image allocation vector j_i is one-to-one correspondent, so vector $\{j_1, j_2, \cdots, j_n\}$ is a permutation of vector $\{1, 2, \cdots, n\}$. We denote this permutation as $j_i = \pi(i)$ and its inverse as $i = \pi^{-1}(j_i)$.

Definition 2 *Peer assignment vector* $\{\Delta_i, i = 1, 2, \cdots, n\}$ *is the vector in which the* i*th element,* $\Delta_i = x_{j_i} - x_{j_i-1}$, *defines the size of the image segment assigned to peer* i.

Given permutation $\pi(i)$, there is a one-to-one correspondence between an image allocation vector and a peer assignment vector: $x_i = \sum_{k=1}^{i} \Delta_{\pi^{-1}(k)}$ and $\Delta_i = x_{j_i} - x_{j_i-1}$ for $i = 1, 2, \cdots, n$.

Based on the above definitions, image transmission time (also called downloading time) is calculated as the maximum of $\{\Delta_i/b_i, i = 1, 2, \cdots, n\}$. The goal of an peer assignment algorithm is to find a peer assignment vector (or an image allocation vector) and its permutation function π.

Let us study a small example to understand the above notations and how different peer assignment solutions affect image transmission time. Fig. 1 shows the image transmission request in a peer-to-peer system. The requesting peer contacts the directory server to find out who have the *lena* image of size 200 kbits. The directory server responds with four possible supplying peers, p_1, p_2, p_3, and p_4. The four peers hold coded images of size 60 kbits, 120 kbits, 200 kbits, and 210 kbits, and supply them using bandwidth 20 kbps, 40 kbps, 10 kbps, and 30 kbps, respectively. Fig. 2 compares two peer assignment solutions.

In solution one (bottom left of Fig. 2), p_1 is assigned to transmit the image bit stream between (40, 60] kbits, p_2 to transmit between (0, 40] kbits, p_3 to transmit between (60, 160] kbits, and p_4 to transmit between (160, 200] kbits. This transmission

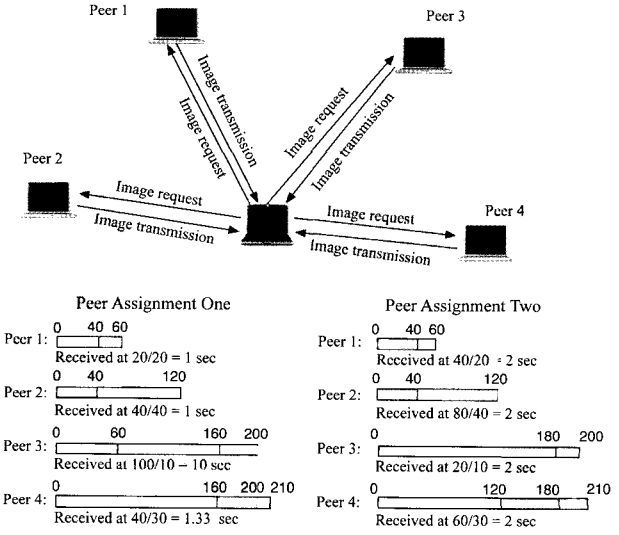


Fig. 2. Comparisons of two peer assignment methods for a scalable coded image.

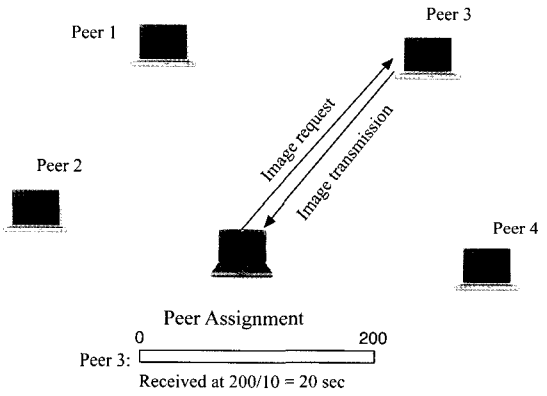


Fig. 3. Image transmission of a non-scalable coded image.

method results in an image allocation vector of $\{x_0 = 0, x_1 = 40, x_2 = 60, x_3 = 160, x_4 = 200\}$ and a peer assignment vector of $\{\Delta_1 = x_2 - x_1 = 20, \Delta_2 = x_1 - x_0 = 40, \Delta_3 = x_3 - x_2 = 100, \Delta_4 = x_4 - x_3 = 40\}$. Its permutation function is $\pi(1) = 2$, $\pi(2) = 1$, and $\pi(3) = 3$, and $\pi(4) = 4$. As a result, the downloading time is equal to $\max\left\{\frac{20}{20}, \frac{40}{40}, \frac{100}{10}, \frac{40}{30}\right\} = 10$ seconds.

In solution two (bottom right of Fig. 2), p_1 is assigned to transmit between (0,40] kbits, p_2 to transmit between (40,120] kbits, p_3 to transmit between (180,200] kbits, and p_4 to transmit between (120,180] kbits. This corresponds to an image allocation vector of $\{x_0 = 0, x_1 = 40, x_2 = 120, x_3 = 180, x_4 = 200\}$ and a peer assignment vector of $\{\Delta_1 = x_1 - x_0 = 40, \Delta_2 = x_2 - x_1 = 80, \Delta_3 = x_4 - x_3 = 20, \Delta_4 = x_3 - x_2 = 60\}$. The resulting permutation function is $\pi(1) = 1$, $\pi(2) = 2$, and $\pi(3) = 4$, and $\pi(4) = 3$. In this case, the downloading time is equal to $\max\left\{\frac{40}{20}, \frac{80}{40}, \frac{20}{10}, \frac{60}{30}\right\} = 2$ seconds.

Let us now suppose the requested image is *not* scalable coded. In this case, there exists only *one* supplying peer: p_3, as the bit streams held by p_1, p_2, and p_4 are totally different from the requested one with size 200 kbits. As shown in Fig. 3, the time needed to transmit the image is $200/10 = 20$ seconds.

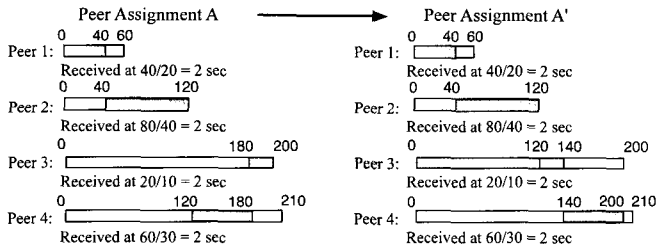


Fig. 4. Conversion from assignment $\mathcal{A}$ to $\mathcal{A}'$.

The above example illustrates two points. First, employing scalable coding results in a larger set of supplying peers compared to non-scalable coding. Therefore, the image transmission time is reduced significantly, from 20 to 2 seconds. Second, when requesting scalable coded images, peer assignment algorithms have big impact the latency of image transmission. In the above example, solution two results in a much shorter end-to-end transmission time than solution one.

V. OPTIMALITY OF PEER ORDERING

Given a scalable image transmission request, optimal solutions to peer assignment may not be unique. In Fig. 4, both solution $\mathcal{A}$ (also Peer Assignment Two in Fig. 2) and solution $\mathcal{A}'$ are optimal with the minimum transmission time. In both $\mathcal{A}$ and $\mathcal{A}'$, the four supplying peers start and finish simultaneously. Therefore, the aggregate bandwidth has been maximally utilized during transmission, resulting in the minimum transmission time.

In this example, solutions $\mathcal{A}$ and $\mathcal{A}'$ have the same peer assignment vector $\{\Delta_1 = 40, \Delta_2 = 80, \Delta_3 = 20, \Delta_4 = 60\}$ but different permutation functions: $\mathcal{A}$ has $\pi(1) = 1$, $\pi(2) = 2$, $\pi(3) = 4$, and $\pi(4) = 3$. while $\mathcal{A}'$ has $\pi(1) = 1$, $\pi(2) = 2$, $\pi(3) = 3$, and $\pi(4) = 4$. We can see that different permutation functions result in different peer assignment solutions with the same transmission time.

Questions naturally arise as to how to choose a permutation function and whether the chosen permutation function can lead to an optimal peer assignment solution. The following theorem states that the optimal peer assignment solution exists for an identity permutation function, $\pi_i(i) = i$. The permutation function, $\pi_i(i) = i$, implies that we allocate image portions in the increasing order of the image sizes of peers, *i.e.,* allocating the image between $(0, \Delta_1]$ to peer 1 that has the smallest image size, and the image between $(\Delta_1, \Delta_2]$ to peer 2 that has the second smallest image size, and so on.

Intuitively, if an optimal peer assignment is found for a non-identity permutation, we can always perform pairwise swap of the peer assignment until we have an identity permutation. For example, from assignment $\mathcal{A}$, we make a swap of Δ_3 and Δ_4 to get the assignment $\mathcal{A}'$: $\Delta_3' = \Delta_4 = 20$ kbits and $\Delta_4' = \Delta_3 = 60$ kbits, and this results in an identity permutation. In addition, we need to verify that the new peer assignment vectors fall within the peers' image boundaries and can be supplied by those peers. The following theorem formalizes the above analysis to the general case, when multiple pairwise swaps are needed.

Theorem 1 *Given the identity permutation function $\pi_i(i) = i$, there always exists an optimal peer assignment solution with the minimum transmission time.*

Proof: Suppose there exists an optimal peer assignment solution $\mathcal{A}$ with permutation function $\pi_{k_0}(i) = i'$. If $i' = i$ for every $i = 1, 2, \cdots, n$, then we are done. Otherwise, at least one i' is not equal to i, and we show that there exists an alternative optimal peer assignment solution $\mathcal{A}'$ with the identity permutation function $\pi_i(i) = i$.

In solution $\mathcal{A}$, the first image portion $(0, \Delta_{\pi_{k_0}^{-1}(1)}]$ is transmitted by peer $\pi_{k_0}^{-1}(1)$ (with image size $s_{\pi_{k_0}^{-1}(1)}$), the second image portion $(\Delta_{\pi_{k_0}^{-1}(1)}, \Delta_{\pi_{k_0}^{-1}(2)}]$ is assigned to peer $\pi_{k_0}^{-1}(2)$, and i^{th} image portion $(\Delta_{\pi_{k_0}^{-1}(i-1)}, \Delta_{\pi_{k_0}^{-1}(i)}]$ is assigned to peer $\pi_{k_0}^{-1}(i)$, for $i = 1, 2, \cdots, n$. All the assignments need to satisfy the following constraints:

$$\begin{aligned} \Delta_{\pi_{k_0}^{-1}(1)} &\le s_{\pi_{k_0}^{-1}(1)}, \\ &\cdots \\ \Delta_{\pi_{k_0}^{-1}(1)} + \cdots + \Delta_{\pi_{k_0}^{-1}(i-1)} + \Delta_{\pi_{k_0}^{-1}(i)} &\le s_{\pi_{k_0}^{-1}(i)}, \\ &\cdots \\ \Delta_{\pi_{k_0}^{-1}(1)} + \cdots + \Delta_{\pi_{k_0}^{-1}(n-1)} + \Delta_{\pi_{k_0}^{-1}(n)} &\le s_{\pi_{k_0}^{-1}(n)}. \end{aligned} \quad (1)$$

We construct solution $\mathcal{A}'$ from $\mathcal{A}$ through a sequence of intermediate permutation functions, π_{k_j} for $j = 1, 2, \cdots, m$, so that the last permutation function satisfies the following condition: $\pi_{k_m}^{-1}(1) < \pi_{k_m}^{-1}(2) < \cdots < \pi_{k_m}^{-1}(n-1) < \pi_{k_m}^{-1}(n)$. Here m is the number of steps. As $\pi_{k_m}^{-1}(.)$'s are the indices of peers, they are essentially a permutation of $\{1, 2, \cdots, n\}$, so we have $\pi_{k_m}^{-1}(i) = i$. Therefore, $\pi_{k_m}(\cdot)$ is the identity permutation function.

After step j, if the two indices of peers satisfy $\pi_{k_j}^{-1}(i) > \pi_{k_j}^{-1}(i+1)$, *i.e.,* the image portion i is assigned to peer $\pi_{k_j}^{-1}(i)$ and image portion $i+1$ is assigned to peer $\pi_{k_j}^{-1}(i+1)$, then in step $j+1$ we define a new permutation function $\pi_{k_{j+1}}(\cdot)$ by swapping the peer assignment so that image portion i is assigned to peer $\pi_{k_j}^{-1}(i+1)$ and image portion $i+1$ is assigned to peer $\pi_{k_j}^{-1}(i)$. Therefore, the inverse of the new permutation function $\pi_{k_{j+1}}(\cdot)$ satisfies the following condition.

$$\pi_{k_{j+1}}^{-1}(l) = \begin{cases} \pi_{k_j}^{-1}(l+1), & l = i, \\ \pi_{k_j}^{-1}(l-1), & l = i+1, \\ \pi_{k_j}^{-1}(l), & l \neq i, i+1. \end{cases}$$

We need to ensure that conditions in (1) are satisfied in every step. If after step j, we have

$$\Delta_{\pi_{k_j}^{-1}(1)} + \cdots + \Delta_{\pi_{k_j}^{-1}(i-1)} + \Delta_{\pi_{k_j}^{-1}(i)} \le s_{\pi_{k_j}^{-1}(i)},$$

$$\Delta_{\pi_{k_j}^{-1}(1)} + \cdots + \Delta_{\pi_{k_j}^{-1}(i-1)} + \Delta_{\pi_{k_j}^{-1}(i)} + \Delta_{\pi_{k_j}^{-1}(i+1)} \le s_{\pi_{k_j}^{-1}(i+1)}.$$

Then after pairwise swapping in step $j+1$, we have

$$\begin{aligned} &\Delta_{\pi_{k_{j+1}}^{-1}(1)} + \cdots + \Delta_{\pi_{k_{j+1}}^{-1}(i-1)} + \Delta_{\pi_{k_{j+1}}^{-1}(i)} \\ &= \Delta_{\pi_{k_j}^{-1}(1)} + \cdots + \Delta_{\pi_{k_j}^{-1}(i-1)} + \Delta_{\pi_{k_j}^{-1}(i+1)} \\ &\le s_{\pi_{k_j}^{-1}(i+1)} = s_{\pi_{k_{j+1}}^{-1}(i)}, \end{aligned}$$

$$\Delta_{\pi^{-1}_{k_{j+1}}(1)} + \cdots + \Delta_{\pi^{-1}_{k_{j+1}}(i-1)} + \Delta_{\pi^{-1}_{k_{j+1}}(i)} + \Delta_{\pi^{-1}_{k_{j+1}}(i+1)}$$
$$= \Delta_{\pi^{-1}_{k_j}(1)} + \cdots + \Delta_{\pi^{-1}_{k_j}(i-1)} + \Delta_{\pi^{-1}_{k_j}(i+1)} + \Delta_{\pi^{-1}_{k_j}(i)}$$
$$\leq s_{\pi^{-1}_{k_j}(i+1)} \leq s_{\pi^{-1}_{k_j}(i)} = s_{\pi^{-1}_{k_{j+1}}(i+1)}.$$

We see that conditions are still satisfied after step $j+1$. Therefore, after m steps of such pairwise swaps, we reach a peer assignment solution with the identity permutation function $\pi_i(i) = i$.

Both solutions $\mathcal{A}$ and $\mathcal{A}'$ have the same transmission time, which is equal to $\max\{\frac{\Delta_i}{b_i}, i = 1, 2, \cdots, n\}$, since in constructing $\mathcal{A}'$ we do not alter the values of $\Delta_i, i = 1, 2, \cdots, n$. □

To illustrate how to construct solution $\mathcal{A}'$ from $\mathcal{A}$, let us revisit the example in Fig. 4, which illustrates the conversion of $\mathcal{A}$ to $\mathcal{A}'$. In $\mathcal{A}$, the image allocation vector is $\{x_0 = 0, x_1 = 40, x_2 = 120, x_3 = 180, x_4 = 200\}$ and the permutation function is $\pi(1) = 1$, $\pi(2) = 2$, $\pi(3) = 4$, and $\pi(4) = 3$. After conversion, $\mathcal{A}'$ has the image allocation vector $\{x_0 = 0, x_1 = 40, x_2 = 120, x_3 = 140, x_4 = 200\}$ and permutation function $\pi(1) = 1, \pi(2) = 2, \pi(3) = 3$, and $\pi(4) = 4$.

VI. PEER ASSIGNMENT ALGORITHMS

In Section V, we have shown that there always exists an optimal peer assignment solution using identity permutation function, $\pi(i) = i$. In this section, we will discuss how to drive such an optimal peer assignment algorithm, and how to address dynamic peer departures in peer assignment. We will also discuss the issues and challenges when extending this algorithm to media streaming applications.

A. Optimal Peer Assignment Algorithms

Using an identity permutation function means that peer i is assigned to transmit the image portion between $(x_{i-1}, x_i]$. For this peer assignment to be feasible, the assigned image portion has to reside within the image boundary of peer i, where $x_i = \sum_{k=1}^{i} \Delta_k$ and $\Delta_i = x_i - x_{i-1}$. Considering this constraint, we can formulate the peer assignment problem as an optimization problem.

$$\begin{aligned} \min_{\{\Delta_1, \Delta_2, \cdots, \Delta_n\}} \quad & \max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \cdots, \frac{\Delta_n}{b_n}\right\} \\ \text{subject to} \quad & \Delta_1 \leq s_1, \\ & \Delta_1 + \Delta_2 \leq s_2, \\ & \cdots \\ & \Delta_1 + \Delta_2 + \cdots + \Delta_n \leq s_n, \\ & \Delta_1 + \Delta_2 + \cdots + \Delta_n = M, \\ & \Delta_i \in \{0\} \bigcup Z^+, \ i = 1, 2, \cdots, n. \end{aligned} \tag{2}$$

In this formulation, we try to minimize the downloading time ($\max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \cdots, \frac{\Delta_n}{b_n}\right\}$) while satisfying the constraints that the allocated image portion lies within image boundary for every peer. The last equality constraint specifies that the requested image should be satisfied by the set of supplying peers. Every variable, Δ_i, is a non-negative integer.

This formulation has linear constraints but nonlinear objective as max is a nonlinear function. By introducing a new variable $y = \max\left\{\frac{\Delta_1}{b_1}, \frac{\Delta_2}{b_2}, \cdots, \frac{\Delta_n}{b_n}\right\}$, we can convert (2) into a mixed-integer linear programming problem:

$$\begin{aligned} \min_{\{\Delta_1, \Delta_2, \cdots, \Delta_n, y\}} \quad & y \\ \text{subject to} \quad & \Delta_1 \leq s_1, \\ & \Delta_1 + \Delta_2 \leq s_2, \\ & \cdots \\ & \Delta_1 + \Delta_2 + \cdots + \Delta_n \leq s_n, \\ & \Delta_1 + \Delta_2 + \cdots + \Delta_n = M, \\ & \Delta_i / b_i \leq y, \ i = 1, 2, \cdots, n, \\ & y \in R^+, \\ & \Delta_i \in \{0\} \bigcup Z^+, \ i = 1, 2, \cdots, n. \end{aligned} \tag{3}$$

Existing techniques to solve mixed-integer linear programming problems include branch and bound, cutting planes, clustering methods, and stochastic algorithms [24], [25]. However, these methods are computational expensive with large number of variables, and sometimes can not guarantee to find an optimal solution, even a feasible solution. In our experiments of peer-to-peer image transmission, we also observed that it is difficult to directly solve the mixed-integer formulation.

Because of this inefficiency to directly solve such a mixed-integer problem, we propose to find a sub-optimal solution in two steps: (a) Find an optimal solution to (3), assuming each Δ_i to be a non-negative continuous variable. Note it is easy and efficient to find an optimal solution to a continuous linear programming problem [26]; (b) round this optimal continuous solution to its integer values. Two questions arise for this approach:

1. How to round an optimal continuous solution to its integer version?
2. How good is this approximate solution obtained by rounding?

To answer the first question, let us denote $\{\hat{\Delta}_1, \hat{\Delta}_2, \cdots, \hat{\Delta}_n\}$ as an optimal continuous peer assignment vector, $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n\}$ as its corresponding image allocation vector, and u as a basic rounding unit in terms of bits. Here, u is an integer whose value can be larger than or equal to one, depending on how images are scalable coded and transmitted using packets. For every $\hat{x}_i$, we can either round it to $\lfloor \hat{x}_i/u \rfloor u$ or $\lceil \hat{x}_i/u \rceil u$. The rounding criterion is to minimize the increase in the transmission time compared with the continuous solution. If we round $\hat{x}_i$ to $\lfloor \hat{x}_i/u \rfloor u$, then the increase in transmission time is the time for peer $i+1$ to transmit the portion between $\lfloor \hat{x}_i/u \rfloor u$ and $\hat{x}_i$. Similarly if we round $\hat{x}_i$ to $\lceil \hat{x}_i/u \rceil u$, then the increase in transmission time is the time for peer i to transmit the portion between $\hat{x}_i$ to $\lceil \hat{x}_i/u \rceil u$. Based on this analysis, we present our rounding algorithm in Fig. 5.

To answer the second question regarding the quality of this rounded solution, we have derived an upper bound on the distance between this rounded solution and the optimal integer solution, *i.e.*, $u/\min\{b_1, b_2, \cdots, b_n\}$, where u is the basic unit of rounding in terms of bits and b_is are outgoing bandwidths of supplying peers. As the bandwidth values, b_i ($i = 1, 2, \cdots, n$), are in the range of kbits per second, this upper bound is a very

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

Fig. 5. Rounding to integer solutions.

small fractional value. Therefore, we can conclude that the quality of the rounded solution is very close to that of the optimal integer solution. Proof of this bound can be found in [23].

B. Peer Assignment with Dynamic Peer Departures

The optimal peer assignment derived previously produces an optimal peer assignment vector, $\{\Delta_1, \Delta_2, \cdots, \Delta_n\}$, for the n supplying peers. If every peer i reliably finishes transmitting its assigned image segment, Δ_i, the requesting peer will receive the image bit stream with the minimum latency. However, peers in the network may leave the network at any time. When a supplying peer fails to complete its assigned segment, the requesting peer needs to allocate the missing segment to the remaining peers, who also have the missing segment.

If there exist more supplying peers with the missing image segment, the requesting peer will experience shorter image transmission latency. Let us compare two ways that a supplying peer transmits its assigned image segment and see how the transmission strategies affect the remaining set of supplying peers in case of peer departures. On the left of Fig. 6, peers transmit their assigned Δ_i's, from the start of the segments. After peer 2 transmits 70 kbits of its assigned segment, *i.e.*, between (40, 110] kbits, it leaves the network. In the remaining supplying peers, peer 3 and peer 4 still carry the missing segment between (110, 120] kbits, so they can help to fill the request. On the other hand, peers may start to transmit their assigned segments from the end, as shown on the right of Fig. 6. In this case, when peer 2 leaves the network after transmitting 70 kbits of the bit stream between (50, 120] kbits, it fails to supply the segment between (40, 50] kbits. In the network, all the remaining peers, 1, 2, and 4, carry the segment, and can work as supplying peers.

Obviously if peers adopt backward transmission method, they may help to reduce image transmission time in case of peer departures. This is because in our peer-to-peer network model, less peers carry the higher portions of the bit streams. When they transmit image segments backward from the end to the start, the unfinished segments tend to be located towards the lower end of the image bit stream. Therefore, it will likely result in a larger set of eligible supplying peers to finish the missing image segment.

To summarize, we need to extend our peer assignment algorithms in two ways to address peer transience in the network. First, as discussed above, supplying peers should employ backward transmission strategy to send their assigned segments. Second, image transmission request may not be completed in one round in case of peer departures. When a supplying peer leaves the network without finishing its transmission, the requesting

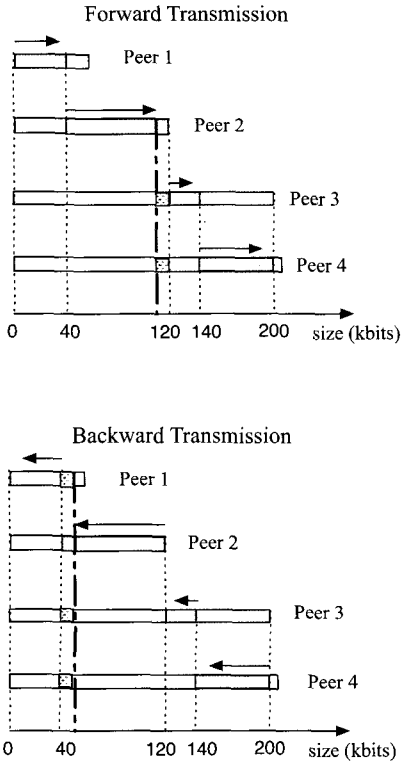


Fig. 6. Image transmission in case of peer departures.

peer will seek the missing segment from the remaining supplying peers, who carry the same segment. It treats this request similarly as the original request and uses the optimal peer assignment algorithm derived previously to allocate image transmission to this reduced set of supplying peers. This process repeats until the complete image is received.

C. Peer Assignment for Media Streaming

Image transmission is inherently a "play-after-downloading" type of application. For streaming applications, let us inspect how video coding algorithms affect peer assignment. We assume that the requesting peer can retrieve the information on video frame size and supplying bandwidth prior to frame playback.

In the first case, the video to be streamed is coded without motion estimation and compensation. In other words, every frame in the video is encoded independently of other frames, by exploiting spatial redundancy only. Motion-JPEG [27] may produce such video streams. Our proposed peer assignment algorithms can be easily applied to streaming this type of videos. The requesting peer can calculate the optimal peer assignment strategy for each frame and request the frame transmission accordingly. As the time needed to calculate optimal peer assignment is usually negligible, it does not introduce much overhead in real-time playback.

In the second case, the video to be streamed is coded using a hybrid transform codec with motion estimation and compensation, such as MPEG-2 [7], H.263 [6], and H.264 [28]. In the scalable extensions to MPEG-2 [7], MPEG-4 [29], and H.264 [22], three scalability modes are defined: Temporal scal-

Table 1. Comparison of two methods, M1 and M2, when the requested coded image size is equal to 16 kbytes.

#Peers	M1			M2		
	Solution	Time	Success ratio	Solution	Time	Success ratio
2	4.7383	0.00028	1.00	4.7386	0.00016	1.00
4	2.2070	0.00089	1.00	2.2076	0.00024	1.00
8	1.0374	0.00887	1.00	1.0382	0.00047	1.00
12	0.6789	0.06471	1.00	0.6800	0.00079	1.00
16	0.5058	0.37441	1.00	0.5073	0.00116	1.00
20	0.4044	1.61017	1.00	0.4059	0.00167	1.00
24	0.3331	7.89965	1.00	0.3351	0.00231	1.00

Table 2. Comparison of two methods, M1 and M2, when the requested coded image size is equal to 32 kbytes.

#Peers	M1			M2		
	Solution	Time	Success ratio	Solution	Time	Success ratio
2	13.06070	0.00024	1.00	13.06094	0.00018	1.00
4	6.93387	0.00083	1.00	6.93437	0.00026	1.00
8	6.64654	0.06298	0.91	6.64744	0.00046	1.00
12	6.89112	1.24115	0.74	6.89182	0.00079	1.00
16	3.55968	3.18436	0.64	3.56089	0.00120	1.00
20	3.88057	43.25981	0.55	3.88138	0.00167	1.00
24	4.74737	35.58403	0.31	4.74806	0.00243	1.00

ability, spatial scalability, and SNR scalability. These scalability modes generate a base layer and a number of enhancement layers. An entire enhancement layer needs to be received completely for quality improvement, and the reception of a partial enhancement layer does not bring any quality improvement. For peers with different enhancement layers, the peer assignment needs to be done in the granularity of layers, consisting of a block of bits. In this paper, we have focused on how to perform peer assignment when video quality improves with the reception of every additional bit, *i.e.,* for videos coded using fine granularity scalability (FGS). How to perform optimal assignment in the granularity of layers remains to be a challenging research topic and is beyond the scope of this paper.

In addition, MPEG-4 defined a FGS extension, in which the base layer uses non-scalable coding to generate its bit stream, and the enhancement layer codes the difference between the original picture and the reconstructed base layer picture using bit-plane coding of DCT coefficients. The bit stream of the FGS enhancement layer can be truncated into any number of bits *per picture*. In this case, our peer assignment algorithm can be directly applied to schedule transmissions of the enhancement layer on a picture-by-picture basis.

VII. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our proposed peer assignment algorithms. First, we compare the quality and computational cost of the two methods: (M1) Applying a mixed-integer programming solver package, *lp_solve* (version 4.0) [31], to find the optimal integer solution; (M2) finding a sub-optimal solution by first obtaining the optimal continuous solution (step (a) in Section VI) and then applying the rounding algorithm (step (b) in Section VI). The optimal continuous solution is obtained by the same package. The purpose of comparison is to evaluate the solution quality and computational time of M2 compared to M1. Second, we compare algorithm M2 with two simple heuristic peer assignment schemes.

We wrote our own simulations to conduct the experiments. The input parameters to the simulations include the number of peers, bandwidth of each peer, image size of each peer, whether a peer leaves early or not, and if so, when it leaves the system. We implemented M1, M2, and three heuristic schemes to perform peer assignment. We evaluated these peer assignment schemes based on image transmission time and computational overhead.

In our experiments, we set the range of bandwidth to be between 64 bytes/sec and 4 kbytes/sec, and consider the images of size 512×512 and $1,024 \times 1,024$ coded in either 0.5 bpp (bit per pixel) or 1 bpp. Therefore, for an image of 512×512, the size of a requested coded image is either 16 kbytes or 32 kbytes, and for an image of $1,024 \times 1,024$, the requested image is either 64 kbytes and 128 kbytes. Since the supplying peers can have images with sizes either less or greater than the requested image, we set their image sizes to be between $[4, 32]$ kbytes for 512×512 images and between $[16, 128]$ kbytes for $1,024 \times 1,024$ images.

For each of the requested image size, 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 1.8 GHz CPU and 512M memory. All the reported results are calculated as the average of successful runs of 100 setups.

A. Quality and Time Comparison of M1 and M2

In this subsection, we compare both the quality and time of method M1 that finds optimal solutions and method M2 that finds sub-optimal solutions. In our experiments, we set CPU time limit to be 300 seconds for each run. If package *lp_solve* can not find an optimal solution within this limit, we consider this run as a failure.

Tables 1 and 2 show the comparison results when the re-

Table 3. Comparison of two methods, M1 and M2, when the requested coded image size is equal to 64 kbytes.

#Peers	M1			M2		
	Solution	Time	Success ratio	Solution	Time	Success ratio
2	19.71571	0.00028	1.00	19.71603	0.00018	1.00
4	9.28498	0.00079	1.00	9.28597	0.00026	1.00
8	4.18109	0.00960	1.00	4.18198	0.00044	1.00
12	2.75602	0.06813	1.00	2.75732	0.00075	1.00
16	2.05009	0.40652	1.00	2.05173	0.00114	1.00
20	1.62518	1.94033	1.00	1.62664	0.00163	1.00
24	1.34693	9.49448	1.00	1.34867	0.00224	1.00

Table 4. Comparison of two methods, M1 and M2, when the requested coded image size is equal to 128 kbytes.

#Peers	M1			M2		
	Solution	Time	Success ratio	Solution	Time	Success ratio
2	113.52194	0.00016	1.00	113.52204	0.00018	1.00
4	78.14524	0.00034	1.00	78.14542	0.00026	1.00
8	33.74424	0.00239	0.97	33.74474	0.00048	1.00
12	23.23318	0.05882	0.89	23.23373	0.00079	1.00
16	6.09379	2.09358	0.92	6.09499	0.00120	1.00
20	4.84248	32.55980	0.94	4.84386	0.00177	1.00
24	4.09679	64.97960	0.58	4.09848	0.00247	1.00

quested images are of size 512×512 and are coded in 0.5 bpp (*i.e.,* coded image size = 16 kbytes) and 1 bpp (*i.e.,* coded image size = 32 kbytes), respectively. Similarly Tables 3 and 4 show the results for the images of size $1,024 \times 1,024$, which are coded in 0.5 bpp (*i.e.,* coded image size = 64 kbytes) and 1 bpp (*i.e.,* coded image size = 128 kbytes), respectively. We have the following observations on the comparison results.

1. The sub-optimal solutions found by M2 are very close to the optimal solutions by M1 with difference less than 0.6%. In Section VI, we have derived an upper bound characterizing distance between these two solutions, which is equal to $\frac{u}{\min\{b_1,b_2,\cdots,b_n\}}$, where u is set to 8 bits (1 byte) here. Therefore, the difference should be smaller than $1/64 = 0.015625$ second, and this is verified by these experiments.
2. In terms of computational time, M2 is able to find sub-optimal solutions in the order of one to two milliseconds, independent of the sizes of requested images. For all the cases, its CPU time only grows from 0.2 to 2.5 millisecond as the number of peers increases from 2 to 24. However, for M1, computational time grows rapidly as the number of peers increases. For example, when the requested image is of size 16 kbytes (resp. 128 kbytes), its CPU time increases from 0.3 millisecond to 7.9 seconds (resp. 0.2 millisecond to 65.0 seconds) as the number of peers increases from 2 to 24.
3. M1 has difficulties finding solutions for two groups of experiments in which all the peers hold coded images less than or equal to the requested image size. For example, its success ratio is less than 100% when the number of peers is greater than or equal to 8 in Tables 2 and 4. The success ratio can be as low as 31% in Table 2 and 58% in Table 4. For the other two groups of experiments where some peers hold images greater than the requested image size, M1 can find all the solutions with 100% success ratio. This indicates that the former case represents more challenging scenarios for mixed-integer linear programming. In contrast, M2 can successfully find solutions in all sets of experiments.

In summary, we can conclude that M2 can find near-optimal solutions very efficiently. Besides, it enjoys 100% success ratios in all the scenarios. Therefore, in practice M2 is a much better candidate than M1 to perform peer assignment.

B. Comparing M2 with Simple Heuristic Schemes

In this subsection, we study how M2 compares with other heuristic peer assignment schemes. As we haven't found any previous work performing peer assignment for transmission of scalable coded images, we define the following three simple heuristics.

H1. A length-based assignment scheme in which image segments are allocated to peers based on the image size of each peer. First, we select peer 1 to transmit the bit stream between [0, s_1] kbits, and peer 2 to transmit between (s_1, s_2], and so on. In general, peer i would transmit the bit stream between (s_{i-1}, s_i], for $i = 1, 2, \cdots, n-1$, and peer n would transmit between (s_{n-1}, M]. For this scheme, the requesting peer only needs to do simple substractions to come up with peer assignment.

H2. A random assignment scheme in which image segments are allocated to peers by random selection. The requesting peer starts its assignment from the beginning of the bit stream, and randomly selects a peer, *e.g.*, peer r, to transmit the portion from the start to the size of this peer's coded bit stream, *i.e.,* (0, s_r]. Then it randomly selects another peer t and allocates the bit stream between (s_r, s_t] to peer t. It follows this procedure until it finishes allocating all the bit stream to the peers.

H3. A bandwidth-based assignment scheme in which image segments are allocated to peers based on the bandwidth of each peer. The requesting peer first selects the peer with the

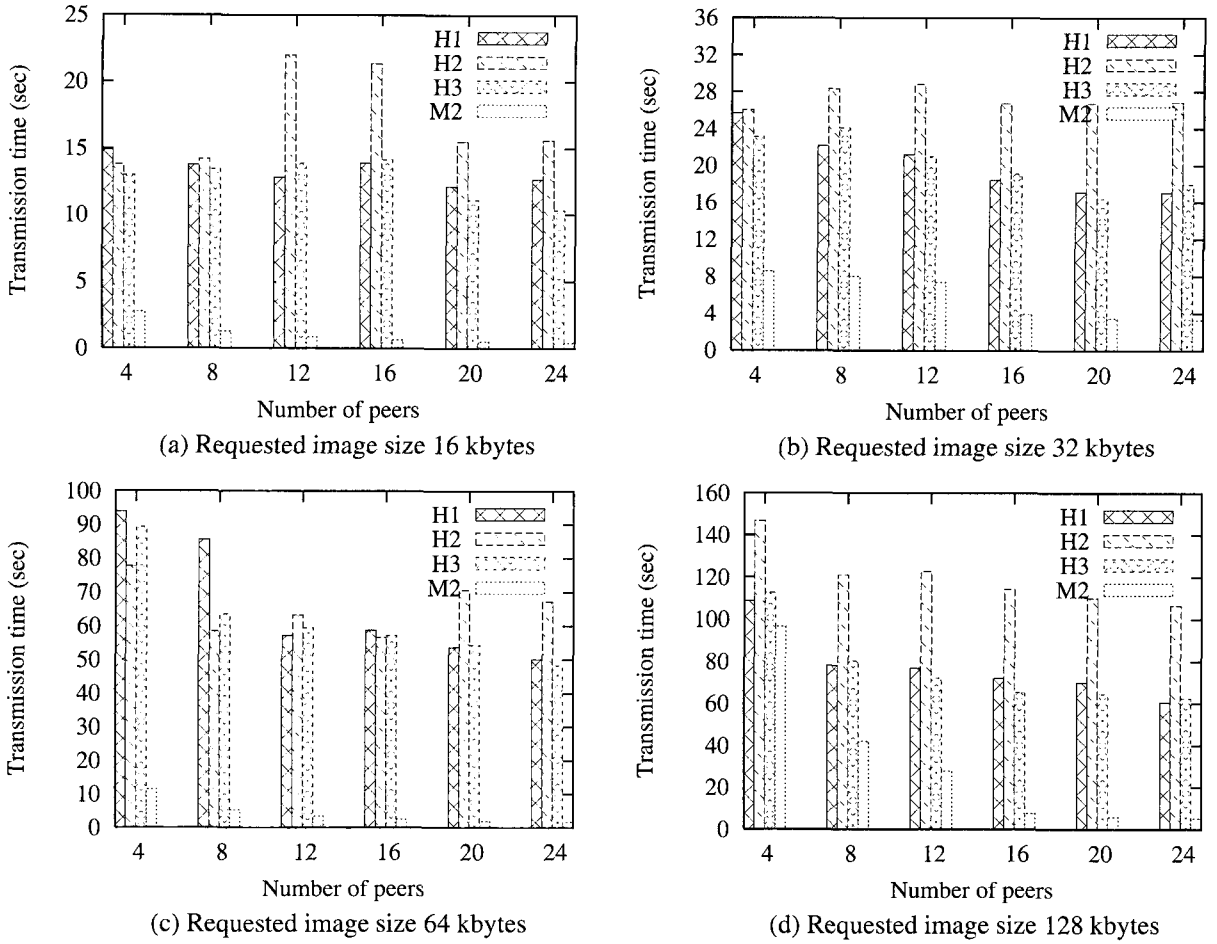


Fig. 7. Performance comparisons of three peer assignment algorithms when image size is equal to (a) 16 kbytes, (b) 32 kbytes, (c) 64 kbytes, and (d) 128 kbytes, respectively.

largest bandwidth, *e.g.*, peer i, to transmit the portion between $(0, s_i]$, then it selects the peer with the second largest bandwidth, *e.g.*, peer j, to transmit the image segment between $(s_i, s_j]$. This procedure continues until it has mapped all the bit stream to the peers.

In our experiments, peers employ backward transmission strategy (discussed in Section VI-B) in all four algorithms, H1, H2, H3, and M2. We assume that 20% peers may leave the network before they finish their transmissions. Their network departure time is uniformly distributed in the time interval of image transmission.

Fig. 7 shows the comparison results when the size of the requested image is equal to 16 kbytes, 32 kbytes, 64 kbytes and 128 kbytes, respectively. Fig. 7 shows that M2 results in much shorter transmission latency than the three heuristics. In Fig. 7(a), the speedup ranges from 6 to 30 times compared to H1, from 6 to 38 times compared to H2, from 6 to 25 times compared to H3. In other image transmission requests (shown in Figs. 7(b), 7(c), 7(d)), the speedups are also very significant.

Intuitively, the peer assignment algorithm is largely dependent on two parameters: image sizes and bandwidths of peers. Among the four algorithms, H1 performs peer assignment based on the image sizes of peers, H3 based on bandwidths, and H2 based on random selection, while M2 optimizes transmission time based on both image size and transmission bandwidth. We observe similar performance of two heuristics, H1 and H3, and M2 outperforms both of them. All the three algorithms, M2, H1, and H3 perform better than random-based peer assignment.

We can also observe that the transmission time of M2 decreases with increasing number of supplying peers, which is a desirable property of a good peer assignment algorithm. In contrast, none of the heuristic algorithm demonstrates this property.

In terms of computational time, all the heuristic algorithms H1, H2, and H3, take about 0.1 milliseconds to complete, about 10 times faster than M2. However, since M2 is sufficiently fast, in the order of 1 to 2 milliseconds, it is not important to push the algorithm to be even faster.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we described our efforts to understand how scalable coding algorithm and property may influence image transmission on peer-to-peer networks. We first defined peer assignment problem, proved the existence of an optimal solution when using an identity permutation, and subsequently formulated peer assignment with this identity permutation as a mixed-integer linear programming problem. Then we described how to perform peer assignment when peers may leave the network before fin-

ishing transmission and discussed how to adapt the algorithm to video streaming. Finally, we verified the excellent performance of the proposed algorithms through extensive simulations.

In the future, we plan to extend this work to unreliable delivery by studying the effects of packet losses on supplying bandwidth and peer assignment algorithms.

REFERENCES

[1] D. Stolarz, "Peer-to-peer streaming media delivery," In *Proc. First Int. Conf. Peer-to-Peer Computing*, Aug. 2001.

[2] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, June 1996.

[3] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

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

[5] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, Feb. 1992.

[6] ITU-T recommendation H.263: Video coding for low bitrate communication, 1998.

[7] ISO/IEC IS 13818-2 (MPEG-2 Video): Information technology - generic coding of moving pictures and associated audio information, Apr. 1996.

[8] Napster, http://www.napster.com.

[9] Gnutella, http://gnutella.wego.com.

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

[11] eDonkey, http://www.edonkey2000.com.

[12] BitTorrent, http://bitconjurer.org/BitTorrent.

[13] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," Tech. Rep. MSR-TR-2003-11, Microsoft Research, Mar. 2003.

[14] 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.

[15] 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*, Feb. 2003.

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

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

[18] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE J. Sel. Area Commun.*, vol. 22, no. 1, pp. 121–133, Jan. 2004.

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

[20] R. Rejaie and A. Ortega, "PALS: Peer to peer adaptive layered streaming," In *Proc. ACM/IEEE NOSSDAV*, June 2003.

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

[22] H. Huang, W. Peng, T. Chiang, and H. Hang, "Advances in the scalable amendment of h.264/avc," *IEEE Commun. Mag.*, vol. 45, no. 1, pp. 68 – 76, 2007.

[23] X. Su, R. Fatoohi, and T. Wang, "Optimizing transmission time of scalable coded images in peer-to-peer networks," *ACM/Springer Multimedia Systems Journal*, pp. 413–421, Aug. 2005.

[24] P. M. Pardalos and H. E. Romeijn, *Handbook of Global Optimization.* Kluwer Academic Publishers, 2002.

[25] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*, 1st ed. Wiley-Interscience, 1999.

[26] S. J. Wright, *Primal-Dual Interior-Point Methods.* SIAM, 1997.

[27] Morgan M-JPEG2000, http://www.morgan-multimedia.com/M-JPEG2000/.

[28] ITU-T recommendation H.264: Advanced video coding, 2005.

[29] W. Li, "Overview of fine granularity scalability in mpeg-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

[30] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE J. Sel. Area Commun.*, vol. 22, no. 1, pp. 91–106, Jan. 2004.

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

Xiao Su received her Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. She is currently an associate professor in the Computer Engineering Department, San Jose State University, San Jose, CA. Her research interests include network security, multimedia communications, media coding, and mobile computing. She served as co-chair for the Int'l Symposium on Multimedia over Wireless and on technical committees on numerous IEEE sponsored conferences. She is a recipient of the National Science Foundation CAREER award.

Tao Wang received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. He is currently with Synopsys Inc., USA. His research interests include nonlinear optimization, image processing, and networking systems.