Building an Efficient Transcoding Overlay for P2P Streaming to Heterogeneous Devices

DONGYU LIU, Cavium Networks
FEI LI, George Mason University
BO SHEN, Vuclip
SONGQING CHEN, George Mason University

With the increasing deployment of Internet P2P/overlay streaming systems, more and more clients use mobile devices, such as smart phones and PDAs, to access these Internet streaming services. Compared to wired desktops, mobile devices normally have a smaller screen size, a less color depth, and lower bandwidth and thus cannot correctly and effectively render and display the data streamed to desktops.

To address this problem, in this paper, we propose PAT (Peer-Assisted Transcoding) to enable effective online transcoding in P2P/overlay streaming. PAT has the following unique features. First, it leverages active peer cooperation without demanding infrastructure support such as transcoding servers. Second, as online transcoding is computationally intensive while the various devices used by participating clients may have limited computing power and related resources (e.g., battery, bandwidth), an additional overlay, called metadata overlay, is constructed to instantly share the intermediate transcoding result of a transcoding procedure with other transcoding nodes to minimize the total computing overhead in the system. The experimental results collected within a realistically simulated testbed show that by consuming 6% extra bandwidth, PAT could save up to 58% CPU cycles for online transcoding.

Categories and Subject Descriptors: D.4.4 [**Operating Systems**]: Communications Management—*Network communication*

General Terms: Design, Algorithms, Experimentation

Additional Key Words and Phrases: P2P/overlay streaming, meta-transcoding, heterogeneity

ACM Reference Format:

Liu, D., Li, F., Shen, B., and Chen, S. 2012. Building an efficient transcoding overlay for P2P streaming to heterogeneous devices. ACM Trans. Multimedia Comput. Commun. Appl. 8S, 1, Article 10 (February 2012), 22 pages.
DOI = 10.1145/2089085.2089087 http://doi.acm.org/10.1145/2089085.2089087

The work has been supported in part by U.S. AFOSR under grant FA9550-09-1-0071, and by the U.S. National Science Foundation under grants CNS-0746649, CCF-0915681, CNS-1117300, and CCF-1146578.
A previous version of the work has been published in NOSSDAV 2007 [Liu et al. 2007].
D. Liu is currently affiliated with MicroStrategy Inc.
Author's addresses: D. Liu, MicroStrategy Inc., 1850 Towers Crescent Plaza, Tysons Corner, VA 22182; email: dliu@microstrategy.com; F. Li, Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, VA 22030; email: lifei@cs.gmu.edu; B. Shen, Vuclip, 1551 McCarthy Boulevard #213, Milpitas, CA 95035; email: boshen_99@yahoo.com; S. Chen, Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, VA 22030; email: sqchen@cs.gmu.edu.

DOI 10.1145/2089085.2089087 http://doi.acm.org/10.1145/2089085.2089087

1. INTRODUCTION

In recent years, the amount of Internet media traffic has increased exponentially. According to ComScore [Comscore], 16.8 billion online videos were accessed in April 2009 and that is a 40% increase compared with that of April 2008. To efficiently deliver Internet media objects, a number of different P2P/overlay streaming systems, such as PPLive [PPLive], MySee [Mysee], UUSee [UUSee], AnySee [Liu et al. 2006], and PPStream [Ppstream], have been proposed. In practice, PPLive [PPLive] serves hundreds of TV channels daily and has attracted hundreds of thousands of users.

In tandem, recent technology advances in wireless [wireless] and 3G [3G] have made Internet access pervasive. Many users today use PDAs or smart phones to access the Internet instead of traditional desktop computers. According to Nielsen Mobile [Nielsen mobile], a leading provider of performance measurement to the mobile industry, more than 40.4 million US mobile subscribers actively accessed the Internet via wireless devices in May 2008 and these subscribers account for 15.6% of the total 254 million US mobile phone subscribers with Internet access [UUSee].

While the usage of all kinds of mobile devices is increasing on the Internet, delivering streaming media to these devices in overlay streaming systems is challenging, since these mobile devices often differ from desktops in the screen size, the color depth, and the available downloading/uploading bandwidth. Therefore, the streaming data intended for a desktop cannot be directly rendered and displayed on a PDA or a smartphone.

Although some existing studies have considered that participating devices may have different uploading bandwidth in an overlay streaming system [Small et al. 2006], which is not an uncommon situation for most residential broadband users if ADSL or cable is used for Internet connections, they have not considered the heterogeneity in multiple dimensions when various mobile devices join the system. Some existing solutions based on scalable/layered coding [Ghanbari 1989; Nakamura and Sawada 1995] or multiple description coding [Padmanabhan et al. 2003] may not be effective given the fact that most devices do not support these codecs. Furthermore, these coding schemes can mainly satisfy coarse granularity QoS requirements and they cannot support customized content adaptation such as personal logo insertion. On the other hand, in P2P/overlay streaming, precoding requires either streaming of compound objects containing all versions or a separate overlay for each version, both of which imply waste of bandwidth.

In this paper, aiming to more efficiently address the heterogeneity problem, we propose Peer-Assisted Transcoding (PAT) to facilitate P2P/overlay streaming in heterogeneous environments. PAT relies on node cooperation without demanding infrastructure support. Media adaptation is carried out at real-time with online transcoding by selected peers. We show that this approach is effective in satisfying diverse demands from heterogeneous participating nodes in a P2P system.

Although the transcoding solution is more flexible in adapting to heterogeneous environments, one key challenge remains that such a solution is computationally intensive. Some research has been conducted to study online transcoding [Acharya and Smith 2000; Amir et al. 1995; Hess et al. 2000]. Most of these schemes treat the transcoding procedure as a black box with the final transcoded result as the only caching candidates. In our previous study [Liu et al. 2006a], we have introduced the idea of metacaching. Metacaching identifies intermediate transcoding steps from which certain intermediate results (called *metadata*) can be cached and reused later so that a fully transcoded object version can be produced from the metadata with a relatively smaller amount of CPU cycles. Such metadata-assisted transcoding is called meta-transcoding, which is leveraged in PAT. In addition, the sharing of the metadata is facilitated by the construction of an additional overlay, called metadata overlay, which is in parallel with the overlay used for data streaming. On this overlay, the metadata of a transcoding procedure is instantly shared with other transcoding nodes in order to minimize the total computing overhead in the system.

With the assistance of meta-transcoding, PAT effectively improves streaming quality and reduces the overall CPU load. Through extensive real-data-parameterized simulations, we show that the client-perceived streaming quality in PAT is remarkably improved, and with a small amount of additional bandwidth (6%), PAT can significantly reduce the CPU load (up to 58%) for online transcoding.

The remainder of the paper is organized as follows. Section 2 discusses related work. We describe the system in Section 3. A performance evaluation obtained with a simulated network is presented in Section 5. We make concluding remarks in Section 6.

2. RELATED WORK

P2P/overlay streaming systems have attracted considerable attention [PPLive; Castro et al. 2003; Hefeeda et al. 2003; Kostic et al. 2003; Magharei and Rejaie 2007; Padmanabhan et al. 2003]. ESM [Chu et al. 2000] is among the first reported implementation of peer-to-peer video streaming over the Internet. Recently a number of P2P/overlay streaming systems have been deployed, such as PPLive [PPLive], PPStream [PPstream], and SopCast [SOPcast].

Many researchers have studied the overlay streaming based on multicast trees or structured P2P systems. For example, ZIGZAG [Tran et al. 2003] organizes all the nodes into clusters and constructs a multicast tree based on these clusters. Padmanabhan et al. [Padmanabhan et al. 2003] use multiple distribution trees and multiple description coding (MDC) to provide redundancy to the system and enhance robustness of content distribution. Coopnet [Padmanabhan et al. 2002] provides live streaming as well as VoD services using a hybrid approach. Splitstream [Castro et al. 2003] is a high-bandwidth content distribution system where content is divided into k strips in order to distribute the forwarding load among all the participants. P2P/overlay streaming using data-driven unstructured P2P networks has also been studied. For example, Chainsaw [Pai et al. 2005] provides unstructured solutions to high-bandwidth data dissemination systems. A number of studies have also investigated efficient topologies for P2P live streaming. For example, Small et al. [Small et al. 2006] theoretically investigate the scaling law of P2P live multimedia streaming and the result can have practical implications for tree construction. A scalable architecture is proposed for congestion controlled multicast real-time communication by using self organized transcoders in [Kouvelas et al. 1998]. In order to satisfy large and highly dynamic host population, Dagster [Ooi 2005] proposes a novel incentive scheme to encourage nodes to contribute more bandwidth resource to the entire system. Peers in Dagster are assumed to be able to conduct transcoding during streaming. In TeC [Shen et al. 2004], strategies are proposed to selectively cache original or transcoding objects. Shen [2003] introduces the meta-transcoding for server side proxies. In our previous study [Liu et al. 2006a], we have introduced the idea of metacaching into the video on demand streaming. In our recent study [Liu et al. 2009], a general model is proposed to optimize the resource utilization in P2P streaming systems.

3. PEER-ASSISTED TRANSCODING (PAT) DESIGN

In this section, we first briefly introduce the basics of meta-transcoding and metadata matrix. Then we present the design of Peer-Assisted Transcoding (PAT), which consists of two overlays. One is the data (base) overlay for streaming data transmission. The other is the metadata overlay for instant metadata sharing.

3.1 Meta-Transcoding

In Liu et al. [2006a], we have proposed metacaching to effectively balance the resources used for online transcoding between CPU and storage resources in a transcoding proxy or server. The motivation of metacaching is as follows. Many existing schemes that aim to optimize transcoding treat transcoding

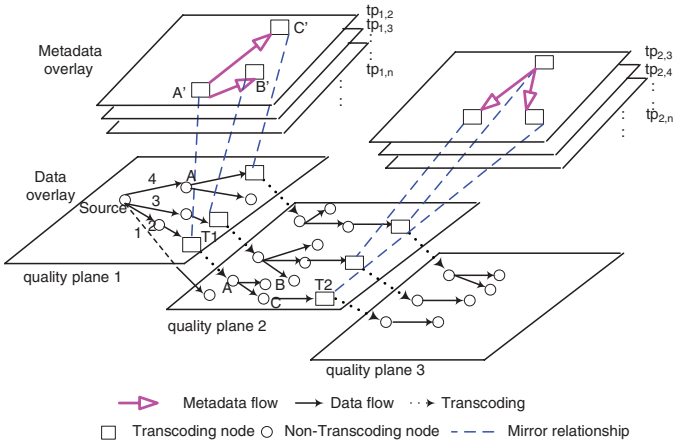

Fig. 1. A bi-overlay example.

as a black box, focusing on the optimization of caching transcoded object versions based on the prediction of future requests. However, a typical transcoding procedure involves multiple steps, among which some intermediate results, also called *metadata*, could be saved to avoid recomputing the same result again for a later transcoding request. If the saved metadata only consumes a small storage size while it takes a significant amount of computing cycles to generate, it is apparently an effective trade-off between the storage and CPU cycles. For example, for bit rate reduction transcoding, the requantization scale factor can be stored rather than being recomputed. In this case, meta-transcoding requires significantly less CPU cycles to produce the new transcoded object. For more details about meta-transcoding, please refer to Liu et al. [2006] and Shen [2003].

In this paper, we combine this approach with P2P/overlay streaming, where a media stream is disseminated to a large population of clients, utilizing the forwarding capacity of the clients. In addition, a client can also instantly share some intermediate transcoding results with other peers which are performing the same transcoding process. Clearly, extra bandwidth is required for metadata sharing.

3.2 Quality Plane and Transcoding Plane

Figure 1 shows that our system consists of two overlays: a data overlay and a metadata overlay. On the data overlay, the nodes that request the same streaming quality form a quality plane. Formally, *quality plane* i *(*qp_i*)* represents the sets of nodes which request quality i. Different quality planes are connected through transcoding nodes and these quality planes form a "quality version falls." With a total of n different quality versions in the system, there are n quality planes.

On the metadata overlay, based on the transcoding input and output versions, we have different transcoding planes for the corresponding transcoding clients. Formally, *transcoding plane (*$tp_{i,j}$*)* represents the sets of nodes which transcode quality i to quality j. For example, in Figure 1, transcoding nodes A', B', and C' perform the transcoding from quality version 1 to quality version 2 and thus are on the same transcoding plane (denoted as $tp_{1,2}$). Other three transcoding nodes conduct another type of transcoding so they are on another transcoding plane ($tp_{2,3}$). With a total of n different quality versions in the system, there are at most $\frac{1}{2}n \cdot (n-1)$ different transcoding planes if we ignore the impractical cases of transcoding from a lower quality to a higher quality stream. To represent availability of the metadata corresponding to transcoding from version i to version j, we use a matrix of size $n \cdot n$, denoted by *mMatrix*, which is an upper triangle matrix. We assume there is a single root (the source of

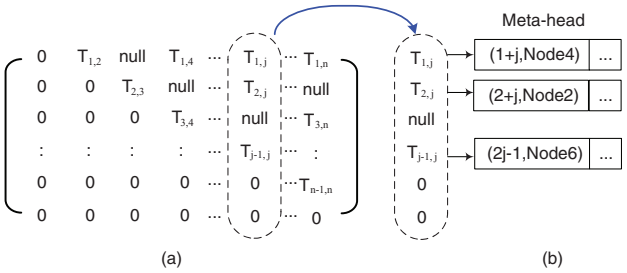

Fig. 2. A mMatrix example on the metadata overlay.

the stream). In addition to providing the original streaming data, the root also maintains *mMatrix*. Figure 2(a) shows an example of mMatrix.

When the corresponding metadata is not available, $T_{i,j}$ in mMatrix is set to `null`. Otherwise, such metadata is available and $T_{i,j}$ denotes the ID of the node which transcodes from version i to version j. We also call this node as the *metahead* and there may be other transcoding nodes that receive metadata from a metahead in order to perform the meta-transcoding task. Figure 2(b) shows an example of column j in mMatrix. The first element in the list represented by each `non-null` element in the mMatrix is referred as *meta-head*.

3.3 Data Overlay and Metadata Overlay Construction

The construction of data overlay in PAT differs from traditional P2P/overlays, mainly because when a node wants to join the streaming service, its parent selection procedure is substantially different from the traditional ones and also affects the construction of the metadata overlay. The node departure in PAT also affects both the data overlay and the metadata overlay. We present the node arrival and departure processes in Section 3.3.1 and Section 3.3.2, respectively.

3.3.1 *Node Arrival.* Assume the system starts with a source node and several bootstrap nodes. When a node wants to join the system, it first contacts the source node or a bootstrap node with its quality requirement. In this paper, the quality requirement is represented by the desired frame rate and bit rate version of the content. We map the quality requirement into a version index. The critical part of node joining is to find an appropriate parent for the joining node, which is conducted as follows.

—*Case* 1.1: *no transcoding required.* Assume the joining node requests object version j, the joining node looks for the node that can directly provide object version j via a candidate parent list containing the nodes that can provide the exact desired object quality. If there are several candidate parents, the joining node selects one of them with the smallest height in the tree in order to get the streaming with the least delay. In this case, only the data overlay is updated.

—*Case* 1.2: *transcoding required.* If the desired object version does not exist or if the node having the desired version cannot accept the joining node as a child, the joining node must look for a parent capable of transcoding. Assuming that the desired version is j, based on the availability of the metadata for transcoding to version j, the parent selection follows these steps.

—*Step* 1: *metadata available and meta transcoding*. The joining node checks the mMatrix from the source node or a bootstrap node by examining the j^{th} column of mMatrix. The possible parent candidates are from $T_{1,j}$ to $T_{j-1,j}$ (a total of $j-1$ candidates). Based on the type of transcoding, two different policies are used to reduce the usage of resources while maintaining similar

quality by considering different characteristics of different transcoding procedures. For this purpose, the joining node starts to send the joining request to the metahead node indicated by $T_{t,j}$ (t is a variable and $0 < t < j$)

—where $(j - t)$ is the smallest, if the transcoding is for frame rate or spatial resolution reduction. This policy is designed to minimize the computing load since it requires less computing load to transcode from a frame rate or spatial resolution version that is closer to the target frame rate or spatial resolution version.

—where $(j - t)$ is the largest, if the transcoding is for bit rate reduction. This policy is designed to minimize the quality degradation since the transcoding between versions that are further away (i.e., smaller number of intermediate versions) from each other in bit rate leads to less generation loss, which refers to video quality loss during the transcoding from one quality to another. On the other hand, the computing load for transcoding between different versions in bit rate is almost the same.

Then the candidate parent P decides whether to accept the joining node based on its bandwidth availability. If yes, P needs to conduct meta transcoding for the joining node and metadata subtree $T_{t,j}$ is updated when the metahead of $T_{t,j}$ looks for a parent for P as follows. If the metahead of $T_{t,j}$ itself has enough bandwidth, it serves as the parent of P; otherwise it will check whether any of its current children has enough bandwidth to accept the new child P using a breadth-first search. If P can find a parent to join the metadata overlay, the joining process finishes. If none of these nodes can accept P, the joining node will try another `non-null` element in mMatrix in ascending (or descending) order of $j - t$ value depending on the transcoding type so that it can find a new transcoding parent, and repeat above parent-children relationship establishing process. If all the `non-null` elements in the jth column of mMatrix have been explored without a transcoding parent node being identified, go to the next step.

—*Step* 2. *metadata unavailable and full transcoding*. If a parent is not found in the previous step, the joining node must look for a parent with enough bandwidth and CPU resource to perform full transcoding for it. To minimize the overall system transcoding overhead, the joining node starts to probe nodes receiving version t, where $T_{t,j}$ is `null` in mMatrix. The joining node probes the node with the smallest or largest $j-t$ value depending on the type of transcoding as discussed in Step 1. If no suitable parent is found, the joining node probes the next node in ascending (or descending) order of $j - t$ value until a parent node is found. The chosen parent will start a full transcoding process. The chosen parent can also share the metadata produced during the transcoding session by joining the metadata overlay as follows.

—After the parent (with its ID denoted as P) that is willing to do transcoding is selected and the joining node joins the data overlay, P also needs to join the metadata overlay. Suppose node P needs to do transcoding from version k to version j. As P needs to do full-transcoding, metadata subtree $T_{k,j}$ is updated as follows: node P becomes the meta-head of the $T_{k,j}$ subtree and the corresponding element in mMatrix is updated.

If a parent node cannot be found, go to the next step.

—*Step* 3. If version j is not the lowest quality version, go back to Step 1 to request version $j + 1$. Otherwise, the joining request is rejected.

During the joining process, the joining node may find several eligible parents and it can save the information about s parents (e.g., $s = 3$) as its backup parents. The reason to do this is to increase the system robustness: the node can quickly connect to its backup parent when its current parent leaves the system.

3.3.2 *Node Departures.* In P2P/overlay streaming, a participating node may depart at any time. Upon node departure, the overlay must be readjusted to adapt to the change. Denote the departing node as D and its parent as P. Such adjustments are performed as follows.

—*Case* 2.1. If D is a leaf node and is receiving version j, (1) if P is not a transcoding node, remove D from P's children list and update its available bandwidth information; (2) if P is a transcoding node conducting transcoding from version i to version j, P needs to depart from the metadata subtree $T_{i,j}$ according to the following *META-LEAVE* protocol.
  —If P is the metahead in the metadata tree, it tries to find a new meta-head from its children list and select the one with the largest available bandwidth. If a child C is selected, $T_{i,j}$ in mMatrix is updated accordingly and the sibling nodes of node C become the children of new metahead C and parent-children relationship of the related nodes is updated. If P is the meta-head with no children in the metadata overlay, $T_{i,j}$ in mMatrix is set to `null`.
  —If P is not a meta-head in the metadata tree, the meta-head deletes node P from the metadata subtree. If node P has children, they become the children of the parent of P in the metadata subtree if the parent of P has enough bandwidth. They will contact the metahead to rejoin the metadata tree if the parent of P does not have enough bandwidth. The corresponding children and parent information of these nodes is updated.

—*Case* 2.2. If D is neither a leaf node nor a transcoding node, the children of D need to find a new parent. If P has enough bandwidth, P becomes their new parents and the children of D take it as its new parent and update the corresponding information. Otherwise, the children of D need to perform a rejoin process as described in Section 3.3.1 and the rejoining process may change the transcoding plane but not the quality plane because these children do not change their quality requirements. If P is a transcoding node, P leaves the metadata subtree by following the above *META-LEAVE* steps. Finally, D is removed from the meta-data overlay.

—*Case* 2.3. If D receiving version i is not a leaf node and is a transcoding node, the adjustment must be done on both the data overlay and the metadata overlay. We assume D is in the metadata tree $T_{i,j}$ and the children of D need version j. All children of D in both the metadata tree and the data tree need to find new parents in the corresponding tree. In the metadata tree, D departs the meta-data tree according to the *META-LEAVE* protocol in Case 2.1. After the metadata tree is updated, the first child of D and its siblings in the data tree find a new parent as follows. In the updated metadata subtree $T_{i,j}$, if there are still some nodes in the metadata tree $T_{i,j}$ after the departure of node D, they can be selected to be new transcoding parents of these children based on their available bandwidth. If the metadata tree $T_{i,j}$ is `null` or no node has enough bandwidth, the children of D need to rejoin the system. In the rejoining process, these nodes have a chance to be the child node of a node that can directly provide version j within the data tree.

4. PERFORMANCE MODELING

In a streaming system, the throughput performance is very important. In this section, we seek to study the throughput of PAT. In PAT, the system throughput is represented by the total number of the peers that a quality plane in the data overlay and a transcoding plane in the metadata overlay can serve.

On a transcoding plane, the transcoding node have three schemes to select: full transcoding, meta transcoding and no-caching transcoding. In full transcoding, a transcoded object version gets cached in full. Meta transcoding caches the intermediate transcoding results, which can be used to construct the transcoded object version in full. No-caching transcoding does not cache any transcoding result. Different schemes have different CPU cycles and bandwidth resource requirements for transcoding. Thus, with certain available bandwidth and CPU cycles, the throughput in a transcoding plane is

Table I. Notations Used in the Analysis

B	the total bandwidth resource for quality version j
C	the total computing cycles for quality version j
v	the number of quality versions of original object
N_j	the total number of peers requesting quality version j
p_j	the percentage of the sessions not served by transcoding in the quality plane qp_j

decided by the transcoding scheme selected by the transcoding node in the corresponding transcoding plane. On a quality plane, the throughput is only decided by the available bandwidth.

Therefore, with given resources, we seek to quantify the throughput for full transcoding, meta transcoding and no caching transcoding for each quality version, and compare their performance.

4.1 Model Construction

In the analysis, we make the following assumptions about client accesses and media objects in the streaming system.

(1) The bandwidth resource occupied by each original object is one bandwidth unit. The bandwidth needed for delivering different versions is different. From the highest quality to the lowest quality, each version consumes bandwidth of 1, $\frac{v-1}{v}$, $\frac{v-2}{v}$, ... , $\frac{1}{v}$.

(2) The CPU resource used for full transcoding is one unit. The bandwidth used to deliver metadata is α $(0 < \alpha < 1)$ percent of the corresponding object. For meta-caching, we assume the CPU resource consumed to produce the final version is β $(0 < \beta < 1)$.

In this section, we conduct our analysis on one quality version j. The notations used in the following analysis are summarized in Table I.

4.1.1 *Bandwidth Constraint.* In our system, we consider how many accesses can be supported by the available bandwidth. We assume that there are v object versions in the system. Among the total N accesses, the number of accesses to each version is $N_1, N_2, \ldots, N_v$ and $N = \sum_{j=1}^{v} N_j$, respectively.

We use the procedure which processes the requests for version j as an example to analyze the system throughput. For quality version j, there are a corresponding quality plane (qp_j) and $v - j$ transcoding planes $(tp_{j,j+1}, \ldots, tp_{j,v})$.

Among all the nodes in qp_j (note that both non-transcoding peers and transcoding peers which require quality j are on quality plane qp_j. Some of these peers can provide the streaming version j without transcoding and some can provide lower quality versions). Assume the percentage of the sessions that are not served by transcoding in the quality plane (qp_j) is p_j $(0 < p_j < 1)$, the number of the sessions which are not served by transcoding is $N_j \cdot p_j$ and the number of the sessions which are served by transcoding is $N_j \cdot (1 - p_j)$.

(1) For full-transcoding, the total bandwidth consumption is composed of two parts: non-transcoding bandwidth consumption and transcoding bandwidth consumption. These non-transcoding peers consume $N_j \cdot p_j \cdot \frac{v-j+1}{v}$ considering that the size of quality version j is $\frac{v-j+1}{v}$. These transcoding peers use $N_j \cdot (1 - p_j) \cdot v_a$ (here v_a represents the average size the versions which are transcoded from version j and $v_a = (\frac{v-j}{v} + \frac{v-(j+1)}{v} + \cdots + \frac{1}{v})/(v - j) = \frac{v-j+1}{2v}$ if we assume different versions are accessed uniformly). Because of the bandwidth constraint of the transcoding head (note that the transcoding head is the node which initially conducts the full transcoding), the transcoded version is transferred to one child node of the transcoding head and then forwarded to the node which originally requests the transcoded version. Thus the total of the bandwidth consumed by the joining node is $2v_a$.

Therefore, the average bandwidth requirement for each session for full-transcoding b_f is:

$$\begin{aligned} b_f &= \frac{N_j \cdot p_j \cdot \frac{v-j+1}{v} + N_j \cdot (1-p_j) \cdot 2v_a}{N_j} \\ &= p_j \cdot \frac{v-j+1}{v} + (1-p_j) \cdot 2 \cdot \frac{v-j+1}{2v} \\ &= \frac{v-j+1}{v}. \end{aligned} \tag{1}$$

(2) For meta-transcoding, in addition to the bandwidth to deliver the transcoded version to the requesting node, metadata also needs to be delivered to the transcoding parent in the transcoding plane. Thus the total bandwidth requirement is $(1+\alpha) \cdot v_a$ (α is the bandwidth unit required to deliver metadata and $0 < \alpha < 1$). So the average bandwidth requirement b_m is:

$$\begin{aligned} b_m &= \frac{N_j \cdot p_j \cdot \frac{v-j+1}{v} + N_j \cdot (1-p_j) \cdot (1+\alpha) \cdot \frac{v-j+1}{2v}}{N_j} \\ &= \left(p_j + (1-p_j) \cdot \frac{(1+\alpha)}{2} \right) \cdot \frac{v-j+1}{v}. \end{aligned} \tag{2}$$

(3) For no-caching transcoding, every transcoded version is delivered to the requesting client directly after the transcoded version is produced, thus the total bandwidth requirement is v_a. Hence, the average bandwidth requirement b_n is:

$$\begin{aligned} b_n &= \frac{N_j \cdot p_j \cdot \frac{v-j+1}{v} + N_j \cdot (1-p_j) \cdot \frac{v-j+1}{2v}}{N_j} \\ &= \left(p_j + \frac{(1-p_j)}{2} \right) \cdot \frac{v-j+1}{v} \\ &= \left(\frac{1+p_j}{2} \right) \cdot \frac{v-j+1}{v}. \end{aligned} \tag{3}$$

Intuitively, we can get that $b_f > b_m > b_n$. Given a bandwidth capacity B, the total number of sessions that can be supported by full-transcoding, meta-transcoding, and no-caching schemes are denoted as n_{fb}, n_{mb}, and n_{nb}, respectively. We have

$$n_{fb} = B/b_f = \frac{B \cdot v}{v-j+1}. \tag{4}$$

$$n_{mb} = B/b_m = \frac{B}{\left(p_j + (1-p_j) \cdot \frac{(1+\alpha)}{2}\right) \cdot \frac{v-j+1}{v}}. \tag{5}$$

and

$$n_{nb} = B/b_n = \frac{B}{\left(\frac{1+p_j}{2}\right) \cdot \frac{v-j+1}{v}}. \tag{6}$$

4.1.2 *CPU Constraint.* Having considered the bandwidth constraint, we continue to study the CPU constraint of these three schemes. In full-transcoding, every session needs one unit of CPU resource for the first access and zero unit for the following accesses requesting the same version. In meta-transcoding, every session needs one unit of CPU for the first access and stored the metadata and β unit for the following accesses requesting the same version (β is the CPU resource used to produce a

fully transcoded version from cached metadata and $0 < \beta < 1$). In no-caching, every session asking for a transcoded version needs one unit of CPU resource.

For full-transcoding, only $v - j$ versions $(j + 1, \ldots, v)$ can be provided by version j and the total CPU resource required is $v - j$ units. The average computing load c_f is

$$c_f = \frac{v - j}{N_j}. \tag{7}$$

For meta-transcoding, only the first $v - j$ sessions need 1 unit CPU and other transcoding sessions need β unit CPU. The average computing load c_m is

$$c_m = \frac{v - j + (N_j \cdot (1 - p_j) - (v - j)) \cdot \beta}{N_j}. \tag{8}$$

For no-caching, every transcoding session needs 1 unit CPU resource and the average computing load c_n is

$$c_n = \frac{N_j \cdot (1 - p_j)}{N_j} = 1 - p_j. \tag{9}$$

Based on these equations, we can get $c_f \leq c_m \leq c_n$. Given an available CPU capacity C, the number of sessions that can be supported by the full-transcoding, meta-transcoding, and no-caching transcoding schemes are denoted as n_{fc}, n_{mc}, and n_{nc}, respectively. We have

$$n_{fc} = C/c_f = \frac{C \cdot N_j}{v - j}. \tag{10}$$

$$n_{mc} = C/c_m = \frac{C \cdot N_j}{v - j + (N_j \cdot (1 - p_j) - (v - j)) \cdot \beta}. \tag{11}$$

and

$$n_{nc} = C/c_n = \frac{C}{1 - p_j}. \tag{12}$$

4.1.3 *Bandwidth and CPU Constraint.* Now we consider the bandwidth and CPU resource constraint jointly. If we use tn_f, tn_m, and tn_n to denote the total session number for full-transcoding, meta-transcoding, and no-caching, respectively, we have

$$tn_f = \min(n_{fb}, n_{fc}) = \min\left(\frac{B \cdot v}{v - j + 1}, \frac{C \cdot N_j}{v - j}\right). \tag{13}$$

$$\begin{aligned} tn_m &= \min(n_{mb}, n_{mc}) \\ &= \min\left(\frac{B}{\left(p_j + (1 - p_j) \cdot \frac{(1+\alpha)}{2}\right) \cdot \frac{v-j+1}{v}}, \frac{C \cdot N_j}{v - j + (N_j \cdot (1 - p_j) - (v - j)) \cdot \beta}\right). \end{aligned} \tag{14}$$

and

$$tn_n = \min(n_{nb}, n_{nc}) = \min\left(\frac{B}{\left(\frac{1+p_j}{2}\right) \cdot \frac{v-j+1}{v}}, \frac{C}{1 - p_j}\right). \tag{15}$$

4.2 Modeling Analysis

According to the model we have presented, we discuss the throughput for these three schemes.

By Equations (4)–(6) and $0 < \alpha < 1$, it is obvious that

$$n_{fb} < n_{mb} < n_{nb}. \tag{16}$$

By Equations (10)–(12) and $0 < \beta < 1$, it is obvious that

$$n_{fc} \geq n_{mc} \geq n_{nc}. \tag{17}$$

When $n_{nb} \leq n_{nc}$ ($B \leq \frac{C}{1-p_j} \cdot (\frac{1+p_j}{2}) \cdot k_1$), no-caching is the scheme that has the largest throughput. Here, $k_1 = \frac{v-j+1}{v}$. Now we focus on the analysis when $n_{nb} > n_{nc}$ ($B > \frac{C}{1-p_j} \cdot (\frac{1+p_j}{2}) \cdot k_1$).

1. When $n_{fb} \geq n_{fc}$, full transcoding has the largest throughput and the largest throughput is n_{fc}. From $n_{fb} \geq n_{fc}$, we get

$$\frac{B}{k_1} \geq \frac{C \cdot N_j}{v-j} \tag{18}$$

When $\frac{B}{k_1} \geq \frac{C \cdot N_j}{v-j}$, full-transcoding has the largest throughput.

2. When $n_{fb} < n_{fc}$, the full-transcoding result is n_{fb} and we need to compare that with the throughput of meta-transcoding.

When $n_{fb} < n_{fc}$, we get

$$B < \frac{C \cdot N_j \cdot k_1}{v-j} \tag{19}$$

—$n_{mb} \geq n_{mc}$: If $n_{mb} \geq n_{mc}$, the throughput of meta-transcoding is n_{mc}.

—If $n_{fb} \geq n_{mc}$, full-transcoding should be selected. When $n_{fb} \geq n_{mc}$, we can get

$$B \geq \frac{C \cdot N_j \cdot k_1}{v-j+(N_j \cdot (1-p_j)-(v-j)) \cdot \beta} \tag{20}$$

—Combining with 19, we can get

$$\frac{C \cdot N_j \cdot k_1}{v-j+(N_j \cdot (1-p_j)-(v-j)) \cdot \beta} \leq B < \frac{C \cdot N_j \cdot k_1}{v-j} \tag{21}$$

If $n_{fb} < n_{mc}$, meta-transcoding should be selected. When $n_{fb} < n_{mc}$, we can get

$$\frac{C \cdot N_j \cdot (p_j + (1-p_j) \cdot \frac{(1+\alpha)}{2}) \cdot k_1}{v-j+(N_j \cdot (1-p_j)-(v-j) \cdot \beta} \leq B < \frac{C \cdot N_j \cdot k_1}{v-j+(N_j \cdot (1-p_j)-(v-j)) \cdot \beta} \tag{22}$$

—$n_{mb} < n_{mc}$: If $n_{mb} < n_{mc}$, the throughput of meta-transcoding is n_{mb}. When $n_{fb} < n_{mb}$, we can get

$$\frac{B}{k_1} < \frac{B}{\left(p_j + (1+\alpha) \cdot \frac{1-p_j}{2}\right) \cdot k_1}. \tag{23}$$

Obviously as $0 < \alpha < 1$, we can get $n_{fb} > n_{mb}$, so full-transcoding should be selected.

Table II summarizes the analysis result briefly.

Based on the results in Table II, a system can use the help of such information and decide which transcoding scheme to use in order to further improve the system throughput.

Table II. Summary of Analysis

Condition	Best scheme
$n_{nb} \leq n_{nc} (B \leq \frac{C}{1-p_j} \cdot (\frac{1+p_j}{2}) \cdot k_1)$	no-caching
$n_{nb} > n_{nc}, n_{fb} > n_{fc} (B > \frac{C \cdot N_j \cdot k_1}{v-j})$	full-transcoding
$n_{nb} > n_{nc}, n_{fb} \leq n_{fc} (B \leq \frac{C \cdot N_j \cdot k_1}{v-j}), n_{mb} \geq n_{mc}$ and $n_{fb} \geq n_{mc}$	full-transcoding
$n_{nb} > n_{nc}, n_{fb} \leq n_{fc} (B \leq \frac{C \cdot N_j \cdot k_1}{v-j}), n_{mb} \geq n_{mc}$ and $n_{fb} < n_{mc}$	meta-transcoding
$n_{nb} > n_{nc}, n_{fb} \leq n_{fc} (B \leq \frac{C \cdot N_j \cdot k_1}{v-j}), n_{mb} < n_{mc}$	full-transcoding

5. PERFORMANCE EVALUATION

In this section, we first perform real transcoding and meta-transcoding in our implemented transcoder for both bit rate reduction and frame rate reduction. With parameters obtained from these experiments, we evaluate the performance of PAT based on ns2 [Network Simulator].

5.1 Experimental Parameter Capturing and Setup

In our implemented transcoder, we conduct both bit rate transcoding and frame rate transcoding experiments. A 1000-second video with a data rate of 500 Kb/s and 30 frames per second (denoted as f/s) is used as the original object for each transcoding experiment. The corresponding PSNR of the original video is 33.85 dB. For bit rate reduction, we have 5 different versions with the corresponding bit rates decreasing from 500 Kb/s to 100 Kb/s with a 100 Kb/s interval. Normalizing the CPU load for a full transcoding session as 1 unit, meta-transcoding only consumes 0.4 for all bit rate reduction cases. The corresponding metadata size is 10% of the original video file size, regardless of bit rate differences.

For frame rate transcoding, we experiment with 5 different quality versions as well. They are 500 Kb/s and 400 Kb/s at 30 f/s, 300 Kb/s and 200 Kb/s at 15 f/s, and 100 Kb/s at 5 f/s. Normalizing the CPU load for a full transcoding session from 30 f/s to 15 f/s as 1 unit, the CPU load for full transcoding from 30 f/s to 5 f/s and from 15 f/s to 5 f/s are 0.44 units and 0.36 units, respectively. For meta-transcoding, the corresponding CPU load is 0.5, 0.27, and 0.19 units, respectively, and the corresponding metadata size is 20%–25% of the original video file size for transcoding between different frame rates, depending on the source bit rate and target frame rate.

With these parameters, we evaluate PAT over a topology of 1000 nodes. Specifically, one source node has all the different versions and its bandwidth capacity is 5 Mb/s. The distribution of uploading bandwidth capacity for the rest of the nodes is as follows: 5 Mb/s (11%), 2 Mb/s (3%), 896 Kb/s (9%), 384 Kb/s (21%) and 256 Kb/s (56%). Assume the version index 1 represents the original version and a larger index indicates a lower quality version. We further dictate the nodes with 5 Mb/s and 2 Mb/s connections to randomly access versions 1 to 3, and nodes with other bandwidth capacities to randomly access versions 1 to 5. The CPU constraints of nodes are randomly distributed from 1 unit to 2 units. The system starts with one source node capable of serving all versions. These 1000 nodes dynamically join and depart from the system. Their arrival pattern follows a Poisson distribution with the mean arrival rate as 1 request per second and the maximum arrival interval of 10 seconds. Nodes depart randomly after receiving the service for a duration ranging from 250 seconds to 1000 seconds.

5.2 Experimental Results Under Moderate Load

We evaluate the performance of our proposed scheme (*PAT-Meta*) by comparing it with two other schemes. *No-Transcoding* indicates the system in which no node can perform transcoding but the source node can serve precoded versions and a joining request is accepted if the requested or a lower quality version is available. *PAT-Normal* indicates the PAT system in which all transcoders perform full transcoding without assistance from metadata.

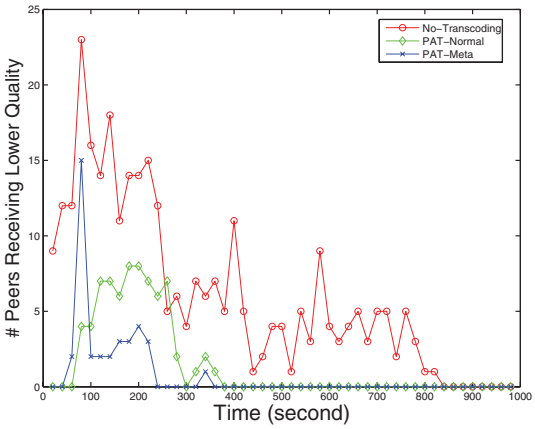

Fig. 3. Peers receiving lower quality (bit rate transcoding).

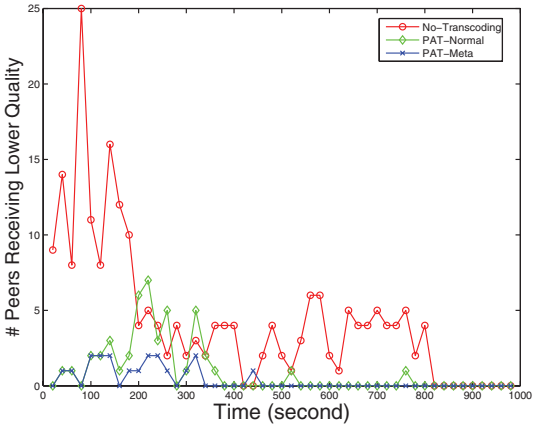

Fig. 4. Peers receiving lower quality (frame rate transcoding).

5.2.1 *Effect on the Client Received Video Quality.* First, we compare the video quality received at the peers. Figure 3 and Figure 4 show the number of nodes that are rejected or accepted but receiving quality versions lower than desired for bit rate transcoding and frame rate transcoding, respectively. The plots are based on a 20-second window. Both transcoding schemes in PAT significantly outperform *No-Transcoding*, and *PAT-Meta* performs better than *PAT-Normal*. Overall, about 30% nodes could not receive their desired video versions in *No-Transcoding*, while it is only about 4% in *PAT-Meta*. The confidence level used here and hereafter is 95%.

Figure 5 and Figure 6 further plot the average PSNR of the received video in increasing order. To simplify evaluations, we assume a PSNR of 15 dB for nodes that are rejected when joining. For both experiments, *PAT-Normal* and *PAT-Meta* achieve better overall quality than *No-Transcoding*. Comparing the number of rejected nodes (PSNR at 15 dB), *No-Transcoding* rejects many more nodes than *PAT-Normal* and *PAT-Meta* and has a low admission rate.

Figure 7 and Figure 8 compare the client received streaming quality along time using the average quality distance. In our experiments, *quality distance* is defined as the version difference of the desired object and the received object. In these figures, the y-axis denotes the average quality distance

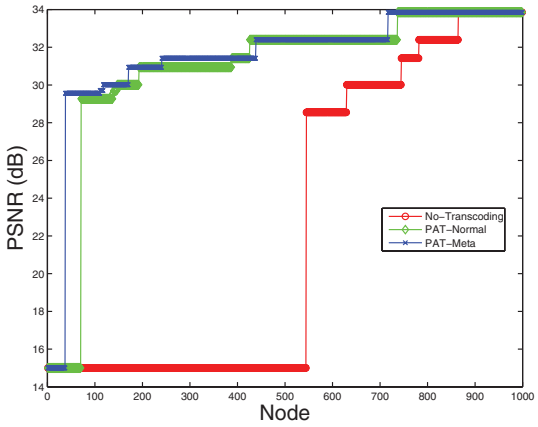

Fig. 5. Streaming quality (bit rate transcoding).

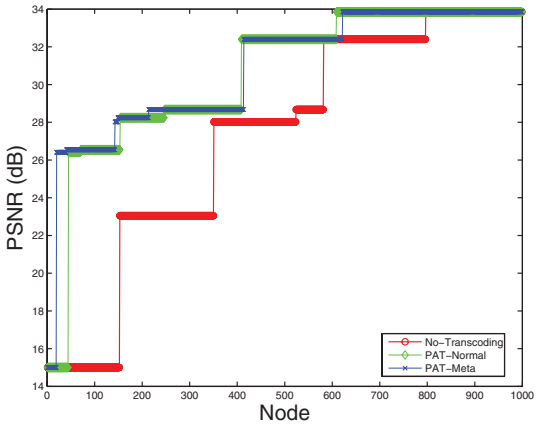

Fig. 6. Streaming quality (frame rate transcoding).

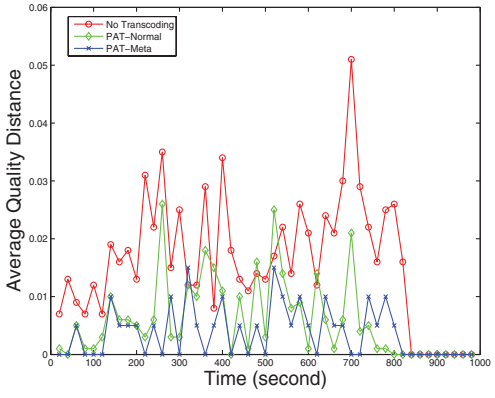

Fig. 7. Quality distance (bit rate transcoding).

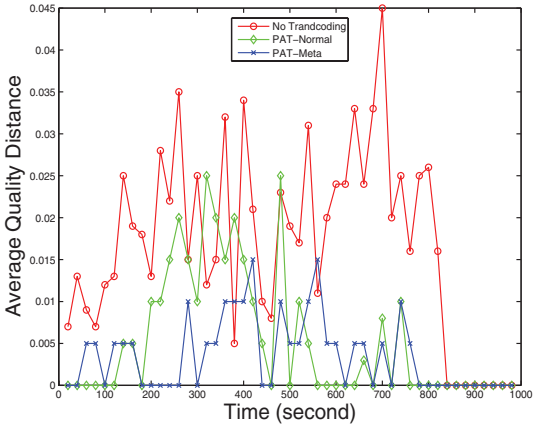

Fig. 8. Quality distance (frame rate transcoding).

Table III. Average PSNR: Client Perceived Video Quality (dB)

	No-transcoding	PAT-Normal	PAT-Meta
Bit	28.14	30.86	31.49
Frame	27.19	30.54	30.82

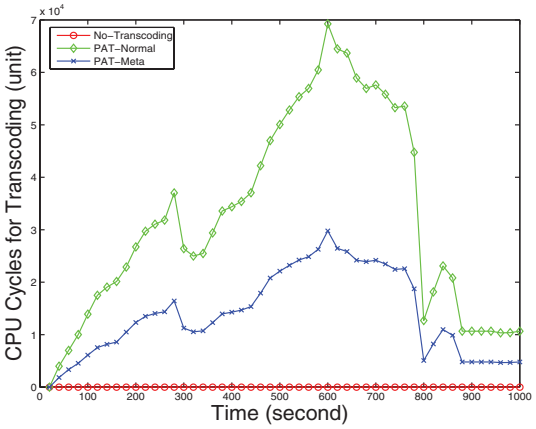

Fig. 9. CPU cycles consumed (bit rate transcoding).

of all nodes along time. Again, the plots are based on a 20-second window in time. Apparently, *No-transcoding* has the largest quality distance in both bit rate transcoding and frame rate transcoding cases. On average, *PAT-Meta* achieves better results than *PAT-Normal*.

Table III summarizes the average video quality that clients receive in two experiments. Overall, the average video quality perceived by clients in *PAT-Meta* is the best. Although *PAT-Normal* achieves similar quality gain over *No-Transcoding*, it is at the cost of much more computing overhead as shown next.

5.2.2 *Effect on CPU and Bandwidth Consumption.* Figure 9 and Figure 10 show the normalized CPU load required for bit rate and frame rate transcoding, respectively. Again, the CPU load is

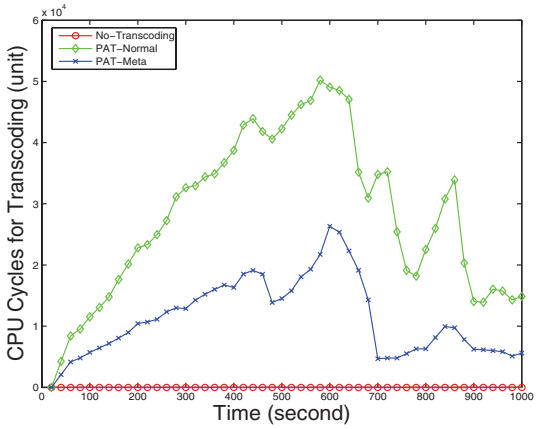

Fig. 10. CPU cycles consumed (frame rate transcoding).

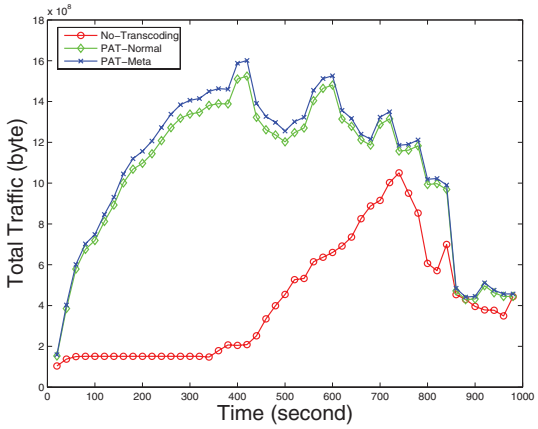

Fig. 11. Total traffic (bit rate transcoding).

computed based on a 20-second window in time. As expected, there is no CPU consumption in *No-Transcoding*. With the assistance of metadata, *PAT-Meta* significantly outperforms *PAT-Normal*. On average, *PAT-Meta* can save 58% and 39% CPU load comparing with *PAT-Normal* for bit rate transcoding and frame rate transcoding, respectively. In addition, frame rate transcoding requires generally more CPU load than bit rate transcoding, which is consistent with our previous analysis.

PAT-Meta significantly reduces the CPU load required for transcoding by sharing metadata in the metadata overlay, which leads to some traffic overhead. To evaluate this traffic overhead, Figure 11 shows the total traffic for bit rate transcoding, and Figure 12 shows the corresponding result for frame rate transcoding. The traffic amount is summed every 20 seconds. With transcoding, both *PAT-Normal* and *PAT-Meta* serve better quality video and a larger number of clients than *No-Transcoding*, leading to a significantly increased total amount of traffic over *No-Transcoding*. The difference between the total traffic in *PAT-Normal* and *PAT-Meta* mostly indicates the additional bandwidth consumed for metadata sharing, which only accounts for at most 6% of the total traffic. Jointly considering the quality improvement and the computing and traffic overhead, we can see that *PAT-Meta* achieves the best quality improvement with the least computing overhead and trivial traffic overhead.

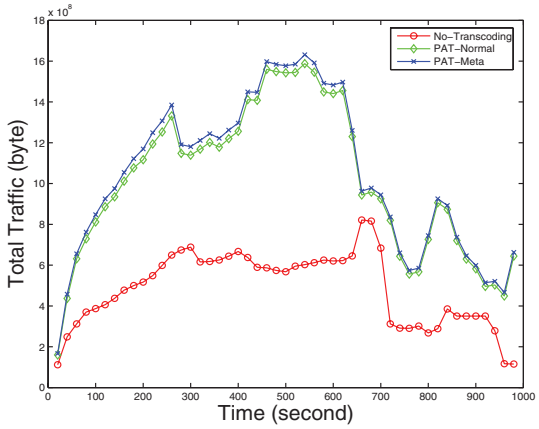

Fig. 12. Total traffic (frame rate transcoding).

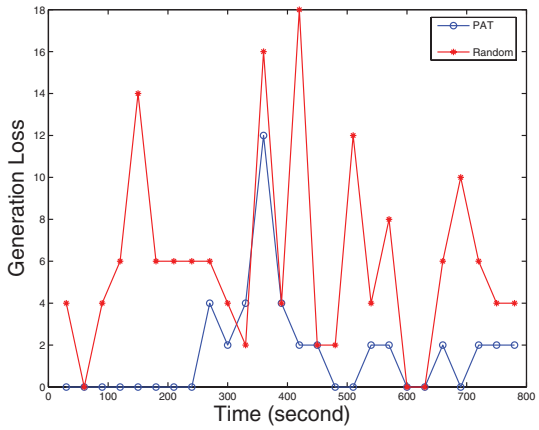

Fig. 13. Generation loss (bit rate transcoding).

5.2.3 *Effectiveness of Transcoding Parent Selection.* We also studied the effectiveness of our transcoding parent selection in our simulations. Apparently, the parent selection algorithm affects the resource consumption and video quality received by the client. As discussed in Section 3.3.1, for bit rate reduction and frame rate reduction, we have different parent selection strategies for different transcoding scenarios, aiming to reduce generation loss (and thus increase PSNR of the streaming) or CPU consumption. We compare parent selection algorithm in PAT with a random parent selection algorithm, denoted as *Random*. Figure 13 and Figure 14 show the generation loss for bit rate transcoding and frame rate transcoding, respectively. In these figures, the y-axis denotes the quality in dB of all nodes along time. Here, the generation loss is also computed on a 20-second window. As expected, PAT has better performance than that of the Random algorithm and is effective to increase the video quality under both bit rate and frame rate transcoding scenarios.

For more results under heavy load, please refer to appendix A.

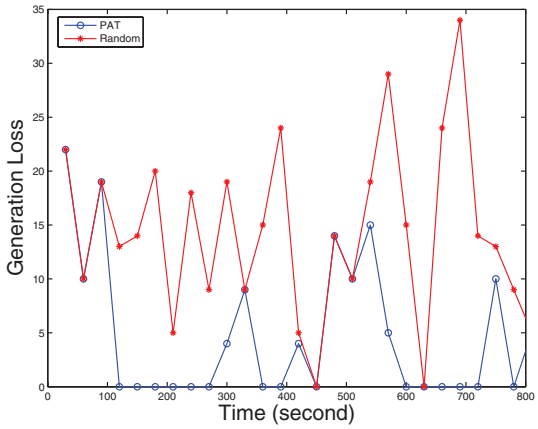

Fig. 14. Generation loss (frame rate transcoding).

6. CONCLUSION

To assist overlay streaming to a mixture of heterogeneous client devices, in this paper, we propose to build PAT that enables effective online transcoding in overlay streaming systems by leveraging node cooperation without requiring additional infrastructure support. To minimize the computing overhead, PAT effectively leverages meta-transcoding with a metadata overlay that can instantly share the metadata with other transcoding nodes. According to our throughput performance analysis, to achieve best throughput, the transcoding nodes in PAT need to select an appropriate transcoding scheme in order to maximize system throughput based on available CPU cycles and bandwidth resources at runtime. Driven by parameters obtained from practical transcoding and meta-transcoding experiments, our simulation shows that PAT improves streaming quality in heterogeneous environments with reasonable computing overhead, and in addition, metadata-assisted transcoding further reduces computing overhead with limited traffic overhead.

APPENDIX

A. EXPERIMENTAL RESULTS UNDER HEAVY LOAD

PAT has shown its effectiveness for both bit rate and frame rate transcoding under a moderate peer arrival rate with a medium number of total participating peers. In this subsection, we further evaluate its performance when both the total number of participating peers and the mean arrival rate of requests are doubled. That is, there are a total of 2000 nodes, and a mean arrival rate of 2 requests per second. Other experimental setup remains the same.

Figure 15 and Figure 16 show the number of nodes that could not be served or served with lower quality versions than desired for bit rate transcoding and frame rate transcoding, respectively. Again, it is shown that *PAT-Meta* performs the best among the three schemes.

Correspondingly, Figure 17 and Figure 18 show the accumulated number of nodes that are receiving the desired quality versions along time for bit rate transcoding and frame rate transcoding, respectively. As these figures show, *PAT-Meta* can always serve more peers with the desired quality than the other two schemes.

Figure 19 and Figure 20 show the normalized CPU load required for bit rate and frame rate transcoding, respectively. Both figures show consistent trend as we have observed before.

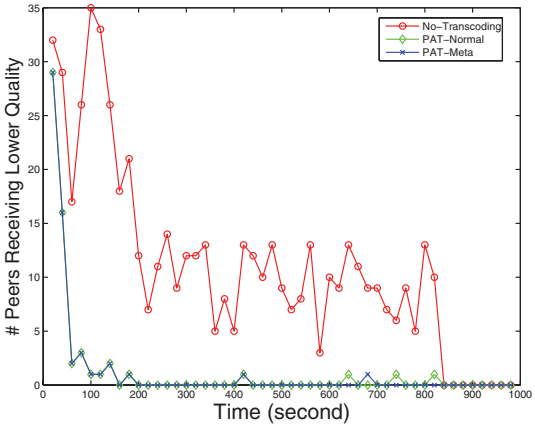

Fig. 15. Peers receiving lower quality (bit rate transcoding).

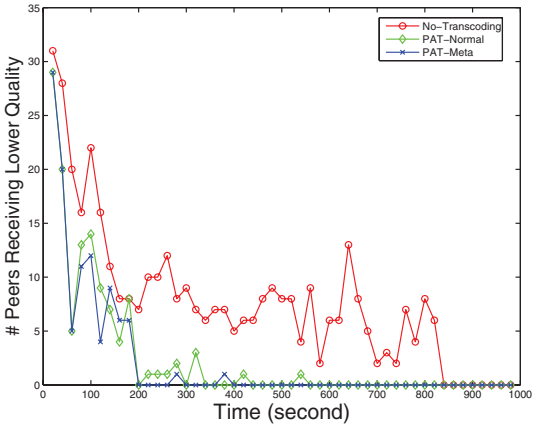

Fig. 16. Peers receiving lower quality (frame rate transcoding).

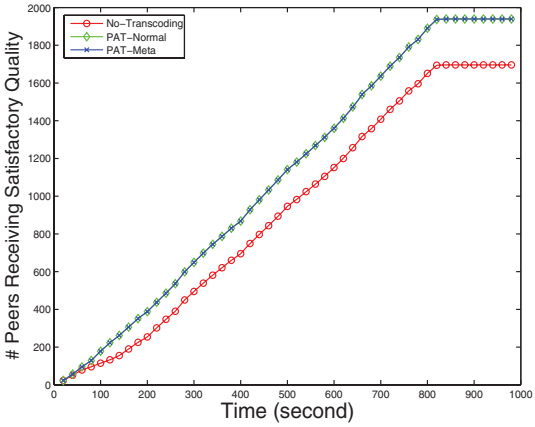

Fig. 17. Cumulative peers receiving satisfactory quality (bit rate transcoding).

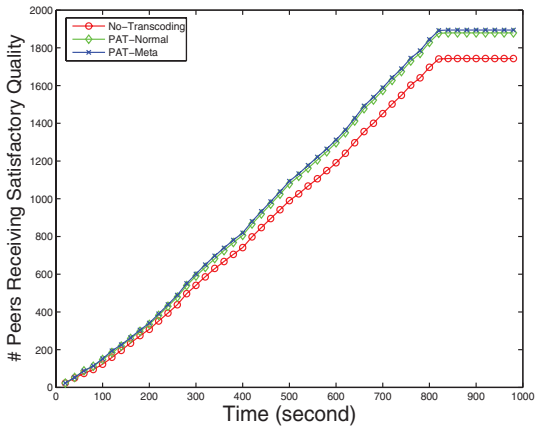

Fig. 18. Cumulative peers receiving satisfactory quality (frame rate transcoding).

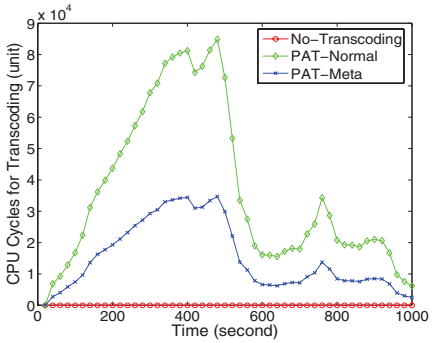

Fig. 19. CPU cycles consumed (bit rate transcoding).

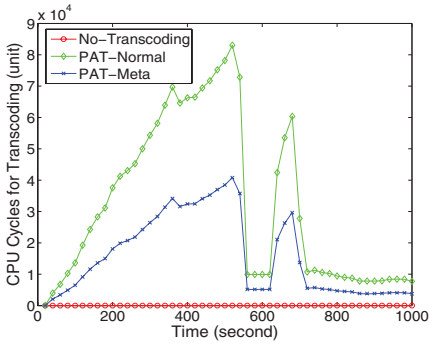

Fig. 20. CPU cycles consumed (frame rate transcoding).

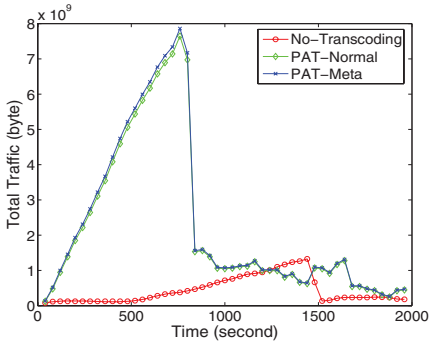

Fig. 21. Total traffic (bit rate transcoding).

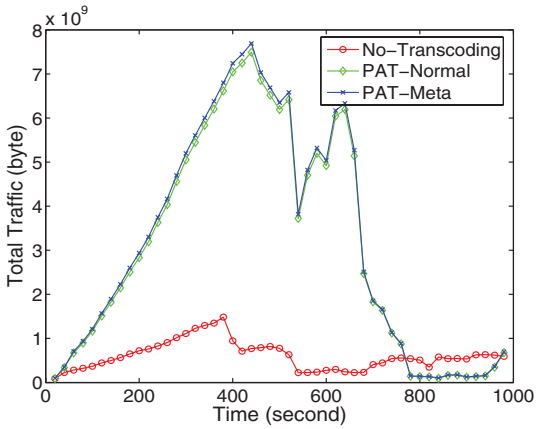

Fig. 22. Total traffic (frame rate transcoding).

Figure 21 further shows the total traffic for bit rate transcoding, and Figure 22 shows the corresponding result for frame rate transcoding. It is shown that the extra traffic overhead of *PAT-Meta* is trivial compared to that of *PAT-Normal*.

ACKNOWLEDGMENTS

We thank Associate Editor Prof. Roger Zimmermann and the anonymous reviewers for constructive comments.

REFERENCES

3G. http:/www.itwire.com.aul content/view/5383/127/.

ACHARYA, S. AND SMITH, B. C. 2000. Middleman: A video caching proxy server. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

AMIR, E., MCCANNE, S., AND ZHANG, H. 1995. An application level video gateway. In *Proceedings of ACM Multimedia*.

CASTRO, M., DRUSCHEL, P., KERMARREC, A., NANDI, A., ROWSTRON, A., AND SINGH, A. 2003. Splitstream: High-bandwidth content distribution in a cooperative environment. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*.

CHU, Y., RAO, S., AND ZHANG, H. 2000. A case for end system multicast. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*.

COMSCORE. http://www.comscore.com/.

GHANBARL, M. 1989. Two-layer coding of video signals for vbr networks. *IEEE J. Select. Areas Comm. 7*.

HEFEEDA, M., HABIB, A., BOTEV, B., XU, D., AND BHARGAVA, B. 2003. Promise: Peer-to-peer media streaming using collectcast. In *Proceedings of ACM Multimedia*.

HESS, C. K., RAILA, D., CAMPBELL, R. H., AND MICKUNAS, D. 2000. Design and performance of mpeg video streaming to palmtop computers. In *Proceedings of the SPIE/ACM Annual Multimedia Computing and Networking Conference*.

KOSTIC, D., RODRIGUEZ, A., ALBRECHT, J., AND VAHDAT, A. 2003. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of the ACM Symposium on Operating Systems Principles*.

KOUVELAS, I., HARDMAN, V., AND CROWCROFT, T. 1998. Network adaptive continuous-media applications through self organized transcoding. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

LIU, D., CHEN, S., AND SHEN, B. 2006. Amtrac: Adaptive meta-caching for transcoding. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

LIU, D., LI, F., AND CHEN, S. 2009. Towards optimal resource utilization in heterogeneous p2p. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*.

LIU, D., SETTON, E., SHEN, B., AND CHEN, S. 2007. PAT: Peer-assisted transcoding for overlay streaming to heterogeneous devices. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

LIU, X., JIN, H., LIU, Y., NI, L., AND DENG, D. 2006. Anysee: Peer-to-peer live streaming. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.

MAGHAREI, N. AND REJAIE, R. 2007. Prime: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.

MOBILE USERS. http://www.nielsenmobile.com/documents/CriticaIMass.pdf.

MYSEE. http://www.mysee.com.

NAKAMURA, M. AND SAWADA, K. 1995. Scalable coding themes based on dct and mc prediction. In *Proceedings of the IEEE International Conference on Image Processing*.

NETWORK SIMULATOR. http://www.isLedu/nsnam/ns.

NIELSEN MOBILE. http://www.nielsenmobile.com/.

OOI, W. 2005. Dagster: Contributor-aware end-host multicast for media streaming in heterogeneous environment. In *Proceedings of the ACM/SPIE Annual Multimedia Computing and Networking Conference*.

PADMANABHAN, V., WANG, H., AND CHOU, P. 2003. Resilient peer-to-peer streaming. In *Proceedings of the IEEE Annual International Conference on Network Protocols*.

PADMANABHAN, V., WANG, H., CHOU, P., AND SRIPANIDKULCHAI, K. 2002. Distributing streaming media content using cooperative networking. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*.

PAI, V., KUMAR, K., TAMILMANI, K., SAMBAMURTHY, V., AND MOHR, A. 2005. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of the 4th International Parallel and Distributed Processing Symposium*.

PPLIVE. http://www.pplive.com.

PPSTREAM. http://www.ppstream.com/.

SHEN, B. 2003. Meta-caching and meta-transcoding for server side service proxy. In *Proceedings of the IEEE International Conference on Multimedia and Expo*.

SHEN, B., LEE, S., AND BASU, S. 2004. Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. *IEEE Trans. Multimedia 6*, 375–386.

SMALL, T., LIANG, B., AND LI, B. 2006. Scaling laws and tradeoffs in peer-to-peer live multimedia streaming. In *Proceedings of ACM Multimedia*.

SOPCAST. http://www.sopcast.org/.

TRAN, D., HUA, K., AND DO, T. 2003. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.

UUSEE. http://www.uusee.com.

WIRELESS. http://www.ieee802.org/11/.

Received February 2010; revised May 2010, August 2010; accepted October 2010