PAT: Peer-Assisted Transcoding for Overlay Streaming to Heterogeneous Devices

Dongyu Liu[1], Eric Setton[2], Bo Shen[2], and Songqing Chen[1]
[1]Dept. of Computer Science
George Mason University
Fairfax, VA 22030
{dliu1, sqchen}@cs.gmu.edu
[2] Mobile and Media Systems Lab
Hewlett-Packard Laboratory
Palo Alto, CA 94304
{eric.setton, bo.shen}@hp.com

ABSTRACT

With the increasing deployment of Internet P2P/overlay streaming systems, more and more clients use mobile devices, such as smart phones, PDAs, to access these Internet streaming services. Compared to wired desktops, mobile devices normally have smaller screen size, 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 a transcoding server. Second, as online transcoding is computationally intensive while the diverse devices used by participating nodes 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

H.4 [**Information Systems**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

P2P/Overlay Streaming, Meta-Transcoding, Heterogeneity

1. INTRODUCTION

NOSSDAV '07 Urbana, Illinois USA

In recent years, the amount of Internet media traffic has increased exponentially. According to ComScore [1], from October 2005 to March 2006, the Internet video traffic doubled every 3 to 4 months, and the number of users watching videos online increased by 18% monthly. To efficiently deliver Internet media objects, a number of different P2P/overlay streaming systems have been proposed [6, 10, 12, 14, 16, 17, 18]. In practice, PPLive [2] regularly provides access to more than 400 TV channels daily and has been used by hundreds of thousands of users.

Recent technology advances in wireless and 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 Telephia [3], more than 34.6 million mobile subscribers accessed the Internet via wireless devices in June 2006 and these 34.6 million subscribers account for 17% of the total 206 million US mobile phone subscribers.

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 to a desktop cannot be directly rendered and displayed on a PDA or a smart phone.

Although some existing studies have considered that participating nodes may have different capabilities in uploading bandwidth in an overlay streaming system (e.g., [18]), 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 [7, 13] or multiple description coding [16] may not be effective given the fact that most devices do not support these codecs. On the other hand, in the context of 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, we propose PAT (Peer-Assisted Transcoding) to facilitate P2P/overlay streaming in heterogeneous environments. The PAT scheme relies on node cooperation without demanding infrastructure support. Media adaptation is carried on by real-time online transcoding at the 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 [4, 5, 8]. Most such schemes treat the transcoding procedure as a black box with the final transcoded result as the only caching candidate. In our previous study [11], we have introduced the idea of meta-caching. Meta-caching identifies intermediate transcoding steps from which certain intermediate results (called *metadata*) can be cached and re-used later so that a fully transcoded object version can be produced from the metadata with a relatively smaller amount of CPU cycles. Meta-caching demonstrates an effective trade-off between the storage and CPU load. Such metadata-assisted transcoding is called meta-transcoding.

This meta-transcoding approach is also 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. This approach instantly shares the intermediate result of a transcoding procedure with other transcoding nodes, aiming to minimize the total computing overhead in the system.

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

To the best of our knowledge, this work is the first to combine transcoding and P2P/overlay streaming. We believe our proposed scheme is generic and it can be used to design both tree and mesh based transcoding assisted overlay streaming systems. In this paper, for illustrative purposes, the tree-based overlay streaming system is used as a base to introduce the scheme.

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 4. We make conclusion remarks in section 5.

2. RELATED WORK

Research on P2P/overlay streaming systems has attracted considerable attention [2, 6, 16]. Padmanabhan et al. [16] use multiple distribution trees and multiple description coding (MDC) to provide redundancy to the system and enhance robustness of the content distribution. Splitstream [6] 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 [17] 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. [18] theoretically investigate the scaling law of P2P live multimedia streaming and their results can have practical implications when constructing tree topologies. A scalable architecture is proposed for congestion controlled multicast real-time communication by using self organized transcoder in [9]. Dagster [15] proposes a novel incentive scheme to encourage nodes to contribute more bandwidth to the whole system. Nodes in Dagster are assumed to be able to conduct transcoding during broadcasting. However, it is not clear how a peer could efficiently conduct transcoding at runtime.

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 base overlay, which is used for streaming data transmission. The other is the metadata overlay, which is used to instantly share metadata, the intermediate results of a transcoding procedure, to efficiently minimize the total overhead for online transcoding by trading a small amount of bandwidth for a large saving on CPU cycles in the system.

3.1 Meta-transcoding

In [11], we have proposed meta-caching to effectively balance the resources used for online transcoding between the CPU cycles and the limited storage space in a transcoding proxy or server. The motivation of meta-caching is as follows. Many existing schemes that aim to optimize transcoding treat the transcoding procedure itself 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*, may be saved to avoid re-computing the same result again for a later transcoding request. If the saved metadata only consumes a small storage size while it takes a significant computing load 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 re-computed each time data is transcoded. In this case, meta-transcoding requires significantly less CPU cycles to produce the new transcoded object.

In this paper, we combine this approach with P2P/overlay streaming, where a media stream is spread to a large population of clients, utilizing the forwarding capacity of the peers. In addition, a node can also instantly share some intermediate transcoding results with other nodes performing same transcoding. This approach reduces the CPU overhead in the system. Clearly, additional bandwidth is required for metadata sharing. In the following context, we propose a protocol that can reduce computing load with small bandwidth overhead.

3.2 Meta-data Matrix

We assume there is a single root (the source of the stream). In addition to providing the original streaming data, the root also maintains a small matrix. The size of this matrix is determined by the number of possible stream qualities. For example, if the video to be streamed has different qualities from 1 to n, then the matrix is a $n \times n$ matrix. Each element in the matrix indicates availability of the metadata corresponding to transcoding from version i to j. By definition, the metadata matrix, denoted as $mMatrix$, is an upper triangle matrix, since it is rare in practice to transcode a

lower quality version to a higher quality version. Figure 1 shows an example of an mMatrix.

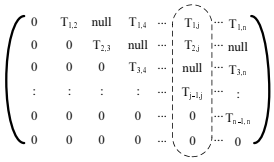

Figure 1: Example of an mMatrix.

When the corresponding metadata is not available, $T_{i,j}$ in mMatrix is `null`. Otherwise, such metadata is available and $T_{i,j}$ denotes the ID of the node which transcodes the stream from version i to version j. We also call this node as the *meta-head* and there may be other transcoding nodes that receive metadata from a meta-head to perform meta-transcoding.

3.3 Base Overlay and Meta-data Overlay Construction

The construction of base 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 traditional ones and also affects the construction of the metadata overlay. The node departure in PAT also affects both the base overlay and the metadata overlay.

3.3.1 Node Arrival

Assume the system starts with a source node and several bootstrap nodes providing the streaming service. When a node wants to join the streaming service, it first contacts the source node or a bootstrap node with its quality requirement. In this study, 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. The algorithm is 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 the information on the source node or a bootstrap node. If there are such candidate parents, the joining node selects one of them with the smallest height in the tree. In this case, only the data-tree is updated.

- **Case 1.2 – transcoding required:** If the desired object version does not exist or if the node having the desired version can't 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**: 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, the joining node starts to send the joining request to the meta-head 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 only for bit rate reduction. This policy is designed to minimize the quality degradation since the transcoding between versions that are further away (i.e., less number of intermediate versions) from each other in bit rate leads to less generation loss. 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. As P needs to conduct meta transcoding, metadata subtree $T_{t,j}$ is updated as follows: the meta-head of $T_{t,j}$ looks for a parent for P. If the meta-head 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 in 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, and repeat the joining process. If all the `non-null` elements in the jth column of mMatrix have been explored without a parent node being identified, go to the next step.

  – **Step 2 – metadata unavailable**: If a parent is not found in the previous step, the joining node must look for a parent with enough bandwidth and CPU resource for transcoding. 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 base overlay, P also needs to join the metadata overlay. Sup-

pose 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 will become the meta-head of the $T_{k,j}$ subtree and the $T_{k,j}$ in mMatrix is updated.

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

– **Step 3** If version j is not the lowest quality version, go back to Step 1 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, participating nodes may depart at any time. Upon node departure, the overlay must be re-adjusted to adapt to the change. Such adjustments are performed as follows.

- **Case 2.1:** If the departing node is a leaf node and is receiving version j, denoting the parent of the departing node as P, (1) if P is not a transcoding node, remove the departing node from P's children list and update its 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* algorithm. If P is the meta-head in the metadata tree, it tries to find a new meta-head from its children 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 meta-head 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`. Otherwise, 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 will become the children of the parent of P in the metadata subtree if the parent of P has enough bandwidth. They will contact the meta-head to rejoin the metadata tree if the parent of P does not have enough bandwidth. The corresponding children and parent fields are updated in these nodes.

- **Case 2.2:** If the departing node is neither a leaf node nor a transcoding node, the children of the departing node need to find a new parent. If the parent of the departing node has enough bandwidth, it becomes their new parent. Otherwise, the children of the departing node check their backup parents. If the backup parents cannot accept these nodes as children, the children of the departing node need to perform a rejoin process as described in Section 3.3.1.

- **Case 2.3:** If the departing node 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 the departing node D is in the metadata tree $T_{i,j}$ and the children of node D need version j. All children of D in both the metadata tree and the data tree need to find new parents. In the metadata tree, node D departs the meta-data tree according to the *META-LEAVE* algorithm 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 based on their available bandwidth. If the metadata tree $T_{i,j}$ is `null` or no node is available, the peers need to rejoin the multicast session. The node has a chance to subscribe to a node that can directly provide version j within the data tree.

4. 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.

4.1 Experimental Parameter Capturing and Setup

In our implemented transcoder, we conduct both bit rate transcoding and frame rate transcoding. A 1000-second video with a data rate of 500 kb/s and 30 frame/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 qualities 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 unit and 0.36 unit, respectively. For meta-transcoding, the corresponding CPU loads are 0.5, 0.27, and 0.19 unit 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 bandwidth capacity for the rest of the nodes is as follows: 5Mb/s (11%), 2Mb/s (3%), 896kb/s (9%), 384kb/s (21%) and 256kb/s (56%). Assuming the version index 1 representing the original version with increased index indicating lower quality versions, we further dictate the nodes with 5Mb/s and 2Mb/s connections to randomly request versions 1 to 3, and nodes with other bandwidth capacities to randomly request versions 1 to 5. The CPU constraints of nodes randomly are distributed from 1 unit to 2 units. The system starts with one source node capable of serving all versions and 1000 nodes dynamically join and

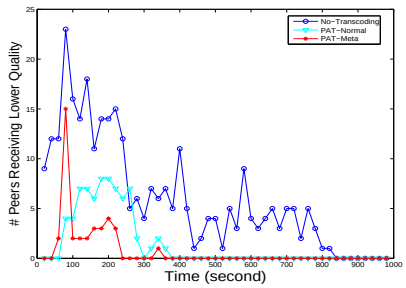

Figure 2: Peers receiving lower quality in bit rate transcoding

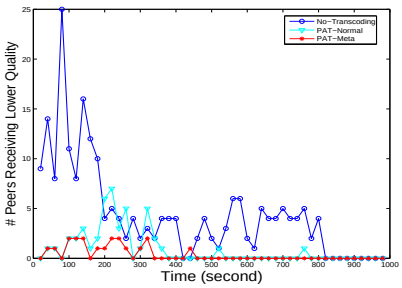

Figure 3: Peers receiving lower quality in frame rate transcoding

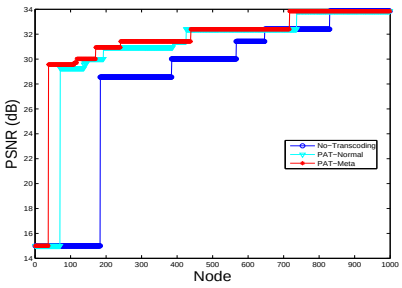

Figure 4: Streaming quality in bit rate transcoding

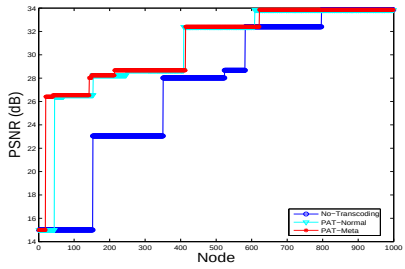

Figure 5: Streaming quality in frame rate transcoding

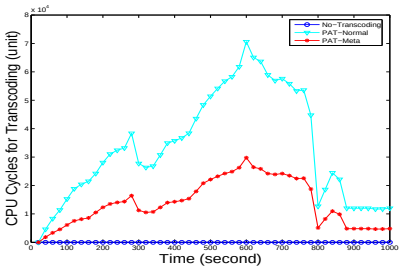

Figure 6: CPU cycles consumed in bit rate transcoding

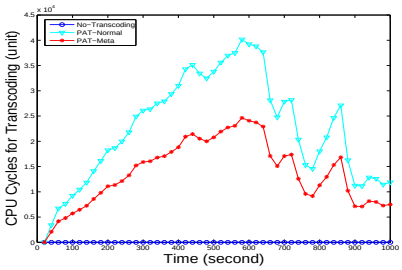

Figure 7: CPU cycles consumed in frame rate transcoding

depart from the streaming service. 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.

4.2 Experimental Results

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.

4.2.1 Effect on the Client Received Video Quality

First, we compare the video quality received at the peers. Figure 2 and Figure 3 show the number of nodes that are rejected or accepted but receiving lower than desired quality versions as a function of time for bit rate transcoding and frame rate transcoding, respectively. The plots are based on a 20-second window in time. Both transcoding schemes in PAT significantly outperform *No-Transcoding*, and *PAT-Meta* performs better than the *PAT-Normal* scheme. Overall, about 30% nodes could not receive their desired video qualities in *No-Transcoding*, while it is only about 4% in *PAT-Meta*.

Figure 4 and Figure 5 further plot the average PSNR of the received video at each node sorted in increasing PSNR values. 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 much more nodes than *PAT-Normal* and *PAT-Meta*.

Table 1 summarizes the average video quality that clients receive in two experiments. Overall, the average video quality experienced 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.

Table 1: 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

4.2.2 Effect on CPU and Bandwidth Consumption

Figure 6 and Figure 7 show the normalized CPU load required for bit rate and frame rate transcoding, respectively. Again, the CPU load is computed based on a 20-second window in time. As expected, there is no CPU consumption for transcoding 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 8 shows the total traffic for bit rate transcoding, and Figure 9 shows the corresponding result

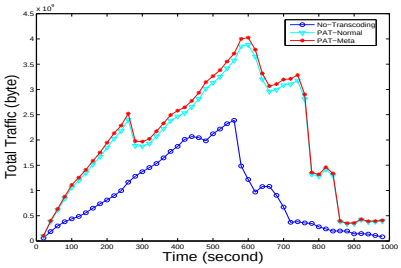

Figure 8: Total traffic in bit rate transcoding

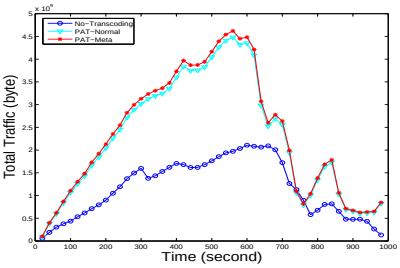

Figure 9: Total traffic in frame rate transcoding

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 improvements and the computing and traffic overhead, we can see that *PAT-Meta* achieves the best quality improvement with least computing overhead and negligible traffic overhead.

5. CONCLUSION

To assist overlay streaming to a mixture of 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 the meta-transcoding with a metadata overlay that can instantly share the metadata with other transcoding nodes. Driven by parameters obtained from practical transcoding and meta-transcoding schemes, our simulation shows the effectiveness of our proposed scheme.

In PAT design, we assume the peers in the system are trusted and cooperative. We plan to investigate the security issues when untrusted/malicious peers present.

6. ACKNOWLEDGMENTS

We would like to thank William L. Bynum and anonymous reviewers for their helpful comments on this paper. The work is supported by NSF grants CNS-0509061 and CNS-0621631.

7. REFERENCES

[1] Comscore. http://www.comscore.com/.

[2] PPLive. `http://www.pplive.com`.

[3] Wireless access. `http://wapreview.com/blog/?p=152`.

[4] S. Acharya and B. C. Smith. Middleman: A video caching proxy server. In *Proceedings of ACM NOSSDAV*, Chapel Hill, NC, May 2000.

[5] E. Amir, S. McCanne, and H. Zhang. An application level video gateway. In *Proceedings of ACM Multimedia*, San Francisco, CA, Nov. 1995.

[6] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth content distribution in a cooperative environment. In *Proceedings of 2nd IPTPS*, Berkeley, CA, Feb. 2003.

[7] M. Ghanbari. Two-layer coding of video signals for vbr networks. In *IEEE Journals on Selected Areas in Communications*, volume Vol 7, Jun. 1989.

[8] C. K. Hess, D. Raila, R. H. Campbell, and D. Mickunas. Design and performance of mpeg video streaming to palmtop computers. In *Proceedings of SPIE/ACM MMCN*, San Jose, CA, Jan. 2000.

[9] I.Kouvelas, V.Hardman, and J.Crowcroft. Network adaptive continuous-media applications through self organised transcoding. In *Proceedings of ACM NOSSDAV*, Cambridge, UK, Jul. 1998.

[10] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of ACM SOSP*, Bolton Landing, NY, Oct. 2003.

[11] D. Liu, S. Chen, and B. Shen. Amtrac: Adaptive meta-caching for transcoding. In *Proceedings of ACM NOSSDAV*, Newport, RI, May 2006.

[12] M.Hefeeda, A.Habib, B.Botev, D.Xu, and B.Bhargava. Promise: Peer-to-peer media streaming using collectcast. In *Proceedings of ACM Multimedia*, Berkeley, CA, Nov 2003.

[13] M. Nakamura and K. Sawada. Scalable coding schemes based on dct and mc prediction. In *Proceedings of IEEE ICIP*, Washington, DC, 1995.

[14] N.Magharei and R.Rejaie. Prime: Peer-to-peer receiver-driven mesh-based streaming. In *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[15] W. Ooi. Dagster: Contributor-aware end-host multicast for media streaming in heterogeneous environment. In *Proceedings of ACM/SPIE MMCN*, San Jose, California, Jan. 2005.

[16] V. Padmanabhan, H. Wang, and P. Chou. Resilient peer-to-peer streaming. In *Proceedings of IEEE ICNP*, Atlanta, GA, Nov. 2003.

[17] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proceedings of 4th IPTPS*, Ithaca, NY, Feb. 2005.

[18] T. Small, B. Liang, and B. Li. Scaling laws and tradeoffs in peer-to-peer live multimedia streaming. In *Proceedings of ACM Multimedia*, Santa Barbara, CA, Oct. 2006.