Streaming To Mobile Users In A Peer-to-Peer Network

Jeonghun Noh
Stanford University
350 Serra Mall
Stanford, CA, USA
jhnoh@stanford.edu

Mina Makar
Stanford University
350 Serra Mall
Stanford, CA, USA
mamakar@stanford.edu

Bernd Girod
Stanford University
350 Serra Mall
Stanford, CA, USA
bgirod@stanford.edu

ABSTRACT

Streaming video to mobile users is rapidly emerging as a crucial multimedia service. One of the stumbling blocks in mobile streaming is the heterogeneity found in mobile devices: diverse display size, computing power, memory, and media capabilities. Given this heterogeneity, video transcoding is often required to satisfy the requirements of different mobile users. In this paper, we propose a peer-to-peer (P2P) method for mobile streaming. In the proposed method, peers other than mobile users, called *fixed nodes*, contribute their computing power for transcoding. We further propose the *interleaved distributed transcoding* (IDT) scheme that allows multiple fixed nodes to perform transcoding for a mobile device. Distributed transcoding not only lowers computation at fixed nodes, but also achieves error resilience against packet loss. The IDT scheme conforms to the H.264/AVC baseline profile, which requires no modification to decoders found in many mobile phones. We analyzed the effect of distributed transcoding under peer churn. Extensive simulations show that mobile streaming based on the proposed scheme is robust to packet loss due to peer churn and adverse wireless channel conditions.

Categories and Subject Descriptors

C.2 [**Computer-communication networks**]: Distributed systems

Keywords

Peer-to-peer network, video streaming, transcoding, mobile device, heterogeneity

1. INTRODUCTION

Live streaming to mobile devices, such as cellular phones and Personal Digital Assistants (PDAs), is a challenging task especially due to their heterogeneity: they differ in display size, main memory, processor, media capability, and network access technology. In typical streaming systems, live media adaptation is performed to meet the requirements of heterogeneous mobile users. For video, media adaptation is often achieved by *video transcoding* [1–3]. Video transcoding converts an original video bitstream to a new bitstream for a different encoding standard, smaller spatial resolution, reduced frame rate, or reduced quality (due to coarser quantization). However, transcoding poses a considerable computational burden on the streaming server because mobile devices often require individually customized transcoding.

In this paper, we propose a peer-to-peer (P2P) streaming method for mobile devices [4]. P2P streaming is a potentially cost-effective alternative to server-based streaming. In P2P systems, users not only consume media contents, but also contribute their uplink bandwidth or local storage. Thus the system can scale well as users bring resources in the system. In this study, we refer to personal computers connected to the network over a wired connection as *fixed nodes*, as opposed to mobile nodes. By harnessing the processing power of the fixed nodes, the transcoding burden of the servers can be reduced or eliminated. On the other hand, due to their limited resources (battery, uplink speed), mobile nodes in the proposed P2P system are treated as *leeches*, i.e., peers that only receive packets but do not relay the packets to other peers.

P2P-based streaming systems often suffer from *peer churn*: when peers leave the system without prior notice, other peers who are connected to the departing peers may experience temporary video disruption and/or disconnection from the system. To address this problem, we propose a distributed transcoding scheme that allows fixed nodes to perform transcoding for a mobile device. After a mobile device connects to multiple fixed nodes as its parents, each parent generates a substream by transcoding the original video. These substreams are transmitted and then assembled at the mobile device as if they were a single stream. If the mobile device loses some of its parents, it still receives substreams from the other parents and decodes the incoming video partially with graceful degradation. In addition, the proposed transcoding scheme distributes transcoding overhead to multiple fixed nodes. We implement the proposed distributed transcoding scheme in a way that conforms to the H.264/AVC baseline profile. This allows any H.264/AVC video decoders to decode the video produced by the proposed scheme.

The remainder of the paper is structured as follows. Section 2 presents the work related to streaming video to mobile devices. Section 3 describes the proposed distributed transcoding and its implementation. In Section 4, we ana-

MOBIMEDIA '09, September 7-9, 2009, London, UK

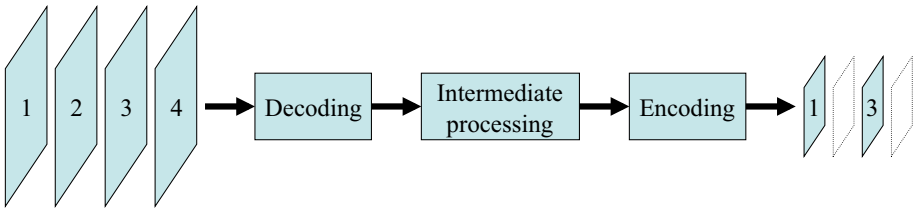

Figure 1: A cascaded transcoder. An original video stream flows into the decoding unit. The intermediate processing unit transforms the decoded stream by performing frame rate conversion, downsampling, cropping, or any other preprocessing before encoding. The output of the encoder is the transcoded bitstream.

lyze the effect of distributed transcoding against peer churn. In Section 5, we provide simulation results.

2. RELATED WORK

Safak et al. propose a server-based adaptive video transcoding system for mobile users in [2]. A video proxy, located at the edge of two or more networks, adaptively transcodes video considering the network conditions and constraints of mobile users in the GPRS network. Warabino et al. also propose a server-based video transcoding scheme for mobile users in [1]. The client proxy collects mobile client profiles and request transcoding to the video proxy. Transcoded videos are then sent back to the client proxy, and the client proxy relays them to each client. Although server-based systems [1,2] are more reliable than P2P-based systems, servers can easily become a bottleneck as the number of mobile users increases.

In [5], several multicasting schemes for streaming to mobile users are surveyed. The authors compare solutions that extend IP multicast to support mobile users. Since IP multicast trees are constructed at the network layer, these solutions often focus on the reduction of the overhead associated with the reconfiguration of the trees due to the mobility of mobile users. The authors then suggest a combination of application layer and network layer multicast solutions.

P2P-based video-on-demand (VoD) mobile streaming was investigated by the authors in [6]. In the system they proposed, the mobile user establishes a one-to-one connection to a proxy peer chosen in the P2P network. Through that proxy peer, the mobile user searches for a pre-encoded video stored somewhere in the P2P network and downloads it. However, there is no discussion in [6] on the needs for video transcoding and error resilience in case of proxy peer failure.

The contribution [7] also studied P2P-based VoD streaming. When a video is downloaded from several source peers, media transcoding is performed by multiple peers to meet the requirements of the destination peer. To combat peer churn, the video quality is adapted by the transcoders based on the feedback from the destination peer. However, no discussion is provided regarding the way video transcoding is performed at multiple peers and how distributed transcoding and peer churn affects the video quality quantitatively.

In [8], a fully collaborative P2P streaming system is proposed. Instead of all users, only a few mobile users pull a video from the video server through base stations. The pulled video is then shared with other neighboring users via a free broadcast channel, such as Wi-Fi or Bluetooth. The proposed system is shown to be scalable with the number of users and is shown to reduce cellular bandwidth usage. However, it works well only when a sufficient number of users watching the same video belong to the same base station, and the neighboring users are physically close to each other to lessen power consumption incurred while the video is being relayed among peers.

3. VIDEO TRANSCODING

In this section, we describe how video transcoding is performed at one or multiple locations for a mobile node in our P2P system. We consider a P2P system that consists of fixed nodes and mobile nodes; fixed nodes are peers that receive and consume the original video emanating from the video source. Mobile nodes are peers that cannot receive the original video due to limited downlink bandwidth, or/and cannot consume the original video due to limited video decoding capabilities. Thus, it is desirable to perform video transcoding to adapt the original video to mobile nodes. Fixed nodes perform transcoding to adapt the original video according to the individual requirements of each mobile node. In this study, we adopt the cascaded transcoding scheme shown in Fig. 1

3.1 Interleaved distributed transcoding

The multiple parent approach is a popular solution to provide robustness to peer churn [9–11]. In this approach, a peer has multiple peers as parents. When a parent disappears, only a subset of video packets are lost, which allows for graceful video degradation. We propose a *distributed transcoding* scheme that allows multiple fixed nodes to perform transcoding for a mobile device. A mobile node selects multiple fixed nodes as parents and the parents perform transcoding collaboratively. A straightforward scheme would be that each parent transcodes the entire original video and delivers a disjoint substream of it to a mobile node. When one of its parents disconnects, the mobile node asks for retransmissions of the missing substream from other parents. However, such a straightforward scheme needlessly wastes computing power at the parents.

To reduce processing redundancy, yet achieve robustness with multiple parents, we propose *interleaved distributed transcoding* (IDT), which is illustrated in Fig. 2. In the illustration, $K = 4$ parents are generating transcoded substreams with a Group of Pictures (GOP) of 12 frames. The original GOP size is assumed to be larger than 14 in Fig. 2. This illustration demonstrates that the GOP size of the transcoded stream can be selected independently of the one of the original stream. In transcoding, the original video frames are first decoded. In this illustration, the decoded bitstream is downsampled to smaller frames in the spatial

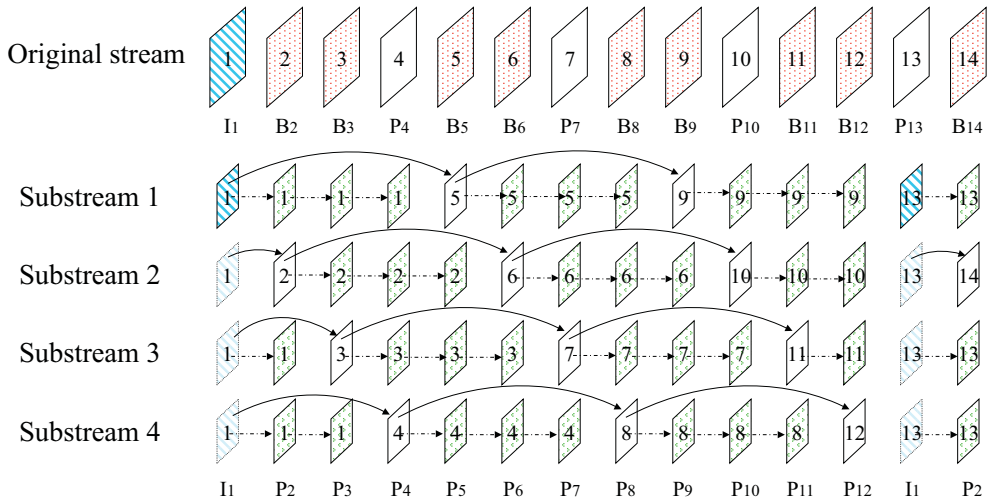

Figure 2: Interleaved distributed transcoding: an example of 4 parents (IDT encoders) with a GOP of 12 frames. The original video stream is divided into 4 substreams. The first frame of a GOP is encoded as an I frame by all parents. To avoid duplicate transmission, only Parent 1 transmits I frames. The I frames that are not transmitted are depicted with dotted boundaries. The solid arrows represent coding dependency. The dotted arrows depict the frame copy operation. The original stream is downsampled before encoding.

domain. The first frame in a GOP is coded as an I frame, and each following frame is coded as a P frame predicted from the frame immediately preceding it in the substream. Parent i codes Substream i, which includes Frame i, $K+i$, $2K+i$, ... , and each parent transmits every K^{th} frame in a disjoint manner. The I frames are encoded and used in prediction by all parents, yet transmitted by only Parent 1 to avoid duplicate transmission. Note that B frames can be employed within each substream to achieve higher coding gain.

The proposed distributed transcoding scheme achieves robustness against peer churn and distributes transcoding workload among multiple fixed nodes. The incurred cost is the redundancy in the transcoding bitstream due to lower temporal correlation between video frames, which will be examined in Section 5 in detail.

3.2 Implementing IDT

We implement the interleaved distributed transcoding (IDT) scheme in such a way that no decoder modification is required. The IDT generates no B frames and utilizes multiple reference frames for encoding P frames. This ensures that any decoder conforming to the H.264/AVC baseline profile can decode transcoded bitstreams. We refer to the H.264/AVC encoder that implements the proposed interleaved distributed transcoding as the *IDT encoder*. Suppose that K parents are involved in transcoding. The IDT encoders at the parents encode the first frame in a GOP as an I frame, which is identical across all the encoders. The remaining frames in a GOP are encoded as P frames. To encode Frame n as a P frame, Frame $n-K$, the previously encoded frame in the same substream, is used as a reference frame for motion-compensated prediction. Therefore, the IDT encoder is required to store K previously encoded frames. For this, we take advantage of the *multiple reference picture motion compensation* specified in the H.264/AVC baseline profile. It allows the short-term reference picture buffer to hold multiple reference pictures, in our case, K previously encoded frames.

We also employ the *reference picture reordering* specified in the H.264/AVC baseline profile to ensure the correct frames are used as a reference picture for motion prediction. The H.264/AVC standard provides the *SKIP mode*, in which the current macroblock (MB) is a copy of the same MB in the previous frame with a motion vector that is the median of the motion vectors in the neighboring MBs. For the *SKIP mode*, the most recent frame in the reference picture buffer is always used as a reference. To allow the *SKIP mode* to work correctly, we move the previous frame in a substream to the front of the picture buffer by *reference picture reordering*. The encoder uses only the most recent frame although there may be up to K pictures available in the buffer.

When the IDT encoder encodes every K^{th} frame, the remaining frames are encoded as an exact copy of the previously encoded frame. In Fig. 2, the IDT encoder at Parent 1 encodes Frames P_2, P_3 and P_4 as a copy of Frame 1. *Frame copy* encodes frames with negligible computational complexity at the cost of about $1 \sim 2$ % of control bits[1] added to the transcoded video. *Frame copy* not only avoids encoding unnecessary frames, but also embeds control bits for error concealment in the bitstream. This allows error concealment to be done at the bitstream level.

3.3 Decoding transcoded video

Substreams generated by multiple parents are transmitted to the destination mobile node. In Fig. 3, the substreams are assembled by the mobile node; it interleaves frames according to their positions in the GOP structure. As the substreams are interleaved, the frame copy bits contained in the other substreams are discarded for frames that are successfully received. The assembled bitstream is then passed to the decoder for playback.

When a parent disconnects, the corresponding substream

[1]The length of the control bits is $6 \sim 7$ Bytes for a QCIF-resolution video. The control bits are about 4 kbps for 4 parents, GOP size = 12 and 30 fps.

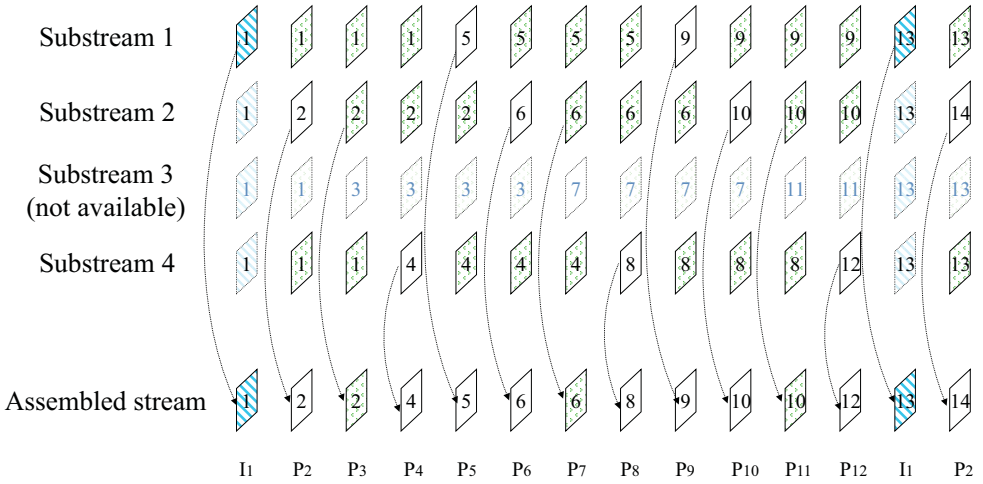

Figure 3: Decoding at a mobile node. In this example, one of 4 substreams is missing (Substream 3). The frame copy bits from Substream 2 are used in constructing the assembled bitstream.

becomes not available at the mobile node. If Parent 1 disconnects, then the mobile node requests the missing I frames from one of the other parents (recall that I frames are encoded at every parent). For the frames of the missing substream, the frame copy bits from the available substream preceding the missing substream are used as a replacement. In Fig. 3, Parent 3 disconnects and Substream 3 becomes not available. To replace Frames 3, 7, 11, 15 in the missing substream, the frame copy bits in Substream 2 are used. Note that the redundancy of frame copy bits in multiple substreams allows the assembled bitstream to be correctly played back even when more than one bitstream is missing. Since the assembly of substreams and the selective insertion of frame copy bits are performed at the bitstream level, we can avoid any modification to the decoder.

4. PEER CHURN ANALYSIS

In this section, we discuss the effect of distributed transcoding against peer churn. For simplicity, we assume that a parent node's lifetime is exponentially distributed with an average of $1/\mu$ seconds. We further assume that a missing parent node can be replaced with another node in a recovery time that is also exponentially distributed with an average of $1/\lambda$ seconds. Each parent node can fail independently from any other parent and the mobile node's recovery process of finding another parent is independent from the state of all other parents. We consider a Markov chain, where a state represents the number of live parents [12]. Fig. 4 illustrates a state-transition-rate diagram, where K represents the total number of parents.

When a parent arrives, the system state jumps from State i to $i+1$. For a transition from State i to $i+1$ ($0 \le i \le K-1$), $K-i$ concurrent recovery processes are performed; the transition rate is therefore $(K-i)\lambda$. When a parent leaves, the system state jumps from State i to $i-1$. For a transition from State i to $i-1$ ($1 \le i \le K$), i parents are alive and have the same failure rate of μ; the transition rate is therefore $i\mu$. We can now compute the stationary state probability distribution of this Markov chain by using the following relationship:

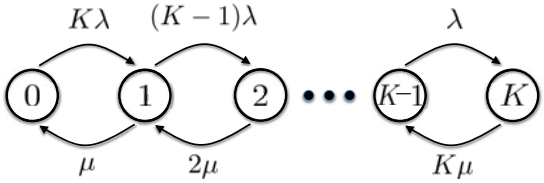

Figure 4: State-transition-rate diagram for a mobile node with K parents. $1/\lambda$ is the average recovery time. $1/\mu$ is the parent average lifetime.

$$\begin{aligned} K\lambda\pi_0 &= \mu\pi_1, \\ (K-1)\lambda\pi_1 &= 2\mu\pi_2, \\ &\cdots \\ \lambda\pi_{(K-1)} &= K\mu\pi_K, \\ \sum_{i=0}^{K}\pi_i &= 1, \end{aligned} \tag{1}$$

where *probability flow conservation* and *probability conservation* apply.

By expressing π_k ($k \neq 0$) with π_0, we obtain

$$\begin{aligned} \pi_0 &= \left\{\sum_{i=0}^{K}\binom{K}{i}\left(\frac{\lambda}{\mu}\right)^i\right\}^{-1} \\ &= \left(\frac{\lambda}{\mu}+1\right)^{-K}. \\ \pi_i &= \left(\frac{\lambda}{\mu}+1\right)^{-K}\binom{K}{i}\left(\frac{\lambda}{\mu}\right)^i, 0 \le i \le K. \end{aligned} \tag{2}$$

We can relate the states depicted in Fig. 4 with the effective frame rate of the decoded video. Suppose that f is the frame rate of the transcoded video. When the mobile node is in State $K-1$, then one substream is missing. With the copy error concealment, previously decoded frames are displayed in lieu of the frames of the missing substream. Thus

the effective frame rate becomes $(\frac{K-1}{K})f$ (no packet loss over the wireless channel is assumed for this discussion). For instance, when $K = 2$, if the mobile node is in State 1, then the effective frame rate is $0.5f$. When $K = 4$, then if the mobile node is in State 3, the effective frame rate is $0.75f$.

Table 1 shows the average fraction of time during which a particular number of parents (substreams) of a mobile node are missing under different peer churn rates. λ/μ, denoted by α, indicates the ratio of the parent average lifetime to the parent average recovery time. As α increases, the state probability of State K (no parents are missing) also increases and the state probability of all the other states decreases. It is obvious because the mobile node is more likely to have all parents alive when the recovery time is shorter or the average lifetime is longer.

In Table 2, the effect of peer churn on the number of missing parents is shown with the total number of parents K varied from 1 to 4. As a mobile node increases the total number of parents it connects to, the probability that all parents are missing approaches 0, whereas the probability that at least one parent is missing increases (when α is large, the probability of missing one parent (State $K-1$) increases linearly as K increases). We also found that regardless of K, the average fraction of missing frames is $\frac{1}{\alpha+1}$ (proof is omitted due to space constraint). This indicates that although connecting to multiple parents does not reduce the average number of missing packets, it effectively alleviates the degree of video degradation (reduced effective frame rate) at the expense of more frequent video degradation.

Table 1: The average fraction of time during which a particular number of parents of a mobile node are missing. The total number of parents is set to 4. λ/μ is the ratio of the parent average lifetime to the average recovery time.

λ/μ	Number of missing parents				
	0	1	2	3	4
10	0.683	0.273	0.041	0.003	0
30	0.877	0.117	0.006	0	0
50	0.924	0.074	0.002	0	0

Table 2: The average fraction of time during which a particular number of parents are missing given different total number of parents (α is set to 30.)

Total number of parents	Number of missing parents				
	0	1	2	3	4
1	0.968	0.032	N/A	N/A	N/A
2	0.936	0.062	0.002	N/A	N/A
3	0.906	0.091	0.003	0	N/A
4	0.877	0.117	0.006	0	0

5. EXPERIMENTAL RESULTS

We conducted computer simulations to evaluate the proposed distributed transcoding scheme in a P2P environment. We implemented the IDT algorithm by modifying the x264 encoder [13]. The x264 encoder is an open source library for encoding videos in the H.264/AVC syntax. The *Foreman* sequence and the *Mother & daughter* sequence in CIF resolution were used as the original videos, coded at 590 kbps and 223 kbps, respectively, using the H.264 main profile. The GOP size was 15 frames, and two B frames were generated between anchor frames. The frame rate of the original video was 30 fps. This original video was streamed live to a population of peers from the video source. Peers incrementally constructed the overlay as they joined the system [11] . When a fixed node was connected to the system, it received the original video as long as its downlink capacity was higher than the video bitrate. For simplicity, we assume that all fixed nodes had downlink capacity larger than the video bitrate.

When a mobile user joins the system, it searches for K fixed nodes that have available uplink bandwidth and processing power. We assumed that the number of fixed nodes exceeds K. After the mobile user finds K fixed nodes as parents, it assigns them unique Parent IDs (from 1 to K). Then, it requests them to trancode disjoint sets of video frames (substreams). For the synchronization of substreams, parents add meta data to substreams, such as the time stamp of a GOP. During the parent-coordination process, the mobile node examines its device-specific profile, such as the media decoding capability, display size, and user's preference. It also detects time-varying parameters including the remaining battery capacity and the maximum downlink bandwidth of the wireless channel. Based on the collected information, the mobile node determines the video quality (e.g., quantization parameter), frame rate, and spatial resolution. In our simulations, we kept the frame rate of the original video. Spatial resolution was reduced from CIF to QCIF. For a transcoded video, the GOP size was set to 24. The quantization parameter and the number of parents were varied across different simulations.

The fixed node's lifetime is exponentially distributed with an average of 90 seconds. When a fixed node serving as a parent leaves the system, its child node finds a different fixed node to recover the missing substream. The recovery time is exponentially distributed with an average of 3 seconds. Although the IDT provides error resilience on its own, it is sensitive to I frame loss. When Parent 1 failure is detected, the mobile node selects one of its available parents as the new Parent 1. When I frames are lost due to the lossy channel, retransmission is requested for the missing I frames. To avoid self-congestion, retransmissions of P frames are not requested.

Figs. 5 and 6 show simulation results for single-parent trans-coding and the interleaved distributed transcoding (2 and 4 parents). A distortion measured in PSNR is computed between the input stream to the IDT encoders (the bistream after the intermediate processing unit) and the assembled stream. The *Foreman* sequence contains much more motion than the *Mother & daughter* sequence and requires more bits to encode, and concealment of lost frames is more difficult. The solid curves show the rate-distortion performance without packet loss and without peer churn. The single-parent transcoding shows the highest coding efficiency in this scenario. When the number of parents increases, the distance between frames in a substream increases and this lowers inter-frame temporal correlation.

Now we consider the lossy scenario, where video degradation is incurred by two different sources: packet loss over the wireless channel and peer churn. The wireless channel is modeled by the Gilbert-Elliot model, as a discrete-time Markov chain with two states. Table 3 shows the Gilbert

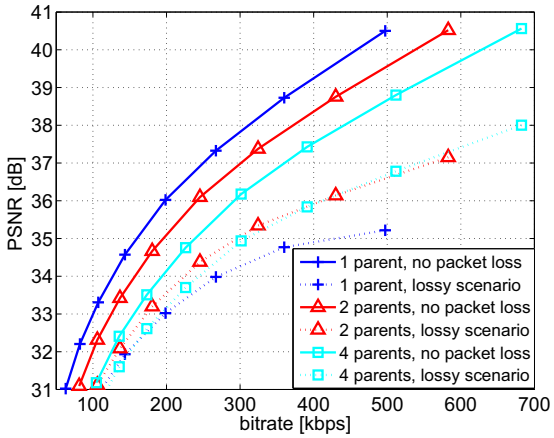

Figure 5: *Foreman* sequence: Rate-distortion curves with and without packet loss for conventional transcoding (1 parent) and interleaved distributed transcoding (2 and 4 parents).

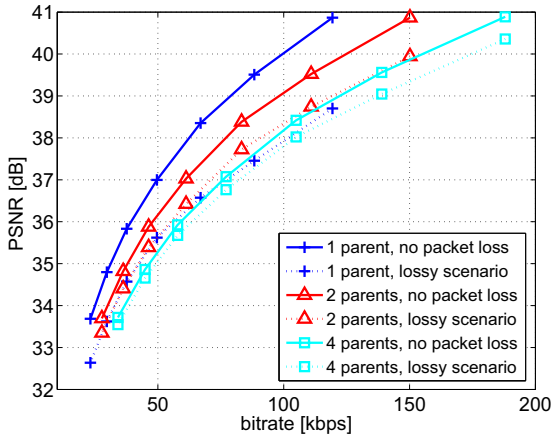

Figure 6: *Mother & Daughter* sequence: Rate-distortion curves with and without packet loss for conventional transcoding (1 parent) and interleaved distributed transcoding (2 and 4 parents).

model transition probabilities based on the GSM network, presented in [14]. During the good state there is no packet loss, whereas during the bad state the channel produces packet loss with probability 0.5. State transitions occur according to Table 3 at the transmission of each packet. For peer churn, a burst of packets that belongs to a substream is lost when the corresponding parent leaves. The dashed curves in Figs. 5 and 6 indicate the degraded video quality in the lossy scenario.

Table 3: Probabilities used in the Gilbert-Elliot model.

State i	$\Pr(i)$	$\Pr(1\|i)$	$\Pr(0\|i)$
0 (Good)	0.9449	0.0087	0.9913
1 (Bad)	0.0551	0.8509	0.1491

Figs. 5 and 6 also show that single-parent transcoding is vulnerable to the impairment of the received stream. In the case of distributed transcoding without packet loss, two parents outperformed four parents. However, the performance difference between two parents and four parents is negligible. This implies that the variance of the video quality is smaller with more parents because the adverse impact of packet loss is alleviated with more substreams. This result is analogous to that from the analysis of peer churn. In Section 4, we showed that the impact of parent disconnect, in terms of the effective frame rate, is smaller with more parents at the cost of longer video degradation periods.

We compared the performance of IDT against *Multiple Description Coding* (MDC) [15]. The MDC in [15] is similar to our IDT scheme, however, each interleaved video frame sequence contains its own I frames, and thus each substream can be decoded independently. For a fair comparison with IDT, the GOP size of each substream is made identical to the GOP size used for IDT. This ensures the ratio of the number of I frames and the number of P frames in the bitstream is identical for both MDC and IDT. Figs. 7 and 8 show the performances of MDC and IDT. IDT consistently outperforms MDC for both the lossy scenario and the scenario without packet loss. A plausible explanation is that in MDC, a GOP in a substream spans a larger duration of the original video than a GOP in the assembled stream. For instance, for K=4 parents and a GOP of 24 frames, IDT encodes every 24^{th} frame as an I frame. In contrast, MDC encodes every 96^{th} frame as an I frame. Reducing the GOP size may alleviate this slow refresh, but it results in a worse rate-distortion performance due to the higher ratio of the number of I frames to the number of P frames.

Next, we investigate the computation required to perform real-time transcoding. We measure the time spent by the modified x264 encoder. Since a fixed node performs decoding for its own playback, we do not include decoding time. We also ignored the intermediate processing time which is much smaller than the encoding time. We averaged the encoding times from 100 experiments[2] (Table 4). Encoding a frame of the *Foreman* sequence using single-parent transcoding required about 6.6 ms on average. When the frame rate of the transcoded bitstream is 30 fps, then about 20% of the CPU is consumed for a single mobile user. As more parents are involved in transcoding, each parent spends less time because the generation of frame copy bits has low complexity.

Table 4: Average time for encoding a frame.

	1 parent	2 parents	3 parents	4 parents
Foreman	6.6 ms	4.1 ms	3.2 ms	2.7 ms
Mthr & Dthr	4.4 ms	2.9 ms	2.4 ms	2.1 ms

6. CONCLUSIONS

In this paper, we propose a peer-to-peer based method for streaming video to mobile users. To satisfy different requirements of each heterogeneous mobile device, peers in the P2P network contribute their processing power to perform video

[2]The benchmark computer used in the experiment runs Linux OS and its CPU is an Intel Pentium 4 (single core) with a clock speed of 2.80 GHz.

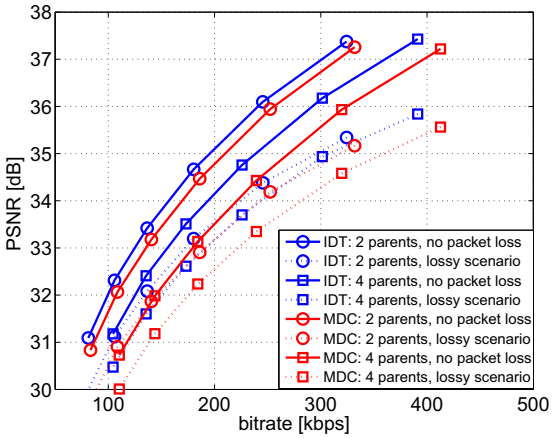

Figure 7: *Foreman* sequence: Comparison of IDT and MDC.

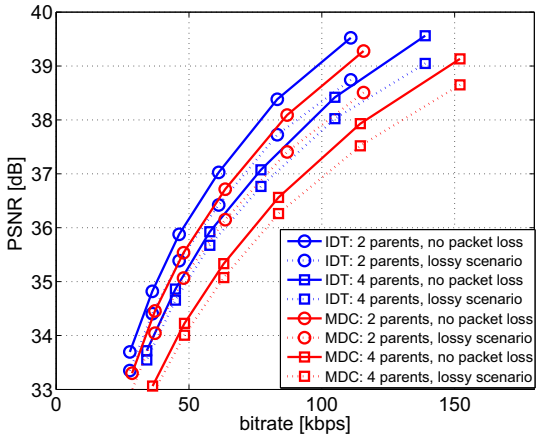

Figure 8: *Mother & Daughter* sequence: Comparison of IDT and MDC.

transcoding. The proposed *interleaved distributed transcoding* (IDT) is shown to lower the computation that each parent has to perform. The IDT scheme also provides error resilience against packet loss due to peer churn and to adverse wireless channel conditions. By analyzing the effects of peer churn, we showed that distributed transcoding balances the degree of video degradation with the occurrence of video degradation. Simulation results show that IDT outperforms multiple description coding (MDC) by 1 to 1.5 dB. We also implemented a real-time IDT encoder and verified the results of our analysis and simulations.

7. ACKNOWLEDGMENTS

Several schemes for distributed video transcoding for mobile devices were explored by Dr. Frank Hartung while he visited our group during the spring of 2008 on leave from Ericsson Eurolab. Support of this work through a gift from Deutsche Telekom Laboratories is gratefully acknowledged.

8. REFERENCES

[1] A. Warabino, S. Ota, D. Morikawa, and M. Ohashi, "Video transcoding proxy for 3G wireless mobile internet access," *Communications Magazine*, pp. 66–71, Jan. 2000.

[2] S. Dogan, A. Cellatoglu, M. Uyguroglu, A. Sadka, and A. Kondoz, "Error-resilient video transcoding for robust internetworkcommunications using GPRS," *IEEE Transactions on Circuits and systems for video technology*, vol. 12, no. 6, pp. 453–464, 2002.

[3] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, 2003.

[4] J. Noh, P. Baccichet, F. Hartung, A. Mavlankar, and B. Girod, "Stanford Peer-to-Peer Multicast (SPPM) – overview and recent extensions," *Proc. of International Picture Coding Symposium (PCS), Chicago, Illinois, USA, invited paper,*, May 2009.

[5] A. Dutta, J. Chennikara, W. Chen, O. Altintas, and H. Schulzrinne, "Multicasting streaming media to mobile users," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 81–89, 2003.

[6] S. Venot and L. Yan, "On-demand mobile peer-to-peer streaming over the JXTA overlay," *Mobile Ubiquitous Computing, Systems, Services and Technologies, (UBICOMM)*, pp. 131–136, Nov. 2007.

[7] F. Chen, T. Repantis, and V. Kalogeraki, "Coordinated media streaming and transcoding in peer-to-peer systems," in *19th IEEE International Parallel and Distributed Processing Symposium, 2005. Proceedings*, 2005.

[8] M. Leung and S. Chan, "Broadcast-based peer-to-peer collaborative video streaming among mobiles," *IEEE Transactions on Broadcasting*, vol. 53, pp. 350–361, 2007.

[9] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment," *Proc. of IPTPS'03, Berkeley, CA*, pp. 298 – 313, Feb. 2003.

[10] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," *Proc. of ACM NOSSDAV, Miami Beach, FL*, pp. 177 – 186, May 2002.

[11] E. Setton, J. Noh, and B. Girod, "Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming," *Workshop on Advances in Peer-to-Peer Multimedia Streaming at ACM Multimedia, Singapore*, pp. 39–48, Nov. 2005.

[12] L. Kleinrock, "Queueing systems: Volume 1 theory," *John Wiley & Sons, Inc.*, 1975.

[13] x264, "http://www.videolan.org/developers/x264.html."

[14] A. Konrad, B. Zhao, A. Joseph, and R. Ludwig, "A Markov-based channel model algorithm for wireless networks," *Wireless Networks*, vol. 9, no. 3, pp. 189–199, 2003.

[15] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *IEEE INFOCOM 2002. Proceedings*, vol. 3, 2002.