Robust mobile video streaming in a peer-to-peer system

Jeonghun Noh a,*, Bernd Girod b

a ASSIA Inc., 333 Twin Dolphin Drive, Redwood City, CA, USA
b Stanford University, 350 Serra Mall, Stanford, CA, USA

A R T I C L E   I N F O

Available online 16 February 2012

Keywords:
Peer-to-peer streaming
Mobile users
Robust transmission
Transcoding
Distributed multimedia

A B S T R A C T

In a peer-to-peer (P2P) video streaming system, peers not only consume video, but also route it to other peers in the system, where ordinary peers are assumed to have sufficient downlink speed and media capability. This assumption often fails when the P2P system consists of peers that are heterogeneous in their computing power, hardware, and media capability.

In this paper, we address a problem of streaming video to mobile devices, which are less capable than ordinary peers. In order to stream video to mobile devices, transcoding is often required to render video suitable for their small display, limited downlink speed, and limited video decoding capability. However, performing transcoding at a single peer is vulnerable to peer churn, which leads to video disruption. We propose interleaved distributed transcoding (IDT), a robust video encoding scheme that allows peers more capable than mobile devices to perform transcoding in a collaborative fashion. IDT is designed in such a way that transcoded substreams are assembled into a single video stream, which can be decoded by any H.264/AVC baseline profile compliant decoder. Extensive simulations and its implementation in a real P2P system demonstrate that the proposed scheme not only reduces computational load at a peer, but also achieves robust streaming in the case of peer failure or packet loss due to adverse wireless channel conditions. We confirm this finding by analyzing the effect of distributed transcoding under peer failure.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The peer-to-peer (P2P) architecture is a promising cost-effective solution to live video streaming to a large population. The P2P systems are self-scalable as peers not only consume video contents, but also relay them to other peers who want the same contents. Although peers may exhibit heterogeneous downlink and uplink bandwidths due to the access networks that peers connect to, it has been often assumed that every peer in the system wants and/or is able to consume the original video transmitted from the video source. In the last decade, mobile devices, such as smartphones and Personal Digital Assistants (PDAs), have become ubiquitous in our daily lives, expediting the heterogeneity among multimedia-capable devices. As most of the mobile devices are equipped with small displays or have access to the Internet with limited download speed, live media adaptation is performed to meet the streaming requirements of heterogeneous mobile users. For video, media adaptation is often achieved by video transcoding [1–4]. For mobile devices, the original video is converted to a new bitstream for a different video 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 may require individually customized transcoding.
In this paper, we propose a P2P-based mobile streaming that leverages the computing resources contributed by peers. Henceforth, we refer to personal computers or set-top boxes as fixed nodes, as opposed to mobile nodes. Fixed nodes usually have continuous power supply, so there is no risk of exhausting their power source. By harnessing the processing power of the fixed nodes, the transcoding burden of the servers can be reduced or eliminated. Due to their limited resources (e.g., battery, uplink speed), mobile nodes in the proposed architecture are treated as leeches, i.e., peers that only receive packets but do not relay the packets to other peers. Moreover, videos customized for individual mobile devices make relaying video less appealing.

However, 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 adversarial effects due to peer churn, the interleaved distributed transcoding scheme introduced in our previous work [5] allows more than one fixed node to perform transcoding for a mobile node. Even if a mobile device loses some of its parents, it may still receive substreams from the other parents. Then, the proposed scheme allows the mobile device to decode the incoming video partially, thereby achieving robust video transmission. In addition, our scheme distributes transcoding overhead to multiple fixed nodes. We design the distributed transcoding scheme in a way that conforms to the H.264/AVC baseline profile. This allows any H.264/AVC decoder to play back the transcoded video. The proposed video codec was integrated with the Stanford Peer-to-Peer Multicast system (SPPM) [6].

The remainder of the paper is structured as follows. Section 2 surveys the prior work in mobile streaming systems, transcoding, and robust video transmission. Section 3 describes the proposed distributed transcoding and its implementation. Section 4 introduces SPPM and discusses the integration of IDT with the original SPPM. In Section 5, we analyze the effect of peer churn on distributed transcoding. In Section 6, we provide simulation results and compare the performance of IDT with MDC (Multiple Description Coding).

2. Related work

2.1. Mobile streaming systems

In typical streaming systems, video transcoding is performed at servers to adapt video to mobile users. In [7,8], server-based adaptive video transcoding systems for mobile users are proposed. 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. The authors of [9] also propose a server-based video transcoding scheme for mobile users. The client proxy collects mobile client profiles and requests transcoding to the video proxy. Transcoded videos are then sent back to the client proxy, and the client proxy relays them to each client. The authors of [10] 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.

Although server-based systems work reasonably for a moderate number of users, transcoding servers can easily become a bottleneck because transcoding poses a considerable computational burden. Thus, P2P-based mobile streaming has been actively studied in order to reduce server load and increase service capacity. The authors in [11] investigated P2P-based video-on-demand (VoD) mobile streaming. In the system they proposed, the mobile user establishes a one-to-one connection to a proxy peer chosen in the P2P network. Through the 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 [11] on the needs for video transcoding and error resilience in the case of proxy peer failure. The contribution [12] 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 [13], 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 scale with the number of users, thereby reducing 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.

2.2. Video transcoding

Video transcoding converts an original video bitstream to a new bitstream for a target video player [1–4]. In [14], temporal transcoding, by which video frames are skipped and the motion vectors in the following frame are modified in order to reduce the bitrate, is conducted according to a channel condition. For rate control, rate-distortion-based frame transcoder [15] and frame skipping transcoders with motion vector update [16,8] have been proposed. A centralized transcoding algorithm is proposed in [17,18] for multi-user video streaming over a wireless LAN. In [17,18], the rate of transcoded video bitstream is dynamically adjusted by the central server in order to maximize the rate-distortion performance by considering the video characteristics and network conditions. Scalable video coding has its obvious merits when an encoded video signal needs to be transmitted to mobile
users with different bitrates or reception capabilities and/or the encoding cannot be done or is not economically viable for each and every receiver [19,20].

2.3. Robust video transmission

Feedback-based error control is a very effective approach to robust video transmission when a feedback channel is available and the extra delay incurred by such schemes is permissible. In this approach, a sender retransmits a lost or damaged packet when the sender receives a feedback message from the receiver [21–23]. In a more sophisticated framework known as RaDiO (Rate-Distortion Optimization) [24], proactive packet scheduling is performed to determine when and which packet to send to maximize the expected video quality at the receiver. In a related approach named CoDiO (Congestion-Distortion Optimization) [25], self-congestion due to retransmission is taken into account for packet scheduling. Our prior work in [26] extended CoDiO to tree-based P2P video streaming.

If the delay required for the feedback-based schemes is prohibitive, forward error correction (FEC) may be performed, in which a portion of the total transmitted bitrate is dedicated to parity information which enables the decoder to correct transmission errors. The most popular channel code used to provide FEC is the systematic Reed–Solomon code [27], in which packets of parity symbols are appended to the video packets prior to transmission. FEC, however, may suffer from the rapid reduction in picture quality, called the “cliff effect,” when the number of symbol errors is too high. To alleviate the cliff effect, the authors of [28] introduce Systematic Lossy Error Protection (SLEP), which transmits an additional bitstream generated by Wyner-Ziv coding.

Multiple Description Coding (MDC) exploits the availability of multiple paths between the source and the receiver [29–31]. In MDC, a signal is encoded into multiple descriptions that are independently decodable, which may lead to better quality if more descriptions are available in decoding. The performance of MDC is examined in [31] when applied to CDN, or in the context of P2P [32]. Layered coding and MDC are used together to achieve better system performance in [33–35]. The authors of [36] survey adaptive techniques used in video multicast.

When some packets miss the playback deadline, the receiver may not be able to decode video frames. Then, the receiver conceals the lost portion of the video based on several methods. Error-concealment minimizes the negative impact of late or missing frames as well as the visual quality degradation caused by the missing frame. Some error concealment schemes use estimation of coding modes and motion vectors [37,38] and spatial or spatio-temporal interpolation [39,40].

3. Interleaved distributed transcoding

In this section, we describe how video transcoding is performed at one or multiple locations for a mobile node in the proposed 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 encoding

The multiple parent approach is a popular solution to provide robustness to peer churn [41,42]. 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. Thus, we devise 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 depicted in Fig. 2. In this 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

Fig. 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.
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 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 $i$, $K+i, 2K+i, \ldots$, and each parent transmits every $K$th frame in a disjoint manner. Parent $i$ then codes the frames other than Frame $i$, $K+i, 2K+i, \ldots$, as frame copy bits. Frame copy bits are independent of the actual video content as they are control bits embedded in a compressed video signal that instructs a decoder to copy the previous frame decoded and available in the video buffer.\footnote{As the frame copy bits are independent of contents, it is possible to reconstruct the control bits at the decoder. When additional complexity at the decoder is acceptable, removing those control bits can eliminate redundancies in the substreams.} This allows error concealment to be done at the bitstream level and ensures that any standard-compliant decoders can decode the assembled bitstream without having to detect a parent or packet loss by itself. This feature is video compression technology dependent, which is available in the H.264/AVC standard and allows content-independent frame copy. For any other video technologies that have no such features, similar error concealment can be implemented accordingly. Note that the I frames are encoded and used in prediction by all parents, yet transmitted by only Parent 1 to avoid duplicate transmission. 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 6 in detail.

### 3.2. Decoding transcoded video

As substreams generated by multiple parents are transmitted to the mobile peer, the peer starts an assembling and decoding process. When there are no lost frames and every parent is available, the assembling process takes frames from each substream according to their interleaving order and places them in the final data stream. In particular, the peer 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. For example, the Frames 1, 5, 9 and 13 are taken from Substream 1 and used as Frames 1, 5, 9 and 13 in the final data stream. For frames that are not used in the final data stream are then discarded. The assembled bitstream is passed to the decoder for playback.

When a parent disconnects, the corresponding substream becomes unavailable at the mobile node. If Parent 1 disconnects, then the mobile node requests 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. Fig. 3 depicts one example, where Parent 3 disconnects and Substream 3 becomes unavailable. Therefore, Frames 3, 7, and 11 become unavailable to the mobile peer. In assembling the final data stream, the frame copies (Frames 2, 6, and 10) in Substream 2, which correspond to Frames 3, 7, and 11 in the missing substream (i.e., Substream 3), are used as Frames 3, 7, and 11 in the final data stream. Alternatively, the frame copies from other substreams (e.g., Substream 1 or 4) can also be used to substitute the missing frames in final data stream.

---

Fig. 2. Interleaved distributed transcoding: an example of four parents (IDT encoders) with a GOP of 12 frames. The original video stream is divided into four 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.
3.3. Implementing IDT

We implement the interleaved distributed transcoding (IDT) scheme in such a way that no decoder modification is required. The IDT encoder 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 P-SKIP mode. For this coding type, neither a quantized prediction error signal, nor a motion vector or reference index parameter is transmitted. The reconstructed signal is obtained similar to the prediction signal of a P-16 × 16 macroblock type that references the picture which is located at index 0 in the reference picture buffer. The motion vector used for reconstructing the P-SKIP macroblock is similar to the motion vector predictor (MVp) for the 16 × 16 block, where MVp is computed as the median of the motion vectors of the macroblock partitions or sub-partitions immediately above, diagonally above and to the right, and immediately left of the current partition or sub-partition. Since the P-SKIP mode always refers to the frame located at index 0 in the buffer, we move the previous frame in a substream to the index 0 location by the reference picture reordering. As a result, the encoder refers to only the frame at the same location for prediction, although there may be up to K pictures available in the buffer.

When the IDT encoder encodes every Kth 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 P2, P3, and P4 as a copy of Frame 1. Frame copy encodes frames with negligible computational complexity at the cost of about 1–2% of control bits added to the transcoded video.

4. Stanford Peer-to-Peer Multicast with mobile extension

We implemented the real-time IDT encoder and decoder, and integrated them into our P2P system, called Stanford Peer-to-Peer Multicast (SPPM). SPPM is a peer-to-peer system designed for low latency and robust transmission of live media. Similar to CoopNet and SplitStream, SPPM organizes peers in an overlay of multiple complementary trees. Every tree is rooted at the video source. The media stream, originating from the video source, is packetized and distributed on different substreams.
trees such that there are no duplicate packets across the
trees. Peers subscribe to every tree in order to receive the
media stream contiguously.

4.1. SPPM

Fig. 4 illustrates an example overlay in which two
complementary trees are constructed among a small
group of peers. Typically, we use 4–8 trees, and peer
groups might be much larger. We have run experiments
with up to 10,000 peers. As peers join the system, the
trees are incrementally constructed in a distributed man-
ner. When a new peer contacts the video source, the video
source replies with session information, such as the
number of multicast trees and the video bit rate. It also
sends a list of candidate parents randomly chosen from
the table of participating peers it maintains. The new peer
then probes each candidate parent to know about their
current status. After receiving probe replies, the best
candidate parent is selected and contacted for each tree
by minimizing the height of the distribution tree. Once
the selected candidate parent accepts the attachment
request, a data connection is established between the
parent and the new peer. After data transmission starts,
each child peer periodically sends hello messages to their
parents. When a peer leaves the system ungracefully, its
parents detect it by observing consecutive missing hello
messages, and stop forwarding video to the child. The
departing peer’s children notice that neither video pack-
ets nor a response to hello messages arrive. Each aban-
donated child then initiates procedure to connect to a
different parent node in the tree.

To provide the best video quality despite congestion
and peer churn, SPPM employs sender-driven packet
scheduling [26] in conjunction with receiver-driven
retransmission requests [45]. Acting as a relay, each peer
schedules the next packet to forward by comparing the
importance of packets in its output queue. As a receiver,
each peer evaluates the importance of missing packets by
computing the expected video quality improvement if the
corresponding frame is received before its playout dead-
line. Retransmission requests of selected missing packets
are then sent to alternative parents. Since packet losses
typically occur in rare bursts, such feedback error control
is generally superior to forward error control. Feedback
implosion, prohibiting feedback error control in network-
layer multicast, is not an issue for a P2P overlay because
retransmissions are handled locally between a parent and
a child peer.

4.2. Mobile extension

We implemented the IDT encoder for the SPPM fixed
peer, which performs distributed transcoding. The SPPM
mobile peer was developed on the Nokia N96 platform
[46]. The original SPPM protocol was extended to allow
the fixed and mobile peers to communicate with each
other as well as with the video source.

4.2.1. SPPM fixed peer

As a fixed peer, the SPPM peer is extended to include a
mobile child manager, a decoder, an intermediate pro-
sessing unit, a set of encoders, a packetizer, and a queue
for transcoded video, as shown in Fig. 5(a). The mobile child
manager creates, maintains, and terminates a service for a
mobile peer. When a new mobile peer requests a connec-
tion, it creates a dedicated encoder. The video unit
decodes the original video stream. The decompressed
video is stored in the Raw Video Buffer. Due to packet loss
or internal errors in decoding, some frames may be
missing in the buffer. Each frame in the buffer is tagged
with both the global Group-of-Picture (GOP) ID and local
frame ID. These metadata are tagged in the video packets
and referred to by the interleaver at the mobile peer. The
intermediate processing unit is used for modifying raw
frames before they are passed into the encoder. Resizing
(e.g., down-sampling or changing the ratio of the number
horizontal and vertical pixels), cropping, and/or frame
rate reduction are performed at this unit. For each mobile
peer, a different intermediate processing may be per-
formed. The encoder encodes the modified raw video

4.2.2. SPPM mobile peer

The mobile peer consists of the input/output NIC, a
control unit, a video unit, and an interleaver. The control
unit processes incoming control messages and generates
response messages or new control messages to trigger
events, such as parent coordination or rejoin after
detecting a missing parent. The component diagram is shown in Fig. 5(b).

The video unit requests the next video packets and processes incoming video packets. Each substream is a bitstream of an H.264/AVC video signal. The video unit divides a substream into frames by detecting their boundaries (e.g., NAL units). It also marks each frame with relevant metadata, such as POC (Picture Order Count), GOP ID, Frame ID, and the Substream ID. The video unit signals the control unit for retransmissions of missing I frames. It also detects parent disconnect or evaluates the download channel status, and signals the control unit. The interleaver assembles the frames of the substreams into a single bitstream. When there is a missing frame due to a missing packet or missing substream, corresponding copy frame control bits are substituted in order to conceal missing frames. The interleaver discards old frames or unnecessary frames in the queues for the substreams. It starts assembling after a time-out. The time-out interval can be adjusted so that the initial buffering time and the packet reception ratio (excluding late or missing packets) are balanced.

4.2.3. Protocol extension

In order to join the system, a mobile peer first contacts the video source and sends the video source information, such as its device type, properties (e.g., display size, multimedia codecs), maximum downlink speed. The video source replies with information, such as the bitrate of the original video and peer list. The mobile peer then collects information from parent candidates, such as the available bandwidth, depth in the overlay tree, Frame ID of the recently decoded frame. Based on the collected information, the peer selects several parent candidates to connect to. When the mobile peer sends a connection request to the parent candidates it selects, it assigns the Substream ID for them. Note that the parent candidates have no need to directly communicate with the others.
The connection request message also contains the downsampling ratio, total number of parents, video quality parameterQP, GOP size, global frame ID used to synchronize parents. Once the mobile peer connects to its parents, it continues to exchange messages with its parents for status report or change in settings. When a parent failure is detected by noticing no video reception, the mobile peer triggers the rejoin procedure. If Parent 1, who is responsible for I-frames, fails, the mobile peer selects one of the other parents as Parent 1. The rest of the rejoin procedure is similar to the rejoin procedure of the basic SPPM protocol. When a mobile peer failure is detected by noticing no hello messages, the fixed peer discards the mobile peerÔs information and terminates the dedicated encoder.

5. Peer churn analysis

In this section, we discuss how distributed transcoding mitigates the effect of peer churn. For simplicity, we assume that a parent node’s lifetime is exponentially distributed, with an average of 1/μ 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/λ 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 [47]. Fig. 6 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 ≤ i ≤ K – 1), K – i concurrent recovery processes are performed; the transition rate is therefore (K – i)λ. When a parent leaves, the system state jumps from State i to i – 1. For a transition from State i to i – 1 (1 ≤ i ≤ K), i parents are alive and have the same failure rate of μ; the transition rate is therefore iμ. We can now compute the stationary state probability distribution of this Markov chain by using the following relationship:

$$\sum_{i=0}^{K} \pi_i = 1,$$

where probability flow conservation and probability conservation apply. By expressing π_k (k ≠ 0) with π_0, we obtain

$$\pi_0 = \frac{\frac{K\lambda}{\mu}}{\sum_{i=0}^{K} \left(\frac{K}{i} \left(\frac{\lambda}{\mu}\right)^iight)^{-1}} = \left(\frac{\lambda}{\mu} + 1\right)^{-K},$$

(3)

![Fig. 6. State-transition-rate diagram for a mobile node with K parents.](image)

$$\lambda \pi_{K-1} = K\mu \pi_K,$$

(1)

Note that the binomial theorem is applied in Eq. (3).

We can relate the states depicted in Fig. 6 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 copy error concealment, previously decoded frames are displayed in lieu of the frames of the missing substream. Thus, the effective frame rate becomes ((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 \(\alpha\), indicates the ratio of the parent average lifetime to the parent average recovery time. As \(\alpha\) increases, the state probability of State K (no parents are missing) also increases and the state probability of all the other states decreases. This 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 0 to 4. As a mobile node connects to more parents, the probability that all parents are missing approaches 0, whereas the probability that at least one parent is missing increases (when \(\alpha\) is large, the probability of missing one parent (State K – 1) increases linearly as K increases). Note that regardless of K, the average fraction of missing frames is a constant 1/(α + 1);

**Table 1**

<table>
<thead>
<tr>
<th>Number of missing parents</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parents</td>
<td>10</td>
<td>30</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.683</td>
<td>0.273</td>
<td>0.041</td>
<td>0.003</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.877</td>
<td>0.117</td>
<td>0.006</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0.924</td>
<td>0.074</td>
<td>0.002</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2**

The average fraction of time during which a particular number of parents is missing given a different total number of parents (\(\alpha = \lambda/\mu\) is set to 30).

<table>
<thead>
<tr>
<th>Total number of parents</th>
<th>Number of missing parents</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0.968</td>
<td>0.032</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.936</td>
<td>0.062</td>
<td>0.002</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.906</td>
<td>0.091</td>
<td>0.003</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.877</td>
<td>0.117</td>
<td>0.006</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
in State $i$, $(K-i)/K$ fraction of packets are lost due to $K-i$ missing parents. Then, $\Psi(K)$, the average fraction of packets lost given $K$ parents is expressed as

$$
\Psi(K) = \sum_{i=0}^{K} \left( \frac{K-i}{K} \right)^{\frac{z}{\mu} + 1} \left( \frac{K}{i} \right)^{\frac{z}{\mu} + 1} \left( \sum_{j=0}^{K-1} \left( \frac{K-1-i}{j} \right)^{\frac{z}{\mu} + 1} \right)
$$

$$
= \left( \frac{z}{\mu} + 1 \right)^{\frac{z}{\mu} + 1} \left( \frac{K-1}{0} \right)^{\frac{z}{\mu} + 1} \left( \sum_{j=0}^{K-1} \left( \frac{K-1-j}{j} \right)^{\frac{z}{\mu} + 1} \right)
$$

$$
= \left( \frac{z}{\mu} + 1 \right)^{\frac{z}{\mu} + 1} \left( \frac{K-1}{0} \right)^{\frac{z}{\mu} + 1} \frac{1}{z+1} \quad (5)
$$

This result shows that connecting to multiple parents does not affect the average number of missing packets. Instead, it alleviates the degree of video degradation at the expense of more frequent video degradation\footnote{We assumed no packet loss in the analysis. In reality, packet loss may be affected by video bitrate, which is subject to $K$.}

### 6. 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 \[49\]. The x264 encoder is an open source library for encoding videos in the H.264/AVC syntax. Six video sequences in CIF resolution were used as the original videos, encoded by using the H.264/AVC main profile. The GOP size was set to 24 frames, and two B-frames were generated between the anchor frames. The frame rate of the original video was 30 fps. The original video was streamed live to a population of SPPM fixed peers from the video source. Peers incrementally constructed the overlay as they joined the system. When a fixed peer 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 assumed 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 exceeded $K$. 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 transcode disjoint sets of video frames (substreams). For the synchronization of substreams, parents add metadata 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 s. 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 s. 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 a lossy channel, retransmission is requested for the missing I frames. To avoid self-congestion, retransmissions of P frames are not requested.

Fig. 7 shows the simulation results for single-parent transcoding and the interleaved distributed transcoding (two and four parents). A distortion measured in PSNR is computed between the input stream to the IDT encoders (the stream after the intermediate processing unit) and the assembled stream. The solid curves show the rate-distortion performance without packet loss and without peer churn, in a case called “no-packet-loss”. The single-parent shows the highest coding efficiency in this case because temporal redundancy is removed most effectively. When more parents are involved with transcoding, 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 caused by two different sources: packet loss over the wireless channel and peer churn. The wireless channel is modeled by the Gilbert–Elliott model, as a discrete-time Markov chain with two states. Table 3 shows the Gilbert model transition probabilities based on the GSM network, presented in \[50\]. 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 Fig. 7 indicate the degraded video quality in the lossy scenario.

Some sequences, such as Foreman sequence, contain more motion than other sequences, such as Mother & Daughter, and the concealment of lost frames is often more difficult. Fig. 7 also shows that the single-parent case suffers from a significant quality drop from the no-packet-loss case. This indicates a huge fluctuation in video quality occurs when the mobile node loses its only parent. On the contrary, distributed transcoding (both two and four parents) exhibit a lower quality degradation than single-parent, which implies a lower variance in video quality. We also observed that the performance difference between two parents and four parents is negligible except for the Mobile sequence. Recall that two parents outperformed four parents in the no-packet-loss case. This shows that the variance of the video quality is smaller by 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 5, 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) \[31\]. The MDC in \[31\] 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

![Rate-distortion curves with and without packet loss for conventional transcoding (one parent) and interleaved distributed transcoding (two and four parents). (a) Foreman sequence; (b) Mother & Daughter sequence; (c) Container sequence; (d) Mobile sequence; (e) News sequence; and (f) Paris sequence.](image)

**Table 3**

| 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 |
Fig. 8. Video quality comparison of IDT and MDC. (a) Foreman sequence; (b) Mother & Daughter sequence; (c) Container sequence; (d) Mobile sequence; (e) News sequence; and (f) Paris sequence.

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 24th frame as an I frame. In contrast, MDC encodes every 96th 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 measured 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 (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 cycles are required for a mobile user. As more parents are involved in transcoding, each parent spends less time because the generation of frame copy bits has low complexity.

7. Conclusions

In this paper, we proposed a robust video streaming scheme for mobile devices by leveraging the peer-to-peer architecture. In the proposed scheme, while ordinary peers consume the original video transmitted from the video source, less-capable peers, such as mobile phones, consume video adapted by the interleaved distributed transcoding (IDT). IDT is shown to provide graceful video degradation against packet loss due to peer churn and to adverse wireless channel conditions. It is also shown to reduce the amount of computation that each parent has to perform. Our theoretical analysis shows that IDT balances the degree of video degradation with the occurrence of video degradation. We also implemented the proposed scheme in a real-time P2P streaming system. For a certain video sequence, it was shown that the IDT achieves 67% reduction in the CPU time required for a parent peer by using four parents.

Much interesting research may be carried out to extend this study. Instead of treating mobile peers individually, they may be clustered into a limited number of profile groups, thus improving system scalability in some scenarios where a large population of mobile peers is present. At the writing of this paper, virtually no SVC decoders for mobile devices are available. However, when SVC for mobile devices becomes prevalent, it will be interesting to compare on-demand transcoding, such as IDT, with SVC-based streaming in the P2P context.

References


Table 4
Average CPU time required for encoding a frame at one of parents.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>One parent (ms)</th>
<th>Two parents (ms)</th>
<th>Three parents (ms)</th>
<th>Four parents (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>6.60</td>
<td>4.14</td>
<td>3.19</td>
<td>2.74</td>
</tr>
<tr>
<td>Mother &amp; Daughter</td>
<td>4.39</td>
<td>2.86</td>
<td>2.41</td>
<td>2.14</td>
</tr>
<tr>
<td>Container</td>
<td>4.38</td>
<td>2.70</td>
<td>2.20</td>
<td>1.96</td>
</tr>
<tr>
<td>Mobile</td>
<td>7.25</td>
<td>4.03</td>
<td>2.49</td>
<td>2.42</td>
</tr>
<tr>
<td>News</td>
<td>3.08</td>
<td>2.54</td>
<td>2.06</td>
<td>1.82</td>
</tr>
<tr>
<td>Paris</td>
<td>4.48</td>
<td>2.77</td>
<td>2.28</td>
<td>1.98</td>
</tr>
</tbody>
</table>

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.


