

Accelerated IPTV Channel Change with Transcoded Unicast Bursting

Zhi Li[†], Ali C. Begen[§], Xiaoqing Zhu[§] and Bernd Girod[†]

[†]Dept. of Electrical Engineering, Stanford University

[§]Cisco Systems, Inc.

[†]{leeoz,bgirod}@stanford.edu, [§]{abegen,xiaoqzhu}@cisco.com

ABSTRACT

We study video transcoding for accelerated channel changes in IPTV systems. Video transcoding at the Retransmission Server not only reduces the channel change latency, but also reduces the duration and data size of the unicast burst stream used for rapid acquisition. We develop an analytical model to capture the fundamental trade-offs in this system. This model is then used to characterize the potential savings from transcoding the unicast stream. Analysis and simulation results show that the stream compression factor affects linearly the saving in the channel change latency, and super-linearly the saving in unicast burst duration (or data size).

Categories and Subject Descriptors

C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms

Design, Performance

Keywords

IPTV, accelerated channel change, transcoding

1. INTRODUCTION

Channel change latency is crucial for the user's quality of experience in a digital TV system. As a rule of thumb, a latency of less than 500 ms is considered as instantaneous whereas more than two seconds is perceived as annoying [4]. Compared to traditional analog TV and digital cable, the channel change latency in multicast-based IPTV is particularly challenging, as it is affected by not only video compression and encryption, but also network operations.

Reduction of the IPTV channel change latency can be addressed at different delay components [4]. *Acquisition delay* and *decoder buffering delay* are the two dominating delay components. Acquisition delay refers to the waiting time until information in the stream necessary for decoding is acquired, such as program specific information (PSI), encryption

keys and a random access point (RAP). In addition, the decoder needs to buffer data before playout can commence. There are also other delay components, such as the delay for stopping the previous multicast stream, the delay for joining the new multicast session, and network buffering delay. The latency of a baseline IPTV scheme includes all the aforementioned delay components.

Various accelerated channel change (ACC) schemes have been proposed to reduce such latency. One approach is to send along the primary stream a *companion stream* of lower quality but with more frequent RAPs, so that the acquisition delay can be effectively reduced [4, 6, 5]. Another approach is to use multiple *time-shifted multicast* sessions for each TV channel and let the set-top box (STB) join one session that leads to a minimum acquisition delay [7, 3]. The companion stream approach and the time-shifted multicast approach impose extra burden on either the multicast router (MR) data plane and the access network downlinks, or on the MR control plane. A different ACC approach is to use a dedicated Retransmission Server (RS) to burst a unicast stream to achieve *rapid acquisition* of the decodable video stream [8, 4, 2]. This approach can avoid the acquisition delay, as the unicast stream always starts with an RAP.

In this work, we study the effect of video transcoding for rapid acquisition. Transcoding can reduce the stream bitrate, thereby allowing the unicast burst to catch up with the multicast stream faster. We show that under realistic settings, its effect on reducing the channel change latency is linear, and that on reducing the unicast stream duration/data size is superlinear with respect to the stream compression factor. After a brief overview of the channel change procedure in Section 2, we model and analyze the rapid acquisition-based ACC with server transcoding in Section 3. In Section 4, we present simulation results based on H.264/AVC and ns-2, and evaluate the perceptual video quality against channel change latency and unicast burst duration.

2. CHANNEL CHANGE PROCEDURE

In the baseline IPTV scheme, which we will refer to as non-accelerated channel change (NCC), the STB responds to a user request of channel change by sending an IGMP Leave message for the old channel and an IGMP Join message for the new channel [1] to the upstream MR. After a period of time, the new multicast stream arrives, but the STB has to wait for the next RAP and then for the decoder buffer to fill up before video playout can start.

In the accelerated channel change (ACC) scheme shown in Figure 1, the RS is situated at the edge of the distri-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'10, October 25–29, 2010, Firenze, Italy.

Copyright 2010 ACM 978-1-60558-933-6/10/10 ...\$10.00.

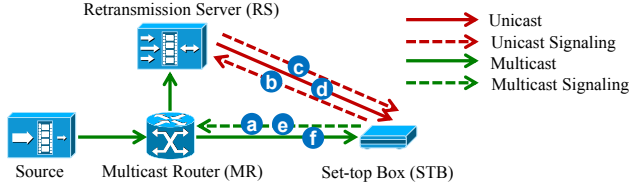


Figure 1: System diagram of accelerated channel change (ACC) based on rapid acquisition.

bution network and continuously caches the data multicast in the last 5 to 10 seconds. When the user requests a new channel, the STB sends an IGMP Leave message to the MR (corresponding to (a) in Figure 1), and at the same time, a rapid acquisition request message to the RS (b). The RS responds with a message announcing whether the request would be served (c), and if served, the RS starts sending the unicast stream (d). If the request is rejected, the STB joins the multicast as in NCC. After some period of delay (which, depending on the implementation, may be calculated at the RS side and signalled to the STB), the STB sends an IGMP Join message to the MR for the new channel (e). After a short delay, the multicast stream arrives at the STB (f), and eventually the unicast stream ceases.

When server transcoding is applied on top of ACC (termed as ACC-ST), the network protocol and the video decoding procedure remain identical to the ACC case, thus the network infrastructure and the STBs are left intact. Only the RS requires modification. When caching the multicast stream, it transcodes the incoming stream on-the-fly, generates a coarsely quantized lower-bitrate stream, and sends this stream to the STBs on-demand. We will discuss a lightweight implementation of the transcoding scheme in Section 4.

3. MODELING AND ANALYSIS

The model we introduce is based on a deployed system [2]. The goal is to capture the fundamental trade-offs without too many implementation details. The main simplifications are listed below.

- Three delay components are considered – multicast join delay, acquisition delay (due to I-frame RAP) and decoder buffering delay.
- The signaling messages among the STB, the RS and the MR are assumed to be lossless and instantaneous.
- The RS can lower its unicast rate in perfect coordination with the multicast arrival to avoid congestion.
- The unicast stream always starts from the immediate preceding I-frame in the cached stream.¹

We use the following criteria to evaluate each channel change scheme: (i) the channel change latency T_P – the duration between request and actual playout of the new channel, (ii) duration of the unicast burst T_U , and (iii) data size (in bits) of the unicast burst B_U . Note that T_U and B_U apply to the case of ACC only, capturing the burden on the RS control plane and data plane, respectively.

3.1 Non-Accelerated Channel Change (NCC)

The channel change latency of NCC consists of three components occurring in tandem: join delay T_J , time to wait

¹The RS could start streaming from an earlier instead of the immediate I-frame. While this may help lower the channel change latency, it leads to a longer unicast burst duration.

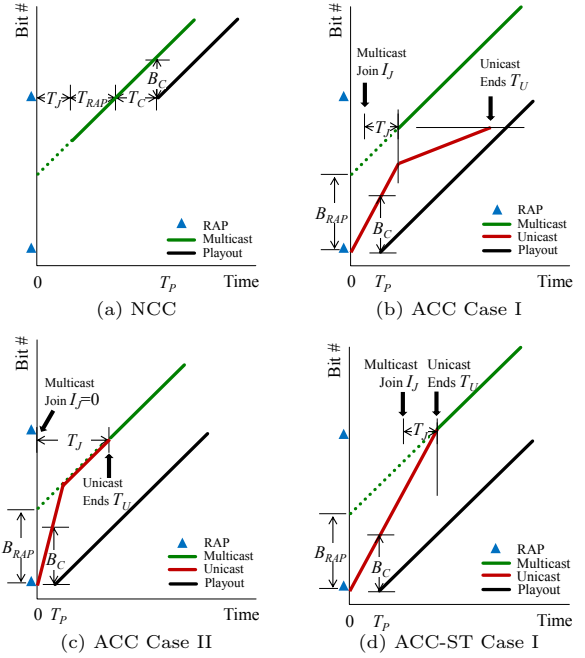


Figure 2: Bits received at STB vs. time for (a) Non-accelerated channel change (NCC); (b) Accelerated channel change (ACC) Case I; (c) ACC Case II; (d) ACC with server transcoding (ACC-ST) Case I.

for the next RAP T_{RAP} and client buffering delay T_C^{NCC} as shown in Figure 2(a). The channel change latency T_P^{NCC} is

$$\begin{aligned} T_P^{NCC} &= T_J + T_{RAP} + T_C^{NCC} \\ &= T_J + T_{RAP} + \frac{B_C}{R} \end{aligned} \quad (1)$$

where R is the nominal video streaming rate and B_C is the number of decoder buffering bits.

3.2 Accelerated Channel Change (ACC)

At time 0, after the STB sends a request, the RS starts bursting the unicast stream from the immediate preceding RAP at the bitrate $(1+e)R$, where $e > 0$ is the bandwidth expansion factor, or the e -factor. This higher-than-usual bitrate allows the unicast burst to catch up with the multicast stream. We discuss two cases: (i) if the multicast arrives before the unicast catches up, we have Case I (See Figure 2(b)); (ii) otherwise, we have Case II (See Figure 2(c)). In Case I, to make room for the multicast, the RS lowers its unicast bitrate to eR at the moment the multicast stream arrives. The unicast then keeps on bursting until T_U , when the gap between the unicast and multicast stream is closed. At some instant I_J , the STB sends an IGMP Join message to the MR; after setup delay of T_J , the STB starts receiving the real-time multicast stream of bitrate R from the MR. In Case II, after the unicast catches up, it should not pass the multicast stream, thus the unicast has to lower its bitrate from $(1+e)R$ to R . The RS terminates the unicast at the moment the multicast begins. In either Case I or II, the decoder starts decoding after duration T_C^{ACC} , when B_C bits are completely filled.

Analysis

First we discuss the condition inducing ACC Case I and Case II. The duration needed for the unicast to catch up

with the multicast, denoted by T_{CAT} , can be expressed as

$$T_{CAT} = \frac{B_{RAP}}{(1+e)R - R} = \frac{B_{RAP}}{eR}. \quad (2)$$

where B_{RAP} is the number of bits the unicast needs to catch up. In Case I, when $T_J \leq T_{CAT}$, the multicast stream should arrive at a moment before T_{CAT} . In Case II, when $T_J > T_{CAT}$, the STB should be instructed (by the RS) to send the multicast join message immediately at time 0. After T_J , the multicast arrives and the unicast terminates.

The channel change latency of ACC only consists of the client buffering delay T_C^{ACC} :

$$T_P^{ACC} = T_C^{ACC}. \quad (3)$$

To find T_C^{ACC} , we need to discuss two cases. If $B_C \leq (1+e)R \cdot T_{CAT}$, the client buffer is completely filled with the full rate $(1+e)R$; otherwise after T_{CAT} , the client buffer is filled with R . Overall,

$$T_C^{ACC} = \begin{cases} \frac{B_C}{(1+e)R}, & B_C \leq (1+e)R \cdot T_{CAT} \\ T_{CAT} + \frac{B_C - (1+e)R \cdot T_{CAT}}{R}, & \text{otherwise.} \end{cases} \quad (4)$$

In ACC, it turns out that the optimization of B_U and T_U boils down to choosing an optimal instant (by the RS) for the STB to send the IGMP Join message to the MR. Next we derive this optimal I_J .

In Case I, there are a few observations for choosing I_J . (i) I_J should be as early as possible, otherwise it would lead to large B_U and T_U . (ii) However, I_J should not be too early as well, for two reasons: If I_J is chosen such that the multicast stream is received earlier than both T_P^{ACC} and T_{CAT} , it will delay the new channel playout. To avoid this, we must ensure

$$I_J + T_J \geq \min(T_P^{ACC}, T_{CAT}). \quad (5)$$

(iii) If I_J is too early, it may result in a gap between the multicast and unicast streams, thus interrupting the playout. To avoid this, we must ensure

$$(T_U - T_P^{ACC})R \leq B_{RAP} + (I_J + T_J)R. \quad (6)$$

Overall, we take

$$I_J = I_J^* = \min I_J \quad \text{subject to (5) and (6)}. \quad (7)$$

In Case II, only (i) needs to be met, thus the STB should join the multicast session immediately, *i.e.*, $I_J = 0$.

Recall that $T_U = T_{CAT}$ in Case I. In Case II, T_U is at the moment the multicast starts arriving, *i.e.*, T_J . Overall,

$$T_U = \max(T_{CAT}, T_J). \quad (8)$$

Combining (3), (4), (7) and (8), we can find the expression for I_J . The other quantity we are interested in, B_U , is

$$B_U = B_{RAP} + (I_J + T_J)R. \quad (9)$$

3.3 ACC with Server Transcoding (ACC-ST)

With ACC-ST, the RS performs transcoding on-the-fly and generates a lower-bitrate stream with compression factor $\beta < 1$.

Analysis

The transcoded stream has bitrate βR . This is equivalent to having an effective e-factor $e_{eff} = (1+e)/\beta - 1$ when the unicast bursts at bitrate $(1+e)R$, and $e'_{eff} = e/\beta$ when the unicast bursts at bitrate eR . Since $\beta < 1$, we always have $e_{eff} > e'_{eff}$. The implication is that the unicast catches up with the multicast more quickly when it uses the full bitrate $(1+e)R$. Suppose that we would like to minimize

Variable	Range	Default Value	Description
T_J		0.1 s	Multicast join delay
T_{RAP}	0~1 s	0.5 s	Time until the next RAP
B_{RAP}	0~1 R Mbits	0.5 R Mbits	#bits to the previous RAP
B_C		0.5 R Mbits	Decoder buffering #bits
R		5 Mbps	Nominal streaming rate
e		0.2	e-factor
β	0.1 ~ 1	0.5	Stream compression factor

Table 1: Analysis and experiment parameters.

the unicast duration B_U , the optimal strategy in ACC-ST is that we burst until the unicast catches up with the multicast (See Figure 2(d) for Case I; Case II is identical to ACC).

Let $T_{CAT}(e, 1)$ and $T_P^{ACC}(e, 1)$ denote the unicast catch-up time and channel change latency, respectively, for excess bandwidth e and $\beta = 1$, *i.e.*, the case without stream compression. For ACC-ST with stream compression factor β , the unicast catch-up time is

$$T_{CAT}(e, \beta) = T_{CAT}(e_{eff}, 1) = \frac{B_{RAP}}{\left(\frac{1+e}{\beta} - 1\right)R}. \quad (10)$$

Similarly, the channel change latency $T_P^{ACC}(e, \beta)$ is

$$T_P^{ACC}(e, \beta) = T_P^{ACC}(e_{eff}, 1). \quad (11)$$

The unicast bursts either until the unicast catches up (Case I), or until T_J when the multicast arrives (Case II), thus the unicast duration is

$$T_U(e, \beta) = \max(T_{CAT}(e, \beta), T_J). \quad (12)$$

The corresponding unicast data size is

$$B_U(e, \beta) = \beta \cdot \max[T_{CAT}(e, \beta) \cdot (1 + e_{eff})R, B_{RAP} + T_J R]. \quad (13)$$

3.4 Numerical Results

Using the parameters listed in Table 1, we plot the numerical results in Figure 3. In (a), as expected, we observe that for NCC, T_P increases linearly with the RAP delay T_{RAP} . For ACC and ACC-ST, except at low T_{RAP} ,² the channel change latency is kept constant. In (b), we observe that for ACC-ST, the gain in T_P is *linear* w.r.t. β . In (c) and (d), the gain in T_U and B_U respectively are both *superlinear* w.r.t. β . This implies that a slight compression of the unicast stream can lead to a significant reduction in unicast duration and data size. Furthermore, in (b)-(d), it is evident that the gain is more significant for smaller e-factors.

4. SIMULATION RESULTS

We implement a simulation program based on the Network Simulator (ns-2) and a modified H.264/AVC JM coder. The primary simulation parameters are listed in Table 1. We consider the scenario where the RS supports five downstream STBs. The RS bitrate is set to 70% of the peak demand, so that moderate contention exists among the STBs. The RS-MR link delay is set to be 5 ms and MR-STB link delay is 20 ms. We assume that each STB initiates its channel change request at a random instant, uniformly distributed within a group of picture (GOP).

To evaluate the received video quality, we use the SOCCER sequence with a spatial resolution of 704×576 pixels and a frame rate of 30 frames per second. The GOP structure is IPPP and the two simulated GOP sizes are 15 (short) and 60 (long). We use a lightweight transcoding scheme where

²This is due to the fact that the unicast burst catches up too quickly without the chance of bursting at the full rate.

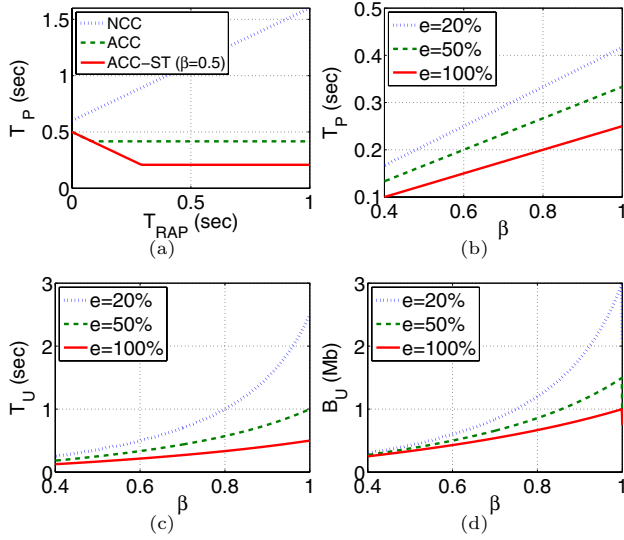


Figure 3: Analytical results: (a) Channel change latency T_P as function of RAP delay T_{RAP} ; (b) T_P as function of stream compression factor β ; (c) Unicast duration T_U as function of stream compression factor β ; (d) Unicast data size B_U as function of stream compression factor β . $T_{RAP}=0.5$ sec.

the transform coefficients in each slice (*i.e.*, one packet) are coarsely requantized to generate a smaller packet, without modifying the GOP structure. When switching from the unicast to the multicast stream, the two streams are spliced together, leading to some drift until the next I-frame.

We plot the CDFs of the channel change latency, unicast duration and unicast data size in Figure 4. The observations are consistent with our analytical results. For NCC, the GOP duration greatly affects the channel change latency. For ACC and ACC-ST, the channel change latency is immune from the GOP duration, since the playout can start as long as the decoder buffer is filled. Nevertheless, the GOP duration still has a significant impact on the unicast duration and data size. The longer the duration, the longer it takes for the unicast to catch up with the multicast. In this regard, server transcoding is very effective in reducing the unicast data size and duration. This suggests that transcoding at the server can significantly reduce the burden per rapid acquisition, thereby allowing each RS to support more STBs.

Figure 5 plots the resulting channel change latency and unicast duration vs. the decoded video PSNR as we change (a) the stream compression factor β and (b) the resulting decoded video quality. The linearity/superlinearity between T_P , T_U and β shown in (a) are consistent with the result predicted by the model.

5. CONCLUSION

We have investigated server transcoding within a server-assisted accelerated channel change framework for IPTV. Analytical and simulation results suggest that the stream compression factor affects linearly the saving in the channel change latency, and superlinearly the saving in unicast duration (or data size). This suggests that potentially large gains in system scalability can be achieved by moderately transcoding the unicast burst streams.

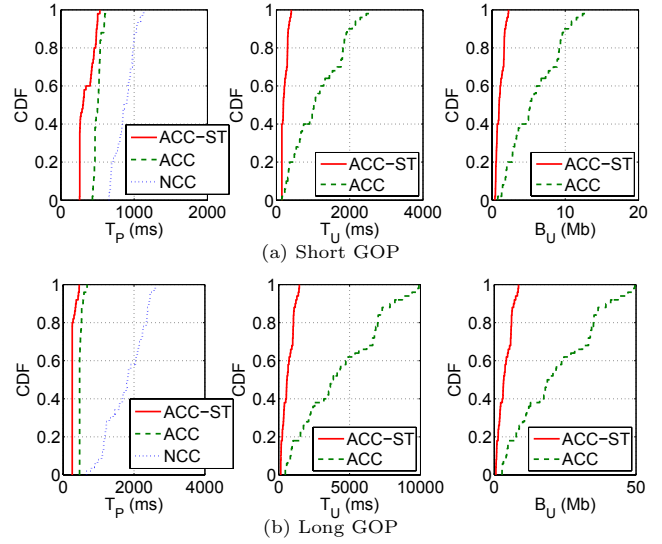


Figure 4: Simulation results: CDF of channel change latency T_P , unicast duration T_U and unicast data size B_U at e -factor $e = 0.2$ and (a) GOP size 15 (short), (b) GOP size 60 (long).

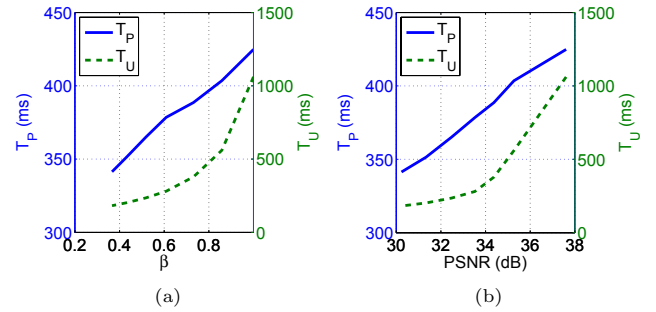


Figure 5: Simulation results: Resulting channel change latency T_P and unicast duration T_U as we change the stream compression factor β (a) and the resulting decoded video quality (b).

6. REFERENCES

- [1] RFC 3376: Internet group management protocol, version 3. Available online: <http://www.ietf.org/rfc/rfc3376.txt>.
- [2] A. C. Begen, N. Glazebrook, and W. V. Steeg. A unified approach for repairing packet loss and accelerating channel changes in multicast IPTV. In *Proc. IEEE Consumer Communications and Networking Conference (CCNC)*, 2009.
- [3] Y. Bejerano and P. V. Koppol. Improving zap response time for IPTV. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2009.
- [4] H. Fuchs and N. Färber. Optimizing channel change time in IPTV applications. In *Proc. IEEE Int. Symp. Broadband Multimedia Systems and Broadcasting*, 2008.
- [5] H. Fuchs, U. Jennehag, and H. Thoma. Subjective evaluation of low resolution tune-in streams for IPTV fast channel change. In *Proc. IEEE Int. Symp. Broadband Multimedia Systems and Broadcasting*, 2009.
- [6] Y. Lee, J. Lee, I. Kim, and H. Shin. Reducing IPTV channel switching time using H.264 scalable video coding. *IEEE Trans. on Consumer Electronics*, 54(2):912–919, May 2008.
- [7] C. Sasaki, A. Tagami, T. Hasegawa, and S. Ano. Rapid channel zapping for IPTV broadcasting with additional multicast stream. In *Proc. IEEE Int. Conf. Communications (ICC)*, 2008.
- [8] B. VerSteeg, A. Begen, T. VanCaenegem, and Z. Vax. Internet draft: Unicast-based rapid acquisition of multicast RTP sessions. Available online: <http://tools.ietf.org/html/draft-ietf-avt-rapid-acquisition-for-rtp>.