Reducing Channel-Change Times with the Real-Time Transport Protocol

In a multicast IPTV distribution network, each channel is offered in a different multicast session, and the IP set-top box joins the respective session when the viewer tunes to a new channel. Due to delays associated with network components and encoding schemes, the time difference between the channel-change request and when the new channel shows up on the screen can be annoyingly large. This article examines the Real-Time Transport Protocol (RTP) in IPTV networks and describes how RTP and its control protocol can help reduce channel-change times.

Ali C. Begen, Neil Glazebrook, and William Ver Steeg
Cisco

Service providers are continually increasing the number of TV channels they offer their subscribers. Despite digital video recorders' widespread use, the consumption of linear TV content has continued to increase as more channels become available. Recent data collected from a major cable TV deployment in the US show that viewers change channels on average 25 times a day. The data also show that some viewers change the channel almost every minute and others several times a minute during peak channel-change times, such as at the top of the hour. As channel line-ups get richer, we could very well observe increasing channel-change frequencies.

In conventional analog broadcast, TV tuners can complete a channel change in a single frame time and render the next frame of the new channel immediately. Thus, viewers usually feel as though zapping, or channel-change, times are virtually instant. Because conventional analog television is a broadcast medium, zapping times don't vary, no matter how many viewers change channels at the same time. In digital cable and satellite broadcast, the zapping demand — that is the aggregated number of channel-change requests — doesn't have an impact on zapping times, either. However, compared to its analog counterpart, digital broadcast does have longer zapping times due to compression and encryption applied to the content.

 1089-7801/09/$25.00 © 2009 IEEE

In IPTV — the delivery of broadcast-quality television programming over IP — the situation is even more complicated because the content isn't actually broadcast, but is instead delivered using multicast. Zapping times increase due to several additional network operations, such as admission control and multicast distribution. The load on network elements such as routers and switches as well as video sources varies the delay the network introduces. Thus, the zapping demand can affect channel-change times in IPTV. This is a major obstacle in IPTV services' wide adoption.

Here, we examine how the Real-Time Transport Protocol (RTP) can help reduce channel-change times.

The Channel-Change Problem

Since its early days, a critical challenge in the IPTV space has been to provide short and near-constant zapping times despite the high demand during peak viewing hours. Both industry and academia have approached this issue in different ways (a comprehensive list is available elsewhere[1]). Proposals so far include solutions that are implemented purely at the video or network level, although hybrid proposals that combine video- and network-level solutions have recently gained traction. Some of these proposals have been implemented commercially, although operators are reporting performance issues in large-scale IPTV deployments.

IPTV providers need a scalable solution for accelerating channel changes. They want to provide the highest quality of video experience for their subscribers to help reduce quality-related churn. At the same time, they want to minimize their capital and operational expenditures as they grow their service areas. To address scalability concerns, the IETF's Audio/Video Transport (AVT) Working Group is developing a standard solution that lets receivers rapidly acquire RTP-based multicast sessions. Because emerging IPTV networks already support RTP transport, the new standard will let RTP readily provide channel-change acceleration for IPTV in addition to features such as instrumentation and loss repair. Using existing protocols and mechanisms in a versatile way will enable faster deployment and better interoperability.

RTP and RTP transport already benefit IPTV networks by supporting packet-loss repair in real time and monitoring reception and transport quality. We can naturally extend RTP's existing loss-repair concepts to provide accelerated channel-change services.

Overview of RTP

The IETF first specified RTP in 1996 to provide end-to-end transport for real-time data services over unicast and multicast networks. Over the years, the IETF has uncovered several issues regarding RTP's initial rules and algorithms. To improve its scalability, the IETF updated the RTP specification with RFC 3550 in 2003.[2] With several additional specifications to date, RFC 3550 is currently used in most audio and video communications applications and emerging large-scale IPTV deployments.

Although it isn't required, RTP typically runs on top of the User Datagram Protocol (UDP)[3] and benefits from its checksum support to identify corrupted packets. Compared to a plain UDP transport, RTP provides the following primary services:

- *Payload type identification* lets applications identify the payload format — for example, the payload might be audio data encoded in the G.711 format or video data encoded in the H.264 format.
- *Sequence numbering* lets RTP receivers detect packets that have been lost during transport. Loss-repair methods use sequence numbers to restore the packet sequence.
- *Time-stamping* lets RTP senders and receivers match their clocks and enables synchronization among them. Time stamps are also useful for calculating delay jitter.

RTP provides these services through its default 12-byte header. However, applications can extend RTP functionality by using the header-extension mechanism.[4]

RFC 3550 also specifies the RTP Control Protocol (RTCP), which, similar to RTP, is independent of the underlying transport- and network-layer protocols and typically runs on top of UDP. RTCP's main purpose is to provide RTP applications with minimal control and identification functionality as well as a scalable quality-monitoring service for RTP transport. RTCP achieves the latter via sender, receiver, and extended reports.[5] Sender reports convey the basic transmission and reception statistics from the active senders in an RTP session. Similarly, re-

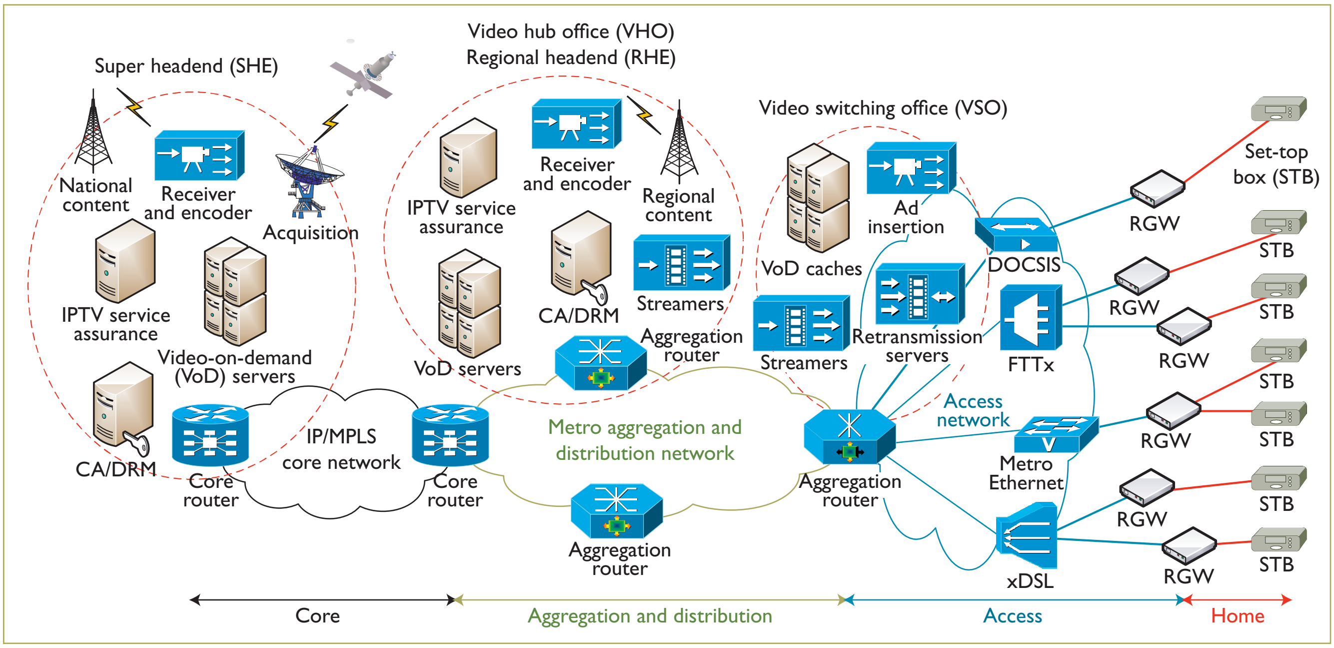

Figure 1. End-to-end IPTV architecture. We can see how the super headends, video hub offices, and video switching offices are interconnected to each other to deliver national, regional, and local content to customer premises.

ceiver reports convey the same from the receivers in the same RTP session. Extended reports, on the other hand, provide supplementary information for applications that desire more detailed statistics.

IPTV Transport Architecture and RTP Use

Unlike Internet video — often referred to as over-the-top video — which runs on top of the best-effort Internet, IPTV is a provider-offered, managed service with certain quality-of-service requirements. In IPTV, the provider doesn't simultaneously broadcast all channels to all viewers due to limited bandwidth on the access networks. Instead, each IP set-top box (STB) receives streams only for the requested channels or video-on-demand sessions. A complete IPTV system employs digital rights management, content and billing management, parental controls, and quality assurance.

Architecture

In an IPTV network, content originates from clustered components that are collectively called *IPTV headends*. Three common types of IPTV headends meet national, regional, and local content distribution requirements. *Super headends* (SHEs) receive and ingest content on the national level, typically from satellites and off the air. After processing and encoding this content, the SHEs distribute it to *video hub offices* (VHOs) over a core IP/Multiprotocol Label Switching (MPLS) network. VHOs aggregate national, regional, and local content with on-demand services, and serve metropolitan areas with populations between 100,000 and 500,000 homes. VHOs are connected to *video switching offices* (VSOs) over metro aggregation networks. VSOs distribute IPTV streams to the customer premises over access networks — for example, cable, metro Ethernet, and various fiber and DSL technologies. Figure 1 shows a typical end-to-end IPTV architecture.

Transport

In 2006, the ITU-T updated its specification for the MPEG2 Transport Stream (MPEG2-TS),[6] which has been the most ubiquitous encapsulation method used in broadcasting over both terrestrial and satellite networks. IPTV isn't any different, and most IPTV networks today use MPEG2-TS as well. MPEG2-TS can encapsulate many kinds of media, including the conventional MPEG2 content as well as the content encoded by the emerging MPEG4/AVC (also called H.264) codecs.[6]

MPEG2-TS is a multiplexed stream comprising digital audio and video content and its ancillary metadata. It carries the audio, video, and in-band control information in separate *elementary streams* (ESs) and provides an internal synchronization mechanism to assist audio and video decoders. These ESs are packetized to form *packetized elementary streams* (PESs). The resulting PES packets are then encapsulated inside the fixed-length (188-byte) TS packets. A straightforward way to transmit TS packets over an IP network is to combine seven of them into a UDP packet. With the addition of UDP (8 bytes) and IPv4 (20 bytes) headers, audiovisual content can be carried in 1,344-byte IP packets. Newer standards-compliant video architectures, however, add RTP encapsulation to these UDP packets as specified in RFC 2250[7] to take advantage of RTP's advanced features.

Note that MPEG2-TS was originally designed for low-jitter and low-loss environments, which aren't always possible for IP networks. During IP transport, packet losses might occur in the headends due to hardware and power failures, and in the core and distribution networks due to router, switch, and link failures. However, by far the majority of packet losses occur in access networks due to noise in the lines and in home networks due to contention and buffer overflows in residential gateways and IP STBs. IPTV is intended to provide entertainment-caliber video; losing even a single packet can create visible or audible glitches, so IPTV demands a very low packet-loss rate (for example, on the order of 10^{-7} to 10^{-6}), which requires applications to ensure that TS packets are delivered in order without errors.[8] Due to the real-time delivery requirements, the tolerance for delay jitter is also limited. An IP STB's de-jittering buffer can usually absorb a jitter of a few hundreds of milliseconds. However, any over-the-limit reordering or delay renders the incoming packets essentially useless in the decoding process. Having a limited tolerance to reordering and delay also requires agility in the loss-repair process.

Repairing Packet Loss

RTP transport enables loss-repair methods to recover missing packets in real time. In large-scale multicast distribution applications such as IPTV, reliability is often addressed above the transport layer. This is largely due to the fact that these applications are one-way in nature. Providing closed-loop reliability doesn't scale well at the transport level when the number of receivers is large, delay tolerance is small, or both. The IETF's Reliable Multicast Transport (RMT) Working Group has developed several standardized solutions for repairing packet loss in multicast distribution applications.[9] Inspired by these solutions, IPTV standards are currently using two main approaches for loss repair: *forward error correction* (FEC) and local retransmissions from distributed repair servers.

In FEC-based loss repair, the media source or the FEC encoder sends a certain amount of redundant data that protects the source content to the receivers, either proactively — that is, without receiving an explicit request from the receivers — or on demand. The receivers might fully recover the lost packets, provided that they've received a sufficient amount of source and repair data. The recovery probability increases with the amount of redundant data, but we can't arbitrarily increase this amount and still keep transmission overhead at an acceptable level. When used together with multicast, FEC can provide reliable transmission to a potentially very large multicast group in a scalable manner.[10,11] However, multicasting FEC to the whole group regardless of which receivers actually need FEC creates an exposure problem. In bandwidth-limited access networks, unnecessary overhead from FEC can be prohibitive.

Retransmission is a fundamental loss-repair method that's been widely used in reliable data transfer protocols such as TCP. Using retransmission generally consumes less bandwidth than FEC because the source or the repair server sends a retransmission only upon receiving an explicit request from a receiver. Provided that the retransmission round-trip delay isn't prohibitively large, retransmissions are often preferred for repairing packet loss in access networks with limited bandwidth. In RTP-based single-source multicast (SSM) sessions, the retransmissions can be handled by local repair servers rather than the source itself.[12] This approach not only provides a short retransmission delay but also avoids potential network implosion problems at the source.

Due to imperfections in both FEC and local retransmissions, using the two in combination often provides the best loss-repair performance. A well-engineered hybrid system can repair both random and congestive losses using less bandwidth and in less time than either method alone. However, devising the right balance between them is a difficult problem and open to further research.

Quality Monitoring

Quality assurance is an integral part of IPTV services. Due to strict quality requirements, IPTV providers must continuously monitor their networks, including the headend hardware, networking infrastructure, and software components. When possible, providers should identify potential sources for impending problems before service disruptions occur and take necessary action through infrastructure upgrades and additional provisioning. If unexpected problems occur, the network should be able to isolate the problem's origin and continue to service subscribers through a backup plane until the service provider can fix the problem.

Monitoring specific hardware components' functionality is usually achieved via vendor-specific tools. However, at the transport level, service providers can monitor RTP flows via standard RTCP reports. As we explained previously, RTCP sender, receiver, and extended reports provide detailed transmission and reception statistics in a scalable way. By collecting reports from different components throughout a multicast distribution tree and processing them, IPTV providers can easily locate transmission abnormalities in their distribution networks. Similarly, individual RTCP receiver/extended reports from subscribers extend IPTV transports' visibility beyond the service provider network into the subscriber premises and are invaluable assets in addressing subscriber-specific complaints.

Channel-Change Acceleration with RTP

Two issues that are most relevant to IPTV subscribers are glitch-free audiovisual quality and short zapping times. The former issue is becoming increasingly important given the increase in the penetration of high-definition content and large TV displays. However, dealing with it is relatively easier using the techniques explained in the previous section. The zapping issue, on the other hand, is far more complicated. Fixing it requires optimizing several IPTV system components, from content compression to its encryption, and from multicast operations to buffering requirements.

Consider an SSM session that's delivering a specific channel in an IPTV network as an MPEG2-TS encapsulated in an RTP stream. When an IP STB wants to tune to this channel, it first leaves the multicast session for the current channel it's tuned to and then joins the new multicast session. Leaving a multicast session is usually a fast operation, and the join process often takes less than 200 ms. However, after the join, a finite amount of time exists before the IP STB can start usefully consuming the audiovisual data sent in the multicast session. Before the decoding starts, the IP STB must first acquire certain reference information. This information comprises several items: a description of what the source is sending over the transport stream; encryption information, including keys for the selected program; an anchor frame from which the decoding process can commence; and any other information required to process the data in the multicast flow.[1] As Figure 2 shows, the reference information is conventionally sent periodically in the multicast session at points we refer to as *transport stream random access points* (TSRAPs).

Because the IP STB can't know what information is currently being transmitted in the multicast session before it joins, it might join right before a TSRAP — in which case, the waiting time for the reference information will be minimal — or right after, in which case the IP STB needs to wait for the reference information to appear again in the stream, resulting in a maximal waiting time. In some applications, the reference information might be sent less frequently to save bandwidth or might not be contiguous in the flow but rather dispersed over a large period for various reasons, including encoding preferences. All these factors often increase the time required to obtain the reference information.

Figure 3a shows one way to reduce the waiting time. A local retransmission server has already been receiving the multicast data for potential future retransmissions. The IP STB concurrently joins the multicast session and requests the retransmission server to send a unicast retransmission of the data sent since an earlier TSRAP (including the reference information). Depending on the bandwidth and resources available in the system, the retransmitted data can be sent at a faster-than-natural rate to reduce the total retransmission time. Once the IP STB completely receives the information it was missing when it joined the multicast session, it might start decoding the new content. This approach could significantly reduce the reference information waiting time. However, the amount of reduction is strictly tied to the total amount of bandwidth the IP STB can receive. If the residual bandwidth remaining from the multicast flow is small, it might take a long time to completely retransmit the missing information.

Figure 3b shows a faster and more bandwidth-friendly alternative solution. In contrast to the scenario sketched in Figure 3a, here, the total bandwidth available to the IP STB is first used to acquire information missing since an earlier TSRAP from the retransmission server. Once the unicast burst catches up with the real-time multicast flow, the unicast retransmission ends, and the IP STB joins the multicast session. We can see in the Figure 3b approach that the retransmission server retransmits more data than in the Figure 3a approach, due to the deferred multicast join. However, because the retransmission session uses the total available bandwidth by itself, the IP STB ultimately achieves a faster synchronization with the multicast session.

Subtle differences exist between a traditional retransmission repairing a lost packet and the retransmission scenarios Figure 3 shows. For example, in the latter, the IP STB doesn't know exactly what it's missing. So, it simply asks the retransmission server to send whatever is necessary to get back on track with the new multicast session. Although the IP STB and retransmission server require a new set of protocol encodings to make this work,[13] the signaling mechanism still uses the same fundamental tools that the traditional retransmission-based loss repair requires. From a provider's viewpoint, using the same infrastructure and the same set of RTP/RTCP protocols for both loss repair and channel-change acceleration means less capital and operational costs.

Note that an IP STB can't start decoding content as soon as it receives the data beginning from a TSRAP, as we've assumed throughout this section. To compensate for the time until various buffers — such as de-jittering, loss-repair, and application-level buffers — fill to the required levels, the IP STB needs to wait. These delays usually add up and normally increase channel-change delays. Yet, a design that carefully shapes the unicast burst and minimizes the amount of duplicate data received from both this burst and the multicast session while guaranteeing a gapless transmission can reduce such buffering delays' impact even in the presence of unpredictable multicast join delays.

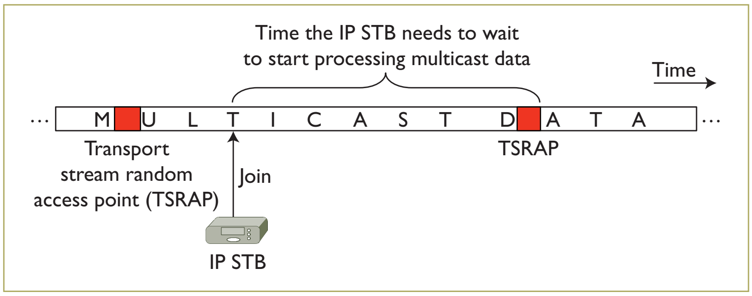

Figure 2. Typical multicast join. This IP set-top box (STB) randomly joins the multicast session but must wait until the reference information is transmitted in the session.

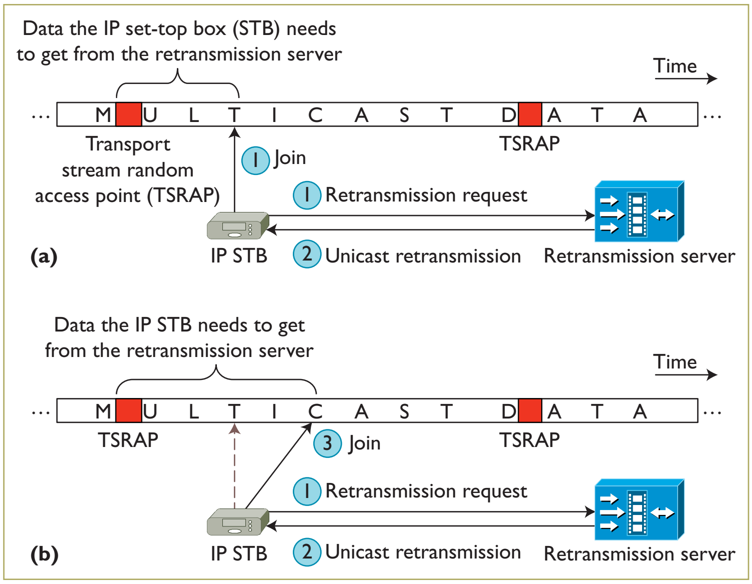

Figure 3. Methods for reducing zapping times. We can reduce these times with (a) concurrent multicast join and unicast retransmission, and (b) unicast retransmission followed by multicast join.

Performance Evaluation

We evaluated the performance of the channel-change acceleration method Figure 3b shows. Recall that when the retransmission server receives a unicast burst request from an IP STB, it must find an earlier TSRAP from which it will start bursting data. Let the *age* of a TSRAP denote how far that point is behind the multicast session at the burst-start time. The retransmission server doesn't necessarily have to — and usually can't — choose the youngest TSRAP in its cache; bursting from an immature TSRAP might not provide the necessary amount of data for filling the STB-side buffers. To avoid un-

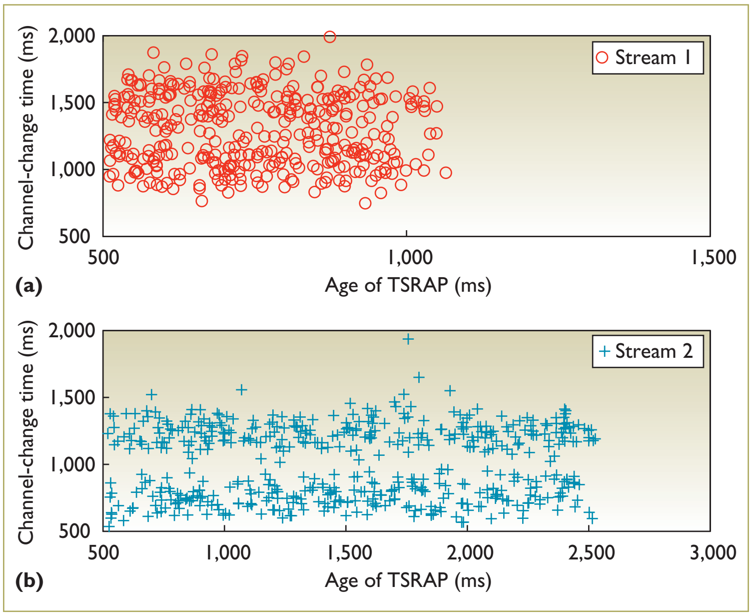

Figure 4. Correlation between transport stream random access point (TSRAP) ages and channel-change times when channel-change acceleration is enabled. We determined this correlation for (a) stream 1 and (b) stream 2.

derflows as well as unnecessary bursting, the server must choose the youngest mature TSRAP that conforms to the IP STB's buffer requirements. The IP STB might let the retransmission server know such requirements via the special retransmission request it sends for channel-change acceleration.[13]

In an ideal world in which an IPTV ecosystem has no random components, channel-change times would be almost deterministic. However, due to the encoding process' stochastic nature, video transport, multicast operations, STB processes, and randomly occurring channel changes, variability in channel-change times is often inevitable. Nevertheless, the approach we've laid out in this article can largely eliminate such variability due to the random time difference between an incoming request and an eligible TSRAP. To demonstrate, we conducted some experiments and analyzed the correlation between the chosen TSRAPs' ages and the channel-change times.

In the experiments, we used two standard-definition AVC streams encoded at 2 Mbps and 30 frames per second (fps). Although both streams were encoded from the same content, they had different characteristics in terms of the distance between TSRAPs. Stream 1 had a TSRAP every half second, whereas stream 2 had a TSRAP every two seconds. In the IP STBs, we allowed a 500-ms buffer for loss-repair operations. In other words, the TSRAP that the retransmission server chose had to provide a burst that would accumulate data worth at least 500 ms. During the unicast burst transmission, we used 20 percent more bandwidth.

Figure 4 shows the age of TSRAP and channel-change time data points corresponding to a sample set of 400 channel changes for each stream. Because the IP STBs performed channel changes independently, we observe a uniform distribution among the ages of the TSRAPs that the retransmission server chose. We can also see that the age values are less than a certain threshold in the ballpark of the loss-repair buffer size plus the period of the TSRAPs in the respective stream. Despite the varying TSRAP ages, however, the channel-change times are piled up within a certain margin that doesn't vary with age values. This suggests that channel-change times aren't correlated with the ages of the TSRAPs that the retransmission server selected, which means that no matter when the IP STB issues the request for channel-change acceleration or how frequently the TSRAPs repeat, the channel-change time variability will be due only to randomness in other components.

To further emphasize this point, let's consider the TSRAP acquisition delays in a conventional channel change – that is, when channel-change acceleration is disabled. As Figure 2 shows, the TSRAP acquisition delay is basically the time it takes for an IP STB to receive all the related information about the transport stream.[1] This delay usually varies within a large range because the source transmits different parts of this information at different times. In Figure 5, we also observe a large variation in TSRAP acquisition delays. Yet, a more important observation this figure shows is that channel-change times are highly correlated with the TSRAP acquisition delays. This means that an increase in the TSRAP acquisition delay will directly manifest itself as an increase in the channel-change time when channel-change acceleration is disabled.

Our results indicate that we can reduce the average and standard variation of channel-change times by up to 65 percent and 60 percent, respectively, when we use the method Figure 3b describes. More detailed results appear elsewhere.[1]

Designing a scalable retransmission server solution that can simultaneously perform loss repair and provide channel-change acceleration for thousands of IP STBs isn't trivial. Retransmission servers generally employ load balancing, high availability, and redundancy techniques to reliably satisfy high demand during peak viewing hours. In addition, retransmission servers use various rate control and policing mechanisms to prevent denial-of-service attacks and provide a proper and fair service to all IP STBs.

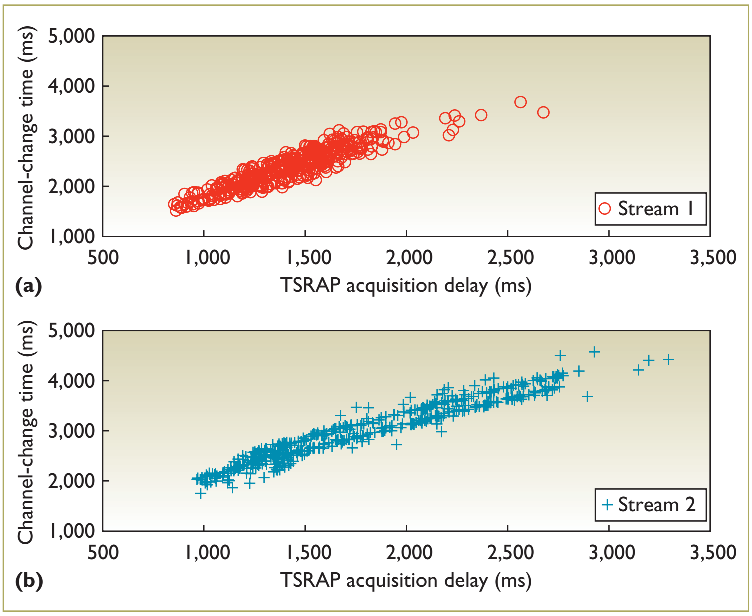

Figure 5. Correlation between transport stream random access point (TSRAP) acquisition delays and channel-change times when channel-change acceleration is disabled. We determined this correlation for (a) stream 1 and (b) stream 2.

Acknowledgments

We thank Eric Friedrich and Yufeng Shan for their contributions to this study.

References

1. A.C. Begen, N. Glazebrook, and W.V. Steeg, "A Unified Approach for Repairing Packet Loss and Accelerating Channel Changes in Multicast IPTV," *Proc. Consumer Comm. and Networking Conf.* (CCNC 09), IEEE Press, 2009, pp. 1–6.
2. *RTP: A Transport Protocol for Real-Time Applications*, IETF RFC 3550, July 2003; www.ietf.org/rfc/rfc3550.txt.
3. *User Datagram Protocol*, IETF RFC 768, Aug. 1980; www.ietf.org/rfc/rfc768.txt.
4. *A General Mechanism for RTP Header Extensions*, IETF RFC 5285, July 2008; www.ietf.org/rfc/rfc5285.txt.
5. *RTP Control Protocol Extended Reports (RTCP XR)*, IETF RFC 3611, Nov. 2003; www.ietf.org/rfc/rfc3611.txt.
6. *Generic Coding of Moving Pictures and Associated Audio Information: Systems*, ITU-T H.222.0, May 2006.
7. *RTP Payload Format for MPEG1/MPEG2 Video*, IETF RFC 2250, Jan. 1998; www.ietf.org/rfc/rfc2250.txt.
8. *IPTV Packet Loss Issue Report*, ATIS-0800005, Alliance for Telecommunications Industry Solutions, 2006; www.atis.org/docstore/product.aspx?id=22659.
9. "NACK-Oriented Reliable Multicast (NORM) Protocol," IETF Internet draft, Mar. 2009; http://tools.ietf.org/html/draft-ietf-rmt-pi-norm-revised-09.
10. A.C. Begen, "Error Control for IPTV over xDSL Networks," *Proc. Consumer Comm. and Networking Conf.* (CCNC 08), IEEE Press, 2008, pp. 632–637.
11. "DVB Application Layer FEC Evaluations," Digital Video Broadcasting document A115-TM 3783, 2007; www.dvb.org/technology/standards/a115.tm3783.AL-FEC_Evaluation.pdf.
12. "RTCP Extensions for Single-Source Multicast Sessions with Unicast Feedback," IETF Internet draft, Mar. 2009; http://tools.ietf.org/html/draft-ietf-avt-rtcpssm-18.
13. "Unicast-Based Rapid Synchronization with RTP Multicast Sessions," IETF Internet draft, Mar. 2009; http://tools.ietf.org/html/draft-versteeg-avt-rapid-synchronization-for-rtp-02.

Ali C. Begen is a software engineer in the Video and Content Platforms Research and Advanced Development Group at Cisco, where he participates in video transport and distribution projects. His interests include networked entertainment, multimedia transport protocols, and content distribution. Begen has a PhD in electrical and computer engineering from the Georgia Institute of Technology. He's a member of the IEEE and the ACM and served as the local chair for the IEEE International Conference on Semantic Computing 2008. Contact him at abegen@cisco.com.

Neil Glazebrook is the product manager for the Visual Quality Experience (VQE) products at Cisco. Contact him at nglazebr@cisco.com.

William Ver Steeg is the chief architect of IPTV for the Service Provider Video Technology Group at Cisco. Contact him at billvs@cisco.com.