Virtual RTCP: A Case Study of Monitoring and Repair for UDP-based IPTV Systems

Alejandra Soni García
Aalto University

Jörg Ott
Aalto University

Martin Ellis
University of Glasgow

Colin Perkins
University of Glasgow

***Abstract*—IPTV systems have seen widespread deployment, but often lack robust mechanisms for monitoring the quality of experience. This makes it difficult for network operators to ensure that their services match the quality of traditional broadcast TV systems, leading to consumer dissatisfaction. We present a case study of *virtual RTCP*, a new framework for reception quality monitoring and reporting for UDP-encapsulated MPEG video delivered over IP multicast. We show that this allows incremental deployment of reporting infrastructure, coupled with effective retransmission-based packet loss repair.**

I. Introduction

Recent years have seen the widespread deployment of high-quality Internet streaming video. Indeed, multimedia traffic is now reported as being the primary use of capacity in many networks, and this use is growing rapidly [15]. Streaming video services generally fall into one of two major categories: 1) unmanaged over-the-top streaming services, generally delivered via HTTP using an inter-domain content distribution network, and primarily focused on catch-up and on-demand services (e.g., the BBC's iPlayer); and 2) operator-managed IPTV systems providing a broadcast-replacement service within an ISP's network. We focus on managed IPTV services in this paper. These services generally deliver relatively high bandwidth (5Mb/s per channel is common) MPEG transport streams (TS) using intra-domain IP multicast. Some implementations encapsulate the MPEG TS frames directly in UDP packets for delivery, while others use the Real-time Transport Protocol (RTP) [14] to provide framing and reception quality feedback.

Given the competitive nature of the market, it is essential that service providers monitor the reception quality of their managed IPTV services. This is needed to ensure the customer experience matches that of their competitors, and of traditional broadcast television. This monitoring is directly supported by RTP-based systems through the use of the RTP Control Protocol (RTCP) [3], but many deployments use a minimal UDP-based encapsulation of MPEG TS frames, gatewayed from non-IP infrastructure, without RTP framing or reporting. It is desirable to have a solution that can provide some of the benefits of RTP-based deployments for UDP-based systems, to ease their transition to a more IP-centric infrastructure.

In this paper, we present a case study of *virtual RTCP*. This new framework shows how RTCP-like reception quality feedback can be provided for UDP-encapsulated MPEG transport streams. A loss estimation algorithm is proposed that uses the MPEG packet identifier (PID) and continuity counter (CC) header fields to estimate IP packet loss rates and loss burst durations. We show that this algorithm gives accurate packet loss rate estimates for networks operating within the regime that delivers usable media quality, and identify several pathological IPTV system behaviors that can disrupt the estimate and user experience. Virtual RTCP also provides equivalents to RTCP extended report packets to enable retransmission-based packet loss repair. We report on initial simulations that show the effectiveness of such repair.

II. Background

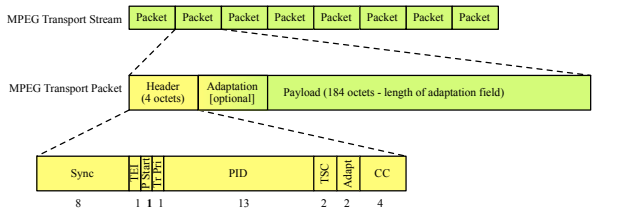
A. IPTV Content and MPEG Transport Streams

Managed IPTV systems have often grown out of existing digital TV services, whether based on cable networks or over-the-air broadcast. In some cases there is a direct business relationship: the ISP is also a cable operator; in others the ISP is competing with cable operators to provide similar service. This legacy has led many IPTV providers to adopt the familiar media formats and transport protocols used in prior digital TV services, primarily MPEG video [6] and transport streams [7].

MPEG transport streams deliver multiple audio and video elementary streams, with associated metadata for error detection, timing recovery, and stream, program, and channel description purposes. A transport stream is a sequence of transport packets, each 188 octets in length, comprising a header and payload (Figure 1). The header is typically 4 octets in length, and is followed by 184 octets of payload. An optional adaptation field can be included, depending on the value of the Adaptation Field Control bits (labeled "Adapt" in Figure 1), increasing the header size and reducing the payload.

Header fields of key interest are the PID, which identifies the particular elementary stream, and the continuity counter, which increments by one (modulo 16) for each packet with the same PID (not counting packets that contain no valid payload). The combination of PID and continuity counter lets us detect packet loss in elementary streams. Also important is the program clock reference, carried periodically in the adaptation field, that allows a receiver to reconstruct timing. See [7] for details of the other headers.

When implementing IPTV systems, it is common for seven MPEG transport packets to be encapsulated into one UDP packet for transmission, with no additional headers, giving a payload 1316 octets. This comfortably fits the typical Ethernet MTU of 1500 octets, even after UDP/IP headers are added.



Name	Size	Description
Sync Byte	8	To identify start of packet
Transport Error (TEI)	1	Packet contains error?
Payload Unit Start (P Start)	1	Packet contains start of data?
Transport Priority (Tr Pri)	1	Packet is high priority?
PID	13	Identifies elementary stream
Transport Scrambling Control	2	Packet is scrambled?
Adaptation Field Control	2	Adaptation field follows header?
Continuity Counter	4	Counts packets with same PID

Fig. 1. MPEG Transport Format and Header Fields

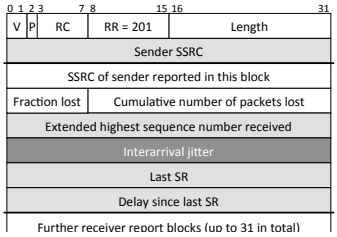


Fig. 2. RTCP Receiver Report format

B. RTP Data Transport and Control Protocols

The RTP framework [14] provides for timely delivery of audio-visual media across an unreliable IP network. There are two parts to the protocol: a data transfer protocol, and a control protocol (RTCP). The data transfer protocol provides framing, sequencing, timing recovery, and adaptation of the codec output to the dictates of a packet network. The mapping of MPEG transport streams onto RTP is defined by [5].

RTCP provides a reporting and control channel for each data channel. RTCP packets are sent every few seconds by all senders and receivers, with the exact interval randomized to avoid synchronization and scaled with the number of participants to avoid congestion. The types of RTCP packets defined in [14] include sender (SR) and receiver reports (RR). RR packets (Figure 2) report on the fraction and total number of packets lost, the highest sequence number received, the variation in interarrival times, the last SR timestamp, and the time since the last SR (the last two allow senders to calculate the RTT to receivers). SR packets extend RR packets with clock synchronization information, and counts of the total number of octets and packets sent. The combination of SR and RR packets provides basic quality of experience (QoE) reporting, suitable for network management and coarse-grained estimates of the user experience. Extensions provide NACK feedback [12] and retransmission [13].

III. Virtual RTCP

RTP/RTCP gives operators a powerful framework for media delivery and QoE management. This comes at a cost of some overhead and complexity, both in terms of a (small) bandwidth overhead for the additional headers and reports, and more importantly in a learning curve for operators who need to learn to make best use of the features of the new protocol. For this reason, some operators use a raw UDP encapsulation of MPEG transport streams to minimize the disruption while they integrate IP-based delivery networks into their operations.

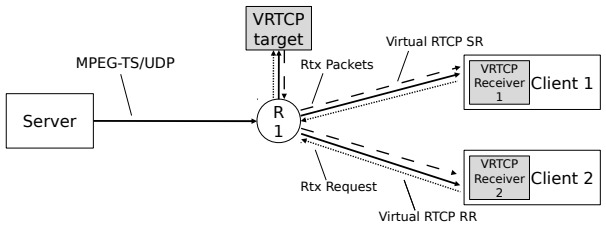


Fig. 3. Virtual RTCP components in an MPEG-TS/UDP streaming system

We introduce *virtual RTCP* (VRTCP). This is a new framework that aims to provide some of the benefits of RTP-based systems to operators using raw UDP encapsulation of MPEG TS. Figure 3 shows the conceptual overview for an IPTV streaming scenario with a server streaming MPEG-TS/UDP on the left and two clients on the right, interconnected by a router R1. Integrated into the client device, a virtual RTCP receiver observes and buffers incoming packets before passing them on to the client. The VRTCP receiver assesses the packet flow and generates RTCP reports on behalf of the unchanged receiver so that any monitoring, inference, and repair infrastructure continue its operation as with RTP streams. The VTRCP receiver may also be realized stand-alone as a bump in the wire: in this case it performs all the monitoring and provides feedback on behalf of the client; it may also buffer media data and splice the original media feed and repair packets together before forwarding the resulting packet stream to the client.

Since the server will not understand RTCP, another entity is required as the destination for the VRTCP traffic: we introduce a node called *VRTCP target*, in analogy to the *feedback target* used for feedback collection in RTP-based IPTV systems [10]. In addition to collecting feedback VRTCP packets for monitoring purposes, the VRTCP target may also receive the multicast media stream and send out repair packets on behalf of the sender. To assist the VRTCP receiver in its statistics calculation, the VRTCP target may also generate VRTCP sender reports on behalf of the origin sender. Similar to feedback targets, many VRTCP targets may be deployed to allow the system to scale [3].

A. Loss Detection

To create RTCP reception statistics, we need to determine if packets are missing. One option could be looking at the inter-arrival times of the packets and inferring losses from gaps. However, this would only work with streams exhibiting a constant packet rate at the source and a sufficiently low jitter (e.g., smaller than the inter-packet spacing). As this cannot be guaranteed in practice (see our packet traces of an IPTV service below), we perform minimal inspection of the packets.

Even though RTP headers are not available, the MPEG-TS headers offer some cues for loss detection as shown in Figure 1. The MPEG-TS header includes a 4-bit sequence number, the

continuity counter (CC) that counts MPEG frames per elementary stream, the latter of which is indicated by a 13-bit *PID*. An IPTV stream comprises multiple elementary streams that are multiplexed into packets in diverse combinations. In ideal operation, the CC field increments by one between adjacent non-empty frames of the same PID, which results in a delta of seven between two consecutive packets if only one elementary stream is contained in a packet. Due to multiplexing, these numbers may vary but we can usually identify a *primary* elementary stream that dominates in frequency (video) and also a *secondary* one (audio) [1].

Loss detection works basically by comparing the (PID,CC) pairs of two consecutively received packets and determining if there is a gap for the primary and (if available) secondary PID.[2] While this indicates a loss of one or more packets, for reporting we also need to determine how many packets were lost. This proves to be complex because we do not know the composition of the missing packets (how many and which elementary streams were included) and therefore need to estimate. We iteratively devised an algorithm that performs this estimate based upon the primary and secondary PIDs, assuming that a missing packet would usually contain 5, 6, or 7 frames of the primary, no more than one frame of the secondary stream, and filled up with frames of other streams. The algorithm picks the three main deltas (the three most common number of frames contained per packet) for the primary PID, and, since the secondary PID appears less often, it only takes the highest most common number of frames (empirical investigation with real-world data shows that including further elementary streams increases complexity, but does not provide additional gain). Then, it performs a search of possible permutations of frames and yields possible combinations of loss runs indicating the number of lost packets. Since multiple combinations are possible, the algorithm is optimistic and returns the lowest loss run length because we assume that the network is well-provisioned and losses will be rather short. Figure 4 summarises the algorithm.

There are two exceptional cases. Firstly, if we do not receive any packets for longer than $T_{thresh} = 1$ s, we report a *long loss* as no sensible guess is possible (obviously for VBR streams, but also for CBR due to timing jitter). Secondly, in some cases, our algorithm will not find a meaningful estimation for the losses and will report an error, but we found this to be rare. These cases are not yet considered in VRTCP.

B. Packet Identification

In addition to loss detection, we need to identify packets to synchronize between the VRTCP receiver and the VRTCP target, e.g., for requesting retransmissions. One option would be using the combinations of (PID,CC) fields across all frames in a packet; but this is insufficient because they repeat too frequently. One could additionally include the UDP checksum,

```
for a given IP packet and its seven MPEG TS Headers
  if (discontinuity found in a CC from the secondary PID)
    for i= 1 .. 5 possible lost packets
      if( Previous CC + First delta == Next expected CC value)
        return number of packets lost
        success secondary PID= true
      else if(success secondary PID==false and i > 5 packets)
        return error

  if(discontinuity found in a CC from the primary PID)
    for i= 1 .. 5 possible lost packets
      for i= 1...3 possible deltas
        look for the next expected CC value resulting from the
        possible permutations of deltas
        if( Previous CC + Delta combination == Next expected CC value)
          return number of packets lost
          success primary PID= true
        else if(success primary PID==false and i > 5 packets)
```

Fig. 4. Loss detection algorithm

but we find this to be unreliable, since the UDP checksum is often not used (set to zero) in IPTV systems. We choose to calculate MD5 hashes over the packet content for which our empirical evaluation shows occasional (but infrequent) duplicates and those are usually spaced far apart. The 160 bit checksums are then used in VRTCP packets.

C. Virtual RTCP Packets

Figure 2 showed a regular RTCP receiver report used for coarse-grained reporting of reception statistics. The fields in white can be populated by the VRTCP receiver from observing the UDP packet stream as discussed above. The inter-arrival jitter field (shaded dark) cannot be calculated since MPEG/UDP packets do not carry timestamp information. The gray fields can be roughly estimated if a VRTCP target complements the media stream with VRTCP sender reports.

VRTCP Sender Reports (VSRs) are regular RTCP SRs complemented by a VSR ID packet in a compound packet. VSR ID packets contain an MD5 hash of the last data packet received by the VRTCP target. The SR packet will provide a time stamp and a receive count reference for the VRTCP receiver to calculate statistics, a virtual sequence number (VSN) assigned to this packet by the VRTCP target, and an SSRC field. This means that the RTT (and possibly loss) will be calculated against the VRTCP target, but for timing repair requests, this is the most interesting segment of the network. VRTCP sender reports are sent using a separate multicast group, establishing a virtual RTP session with the unicast RRs to the VRTCP target (as in [10]).

VRTCP Receiver Reports use the VSNs received and relate them to the packets using the MD5 hash. They can thus precisely determine any difference in the number of packets received at the VRTCP target and the receiver and calculate local loss. In addition, the delta between two VSNs gives an approximation of the number of packets to be received between the two VRTCP SRs, even if there is an—expectedly constant—latency offset between VRTCP SR and data packets. This complements the loss calculation described above, which can also account for loss upstream of the VRTCP target.

VRTCP NACKs are used to signal repair requests to the VRTCP target. The VRTCP receiver includes the IDs of the

[1] If multiple suitable (secondary) audio streams exist, e.g., in multi-lingual programs, one of them is chosen at random.

[2] Our algorithm doesn't check for re-ordered packets, but these are rare [4], and manual review of the larger losses showed they were of different origin.

last packet received before and the first packet received after a loss period, based upon which the VRTCP target (which is caching the received packets) can determine the packets to retransmit. It also carries a NACK sequence number.

Retransmissions are sent via unicast and, in contrast to the original packets, these packets use RTP headers so that proper accounting can be done for the repair channel. Each retransmitted packet carries an RTP extension header that includes the NACK sequence number, the total number of packets to be retransmitted for this NACK, and this packet's number. This information allow the receiver to properly split the original and the retransmission streams in the receiver buffer before delivering the packets to the client.

IV. Trace-based evaluation

We collected traces from an operational IPTV service in Finland using a Soekris net5501 system running FreeBSD, connected to the ADSL access router provided by the ISP. We extended *rtpspy* [11], a tool for capturing (multicast) RTP streams, to support raw UDP encapsulation and extract MPEG header information. We carried out three series of measurements across 14 active IPTV multicast channels: (A) 27 March–9 April 2011 (1 hour per channel), (B) 27 July–21 August 2011 (2 h), and (C) 23–27 November 2011 (2 h).

A. Trace Characteristics

We first describe the characteristics of the IPTV data, and compare these against previously published results [9], [1].

We examine the bit-rates seen over one-second intervals in the traces, as in [9]. Figures 5(a) and 5(b) show the distribution of bit-rates for each channel. Observe that for the channels in Figure 5(a), the distributions are more spread out, indicating variable bit-rate (VBR) traffic, while in Figure 5(b), the distributions are strongly peaked, indicating constant bit-rate (CBR) traffic. These results are consistent with those in [9], showing VBR channels with one dominant bit-rate (referred to as 1-VBR in [9]), as well as one channel with a bimodal bit-rate distribution (referred to as 2-VBR in [9]). The presence of VBR streams makes the process of estimating packet loss non-trivial, as discussed in Section III-A.

Using the receive timestamps in the traces, we can study the packet inter-arrival times (IATs), to identify variations in IATs (which signify disruption due to the network). Moreover, any packet losses will be visible as larger-than-normal values in the series of IATs. Figure 5(c) shows representative complementary cumulative distribution function (CCDF) plots for all the IATs observed for three channels (one each from CBR, 1-VBR, and 2-VBR). The range of data in the plot shows that almost all packets are received with small IATs, under 30ms, while there are a few very large IATs seen for each channel. These clusters of IATs above 30ms are likely periods of packet loss. Note that long outages, as well as short bursts of loss, are present, with 23 of the 47 traces containing loss periods longer than T_{thresh} (across all channel types). This differs from [4], where simulated wide-area IPTV-like flows were sent to 19 different residential hosts over a one year period (2009–2010), and only short outages were seen; the long bursts appear due to problems with the IPTV service, not network packet loss.

Channel	Mean ($\times 10^{-4}$)	SD ($\times 10^{-4}$)	#packets
35 (1-VBR)	157.33	20.29	15053681
110 (1-VBR)	22.41	12.12	8137960
54 (2-VBR)	3.55	3	13344005
50 (1-VBR)	4.80	2.75	14251955
111 (1-VBR)	1.89	1.14	17267010
53 (1-VBR)	0.74	1.06	16499628
52 (1-VBR)	0.22	0.48	1358751
24 (CBR)	405.55	378.72	32370345
95 (CBR)	6.02	16.612	17492325
51 (CBR)	3.44	1.73	14029850
46 (CBR)	1.04	1.18	14213946
47 (CBR)	1.04	1.18	16768522
44 (CBR)	0.66	0.90	13614755
13 (CBR)	0.44	0.89	20270112

TABLE I
Loss Fraction Estimation Per-Channel

The presence of losses is also in contrast to other studies of commercial IPTV, although since our receiver is connected via ADSL (rather than connected within the network core [9], or at a home connected by Ethernet [1]), this is not too surprising.

B. Loss Inference Performance

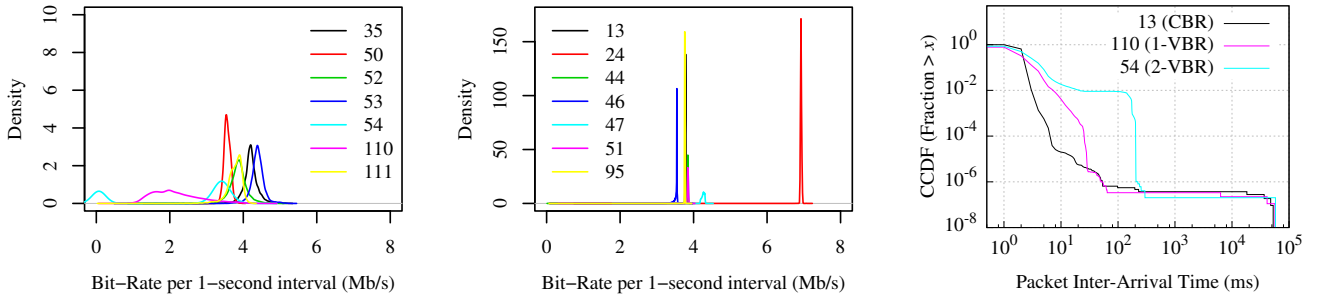
We apply the loss estimation algorithm (Section III) to the IPTV traces. The results show that the algorithm performs well for traces not containing large loss periods or multiplexer errors.[3] Table I shows the mean and standard deviation of the estimated loss rates, with 10 logs per channel from trace series (A) and (C). The errors found in the estimations and the large loss periods (over one second) described in Section IV-A were not taken into account for this calculation. In general, there are two types of channels: those where the loss estimation showed significant variation (e.g., Figure 6(a)), and those where the estimation remained somewhat constant (e.g., Figure 6(b)).

Variations are due to limitations in the estimation accuracy:

a) Lost packets may contain combinations of elementary streams that are not among the three main ones found for the primary PID and hence not considered by the algorithm, which will return an error or make an incorrect estimation of the number of lost packets. Statistics for this case reveal a wide range of such occurrences: 0–12% for the primary and 0–57%(!) for the secondary PID. Fortunately, the former is more important since the primary PID is present in almost all packets.

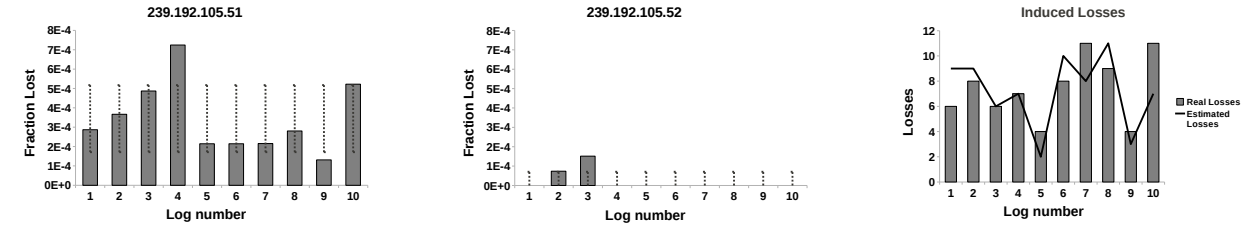
b) A loss burst may include $k \times 16$ frames of the primary PID, which means that no loss is noticeable for this PID. If the same holds for the secondary PID, e.g., because no such frame was sent, the loss goes undetected. Our analysis of trace series A) shows that, on average, a CC in the first frame of a packet repeats every 16–19 packets, but repetitions with shorter distances do occur. Those matter if they are ≤ 5 packets apart as this is the

[3]The latter originate prior to IP encapsulation at the content provider and result, e.g., in CC gaps within an IP packet. They do not reflect network problems and are therefore not considered further in this paper.



(a) Bit-rate Distributions (VBR channels) (b) Bit-rate Distributions (CBR channels) (c) CCDFs of IATs from different bit-rate classes

Fig. 5. Per-Channel Bit-rate / IAT Results



(a) Channel with variable loss estimation (b) Channel with less variable loss estimation (c) Induced Losses

Fig. 6. Behavior of the algorithm in the presence of induced losses

loss burst search bound of our algorithm. These cases occur in 5–40% (mean: 21%) of all repetitions and thus may impact the detection of discontinuities.

c) Similarly to case (b), two bursts of different lengths (e.g., 4 and 20), show the same CC delta, meaning that the actual length cannot be determined for sure. As noted above, our algorithm is optimistic and will estimate the lower burst length. This seems reasonable, given the short loss bursts reported on residential links [4].

We also verify the performance of the algorithm in a controlled environment. We use 10 logs (2 h each) across 7 channels, where the logs show no CC discontinuities, in a trace-driven simulation with controlled random loss generation, and observe how well the algorithm estimates the losses. Figure 6(c) shows the results of ten simulations with different random seeds for each log. We find that the loss estimation algorithm works well if the MPEG stream does not suffer from multiplexer errors and loss bursts are short, as expected in a well-managed network.

The data analysis shows that large bursts happen repeatedly, although a previous study of loss on residential links has shown that loss bursts are typically short [4]. We believe that the large loss bursts observed in the traces are a sign of significant problems with the underlying IPTV service, rather than a problem in the algorithm—it would be able to report their existence and thus indicate severe quality impairment.

V. Evaluation of Repair Performance

Finally, we present initial simulation results showing how VRTCP can be used to facilitate feedback and loss repair within an MPEG/UDP-based IPTV system.

A. Simulation Setup

We use ns-3.11 to simulate a minimal IPTV distribution system. Our simulation comprises a single IPTV sender, a router, the VRTCP target as feedback target and retransmission server, and two IPTV receivers as shown in Figure 3. The simulation system uses the protocols for media transmission, feedback, and retransmissions as described in Section III. The sender sends plain multicast UDP-encapsulated MPEG-TS frames that are read from a trace file but, to simplify the setup, includes (rather than calculates) the MD5 checksums in the packets, along with a packet sequence number for reference.

The router forwards the packets to the VRTCP target and to the receivers. The VRTCP target buffers received packets for five seconds for later retransmissions. It reacts to NACKs by locating the packets specified in the NACK request and retransmitting the ones in-between via unicast to the receiver. Receivers look for discontinuities in the primary and secondary PIDs of each packet, mark those in their receive buffer, and send a retransmission (NACK) request to the VRTCP target with the MD5 hashes of the packets received before and after the loss. When receiving a retransmission, they locate the mark in their buffer to insert the packet(s).

We record the accuracy of NACK in terms of a) errors for mismatching IDs for the retransmissions, and b) for deviations in the “total number of packets expected” per NACK (using the sequence numbers), as well as the repair performance measured in terms of lost packets not recovered. For the evaluation, we run simulations using a group of 7 “lossless” 2 h traces, as per section IV-B, for which we create loss bursts with a probability in the same order of the traces we observed in practice (ignoring traces with long loss bursts).

B. Initial Simulation Results

In our simulations, we find most induced losses are detected and repaired. As a representative example, for one trace comprising almost 2.7 million packets, the two receivers lose 24 and 46 packets, and generate on average one NACK for two lost packets. Only 2 and 3 lost packets, respectively, could not be repaired. The main cause of unrepaired losses was due to issues properly identifying those losses. We found three aspects to this: First, the VRTCP target needs to properly identify the packets to retransmit based upon the two MD5 hashes received: if hashes occur repeatedly (e.g., when packets containing empty frames are sent), the references may not be unique. In such cases, the VRTCP target would pick the most recent matching packet. While this is possible (duplicates exist in the traces), such losses did not occur our simulations.

Second, the receiver is not always able to characterize a loss burst correctly: in its NACK packet, it includes the hashes of the last packet received before, and the first packet after, a loss burst. However, a discontinuity may not be detected immediately: a loss burst may lead to a loss of a secondary PID but does not cause a discontinuity for the primary PID; e.g., because $k \times 16$ primary frames are lost, or because a lost packet did not contain a primary frame. Such losses will only be detected when receiving the next secondary PID. In both cases, the delayed loss detection causes the NACK to identify two consecutive packets with nothing to retransmit in between, while the original loss goes undetected.

Finally, some cases showed that the detection of discontinuities at the receivers did not trigger retransmission requests whenever two consecutive CCs were the same (since there can be frames without payload). This can typically be avoided by inspecting the adaptation field of the MPEG-TS header.

VI. Related Work

Mellia & Meo [9] capture traces from within an operational IPTV network (at the ISP access router, rather than in the home). They report excellent performance, showing that there is little performance degradation up to the last-mile. Baltoglou *et al.* [1] discuss active measurement results captured within a commercial IPTV network serving customers using an Ethernet last-mile, also finding excellent performance, with no packet losses and consistent timing jitter. The difference between their results and those in Section IV is due to the difference at the physical layer, since our receiver is connected via ADSL. A study of "download and play" (D&P) and streaming from residential IPTV subscribers with ADSL, Cable, and Fiber access was conducted by Won *et al*. [17], concluding that D&P is a good interim solution for operators until full-managed multicast networks are deployed.

Mahimkar *et al.* [8] propose a system to automatically diagnose problems within a commercial IPTV network, using log data of various types, including SNMP data, video quality reports from dedicated monitors, STB logs, and customer-service tickets. It is expected that VRTCP could fit into such a framework, to give finer-grained reporting of the quality experienced by end-points. Other work on inferring packet loss comes from work on network tomography [2], although this has mostly focused on correlating across multiple receivers, and relies on being able to identify packets (e.g., using RTP sequence numbers). Inference of QoE, rather than network-level metrics like packet loss and jitter, is discussed in [16], although in that study the packet loss and timing impairments are induced as part of the experiment, and are therefore known.

VII. Conclusions

MPEG transport streams have proved their worth in non-IP media distribution networks over many years. However, the switch to IP-based distribution requires rethinking many issues relating to quality of experience, due to the best-effort nature of the underlying network. We introduce Virtual RTCP to provide incrementally deployable, basic network monitoring and repair capabilities for legacy MPEG/UDP streaming. We find that even a simple algorithm is capable of assessing packet losses and, in conjunction with retransmissions, repairing them for networks operating in a regular regime. While larger loss bursts may not be exactly quantifiable, they (and thus the existence of larger network issues) can be reported. As such, virtual RTCP provides an important stepping-stone between traditional video distribution networks, and a more full-featured RTP-based IPTV distribution infrastructure.

Acknowledgements

This work was supported by Cisco Research and the UK EPSRC. Thanks to David Ros for suggesting the UDP checksum as an additional packet identifer.

References

[1] G. Baltoglou, E. Karapistoli, and P. P. Chatzimisios. Real-World IPTV Network Measurements. In *Proc. IEEE ISCC*, 2011.
[2] A. Begen, C. S. Perkins, and J. Ott. On the Scalability of RTCP-Based Network Tomography for IPTV Services. In *Proc. IEEE CCNC*, 2010.
[3] A. Begen, C. S. Perkins, and J. Ott. On the use of RTP for monitoring and fault isolation in IPTV. *IEEE Network*, 24(2), 2010.
[4] M. Ellis, C. S. Perkins, and D. P. Pezaros. End-to-End and Network-Internal Measurements of Real-Time Traffic to Residential Users. In *Proc. ACM MMSys*, 2011.
[5] D. Hoffman, G. Fernando, V. Goyal, and R. Civanlar. RTP payload format for MPEG1/MPEG2 video. IETF, January 1998. RFC 2250.
[6] ISO. Generic coding of moving pictures and associated audio information: Video. ISO/IEC 13818-2:2000, December 2000.
[7] ISO. Generic coding of moving pictures and associated audio information: Systems. ISO/IEC 13818-1:2007, October 2007.
[8] A. Mahimkar et al. Towards Automated Performance Diagnosis in a Large IPTV Network. In *Proc. ACM SIGCOMM*, 2009.
[9] M. Mellia and M. Meo. Measurement of IPTV traffic from an operative network. *Eur. Trans. Telecomms.*, 21(4), 2010.
[10] J. Ott, J. Chesterfield, and E. Schooler. RTCP Extensions for SSM Sessions with Unicast Feedback. IETF, February 2010. RFC 5760.
[11] J. Ott and D. Kutscher. Drive-thru Internet: IEEE 802.11b for "Automobile" Users. In *Proc. IEEE INFOCOM*, 2004.
[12] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey. Extended RTP Profile for RTCP-Based Feedback. IETF, July 2006. RFC4585.
[13] J. Rey, D. Leon, A. Miyazaki, B. Varsa, and R. Hakenberg. RTP Retransmission Payload Format. IETF, July 2006. RFC4588.
[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. IETF, 2003. RFC 3550.
[15] Cisco Systems. Cisco visual networking index: Forecast and methodology, 2010-2015. http://tinyurl.com/3p7v28, June 2011.
[16] M. Venkataraman and M. Chatterjee. Inferring Video QoE in Real Time. *IEEE Network*, 25(1), 2011.
[17] Y. J. Won et al. Measurement of Download and Play and Streaming IPTV Traffic. *IEEE Commun. Mag.*, 46(10), 2008.