# Cooperative Repair of Wireless Broadcasts

Aaron Harwood[1]**, Spyros Voulgaris[2], and Maarten van Steen[2]

[1] National ICT Australia, Victoria Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
aaron@csse.unimelb.edu.au
[2] Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
{spyros,steen}@cs.vu.nl

**Abstract.** Wireless broadcasting systems, such as Digital Video Broadcasting (DVB), are subject to signal degradation, having an effect on end users' reception quality. Clearly, reception quality can be improved by increasing signal strength. This, however, comes at a significantly increased energy use, with adverse environmental and financial consequences, notably in either sparsely populated rural regions, or overly built and difficult to penetrate dense urban areas. This paper discusses our ongoing work on an alternative approach to improving reception quality, based on the collaborative repair of lossy packet streams among the community of DVB viewers. We present our main idea, the crucial design decisions, the algorithm, as well as preliminary results demonstrating the feasibility and efficiency of this approach.

**Keywords:** Peer-to-Peer, Cooperative Repair, Video Broadcasting

## 1 Background

Wireless broadcasting systems, such as Digital Video Broadcasting [3] (DVB), are subject to interference from the environment, which can result in loss of information and an associated loss of quality for the user. Forward error correction (FEC) and interleaving based schemes involve high overhead on the essentially limited DVB bandwidth. In contrast, a cooperative peer-to-peer repair (CPR) mechanism relies on a *primary channel* being repaired using a CPR protocol on a *secondary channel*. We consider the primary channel to be a video stream and we have chosen the *ISO/IEC 13818-1 Int. Std.* as a case study, which is the standard used for DVB. We propose to use UDP over the Internet as the secondary channel. The ISO/IEC 13818-1 standard describes how to packetize streaming data, such as video and audio streams, for transmission and/or storage. In this case, where the transmission system is unreliable, the standard prescribes the use of Transport Stream (TS) packets of 188 bytes in length. The video and audio

** Research undertaken while on sabbatical at Vrije Universiteit Amsterdam, The Netherlands, Aug. 2010 – Jan. 2011.

MPEG streams (frames) are first encapsulated into a stream of Packetized Elementary Stream (PES) packets, and each PES packet is, in turn, encapsulated into multiple TS packets.

A typical DVB transmitter may be rated at 50kW and supply "adequate" service at up to an 80km radius. In Australia there are over 1300 DVB transmitters totalling over 55.6MW per hour of transmission; or 285k metric tons of $CO_2$ per year. The use of CPR may help to reduce the total number of transmitters and/or the required power, thereby helping the environment.

## 2 Design Decisions and Challenges

### 2.1 Scope of repairing

The cornerstone decision in our work has been the selection of the level at which repair should be applied. Specifically, there are three possibilities. Repairing *frames*, repairing *PES packets*, or repairing *TS packets*. Due to diverse pros and cons, no option constitutes a win-win tradeoff.

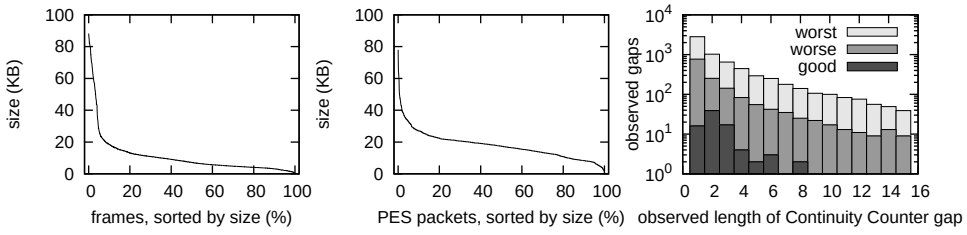
We decided to repair TS packets. While it is tempting to consider working at the PES or frame level because these levels provide more semantic information, there are some drawbacks that we identified: less total reliable information at the higher levels, semantic information tends to be optional, TS packets conveniently fit into a UDP packet, and more processing is required at the higher levels. Furthermore, Fig. 1 (a) and (b) shows the distribution of frame and PES packet sizes from a DVB video sample (video using MPEG2 704x576 which prescales to 1024x576). Even at this relatively low resolution (HD television usually allows up to 1920x1080), frame sizes exceed a single UDP packet payload. The variability of frame/PES sizes and inability to fit within a single UDP packet in general would add further complexity to our overall system that we mitigate by working at the TS level.

On the down side, TS packets do not contain unique identifiers, which would be very beneficial. The lack of unique identifiers at the TS level can be overcome by the use of some convenient, mandatory, semantic information contained at the TS level (explained next).

### 2.2 Transport Stream Packets

A DVB receiver, when tuned and locked to a carrier frequency, produces a stream of TS packets – called the raw TS stream. Among other things, *every* TS packet in the raw stream contains the following key fields: a 13 bit *Program Identifier* ($PID$) that maps the TS packet to an elementary stream, such as a video or audio stream for a given program; a 4 bit *Continuity Counter* ($CC$) that is incremented by 1 modulo 16 for each subsequent TS packet in the elementary stream (certain flagged conditions may arise where the CC is not incremented); and a 1 bit *Transport Error Indicator* ($TEI$) that is set true by the receiver if the TS packet is erroneous.

The fields listed above are insufficient to synchronize two streams for the sake of cooperative repair. There is no uniquely identifying information in each TS




(a) frame size distribution (b) PES size distribution (c) CC gap distribution


(d) *good*: infrequent errors (e) *worse*: frequent errors (f) *worst*: unviewable

**Fig. 1.** a-b: Size distribution of frames and PES packets. c-f: CC gap distribution for three sample streams.

packet, and indeed duplicate TS packets may arise in the stream. However the standard allows a TS packet to contain additional information in an optional Adaption Field, and this optional information contains a 33 bit *Program Clock Reference* (*PCR*). The standard requires this optional information to appear at least every 100 milliseconds. PCR values are not sequential. They are, however, monotonously increasing, therefore *unique* within a stream. The presence of the PCR provides semi-regular, unique stamps on selected TS packets. Thus, we make use of the $PCR$ to synchronize two peers so that cooperative repair can take place.

## 3 The Cooperative Repair Algorithm

A repair algorithm relies on two main components: a mechanism to detect missing information, and a naming scheme to uniquely identify this information in requests to external sources.

### 3.1 Detecting Missing Packets

We use the CC field, described in Section 2.2, to detect missing TS packets. This has a clear limitation, as any sequence of $k$ consecutive missing packets maps to a CC gap of $k$ modulo 16. For instance, it is impossible to distinguish between missing a single packet, or 17, 33, and generally $16i+1$ consecutive packets. Even worse, missing sequences of exactly a multiple of 16 packets will go undetected. To assess the frequencies at which different gaps appear in a realistic setting,

we analyzed a number of DVB streams recorded at locations with diverse reception qualities. Fig. 1(c) shows the count of CC gaps for three recordings of the same duration (circa 1 minute each), yet of very different viewing qualities: infrequent errors, frequent errors and unviewable. The respective snapshots illustrate characteristically the quality of each stream.

As expected, barely tuned streams experience the highest packet loss, and the most CC gaps. Still, the observations of gaps with length 1 (1, 17, 33, etc., consecutive missed packets) are two orders of magnitude more than observations of gaps of length 15 (15, 31, 47 missed packets, etc.). This strongly suggests an inadequacy in the limited length of 4 bits used for the CC to detect missing packets. That, however, occurs only for a negligible fraction of the total losses. Our experiments confirm this observation.

### 3.2 Naming Scheme

In order to be able to name specific units for repair, we introduce the notion of blocks. A *block* is a sequence of TS packets, consisting of all packets between two consecutive PCRs. The starting and ending PCR values of a block are called its *boundaries*, and uniquely identify this block across all nodes. Packets within a block are identified relatively to the starting PCR boundary, based on the CC.

When a node receives a PCR packet, it marks the beginning of a new block. By observing the CC in subsequent TS packets, it keeps track of which packets it received, and which it missed. In doing so, it follows an optimistic approach: a CC gap of $k \in [1, 15]$ is interpreted as a loss of exactly $k$ packets, rather than $16i + k$. A zero gap is interpreted as no lost packet, rather than as $16i$ lost packets. With high probability, the assessment of what has been received is accurate, unless 16 or more *consecutive* packets were lost at some point. The next PCR marks the end of this block and the beginning of a new one.

The record of which TS packets were received and which are missing, constitutes the block's *map*. In our system it is represented as a bitstring; 1 stands for received and 0 for missing packet. Note that missing PCRs leads to concatenated blocks that may later be repaired and split to smaller blocks.

### 3.3 Regular Operation

Upon completion of a block the node stores that block in memory, indexed by its boundaries. Then the node checks whether all packets – are believed to – have been received. If not, it invokes a PULL operation on a random other node that is currently tuned to the same TV channel, requesting the missing TS packets. It sends the block boundaries and the block map.

A node receiving a PULL request, looks up its memory for a block with the specified boundaries. It may have it complete, or it may be missing some TS packets too. In either case, it performs a bitwise operation on the two block maps to figure out which of the requested packets it has. It then sends a PUSH response to the requester, piggybacking zero or more of the requested packets.

Upon receiving a PUSH response, a node adds the received TS packets to the block in question, updates the block's map, and checks if the block is now

complete. If it is still missing packets, it issues a new PULL request on another random node. This is repeated until either the block is complete, or a time threshold called VIEWERTIMEOUT has been reached. At that point, the TS packets of the block are handed to the higher layers for decoding and viewing. A second timer, the PULLTIMEOUT, is associated with each PULL request. If no response is received for that time, a new PULL request is sent to another random node.

### 3.4 Special Cases and Supporting Mechanisms

A number of special cases may arise, which makes the algorithm nontrivial. A node receiving a request may realize that its own copy of the block in question has more or fewer TS packets than the map received. A node receiving a PULL request may be unable to spot the requested block in its local memory. A node receiving a PULL request may figure out that it has both starting and ending PCR values, but they do not belong to a single block. We have solutions to these special cases that cure most (although not all) error scenarios, which we omit for space considerations.

Like many epidemic protocols, the cooperative repair algorithm relies on communication between peers selected *uniformly at random*. To that end, we rely on the family of PEER SAMPLING SERVICE protocols, and specifically CYCLON [5], which provides each node with a regularly refreshed list of links to random other peers, in a fully decentralized manner and at negligible bandwidth cost. We omit discussion of this aspect due to space consideration.

## 4 Evaluation

Our basic results are obtained through simulations using PeerSim, based on a uniform error model. All nodes experience identical constant error probability for each TS packet. Fig. 2 shows, as a function of the TS packet error rate, the percentage of blocks that were received correctly straight away through broadcasting (darkest), blocks that were initially incomplete but were fully repaired (dark), blocks that were still incomplete when passed on to the higher layer – probably partially repaired (light), and blocks which the node never became aware of, due to some lost PCR (lightest). The four cases represent a VIEWERTIMEOUT of 500ms, 1000ms, 2000ms, and 5000ms, while the PULLTIMEOUT was fixed to 600ms for all of them. Clearly, as the VIEWERTIMEOUT increases, the ability to know and repair more blocks increases. In fact, VIEWERTIMEOUT has a dominant effect in this respect, more so than PULLTIMEOUT. E.g., for a mere delay of 5sec, users that would otherwise hardly be able to watch a program, can now enjoy a crystal clear stream. The value of PULLTIMEOUT was chosen to be 600ms in this example because this value keeps the upload bandwidth of the nodes below about 1Mbps, in line with residential ADSL2+ limits.

## 5 Conclusion and Related Work

The work in [6] is closely related to ours – the authors repair satellite broadcasts; however few details are provided in the paper. In [4, 1, 2], some other P2P

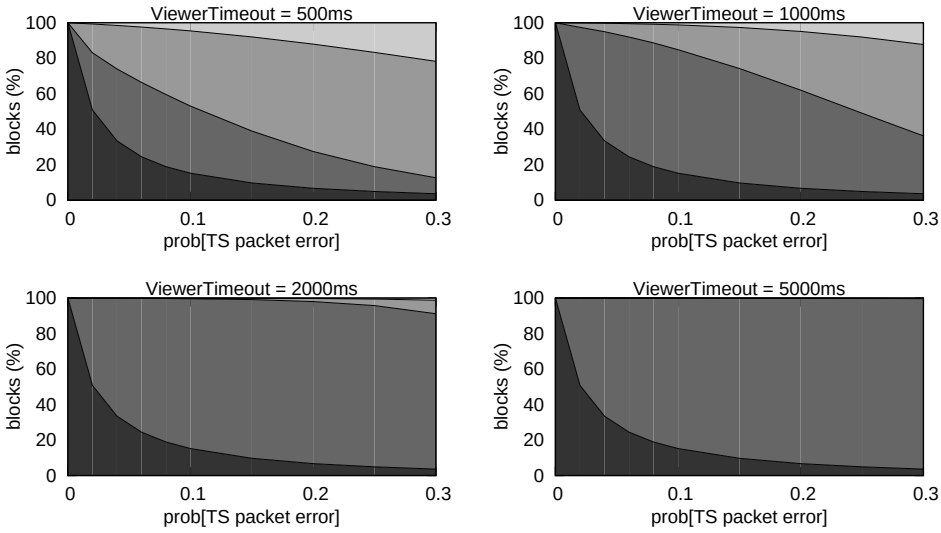



**Fig. 2.** Repair effectiveness vs. node error rates, for different ViewerTimeout values. From bottom up — Darkest: fraction of blocks received correctly; Dark: completed through repairing; Light: known (but incomplete); Lightest: not known blocks.

approaches use a secondary channel such as 802.11 or Bluetooth, with focus on mobile devices. We omit a detailed description of related work for space considerations. Our results thus far, provide significant motivation for us to continue research in this direction.

## References

1. Li, S., Chan, S.H.G.: Bopper: Wireless video broadcasting with peer-to-peer error recovery. In: Proc. IEEE Int. Conf. on Multimedia and Expo. pp. 392–395. IEEE (2007)
2. Raza, S., Li, D., Chuah, C.N., Cheung, G.: Cooperative peer-to-peer repair for wireless multimedia broadcast. In: Proc. IEEE Int. Conf. on Multimedia and Expo. pp. 1075–1078. IEEE (2007)
3. Reimers, U.: DVB-the family of international standards for digital video broadcasting. Proc. IEEE 94(1), 173–182 (jan 2006)
4. Sanigepalli, P., Kalva, H., Furht, B.: Using p2p networks for error recovery in mbms applications. In: Proc. IEEE Int. Conf. on Multimedia and Expo. pp. 1685–1688. IEEE Computer Society, Los Alamitos, CA, USA (2006)
5. Voulgaris, S., Gavidia, D., van Steen, M.: Cyclon: Inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management 13(2), 197–217 (June 2005), http://dx.doi.org/10.1007/s10922-005-4441-x
6. Weigle, E., Hiltunen, M., Schlichting, R., Vaishampayan, V.A., Chien, A.A.: Peer-to-peer error recovery for hybrid satellite-terrestrial networks. In: Proc. IEEE Int. Con. on Peer-to-Peer Computing. pp. 153–160. IEEE Computer Society, Los Alamitos, CA, USA (2006)