Elastic Frame Protocol - an open-source alternative to MPEG-2 TS

Torbjörn Einarsson
Edgeware
Agile Content S.A.
Stockholm, Sweden
torbjorn.einarsson@edgeware.tv

Mikael Wånggren
Edgeware
Agile Content S.A.
Stockholm, Sweden
mikael.wanggren@edgeware.tv

Anders Cedronius
Edgeware
Agile Content S.A.
Stockholm, Sweden
anders.cedronius@gmail.com

***Abstract*—This paper discusses the new open-source Elastic Frame Protocol (EFP) and compares it with MPEG-2 Transport Stream (TS) with regards to protocol overhead, timeline and time stamps, error detection and recovery, (de)multiplexing, complex transport architectures, flexibility and use in modern packet-based transport architectures.**

MPEG-2 TS has the advantage of being almost universally supported in broadcast media transport, but EFP was found to provide significantly lower overhead, 64-bit timestamps and much better possibilities to detect and correct packets that are delivered out of order.

***Index Terms*—MPEG-2, transport stream, EFP, video**

I. Introduction

Video and audio media already constitute the majority of all traffic transported over the internet, and their share continues to grow. They are typically transported compressed into a bit stream format in order to reduce the transport and storage requirements. When transported over a link there is more information needed in order to identify, describe and synchronize the elementary media streams, and this is often handled by a multiplexing protocol. The multiplexing protocol is responsible for aggregating a number of substreams such as video, audio, and subtitle tracks into a single stream, and for adding metadata into this aggregated stream that identifies and describes the constituent substreams as well as how they relate to each other, e.g. regarding timing. There are multiple such protocols originating from different use cases and the technology possibilities and needs at the time when the protocols were conceived.

This paper investigates two such multiplexing protocols, the omnipresent MPEG-2 Transport Stream (TS) [1] protocol and a modern lean protocol called Elastic Frame Protocol (EFP) [2] that was developed in the context of an efficient low-delay link protocol. The focus of this investigation is on the area of contribution/broadcast media transport, where low and predictable latency is required, and error detection and recovery is important.

MPEG-2 TS is a protocol for multiplexing elementary media streams into a format that is suitable for transport. It was first specified in 1995 in the standard ISO/IEC 13818-1 [1], the systems part of MPEG-2, as well as in the ITU-T H.222 specification [3]. As will be discussed further, the design choices made at that time affect its use negatively today.

EFP is an attempt to design a multiplexing protocol that matches today's conditions of media transport over the Internet using modern low latency network resiliency protocols and avoids as much unneeded complexity as possible in its specification. It takes the approach of an open-source implementation rather than a standards-body-driven specification.

Today, IP transport is used everywhere with a strong tendency to build reliable or semi-reliable protocols on top of UDP instead of TCP in order to better control limited delay and out of order delivery. For general purpose data, the IETF QUIC protocol [4] is an example of this. For contribution/broadcast media transport, two such open protocols are Secure Reliable Transport (SRT) [5] and Reliable Internet Stream Transport (RIST) [6]. They both provide a way of combining semi-reliable transport (attempt to recover lost packets up until a deadline) with a fixed end-to-end latency. They are today typically used to transport MPEG-2 TS streams, but are designed to be content agnostic and can therefore also be used to transport EFP streams.

II. Design Considerations and decisions

Each of the multiplexing protocols MPEG-2 TS and EFP have a design that is highly influenced by the intended use at the time of their conception. This section will discuss these conditions and the design decisions that were taken as a result.

A. MPEG-2 TS

The MPEG-2 Transport Stream is specified in the MPEG-2 Systems standard which was developed in the first part of the 1990's. In some aspects, it built upon MPEG-1 Systems which was developed with a focus on making it possible to store MPEG-1 progressive video in SIF resolution (352x240 or 352x288 pixels for NTSC and PAL respectively) and play it from a disc.

MPEG-2 video focused on digital SDTV which has twice the resolution of SIF and is interlaced. MPEG-2 Systems contain both Transport Stream (TS) and Program Stream (PS) and in both cases the actual video and audio samples are framed in Packetized Elementary Stream (PES) packets. The PES packets may have both a Presentation Time Stamp (PTS) and a Decode Time Stamp (DTS). The range and time scale of these timestamps were inherited from MPEG-1 systems to

be 33 bits at 90kHz, which means that they will wrap around approximately every 26.5 hours.

At the time when MPEG-2 TS was conceived, IP transport of media was at its infancy, and the focus was on error-prone channels with constant delay, such as virtual circuit switched Asynchronous Transfer Mode (ATM) networks. The basic building block of a TS stream is the MPEG-2 Transport Stream Packet. It always starts with a 4 byte header and has a length of 188 bytes. This length was chosen to harmonize with the ATM cell payload size. Effects of this early environment can also be seen in the fixed sync byte at the start of every MPEG-2 TS header, and various check-sums in mandatory metadata.

In MPEG-2 TS, the elementary media streams are packetized into PES packets, which carry one or more samples of the media. For video, a PES packet typically carries one video frame or field, while for audio there may be more than one frame's worth of audio samples in a single PES packet in order to achieve lower packetization overhead since audio in general uses much less data than video. The PES packets are the actual payload of the MPEG-2 TS packets, and a flag called "payload_unit_start_indicator" in the MPEG-2 TS packet header signals that a PES packet starts in that particular TS packet.

Furthermore, the PES packets have a PES header starting with a three-byte start code to make it possible to find the PES start even if some packet is missing, or if the PES packet starts in the middle of a TS packet. The mentioned mechanisms make it possible to send a MPEG-2 TS stream without a compatible packet framing, since it is possible to search and find the starts of TS packets and PES packets by looking at byte patterns. However, when using a packet based framing such as UDP or protocols on top of UDP like SRT and RIST the need to handle misaligned or partial packets is not an issue, and such mechanisms can be removed.

Another child of its time is the handling of timing and time stamps in MPEG-2 TS. At the time, it was very hard to synchronize clocks between sites, so MPEG-2 TS carries a System Time Clock (STC) timeline via 42 bit (27MHz) PCR (Program Clock Reference) time stamps. The PCR time stamps are intended to be used to generate a receiver STC that does not drift relative to the sender STC. The PTS and DTS time stamps relate to the PCR time stamps. Not only PTS and DTS but also PCR wraps around after roughly 26.5 hours.

A further restriction is a strict buffer model which was invented to support hardware decoders and set-top-boxes with very small amounts of buffer memory. The standard defines a theoretical system target decoder (T-STD) and a leak or Video Buffer Verifier (VBV) model which should be used to schedule the TS media packets and insert TS stuffing packets in order to neither underflow nor overflow the receiver buffer.

B. EFP

The work on EFP was started in 2020 with the aim of creating a lean protocol to be transported only over packet networks. The sender determines the packet content and its length, and the receiver should receive the same data as a packet with the same length. To avoid segmentation by the network transport layers, the Maximum Transmission Unit (MTU) should be known and used. For Ethernet packets, the MTU is 1500 bytes, but with IP, UDP, SRT or RIST headers, the actual payload MTU may be substantially smaller.

Some concepts in the MPEG-2 TS specification that are not relevant to include in EFP, and therefore allows EFP decreased complexity and overhead, are:

- the System Time Clock and PCR
- the header sync word and PES start code

EFP also improves on a number of concepts, as listed in table I

TABLE I
FIELD RANGES

Field	MPEG-2 TS	EFP
Timestamp	33 bits cyclic	64 bits monotonic
Counter	4 bit cyclic	16 bits fragment 16 bits superframe
Payload size	16 bits (0 for video)	16 bits per fragment

The EFP notion corresponding to an MPEG-2 PES packet is called a Superframe. However, a superframe is not carried inside transport packets as in MPEG-2 TS, but is instead built from *fragments* which are the transport units in EFP. The fragment sizes are not fixed, and there are different types, so the two layers of PES and TS packets are in EFP replaced by one layer of fragments.

There can be up to 256 types of EFP fragments, but currently there are three that are most important and listed in Table II.

TABLE II
EFP FRAGMENT TYPES

Type	header size (bytes)	description
1	8	regular media fragment
2	28	end of superframe incl. time stamps
3	10	regular media fragment with size

A sequence of one or more fragments of different types builds up a superframe that carries a media frame. The last fragment is of type 2 and carries the timing information corresponding to the PTS/DTS time stamps in PES headers for MPEG-2 TS.

Since the fragments have variable size, they can fill out the full MTU if there is enough data. For example, if the MTU is 1450 bytes, a video frame of 10 000 bytes can be split into 6 fragments carrying 1442 bytes of payload and an 8-byte (type 1) fragment header followed by a last fragment containing the remaining 1348 bytes of payload together with a longer 28-byte (type 2) fragment header containing timestamps and substream information.

For smaller payloads, such as signaling data, or trailing bytes of a media frame, it should be possible to pack different substreams into the same MTU-size transport packet, by using other fragment types like type 3.

III. Comparisons

A. Protocol Overhead

All protocols add overhead in some form, such as fixed size headers per packet, padding up to required package sizes, or content specific metadata inlined in the stream for identification and synchronization. When comparing the overhead of multiplexing protocols, there is also a need to factor in the environment in which they will be used. This paper is focused primarily on the environment of broadcast media transport over IP and unreliable networks such as the Internet, so the underlying transport protocols will add fixed size headers for each Ethernet packet that is put on the link, as listed in Table III. Secure Reliable Transport (SRT) [5] is a network resiliency protocol commonly used to protect streaming media over the internet.

TABLE III
Header sizes

Protocol	Header size (bytes)
Ethernet	14
IPv4	20
UDP	8
SRT	16

It is clear that to keep the relative overhead added by these transport headers low, it is beneficial to utilize as much as possible of the payload MTU. Ethernet can carry a payload of 1500 bytes, which gives an MTU of 1456 bytes after IPv4, UDP, and SRT headers are added as can bee seen in Table III.

MPEG-2 TS packets are always 188 bytes long but typically several are aggregated into one IP packet to minimize overhead. The maximum is 7 TS packets constituting a total payload of 1316 bytes. Since EFP does not use fixed size fragments, the fragments are able to fill the full 1456 bytes that is the maximum payload size, and thus have a lower relative overhead. On the other hand, EFP does not currently put more than one Superframe in an IP packet, so in the case of very short Superframes such as for low-latency audio, where the payload is typically in the range of a few hundreds of bytes, the per-Ethernet-packet overhead ratio will be higher.

The fixed size nature of the MPEG-2 TS packet at 188 bytes with 4 bytes header means that if the PES packet size is not divisible by 184, the last TS packet of the PES packet will need to be padded, at least if the PES packet start should be aligned with a TS packet start and not wait until there is data from the next media sample/frame. For video, each frame or field is typically a PES packet, so for a 50 frames per second video there are 50 PES packet endings per second that need to be padded to a 184 byte multiple. For audio the PES packet frequency is typically in the same area (46.8 PES packets per second for 48kHz AAC-LC, for example). Thus the relative impact of this padding is completely dependent on the size of the PES packet. For audio, each PES packet typically contains a large amount of audio samples, as in the case of AAC where 1024 samples are in each PES packet. At 48000 samples per second, this represents about 21 ms of audio. As a theoretical example, let us assume that audio is encoded into AAC at 128 kbps and the resulting PES packets are 379 bytes large. Dividing this into 184-byte units that constitute the MPEG-2 TS packet payload, the result is two full TS packets and a third TS packet with 11 bytes from the end of the PES packet and 173 bytes of padding. That is 173 bytes of padding added to a 379 bytes long PES packet, an additional 45.6% of overhead on the AAC coded audio stream! However, this should be put in relation to the fact that audio bit rates are generally a small part of a stream compared to the video part. One way to reduce the relative impact of padding would be to avoid aligning the PES packet start with the TS packet start, but that would entail waiting for the next PES packet before sending the TS packets off, which would incur an extra 21 ms of latency.

Traditionally MPEG-2 TS streams have been sent using a constant bit rate (CBR), and to achieve this from non-constant elementary streams the multiplexer would add padding in the form of null packets to the TS stream. This was also a source of overhead, typically in the range of 5-10%.

In addition to theoretical discussions on protocol overhead, an experiment was executed to measure the protocol overhead of MPEG-2 TS and EFP when multiplexing a stream consisting of one video and one audio stream. The audio and video sources contained random noise. To achieve a reproducible result, we have used FFmpeg to generate H.264 video at 50 frames per second at a series of different bit rates, while keeping the bit rate of the AAC-LC audio constant at 96kbps. In the script, see Listing 1, the bit rate in kbps is represented by $1 and is varied from 500 to 20000 in steps of 500, while the buffer size is represented by $2 with the value $2 = 1.5 * $1.

Listing 1. ffmpeg script to generate data

```
ffmpeg \
-f lavfi \
-i nullsrc=s=1280x720:rate=50 \
-filter_complex \
"geq=random(1)*255:128:128;\
aevalsrc=-2+random(0)" \
-c:a aac -b:a 96k -ac 2 -ar 48000 \
-c:v libx264 -preset medium -b:v $1k \
-minrate $1k -maxrate $1k -bufsize $2k \
-x264opts "no-scenecut:keyint="50":\
min-keyint="50":nal-hrd=cbr:\
no-open-gop=1:force-cfr=1:aud=1" \
-y -f mpegts -t 25 rnd.ts
```

The result is shown in Fig. 1. The x-axis shows the video bit rate, while the y-axis shows the payload overhead which is calculated as

$$payload\ overhead = \frac{media\ bytes + protocol\ bytes}{media\ bytes} - 1$$

$$media\ bytes = video\ payload + audio\ payload$$

Since all video and audio frames have time stamps and other sample information, the overhead is higher for lower bit rates. For MPEG-2 TS, there is also a high relative

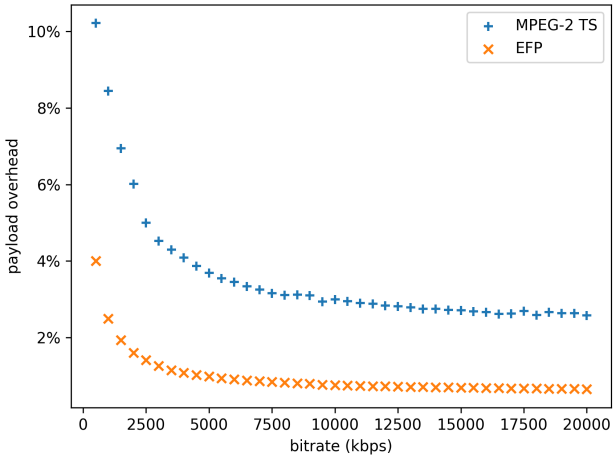

Fig. 1. Payload overhead per protocol

overhead from stuffing the end of each frame. As the bit rate approaches higher values, the payload overheads converge to the asymptotic values where the frame endings are negligible and the overhead can be calculated as

$$asymptotic\ overhead = \frac{header\ size}{packet\ size - header\ size}$$

which results in the values in Table IV.

TABLE IV
ASYMPTOTIC OVERHEAD

Protocol	Packet size (bytes)	Header size (bytes)	Asymptotic overhead
MPEG-2 TS	188	4	$\approx 2.2\%$
EFP	1450	8	$\approx 0.55\%$

In the above, other effects like stuffing to constant bit rate and the headers of the transport packet layers have not been considered, but it should anyway be clear that it is possible to lower the overhead substantially by using a more modern protocol like EFP compared to MPEG-2 TS.

B. Time stamps and clock consistency

As stated in section II, MPEG-2 TS have 33 bit timestamps for PTS and DTS, at a frequency of 90 kHz, and therefore wraps around every 26.5 hours. This makes the absolute values of the timestamps arbitrary and prohibits a monotonically increasing absolute time line. EFP uses a 64 bit timestamp which can be used to carry International Atomic Time (TAI), a monotonic absolute time. The interpretation of the timestamp in EFP is not specified in the protocol, which is both a weakness and a strength. For interoperability purposes, a fixed definition would have been useful, but leaving it implementation defined allows for multiple different uses. In particular, one can choose a timescale that avoids the MPEG-2 TS issue that the DTS values for 59.94 frames/s video increase by 1501.5 ticks in average, resulting in fluctuating DTS step sizes. By using 180kHz, the DTS steps increase evenly by 3003 for each frame. With an monotonically increasing timeline and constant DTS step, one can also make a direct translation to and from MPEG DASH live streams [7] with ISOBMFF [8] segments since they have a monotonic clock 64-bit resolution. Having a constant DTS step translates into a constant sample duration in the ISOBMFF fragments which allows for less overhead by conveying the sample duration as a default value in the Track Fragment Header box.

In MPEG-2 TS, the PCR timestamps are, as described in section II, intended to be used to recover the sender clock in the receiver, but this puts rather tight requirements on the sender and receiver hardware, as well as the jitter characteristics of the channel, as the more jitter that the PCR samples are subjected to over the link, the harder it becomes to recover the STC. There are requirements in the specification on PCR jitter and accuracy that are very difficult to comply with using software implementations and when transporting over public internet. EFP instead expects the clocks in the communicating nodes to be synchronized using some external method, such as Network Time Protocol (NTP), Precision Time Protocol (PTP) or some custom time synchronization protocol. There is also a fragment type (type 0) defined in EFP that can be used for building such a custom synchronization protocol. This removes a large amount of complexity compared to the MPEG-2 TS specification.

C. Error Detection and Recovery

Table I lists the bit width of the packet counters in MPEG-2 TS and EFP. For MPEG-2 TS each elementary stream has its own 4-bit wide continuity counter corresponding to 16 possible values. It can often (but not always) be used to detect that some amount of data has been lost, but reassembling a stream with out of order or duplicated packets is very hard, if not impossible, to do in a reliable way. The fact that up to 7 TS packets from different elementary streams can be combined in each UDP packet only complicates these operations. Therefore, for transporting TS over UDP there is often an extra layer of protocols such as RTP [9] added to get a sequence number for the whole set of 7 MPEG-2 TS packets. However, the RTP header adds another 12 bytes of overhead and only carries a 32-bit timestamp, so it is not an optimal solution.

EFP on the other hand, as can be seen in table I has a 16-bit counter for superframes and another 16-bit counter for fragment within a superframe. This allows for clear identification of exactly how much data was lost, and for sorting out duplicates and reorder out-of-order packets. In fact, EFP's frame numbering and timestamps makes EFP alone go beyond what MPEG-2 TS + RTP achieve together. As a bonus, using the EFP frame numbering it is also possible to insert a thin layer between EFP and the transport stack to do multi-path delivery with either duplication of all packets over several separate paths or distributing the stream over several channels in such a way that one larger combined channel is created. This is possible since identifying each packet uniquely is trivial.

D. Descriptive metadata

In MPEG-2 TS, the Program Association Table (PAT) and Program Map Table (PMT) are periodically repeating metadata that acts as a table of contents for the transport stream. The reason for them repeating regularly is that a receiver shall be able to start parsing the stream at any point, without having to communicate with the sender, and within a reasonable amount of time be able to determine the contents of the stream. The repetition rate is system dependent, but in Digital Video Broadcasting (DVB) the requirement is once every 500 ms [10]. This table-of-contents metadata contains information on all the programs in the transport stream, and all elementary streams in each program. For each type of elementary stream, there are separate specifications that define descriptors, data structures that can be quite lengthy, that also go in the PMT.

EFP also has a method for transporting this kind of table-of-contents metadata. It can be sent in-band in the stream, but taking advantage of modern duplex based communication, can optionally instead be communicated out-of-band using web sockets. If transported in-band, it can be repetitive or only triggered on changes in the stream. Taking even more advantage of the duplex functionality, EFP has also added the possibility to communicate to the sender filtering settings so that not all content is sent over the link. One weakness of EFP in relation to MPEG-2 TS is that while many elementary stream specifications already contain information on how to map them into MPEG-2 transport streams and exactly what the descriptors shall contain, EFP has a more limited set of streams and descriptors as of now.

IV. Possible improvements of EFP

EFP is a new protocol and not all parts have been deployed and used yet. It is defined by its open source code, and the documentation is not fully aligned with the latest changes to the protocol.

As has been shown above, the overhead is relatively low, but by introducing more types of fragments, one could optimize for more cases, and provide even lower overhead.

Some parts that could be further improved with little efforts are

- signaling of the time scale and type of reference
- change the code to use network byte order
- the total fragment count could be removed saving 2 bytes from type 1 and type 2 fragments
- versioning of the protocol

V. Conclusions

MPEG-2 TS is nearly ubiquitous in the broadcast media transport world, and that is its main strength, the fact that almost all equipment can handle it. However, in modern packet switched software based environments, it does suffer from a number of inefficiencies and the full specification is seldom strictly adhered to any longer. EFP offers a number of improvements that align better to the modern reality, but interoperability is its main weakness at this point.

This paper identified the following advantages in using EFP over MPEG-2 TS:

- significantly lower protocol overhead verified through theoretical reasoning and through experiments
- monotonically increasing timeline with 64-bit timestamps compared to 33-bit cyclic timestamps
- repair of out-of-order packet delivery using dual 16-bit instead of 4-bit counters

The EFP protocol is new and not as mature as MPEG-2 TS, but since it is available as open source code it can be used and improved freely and simply together with SRT or RIST to achieve a better media transport link between two nodes.

References

[1] "Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems", International Standard ISO/IEC 13818-1 First Edition, 1995.

[2] Elastic Frame Protocol (EFP) https://github.com/agilecontent/efp

[3] "Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems", Recommendation ITU-T H.222.0 First Edition, 1995.

[4] "QUIC: A UDP-Based Multiplexed and Secure Transport", IETF RFC 9000, 2021.

[5] "Secure Reliable Transport (SRT)" https://github.com/Haivision/srt

[6] "Reliable Internet Stream Transport (RIST), Protocol Specification - Simple Profile", Video Services Forum, 2018.

[7] "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", ISO/IEC IS 23009-1, 2021.

[8] "ISO base media file format", ISO/IEC IS 14496-12, 2021.

[9] "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, IETF, 2003.

[10] "Digital Video Broadcasting (DVB); Measurement guidelines for DVB systems", ETSI TR 101 290 V1.3.1, 2014.