# VIDEO OVER IP PROTOCOLS

## I. INTRODUCTION

The past 10 to 15 years have witnessed a major upheaval in how entertainment content is distributed to, and consumed by, end users. This upheaval is underpinned by the consumers' desire to watch content when they want it, at whatever location they are, and on whatever playback device they have at their disposal. This desire could only be satisfied – on a massive scale – by leveraging the reach and capabilities of the Internet. Almost all playback and display devices – smart TVs, tablets, PCs, smartphones, game consoles, etc. – can be connected to it, from any location, and at any time because they support the core Internet protocols (IP, TCP and UDP).

More recently, this transition towards IP networks for content distribution has also been taking place on the B2B side of the entertainment business. This can be seen in production plants – where cameras are IP-enabled and IP switchers and routers are replacing traditional SDI-based gears [1] – in post-production, and in contribution feeds to Multichannel Video Programming Distributors (MVPDs).

The Internet's traditional transport protocols (UDP and TCP) were designed long before such trends took place. This makes them suboptimal for the various use cases of audio and video transport. This sub optimality was recognized as early as

1990 when the concept of Application Layer Framing (ALF) was first introduced [ALF]. With ALF, applications customize transport layer functionalities (such as error and rate control) at the application layer to suit their specific requirements.

Given the proliferation of applications requiring transport of multimedia content (video, audio, closed captions, and associated metadata) over IP, numerous ALF transport protocols have been developed to achieve optimum performance for each use case. The first widely deployed instance was the Real-time Transport Protocol (RTP) and its companion Real-time Transport Control Protocol (RTCP), first introduced in the mid-1990s. Since then, many others have been developed – some proprietary, others open source, some fully standardized and others still in draft specification form – leading to a confusing alphabet soup of protocols.

It is the intent of this White Paper to decipher this alphabet soup by providing overviews of the more popular protocols and identifying the use cases where they bring the most value. The technical experts at ATEME have implemented most of these protocols within our products. We aim to provide our customers with the widest range of choices and capabilities and help them make decisions that are optimal for their specific requirements.

The sections that follow provide a high-level technical description of various protocols. These include RTP, the popular HTTP adaptive protocols used in OTT distribution, protocols such as Zixi, SRT and RIST that are used for primary distribution, as well as SMPTE standards for video over IP distribution (ST 2022 and 2110) at production head-ends, and the ROUTE protocol that has been integrated into the ATSC 3.0 standard. They also give additional information on deployment status, licensing terms and market acceptance.

[1]Thanks to the increase of CPU power that now can cope with the SDI's data rate (400 Gbps).

## II. RTP & RTCP

### TECHNICAL OVERVIEW

The Real-time Transport Protocol (RTP) was the first successful deployment of an ALF protocol. The first specification was published in 1996 [RFC 1889]. This was superseded by a revised specification in 2003 [RFC 3550].

RTP defined a header, which included two key concepts: a 16-bit sequence number and a 32-bit timestamp. These were considered common parameters that could be used by various applications. The sequence number would be helpful for packet re-ordering as well as packet loss detection. The timestamp would enable synchronization between different media, as well as help the client device to play back media segments at the correct times. RTP would typically run on top of UDP.

The Real-time Transport Control Protocol (RTCP) was specified as a companion control protocol for RTP in the same specifications [RFC 3550]. It allowed for the exchange of (out-of-band) control information between the sender and the receiver such as roundtrip delay measurements, packet arrival and loss statistics and other Quality of Experience (QoE) parameters. The sender could then use this information to, for instance, regulate the transmission rate.

RTP deliberately did not specify how specific media content would be transported. Following the publication of RTP, numerous complementary documents were then developed to define how specific payloads (such as MPEG-2 video, or MP3 audio, etc.) would be carried using RTP. Each document would define additional packet formats with their own specific headers and fields that were carried as part of the RTP payload.

RTP is designed as a stateful protocol: a session is established between sender and receiver through a handshake, and the session is terminated with explicit commands exchanged between the two.

### DEPLOYMENT AND TYPICAL APPLICATIONS

RTP was the first truly successful OTT streaming protocol, used initially for distribution to end users in the late 1990s and early 2000s. It suffered from a few shortcomings, however. One was the fact that network firewalls often blocked the UDP ports on which RTP operated.

A bigger shortcoming was the fact that since RTP was developed just before (or at the same time as) Content Distribution Networks (CDNs), its design did not leverage the capabilities of these networks (such as content caching). This made it impossible for RTP to scale to cases where large numbers of users needed to simultaneously receive the same live stream (such as a popular sporting event). With the advent of HTTP adaptive streaming protocols such as Smooth Streaming and HLS in the late 2000s (see below), RTP was soon replaced by the latter for end user OTT distribution.

Today, RTP is mainly used in contribution or primary distribution (e.g. between a national data center and local affiliates). RTP does not specify error correction protocols, but it is often used with (separately specified) Forward Error Correction (FEC) schemes on error-prone networks. RTP has basic support for encryption but more advanced encryption methods, specified elsewhere, are recommended for content protection.

## III. HTTP ADAPTIVE STREAMING PROTOCOLS

### BACKGROUND

The major shortcomings of RTP were overcome in the late 2000s when adaptive

streaming protocols based on HTTP were introduced.

HTTP was developed in 1996 (version 1.0) and 1997 (version 1.1) [RFC 2068 obsoleted by RFC 2616] primarily to transport the contents of static web pages from a server to a user's browser. Given the exploding popularity of the World Wide Web in the 1990s, an entire infrastructure (mainly CDNs, with a hierarchical network of servers and caches) was developed to enable scaling to numerous users.

HTTP transactions are session-less: A client (browser) typically posts a GET request to a server for a file (be it a web page or a media asset) and the server (or intermediate edge servers that may have cached the content) sends the requested asset. HTTP runs over TCP[2], thereby leveraging the latter's error and flow control mechanisms. Access to web pages through HTTP was so popular – and indeed had become a necessity – that almost all firewalls by default allowed TCP port 80 (used for HTTP communications) through, while blocking most other ports (notably the UDP ports used by RTP).

**MULTI-BITRATE ENCODINGS**

Streaming pioneers in the 1990s had already noticed a significant issue when it came to streaming media content: different users were connected to the Internet with technologies that had widely varying bandwidth: some used dial-up, others DSL or ISDN and yet others T1 connections. This left content distributors with a dilemma: what bitrate should they use to encode their content? If they used a high bitrate, they risked disenfranchising most of their audience. If they used a 'lowest common denominator' bitrate, then their highest paying customers with the best connections would be receiving poor quality video.

*[2]HTTP version 3.0, currently in draft format [HTTP 3.0], runs over UDP.*

A simple, brute force solution was introduced by RealNetworks in the late 1990s: encode the same content at multiple bitrates. At the start of the session, have the end user select the bitrate depending on their connection speed.

This simple solution was an improvement over the single – and sub-optimal –bitrate encoding, but there were still two problems:

1) It required end users to know their connection speed (and to account for various overheads such as TCP, IP, Ethernet and physical layer headers), which was not feasible for the lay customer;

2) It assumed that the bandwidth would remain fixed throughout the duration of the stream – a poor assumption given the public nature of the Internet and the resulting highly dynamic traffic and load.

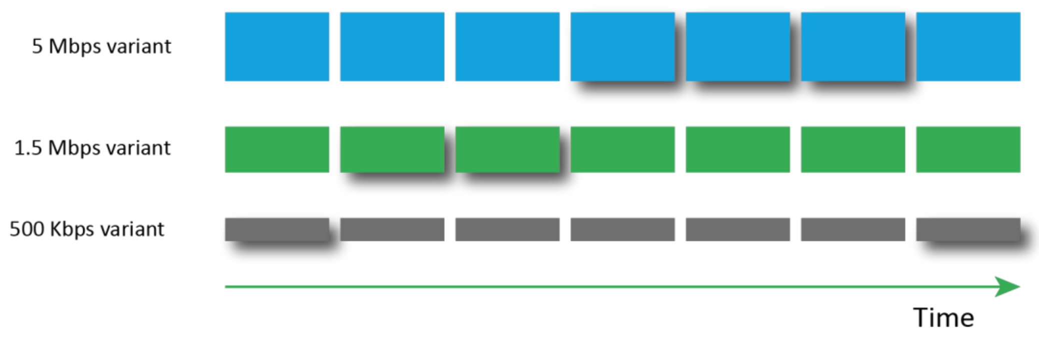
**ADAPTIVE STREAMING PROTOCOLS**

In 2007, Move Networks introduced the concept of adaptive HTTP streaming by adding a couple of twists to the idea of multi-bitrate encoding. Instead of encoding each media asset as a single file, they encoded it as a series of independently decodable mini-files or segments. A two-hour movie could, for instance, be encoded as 720 10-second segments.

In a multi-bitrate encoding, each variant would be encoded as multiple segments with care taken to align the segment boundaries. A manifest file was then created that included information on the number of variants (the different bitrates at which the content was encoded) as well as information about the individual file segments (such as their locations, durations, etc.).

At session initiation, the player device would download the manifest file and request segments (using HTTP GET requests) from a default variant. As segments were downloaded, the player would constantly estimate the bandwidth between itself and the server and move up or down the Adaptive BitRate (ABR) ladder on segment boundaries. The goal of the player would be to receive the highest quality video without exceeding the estimated bandwidth, all while adapting to potential fluctuations in bandwidth. Figure 1 illustrates the concept with an ABR ladder consisting of three variants (0.5, 1.5 Mbps and 5.0 Mbps). The highlighted segments are requested and displayed by the player.

Figure 1 Dynamic bitstream switching in HTTP adaptive streaming



Ironically, this is not a streaming protocol per se, but rather a series of file downloads. The file segments still benefit from CDN caching to enable scalable streaming, and firewall penetration through TCP port 80, with the added benefit of dynamic bandwidth adaptation. The scalability of the scheme is helped by the fact that the intelligence (for estimating bandwidth and adapting to it) is placed in the client, not the server. Unlike RTP, HTTP is session-less and does not include the exchange of RTCP messages (although the underlying TCP protocol does require the server to be responsible for error and flow control).

Move Networks' idea was quickly adopted by the major streaming technology vendors of the time. Microsoft introduced its version of the idea in 2008, naming it Smooth Streaming. In 2009, Apple and Adobe followed suit with HTTP Live Streaming (HLS) and HTTP Dynamic Streaming (HDS) respectively. While these competing technologies relied on the same principles outlined above, they differed in two important aspects: They used different file formats for the audio/visual content (MP4 for Smooth Streaming, MPEG-2 Transport Stream for HLS, F4V for HDS) and they used a different syntax for their manifest files.

This resulted in market fragmentation: Different devices used different proprietary technologies that were not interoperable. Content providers would have to encode the same content in multiple formats, and CDNs would have to distribute and cache the same content in different formats – resulting in wasted storage and bandwidth. To address this issue, the Moving Pictures Experts Group (MPEG) developed the Dynamic Adaptive Streaming over HTTP (DASH) standard [DASH] in 2011, hoping that all parties would use this single method. DASH is based on the fragmented MP4 (fMP4) container format and a standardized manifest file syntax, which allows for advanced features such as Dynamic Ad Insertion (DAI), 3D video, multiple camera angles, subtitles and closed captions, and mixed streaming of live and pre-recorded content.

DASH has now largely replaced both Smooth Streaming and HDS. HLS, however, remains widely used by Apple devices, which constitute a significant market share.

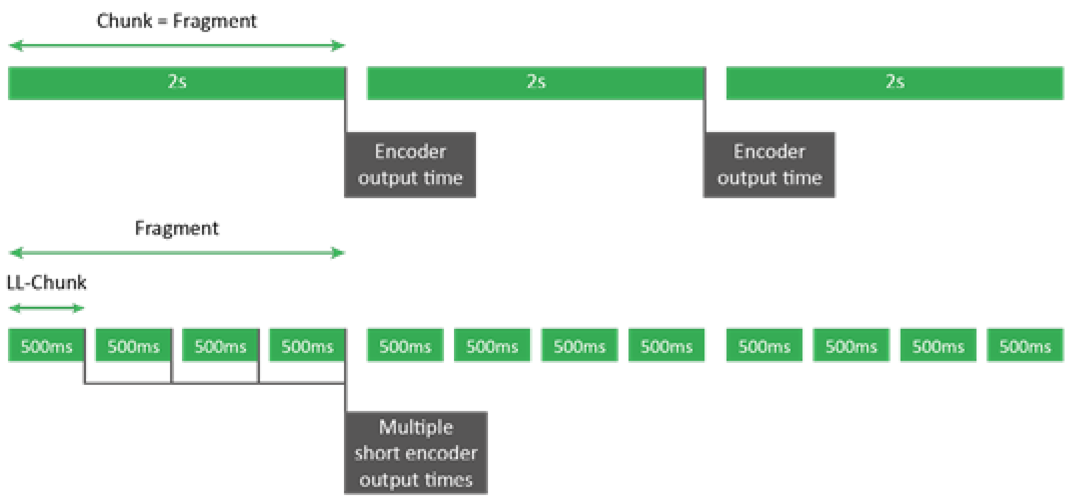
**DASH and HLS currently dominate IP delivery of audiovisual content to the end user.**

**New techniques have emerged – such as chunked encoding and transfer – that promise to reduce end-to-end latency to a few seconds.**

## LOW LATENCY HTTP ADAPTIVE STREAMING

HTTP adaptive streaming protocols did have one shortcoming compared to RTP: a typical live streaming session incurs upwards of 30 seconds of end-to-end latency. New techniques have emerged – such as chunked encoding and transfer – that promise to reduce end-to-end latency to a few seconds. The basic idea is for the encoder to partition file segments (typically 2 to 4 seconds in duration) into smaller segments that are as small as 500 milliseconds. Each chunk can then be individually forwarded to the packager (and from there to the origin server, through the CDN and all the way to the client) without having to wait for the whole segment to be encoded (see Figure 2 below). Note that unlike the segments themselves, these chunks are not independently decodable, so bitrate switching still happens on segment boundaries. Latencies as low as a few seconds have already been demonstrated.

Figure 2 Low-latency chunked encoding and transfer vs. traditional HTTP adaptive streaming



The need for low-latency streaming has opened the door for a unified and efficient solution that encompasses both DASH and HLS. The specification of the Common Media Application Format (CMAF) aims to provide a single container format for assets that are to be streamed by DASH and HLS. Additionally, the concept of requesting media by specifying a range of bytes rather than a media file or segment allows end devices to work with both DASH and HLS.

**RTMP should only be used where support for legacy devices is required.**

## IV. RTMP

The Real-Time Messaging Protocol (RTMP) was developed by Macromedia (now owned by Adobe) to stream multimedia content between a Flash server and client (with the latter typically integrated as a plug-in to a web browser).

RTMP runs over TCP, thus benefitting from the latter's built-in error and flow control capabilities. Audio, video and control messages are sent over separate virtual channels. The media streams are split into fragments, the size of which can be negotiated between the sender and receiver. Typical sizes are 64 bytes for audio and 128 bytes for video. Timing information is added in the RTMP header of each packet to enable synchronization between media.

Content protection is achieved using Secure Sockets (SSL) or RTMPE – a lightweight encryption mechanism.

While Flash was hugely popular in the late 1990s and 2000s for consumer video streaming, it is being phased out and Adobe discontinued its support in January 2021. RTMP itself has been superseded by numerous protocols (see sections below) and should only be considered for use with legacy devices.

## V. ZIXI PROTOCOL

The Zixi protocol (developed by its eponymous corporation) enables live OTT streaming with low latency while providing reliability and content protection. It runs over UDP and provides error resilience using both FEC (Z-FEC, a "dynamic content aware" scheme) and Automatic Repeat reQuest (ARQ) techniques.

The Zixi ecosystem consists of three components: a Zixi Feeder (such as an encoder or a camera), a Zixi Broadcaster (which is the central component performing transcoding, transmuxing, recording and distribution at multiple bitrates and protocols) and a Zixi Receiver (typically a decoder or end device). The Zixi protocol is used between the Feeder and the Broadcaster, or between the Broadcaster and the Receiver.

In low-latency usage (below 1500 ms), error recovery is based on a hybrid approach of FEC and ARQ, with FEC adding up to 30% of overhead. The user can trade-off latency and robustness to errors. It is recommended to set the latency to at least 3 times the Round-Trip Time (RTT).

Zixi can dynamically adapt the transmission bitrate, while content protection is accomplished using both 256-bit AES and DTLS. Zixi supports the use of multiple transmission paths (such as a mobile network and a WiFi connection) for bonding: data transmission can be split over the two paths to achieve a higher aggregate throughput. Alternatively, separate paths can be used for redundancy and hitless failover. Use of bonding requires configuring the Feeder output in Push mode. This mode can also be used between two Broadcasters.

Despite being a proprietary protocol, Zixi has seen significant deployment in the marketplace, specifically for primary distribution (such as to remote facilities or over private multicast-enabled networks).

## VI. SRT

The Secure Reliable Transport (SRT) protocol [SRT] is used for live contribution streams that require low latency, reliability, and security (using encryption). SRT uses UDP as its underlying transport layer and was developed as an alternative to RTMP and HTTP adaptive streaming technologies which rely on TCP for error recovery and congestion control. SRT was created to reduce the latency associated with these protocols (see above). It was also a first step to provide an open-source alternative to Zixi and enable interoperability between multiple vendors, although it falls short of being a standard.

### TECHNICAL OVERVIEW

SRT is an ALF protocol and has its roots in the UDP-based Data Transfer (UDT) protocol. Whereas UDT was aimed at large file transfers, SRT was created mainly for live streaming (its default transmission mode). It includes a few modifications to improve efficiency for use in live streaming, while still maintaining support for file transfers.

Like RTP, SRT is session-based (or connection-oriented): the client and the server establish a session with a handshake; keep the connection open with regular keep- alive messages during streaming; and terminate the session with an explicit close command. This is unlike the session-less HTTP adaptive streaming protocols where each HTTP GET message (retrieval of a media segment) is an independent and state-less transaction between client and server.

Like all ALF protocols, the SRT packet header includes a sequence number (to aid in packet loss detection and re-ordering) and a timestamp (to help with media synchronization and presentation timing).

It includes an ARQ protocol for error recovery, enabling trade-off between latency, reliability, and efficient bandwidth usage. Alternatively, or in addition, FEC schemes may also be used (though no implementation details or mechanism are given in the SRT specifications).

The ARQ scheme uses both positive (ACKs) and negative acknowledgements (NACKs). Three types of ACKs are used. The Full ACKs, which are transmitted every 10 milliseconds, are used for round-trip time (RTT), link capacity and packet reception rate reporting, as calculated by the receiver[3]. These parameters are used by the sender to regulate its transmission rate for file-based transmission. For live encoding cases, no congestion control is applied[4].

NACKs are used to signal lost packets. The sender will hold a packet for the duration of a specified latency in its buffer (in case it needs to be retransmitted) and then drop it at the end of the latency window. Retransmissions are prioritized over first-time transmissions. Likewise, a receiver will wait for a packet to be correctly received for the duration of the latency buffer. If a packet has not been successfully received by then, it will be considered lost. The duration of the latency buffer (expressed in milliseconds) is negotiated during the initial session-establishment handshake.

Content protection is achieved using 128-bit AES encryption in counter mode (AES- CTR). This mode permits random access decryption and tolerates packet loss. SRT encrypts only the payload of SRT data packets, while the header is left unencrypted. The unencrypted header contains the Packet Sequence Number field.

Unlike Zixi and RIST (see below), SRT does not include bonding of multiple connections or channels.

*[3]The sender will emit its own acknowledgement (ACKACK) in response to an ACK to enable RTT calculation by the receiver.*

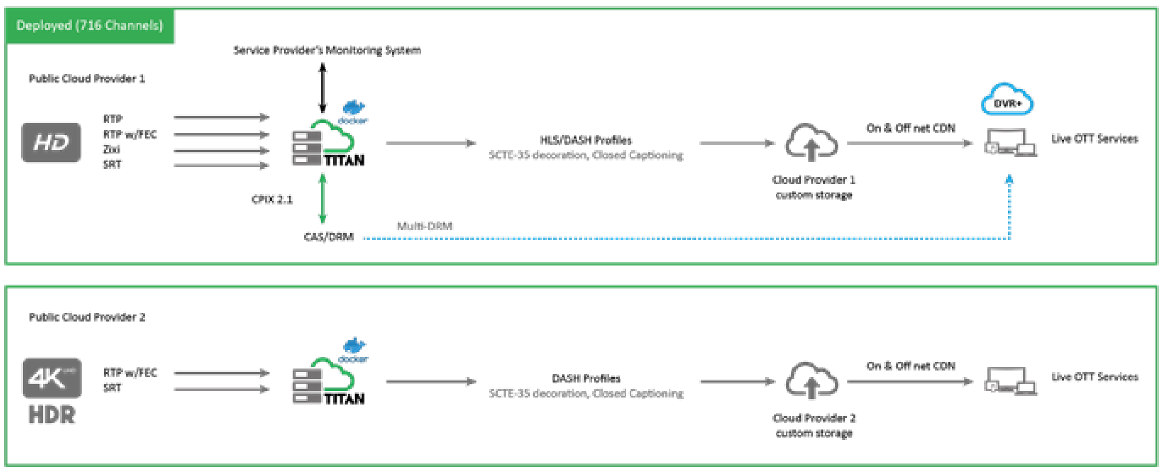
*[4]Technically, this information could be used by the encoder's rate-control algorithm, but that is beyond the scope of SRT.*

DEPLOYMENT AND TYPICAL APPLICATIONS

SRT was developed by Haivision and is promoted by the SRT alliance, which was founded in 2017 by Haivision and includes ATEME among its 400+ members. SRT is gaining acceptance, particularly in price-sensitive markets such as China and more generally the APAC region.

RTP, Zixi and SRT are used primarily in head-ends and for primary distribution. Gateway devices, such as ATEME's TITAN Edge, are often used to convert such streams into HTTP adaptive streams for transmission to end users. A typical scenario for a virtualized head-end is shown in Figure 3 below:

Figure 3 Virtualized headend receiving ALF protocols as input and outputting HTTP adaptive streams



VII. RIST

The Reliable Internet Stream Transport (RIST) protocol was developed by the Video Services Forum (VSF) to provide a standard for reliable streaming over the Internet between products from different companies. The VSF felt that there was a "lack of compatibility between devices" due to the existence of proprietary solutions such as Zixi and SRT. RIST is an example of an ALF protocol and borrows heavily from RTP. To date, two RIST profiles have been specified with a third under development.

### TECHNICAL OVERVIEW

The RIST Simple Profile was published in October 2018 [RIST TR06-1] and uses RTP as the baseline protocol for media transport. It also uses RTCP to exchange control messages between sender and receiver, primarily for error-recovery purposes.

RIST uses an ARQ-based mechanism to recover from packet loss. Negative acknowledgements (NACKs) are sent when packet loss is detected by the receiver (unlike TCP, no positive ACKs are sent for correctly received packets). Selective Retransmission (SR) is then used by the sender to only retransmit lost packets (as opposed to resending all packets after a lost packet, which would be the case in some protocols) to minimize bandwidth usage.

Two types of NACK-based retransmissions are specified: bit masks for individual packets and short bursts of up to 17 lost packets; or range-based NACKS for large bursts of packet loss. Implementation of the detection of packet loss and retransmission request is left to developers.

While RIST allows interoperability between products from different vendors, the actual performance of products in error-prone and/or congested networks can vary from one vendor to another because implementation details are left to the discretion of the product manufacturers.

As in all error-recovery mechanisms relying on retransmissions to recover from packet loss, receivers are expected to implement a buffer to accommodate one or more network round-trip delays as well as packet re-ordering. A packet may be retransmitted multiple times and sufficient buffering is needed to allow a successful retransmission prior to playback, leading to significant latency. RIST should therefore be used in cases where either ultra-low latency is not required or the distance and number of hops between sender and receiver is small (low RTT).

Another attractive feature of RIST is its support for channel bonding in multi-channel transmission environments such as WiFi and LTE. Channel bonding can be used to:

- Divide a high bitrate RTP stream between multiple channels to combine their bandwidths
- Replicate an individual RTP media stream on multiple channels to increase reliability.

Receivers need to combine packets in the right order and eliminate potential duplicates to reconstruct the original stream.

**Main Profile**

The VSF defined additional features and functionalities in March 2020 under a Main

Profile specification [RIST TR06-2]:

- Stream multiplexing support
- Tunnelling support
- Encryption support using DTLS
- Pre-Shared Key encryption support
- NULL packet deletion
- High bitrate/high latency operation

Stream multiplexing and tunneling provide the ability to combine all the communication between two RIST devices into a single UDP port, to which encryption can then be applied. Encryption is achieved with Datagram Transport Layer Security (DTLS), specified in [RFC 6347], and Pre-Shared Key (PSK) encryption, and uses AES Encryption Key and Sequences. The Main Profile also adds authentication (using certificate-based authentication for both clients and servers, and password-based mechanisms for simpler cases). The RIST Simple Profile lacks both encryption and authentication and should not be used in cases where these are high-priority requirements.

MPEG-2 TS NULL Packet deletion works by deleting NULL packets at the sender but signalling their positions within the RTP packet using a bitmask, so the receiver may re-insert them. This can result in 3-5% savings in bandwidth in situations where MPEG-2 TS NULL Packets are used (e.g. in suboptimal statistical multiplexers).

The High bitrate/high latency feature extends the RTP sequence number from 16 to 32 bits to reduce the odds of sequence number wrap around in high bitrate/high latency networks – which could lead to errors in case of packet loss.

RIST Main Profile allows the use of SMPTE-2022-1 FEC to provide a low-latency alternative to error recovery.

**Enhanced Profile**

A third Enhanced profile is under development and aims to add the following features:

- Smart bandwidth optimization
- Common channel session management
- Centralized call home – to control multiple feeds from a centralized location
- Support for hybrid internet/satellite operation (similar to HbbTV and ATSC 3.0 use cases)

The Enhanced profile specifications should become available in the first half of 2021.

**MARKET ACCEPTANCE AND DEPLOYMENT STATUS**

Since the RIST Main Profile was only published a few months ago, there is currently minimal deployment of RIST-capable devices. Given the number of companies that are involved in the development of RIST, however, we expect it to become widely deployed throughout the industry and all regions. RIST offers several attractive features such as multicast and bonding, and it can be enhanced by providers (adding bitrate adaptation, failover solution, etc.).

**VIII. QUIC**

BACKGROUND

In HTTP/1.1, a typical browser would have half a dozen TCP connections to an HTTP server. HTTP/2 introduced the concept of multiplexing multiple logical connections over a single TCP connection. This helped reduce TCP connection overhead and improved the efficiency of TCP's built-in congestion-control algorithm. Multiplexing, however, introduced a new problem: if a packet were lost on the single TCP connection, all logical streams would be blocked until the lost packet was retransmitted successfully.

HTTP/3 AND UDP

The solution to the TCP 'head-of-line blocking' problem was to develop an application-layer protocol (in the spirit of ALF) running over UDP. QUIC aims to provide similar behavior to TCP but with reduced latency. It leverages the multiplexing capability of HTTP/2, but the logical streams run over UDP, with error and congestion control being performed at the application layer.

Since most streaming applications require content protection using encryption, QUIC includes the exchange of keys and other encryption-related setup as part of the initial connection set-up handshake. This reduces start-up time by at least one roundtrip time. QUIC uses TLS 1.3 for encryption and security.

A key attribute of QUIC is that it enables smoother transitions when a client device moves from, say, a mobile network to a home Wi-Fi network. The desire for such transitions reflects the consumers behavior to use the same device (typically smartphones) while traveling and trying to minimize usage of their mobile data – which could be subject to caps, or extra fees or throttling if they reach a threshold. With TCP, connections would have to time out and be re-established, incurring significant delays, which are annoying to the viewer.

Receivers can regulate the flow of data sent by the sender both at a logical stream level and on the connection level (combining the data from all logical streams sharing the same connection). Error control is achieved using ARQ schemes and packet numbering. An FEC scheme is under consideration for QUIC v2.

An earlier version of the protocol developed by Google is supported by Chrome browsers. The IETF version is expected to become widely deployed along with HTTP/3.

**IX. REPLACING SDI WITH IP**

While media distribution to viewers and customers over IP networks has been happening since the mid-1990s, it has taken much longer for video over IP technologies to be deployed at production head-ends. The dominant technology for moving digital video, audio, and associated metadata in a live production facility has been the Serial Digital Interface (SDI) – using coaxial cable – in its various incarnations (SD-SDI, HD-SDI, 3G-SDI, etc.). IP was suspected of not being able to guarantee the predictability and reliability of SDI connections. Over the years, much capital was invested in SDI infrastructures, including expensive switching gear. Replacing that infrastructure with IP was a daunting task.

One shortcoming of SDI, however, is that it requires moving raw video and audio. With video resolutions increasing to 4K and beyond, and frame rates increasing to

60 and even 120 fps, bandwidth greater than 15 Gbps and even 30 Gbps would be required for the video alone. Moreover, SDI is not a practical means of distributing content from broadcasters to content distributors over a long-haul connection.

A final impetus to transitioning to IP was provided by the advent of cloud technologies and the desire to leverage them for a more flexible production system. Connecting to the cloud could only be achieved via IP, and not through SDI.

## SMPTE ST 2022

A first step towards transitioning to an IP infrastructure was the introduction of the ST 2022 set of standards by the Society of Motion Picture and Television Engineers (SMPTE). The first two parts of the standard (published in 2007) defined means of transporting constant bit-rate MPEG-2 Transport Streams (TS) over IP (using RTP encapsulation) with the optional use of FEC for reliability. So ST 2022-1 [ST

2022-1] and 2022-2 [ST 2022-2] were suitable for primary distribution from the broadcasters to the distributors such as cable TV, satellite TV and OTT (v)MVPDs. Typically, video is compressed moderately, using JPEG-2000, MPEG-2 or AVC. These two parts enjoyed commercial success and were widely deployed.

Compressed bit streams, however, are not practical for real-time headend production where frame-accurate splicing is often required for inserting interstitials, as well as logo and other graphic insertions in the broadcast facilities. These operations need to be carried out in the uncompressed domain.

Part 6 of the standard (ST 2022-6) [ST 2022-6] – introduced in 2013 – addressed the carriage of SDI signals over IP (combining uncompressed audio, video and auxiliary data such as closed captions and timing information, as a bundle). ST 2022-6 uses RTP to encapsulate SDI data in packets of 1376 bytes (a size that enables each packet to fit into a single Ethernet frame, once the overhead of RTP/UDP/IP is considered). ST 2022-6 enabled the use of IP routers and switches in head-end production facilities.

Two error-resilience mechanisms were defined. An FEC scheme was specified in ST 2022-5 [ST 2022-5].

Additionally, ST 2022-7 [ST 2022-7] specified a scheme that duplicates the input and sends it over two different paths from sender to receiver. If packets are lost from one of the paths, the receiver will reconstruct the stream using packets from the other path. The receiver must account for delay jitter and different end-to- end delays and allocate a buffer for this purpose. The size of the buffer reflects a tradeoff between latency and resilience.

While this technique doubles the bandwidth usage, it precludes the need for retransmissions, acknowledgements, and the like, as well as sophisticated error- correction algorithms associated with advanced FEC schemes. Moreover, the scheme allows for an automated, autonomous and seamless failover system without the need for supervision software to detect and enact the failover process.

## SMPTE ST 2110

One of the characteristics of ST 2022-6 is the fact that audio, video, and ancillary data are bundled together as one stream. There are many use cases where it is desirable to separate the three from each other (for instance, to change the audio and associated closed captions from one language to another).

In 2015, the Studio Video over IP (SVIP) Activity Group of the VSF defined Technical Recommendation TR-03 [TR-03] that treated each medium as a separate stream or “essence” using RTP as the underlying transport protocol. This recommendation laid the groundwork for the SMPTE ST 2110 standard.

In 2017, the first specifications of the SMPTE ST 2110 standard were published to address this need (among others). Video, audio, and ancillary data are each carried in separate RTP streams. Different parts of the specification define carriage of compressed audio and video. A separate part of the specification is dedicated to system timing to ensure that the now separate media streams are properly synchronized.

Figure 4 Comparison of SDI, ST-2022-6 and ST 2110

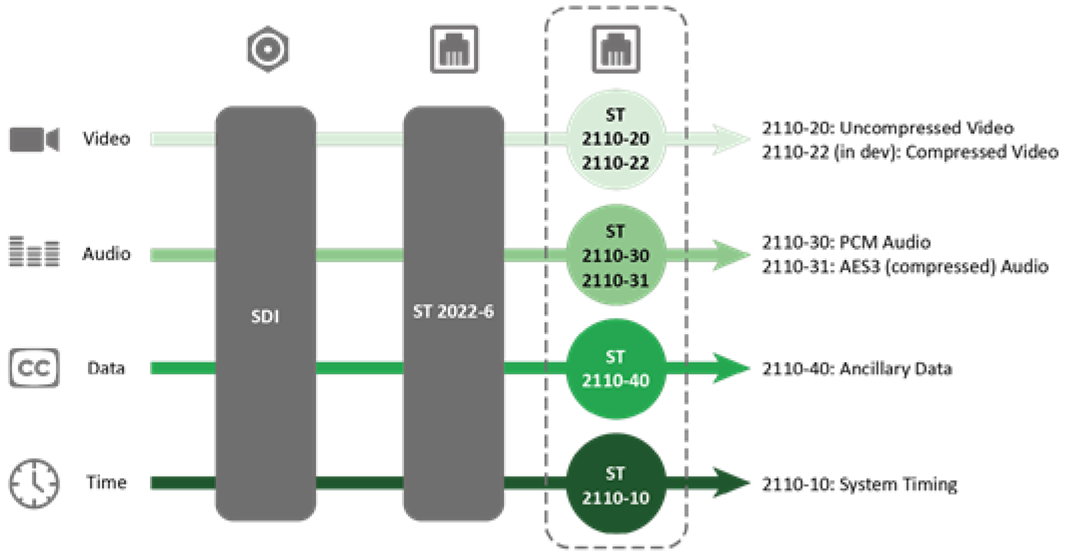


Figure 4 illustrates a high-level comparison of SDI, ST 2022 and ST 2110. At this level, the key difference between SDI and ST 2022 is the physical connection, while the difference between ST 2022 and 2110 is the unbundling of the media in the latter.

Figure 4 also lists some of the specific parts of the 2110 standard and what exactly they pertain to. For instance, ST 2110-22 pertains to the carriage of lightly compressed video (using TICO or JPEG XS[5]). This part of the specification was published in August 2019.

For error resiliency, the ST 2110 standard relies on ST 2022-7. Note that use of ST2110 and ST 2022-7 is limited to the broadcast facility (or inter-facility connections) where bandwidth is not necessarily a scarce commodity.

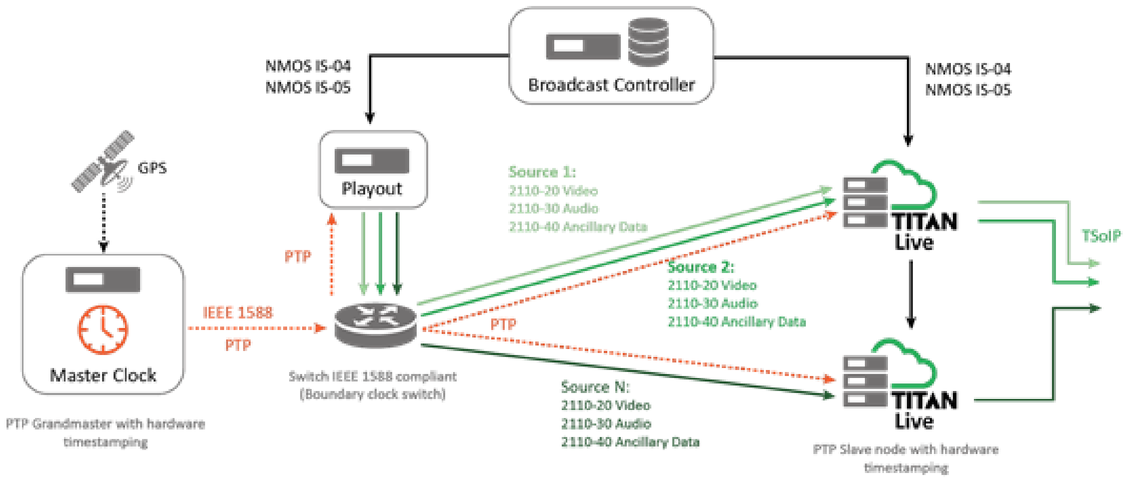
Control and management of ST 2110 compliant devices is defined by the Networked Media Open Specifications (NMOS). NMOS allows devices to automatically register themselves, be discovered by other devices and establish connections, greatly reducing the need for manual set-up and configuration.

[5]JPEG XS typically provides 10:1 compression but at very low latency (~1 millisecond compared to ~60 milliseconds for JP2K).

ST 2110 has only recently been developed (with some parts still under development), but there have already been some deployments.

One such deployment at one of ATEME's customers is shown in Figure 5 below:

Figure 5 Actual use case of ST 2110 protocols



As shown in the figure, a master clock is used to provide a common timing information to multiple media sources (comprising separate video, audio, and ancillary data streams) as well as a playout server. These multiple sources are controlled by a third-party Broadcast Controller using NMOS. The playout server(s) can potentially feed pre-recorded content into the live sources. Note how even the network switch needs to comply with the PTP timing protocol and re-clock the PTP signal to ensure the most accurate timing between all the nodes.

## X. ATSC 3.0 AND ROUTE

### ROUTE OVERVIEW

The Real-time Object delivery over Unidirectional Transport protocol (ROUTE protocol) is specified for delivery of 'application objects' (such as a file, or DASH or HLS segments), including those with real-time constraints, over a unidirectional transport. The ROUTE Protocol also supports low-latency streaming. It relies on underlying protocols to provide security (such as IPSec). It is currently an Internet draft protocol, with the first version of the draft specifications issued in June 2020 [ROUTE].

ROUTE is based on the older File Delivery over Unidirectional Transport (FLUTE) protocol [FLUTE]. FLUTE is geared toward reliable file delivery over a multicast- enabled network to potentially massive numbers of receivers – for instance large software updates to many devices simultaneously. It leverages the efficiencies of IP multicast, but adds reliability (via FEC) and congestion control mechanisms using multiple multicast groups (for different transmission rates). FLUTE runs over UDP.

ROUTE adds chunked encoding and transfer to the FLUTE functionalities it inherited, enabling low-latency delivery of DASH and HLS content with CMAF chunks for live streaming applications. A media-aware packetization scheme enables greater flexibility to optimize object delivery. ROUTE receivers cache application objects, which can then be retrieved by the higher-**layer application using standard HTTP requests. The receiver thus acts as an HTTP server for the calling application.**

### ROUTE DEPLOYMENT

The ATSC 3.0 standard for over-the-air transmission has specified a profile of the ROUTE protocol, as has the DVB Adaptive Media Streaming over IP Multicast specification. The multicast- and broadcast-friendly nature of ROUTE makes it particularly attractive for both over-the-air (OTA) broadcasts and multicast over managed networks. As ATSC 3.0 deployment rolls out across markets in North America and elsewhere, ROUTE capable devices will become more and more ubiquitous.

# XI. TAKEAWAYS AND RECOMMENDATIONS

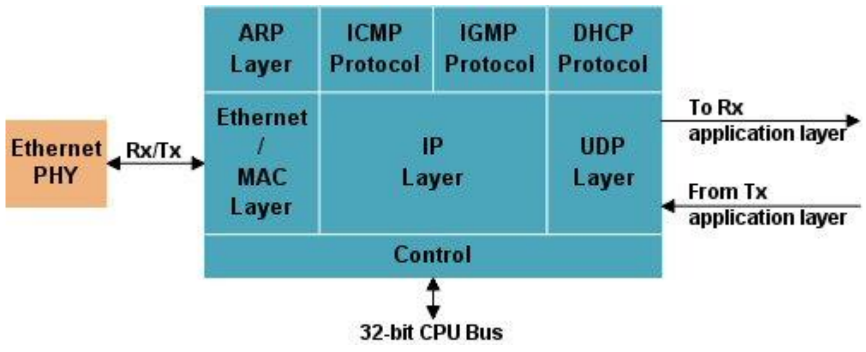
There are many different use cases for carrying audio-visual content over IP networks. The requirements can vary greatly from one use case to another. Different protocols have been specified over the years to address each of these use cases in an optimal fashion. The following table provides a quick guide to determine which protocol(s) are best suited to the main use cases. Before making your final decision though, feel free to reach out to our technical experts for an in-depth analysis of your specific needs and constraints.

| Distribution to end users | Carriage of audiovisual data in the head-end | Primary distribution to affiliates and distributors |
|---|---|---|
| Use **DASH** and **HLS** (with CMAF container format and Byterange addressing) to minimize storage and network usage. For live streams, make sure you use the low-latency variants with support for chunked encoding and transfer.<br><br>Keep an eye open for **HTTP/3 & QUIC** in the near future.<br><br>For **ATSC 3.0 OTA** broadcasts, you must use **ROUTE** however. | **Use ST 2110** (and leapfrog ST 2022 if you are still using SDI).<br><br>Use ST 2022 only if needed for supporting legacy devices. | **RIST is recommended** but support for it is currently limited because it is so new.<br><br>**If you need immediate deployment, then use SRT or Zixi**. RTP is an option but needs to be augmented with external error control (typically FEC) and encryption technologies. |

For end-to-end delivery systems comprising head ends, primary distribution, and delivery to viewers, you will need to use multiple protocols – and, more specifically, convert transport protocols (with or without transcoding content). Therefore, you will need gateway products such as ATEME TITAN Edge to perform the necessary conversions.

**A small overview by 3rd party IP-Cores about implementation of different Streaming relevant technologies/protocols in software (FPGA/SoC)…:**

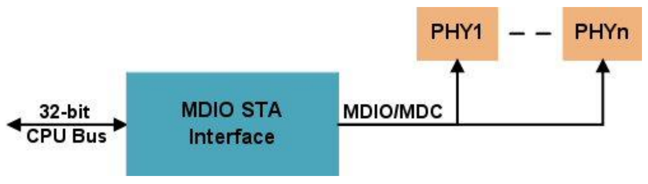
## UDP/IP Stack



- Sends and receives UDP packets on an Ethernet Network
- Includes its own MAC (Ethernet Frame II)
- GMII, RGMII, SGMII (with Xilinx Logicore)
- 10/100/1000 Ethernet modes
- Protocols supported: IPv4, UDP, ARP, ICMP, IGMPv2/v3, DHCP client, VLAN Rx/Tx (supports IEEE 802.1 Q Frame)
- Up to 32 Multicast Address for Reception
- Point to Point UDP Transmission Channel for well-known Link
- Jumbo frames up to 9kB
- Raw Frames
- Differential Services (QoS) for each UDP transmit channel
- TTL IP Layer programmable for each UDP transmit channel

The **UDP/IP Stack** IP core may be used in applications related to Ethernet transmission with Xilinx Technology.

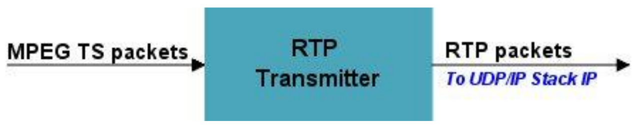
## MDIO Management STA Interface



- Companion core of UDP/IP IP core to control an Ethernet PHY
- Reads and  writes the different registers of the PHY
- Compliant with IEEE RFC 802.3 Clause 22.
- Supports up to 32 devices and up to 32 registers

The ***MDIO Management STA interface*** may be used to access the registers of an Ethernet PHY.
It can also be used for all devices which are MDIO 802.3 Clause 22 Compliant.

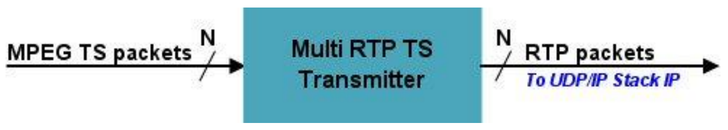
## RTP Transmitter



- Transmits RTP packets
- Companion core of MVD UDP/IP Stack IP core
- RTPv2 encapsulation
- 2D-FEC option available (SMPTE 2022 only for MPEG-TS only - Payload type 33)

The **RTP Transmitter** IP core is specially designed to transmit RTP flows over a routed network. It can be used to send IPTV compliant streams.

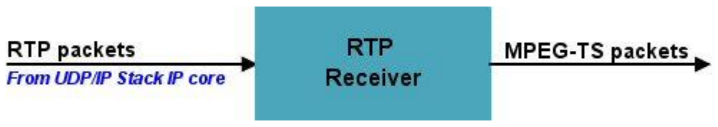
## Multi RTP Transmitter



- Transmits multiple MPEG-TS RTP packets
- Companion core of UDP/IP Stack
- RTPv2 encapsulation
- MPEG-TS payload type only
- From 1 to 7 MPEG-TS Packet per UDP / RTP Packets
- From 4 to 32 independent RTP channels

The **Multi RTP Transmitter** IP core is specially designed to transmit several MPEG-TS RTP flows on a routed network.

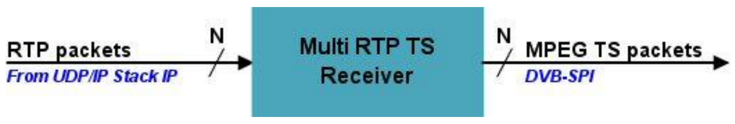
## RTP Receiver



- Receives and process an RTP stream
- MPEG-TS payload type only
- Clock compensation for 90kHz (Clock value for TIMESTAMP Transport Stream) +/-100 ppm
- Efficient packets reordering (for packet loss and packet disorder)

The **RTP Receiver** IP core may be used to receive UDP/RTP packets (or UDP packets only) from an Ethernet Network.

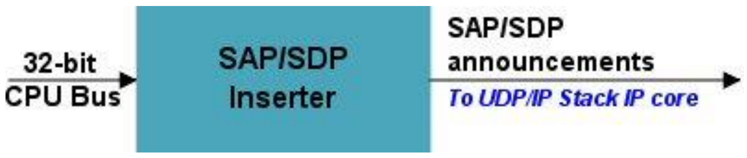
## Multi RTP TS Receiver



- Creates DVB-SPI compliant streams from a UDP/(RTP) CBR Ethernet streams where MPEG-TS packets are not sent regularly (packets come in bursts) or suffer from jitter.
- Up to 32 UDP(RTP) input streams
- Automatic detection of the input stream type (UDP only or RTP)
- Supports any packet length (from 1 to 7 TS, each 188 bytes length, encapsulated in UDP(RTP))
- DVB-SPI compliant transport stream bus output
- Programmable Latency for PCR mode
- Respects PCR Accuracy according to TR 101 290 standard (+/- 500ns) for CBR input streams
- BYPASS mode available for VBR input streams
- Auto-adjustment for a bitrate variation up to 32kbits without any discontinuity of the stream at the output
- For a bitrate variation higher than 32kbits, the PCR calculation is restarted

The **Multi RTP TS Receiver** IP core may be used in applications related to IPTV reception.

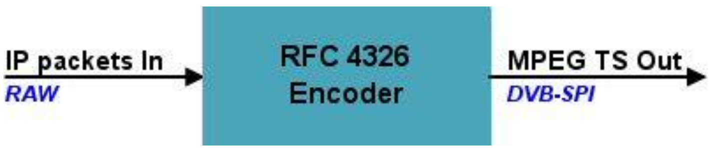
## SAP/SDP Inserter



- Sends periodically SAP/SDP announcements at Multicast address 239.255.255.255 and UDP port 9875.
  These announcements allow the receivers which support SAP/SDP protocol to retrieve all the necessary information to connect to all streaming multicast media present.
- Generates SAP Header and encapsulates SDP data into SAP packets
- Add-on module for the MVD UDP/IP Stack IP core
- SDP description (ASCII) sent through CPU interface
- Up to 128 SAP/SDP Channels

The **SAD/SDP Inserter** IP core allows to send SAP/SDP packets for multicast stream announcements on a local network.

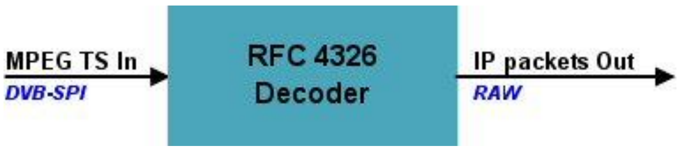
## IP over DVB Encoder (RFC4326)



- Inserts IP packets into user MPEG-2 TS according to RFC4326 standard (Unidirectional Lightweight Encapsulation).
- Completion or no completion mode
- 188 or 204 byte CBR mode output (with RS188/204 encoded byte)
- PAT/PMT generation
- Custom PID for PMT and PAYLOAD frames
- Custom PSI generation delay
- Configurable output rate

The **IP over DVB Encoder (RFC4326)** IP core may be used for transportation of network layer packets over digital television system (DVB-T, DVB-S, DVB-C, ... ).

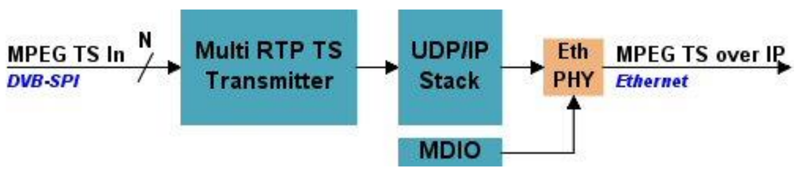
## IP over DVB Decoder (RFC4326)



- Extracts IP packets from user MPEG-2 TS according to RFC4326 standard (Unidirectional Lightweight Encapsulation).
- TS Frame de-encapsulation
- Automatic PAT/PMT analysis and payload recovery
- ETR 101209 Alarms (continuity count error, locked length)

The **IP over DVB Decoder (RFC4326)** IP core may be used for transportation of network layer packets over digital television system (DVB-T, DVB-S, DVB-C, ... ).

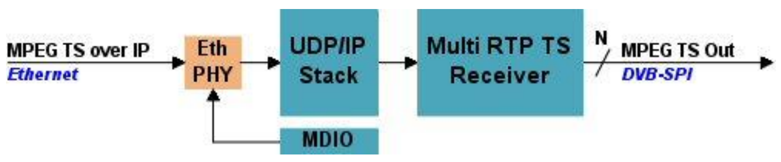
## IPTV Transmitter



- Sends MPEG TS streams in UDP/RTP packets to an Ethernet link
- Built from several MVD IP cores:
- UDP/IP Sack
- MDIO STA Management interface
- Multi RTP TS Transmitter

The **IPTV Transmitter** IP core may be used in TV distribution on an Ethernet network.

## IPTV Receiver



- Extracts one or several MPEG TS streams  (CBR) from UDP/RTP packets from an Ethernet link and sends them to DVB-SPI outputs
- Built from several MVD IP cores:
- UDP/IP Sack
- MDIO STA Management interface
- Multi RTP TS Receiver

The **IPTV Receiver** IP core may be used in TV distribution on an Ethernet network.

## GLOSSARY

| | |
|---|---|
| ALF | Application Layer Framing |
| ARQ | Automatic Repeat request |
| B2B | Business to Business |
| CMAF | Common Media Application Format |
| DASH | Dynamic Adaptive Streaming over HTTP |
| DTLS | Datagram Transport Layer Security |
| FEC | Forward Error Correction |
| FLUTE | File Delivery over Unidirectional Transport |
| HTTP | Hyper Text Transfer Protocol |
| IP | Internet Protocol |
| MMT | MPEG Media Transport |
| NMOS | Networked Media Open Specifications |
| QUIC | This is no longer an acronym and is now the actual name of the protocol |
| RIST | Reliable Internet Stream Transport |
| ROUTE | Real-time Object delivery over Unidirectional Transport |
| RTCP | RTP Control Protocol |
| RTMP | Real-Time Messaging Protocol |
| RTP | Real-time Transport Protocol |
| SRT | Secure Reliable Transport |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |

# REFERENCES

[ALF] Clark, D. D. and Tennenhouse, D. L. (1990), Architectural considerations for a new generation of protocols, ACM SIGCOMM Computer Communication Review, Volume 20, Issue 4 (September 1990).

[DASH] ISO/IEC 23009-1, Information Technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats, January 2012.

[HTTP 3.0] Hypertext Transfer Protocol Version 3 (HTTP/3), IETF

Internet Draft, February 2021 [QUIC] QUIC: A UDP-Based

Multiplexed and Secure Transport, IETF Internet Draft, January

2021. [RFC 1889] RTP: A Transport Protocol for Real-Time

Applications, IETF, January 1996

[RFC 2068] Hypertext Transfer Protocol – HTTP/1.1, IETF, January 1997

[RFC 2616] Hypertext Transfer Protocol – HTTP/1.1, IETF, June 1999

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

[RFC 6347] Datagram Transport Layer Security Version 1.2, IETF, January 2012

[RFC 6726] FLUTE – File Delivery over Unidirectional Transport, November 2012 (obsoletes RFC 3926)

[RIST TR06-1] Video Services Forum (VSF) Technical Recommendation TR-06-1, Reliable Internet Stream Transport (RIST) Protocol Specification – Simple Profile,
October 2018.

[RIST TR06-2] Video Services Forum (VSF) Technical Recommendation TR-06-2, Reliable Internet Stream Transport (RIST) Protocol Specification – Main Profile,
March 2020.

[ROUTE] Real-time Transport Object delivery over Unidirectional Transport (ROUTE), IETF Internet Draft, June 2020.

[SRT] The SRT Protocol, IETF Internet Draft, September 2020.

[ST 2022-1] ST 2022-1 - Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks, 2007.

[ST 2022-2] ST 2022-2 - Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks, 2007.

[ST 2022-5] ST 2022-5 - Forward Error Correction for Transport of High Bit Rate Media Signals over

IP Networks (HBRMT), 2012. [ST 2022-6] ST 2022-6 - Transport of High Bit Rate Media Signals over

IP Networks (HBRMT), 2012.

[ST 2022-7] ST 2022-7 - Seamless Protection Switching of SMPTE ST 2022 IP Datagrams, May 2019.

[TR-03] Video Services Forum (VSF)Technical Recommendation TR-03, Transport of uncompressed Elementary Stream Media over IP, November 2015