ATSC Standard: Scheduler / Studio to Transmitter Link

Doc. A/324:2023-03
28 March 2023

Advanced Television Systems Committee
1300 I Street, N.W., Suite 400E
Washington, D.C. 20005
202-872-9160

Revision History

Version	Date
A/324:2018 Standard approved	5 January 2018
CS Revision of A/324:2018 approved CS Update approved 2nd CS Update approved	23 May 2019 20 September 2019 7 January 2020
A/324:2021 Standard approved Amendment No. 1 approved	24 March 2021 4 October 2021
A/324:2022-03 (references to ATSC documents updated) Amendment No. 1 approved	31 March 2022 13 June 2022
A/324:2022-06 (includes Amendment No. 1 to A/324:2022-03)	13 June 2022
A/324:2023-03 (references to ATSC documents updated)	28 March 2023

Table of Contents

Index of Figures and Tables

ATSC Standard: Scheduler / Studio to Transmitter Link

1 SCOPE

This standard specifies the protocol on the Studio-to-Transmitter Link (STL) from studio side infrastructure to a Single Frequency Network (SFN) of Transmitters. It defines delivery protocols for ALP Transport. The document also defines possible interfaces among the studio infrastructure, for example the interconnection of the ATSC 3.0 Link Layer Protocol (ALP) and a Broadcast Gateway. This document specifies certain constraints on the scheduling of content and signaling on the Physical Layer. The described scheduling process enables Preamble generation and emission time management. It specifies certain aspects of Transmitter behavior and certain parameters of Transmitter operation.

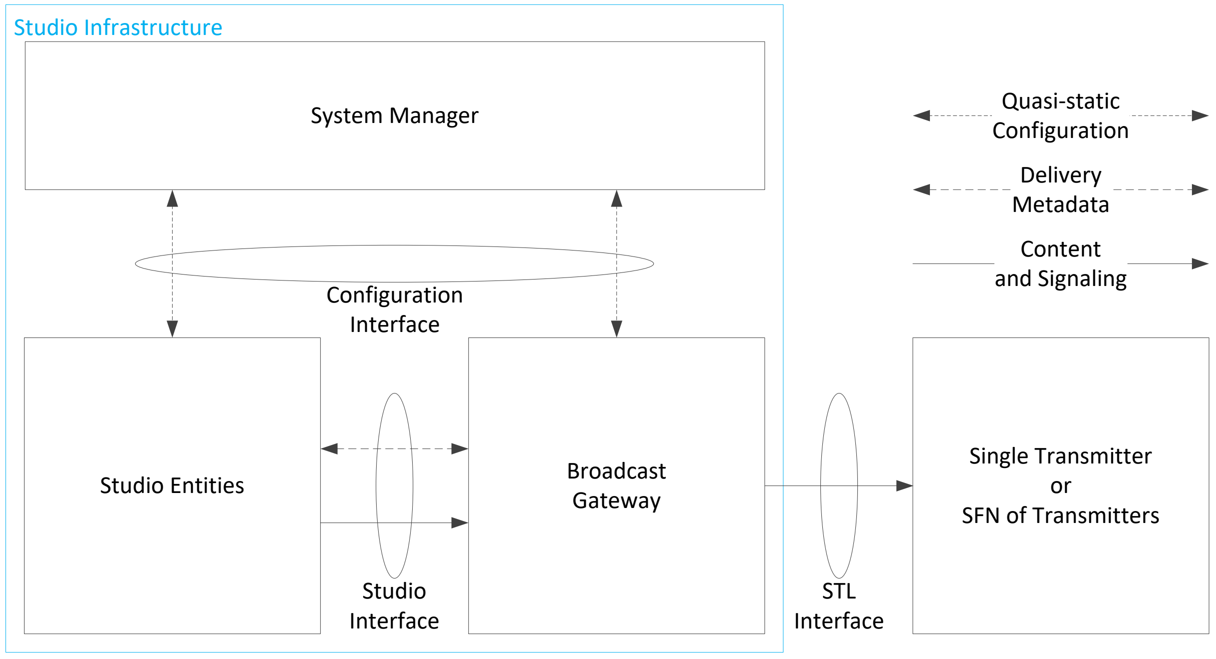


Figure 1.1 In-scope interfaces description

Figure 1.1 depicts an example configuration and connection of these entities. This document provides no specification of broadband delivery of ATSC 3.0 media content.

1.1 Introduction and Background

The ATSC 3.0 system comprises a number of layers that must be connected to one another to construct a complete implementation. Two of the layers that must be interconnected are the transport layer and the Physical Layer. In addition, the Physical Layer is designed to be implemented partially at the studio or Data Source and partially at one or more Transmitters. To enable the necessary interoperation of the layers and system segments, appropriate protocols are necessary so that equipment from multiple suppliers can be assembled into a working system. This document defines four protocols, the ATSC Link-layer Protocol Transport Protocol (ALPTP), the Studio-to-Transmitter Link Transport Protocol (STLTP), the Data Source Transport Protocol (DSTP) and the Data Source Control Protocol (DSCP), for carriage of data through specific portions of the system, as well as a number of operational characteristics of the STL and Transmitter(s). Also defined are a Scheduler to manage operation of the Physical Layer subsystems and two protocols used by the Scheduler (1) to receive high-level configuration instructions from a System Manager and (2) to provide real-time bit-rate control information to Data Sources sending content through the transport layer for emission by the Physical Layer.

1.2 Organization

This document is organized as follows:

- Section 1 – Outlines the scope of this document and provides a general introduction
- Section 2 – Lists references and applicable documents
- Section 3 – Provides definitions of terms, acronyms, and abbreviations for this document
- Section 4 – Provides a system overview
- Section 5 – Defines the Scheduler of Physical Layer resources
- Section 6 – Defines the Common Tunneling Protocol (CTP)
- Section 7 – Defines the Data Source Transport Protocol (DSTP)
- Section 8 – Defines the ALP Transport Protocol (ALPTP)
- Section 9 – Defines the Studio to Transmitter Link Transport Protocol (STLTP)
- Section 10 – Describes Transmitter operation
- Annex A – Describes the parameters used for Physical Layer control
- Annex B – Provides Network Configuration examples
- Annex C – Provides the Scheduler Functional Description
- Annex D – Describes Unicast for Outer Tunneling Packets
- Annex E – Describes Bonded-Channel distribution considerations

2 REFERENCES

All referenced documents are subject to revision. Users of this Standard are cautioned that newer editions might or might not be compatible.

2.1 Normative References

The following documents, in whole or in part, as referenced in this document, contain specific provisions that are to be followed strictly in order to implement a provision of this Standard.

[1] IEEE: "Use of the International Systems of Units (SI): The Modern Metric System," Doc. SI 10, Institute of Electrical and Electronics Engineers, New York, NY.
[2] ATSC: "ATSC Standard: System Discovery and Signaling," Doc. A/321:2023-03, Advanced Television Systems Committee, Washington, DC, 28 March 2023.
[3] ATSC: "ATSC Standard: Physical Layer Protocol," Doc. A/322:2023-03, Advanced Television Systems Committee, Washington, DC, 28 March 2023.
[4] ATSC: "ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection," Doc. A/331:2023-03, Advanced Television Systems Committee, Washington, DC, 28 March 2023.
[5] ATSC: "ATSC Standard: Link Layer Protocol," Doc. A/330:2023-03, Advanced Television Systems Committee, Washington, DC, 28 March 2023.
[6] IETF: "RTP protocol," RFC 3550, Internet Engineering Task Force.

[7] IETF: “RTP Profile for Audio and Video Conferences with Minimal Control,” RFC 3551, Internet Engineering Task Force.

[8] SMPTE: “Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks,” Doc. SMPTE ST 2022-1-2007, Society of Motion Picture and Television Engineers, White Plains, NY, 2007.

[9] SMPTE: “Broadcast Exchange Format (BXF) – Protocol,” Doc. SMPTE 2021-2:2012, Society of Motion Picture and Television Engineers, White Plains, NY, 2012.

[10] ITU-T, “V.41 Data Communication Over the Telephone Network, Code-Independent Error-Control System,” 1993 (or later, if available).

[11] IEEE: “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” Doc. 1588, Institute of Electrical and Electronics Engineers, New York, NY, approved 27 March 2008.

[12] SMPTE: “SMPTE Profile for Use of IEEE-1588 Precision Time Protocol in Professional Broadcast Applications,” Doc. SMPTE ST-2059-2, 2015, Society of Motion Picture and Television Engineers, White Plains, NY, 2015.

[13] IETF: “Network Time Protocol Version 4: Protocol and Algorithms Specification,” RFC 5905, D. Mills, J. Martin, J. Burbank, W. Kasch, Internet Engineering Task Force, June 2010.

[14] ITU-R, “Recommendation ITU-R TF.460-6, Standard-Frequency and Time-Signal Emissions,” Annex 1C "Coordinated Universal Time (UTC)."

[15] ITU-R, “Recommendation ITU-R TF.460-6, Standard-Frequency and Time-Signal Emissions,” Annex 1B "International Atomic Time (TAI)."

[16] IETF: “RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed,” RFC 3095, Internet Engineering Task Force, July 2001.

[17] IETF: “Source-Specific Multicast for IP,” RFC 4607, Internet Engineering Task Force, August 2006.

[18] SMPTE: “SMPTE Professional Media Over Managed IP Networks (ST 2110),” Doc. SMPTE ST-2110, 2018, Society of Motion Picture and Television Engineers, White Plains, NY, 2015.

[19] NIST: “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,” Doc. NIST Special Publication 800-38D, National Institute of Standards and Technology, November 2007. https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=51288

[20] NIST: “Specification for the Advanced Encryption Standard (AES),” Doc. NIST Federal Information Processing Standards Publication 197, National Institute of Standards and Technology, November 26, 2001.

[21] NIST: “Implementation Guidance for FIPS Pub. 140-2 and the Cryptographic Module Validation Program,” Doc. FIPS 140-2IG, National Institute of Standards and Technology and Communications Security Establishment [of Canada], July 3, 2007.

[22] IETF: “OpenPGP Message Format,” RFC 4880, Internet Engineering Task Force, November 2007.

[23] IETF: “Elliptic Curve Cryptography (ECC) in OpenPGP,” RFC 6637, Internet Engineering Task Force, June 2012.

[24] NIST: “Digital Signature Standard,” Doc. FIPS 186-3, National Institute of Standards and Technology, June 2009.

[25] IETF: “Advanced Encryption Standard (AES) Key Wrap Algorithm,” Internet Engineering Task Force, September 2002.

[26] OASIS: "PKCS #11 Cryptographic Token Interface Base Specification Version 2.40 Plus Errata 01," Organization for the Advancement of Structured Information Standards, 13 May 2016.

[27] NIST: "Secure Hash Standard (SHS)," Doc. FIPS 180-3, National Institute of Standards and Technology, October 2008.

[28] W3C: "XML Schema Part 2: Datatypes Second Edition," W3C Recommendation, Worldwide Web Consortium, 28 October 2004. https://www.w3.org/TR/xmlschema-2/

[29] IETF: "The Secure Shell (SSH) Public Key File Format," RFC 4716, Internet Engineering Task Force, November 2006.

[30] ITU-T: "Information Technology – ASN-1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)," Recommendation ITU-T X.690, International Telecommunications Union, August, 2015.

3 DEFINITION OF TERMS

With respect to definition of terms, abbreviations, and units, the practice of the Institute of Electrical and Electronics Engineers (IEEE) as outlined in the Institute's published standards [1] shall be used. Where an abbreviation is not covered by IEEE practice or industry practice differs from IEEE practice, the abbreviation in question is described in Section 3.3 of this document.

3.1 Compliance Notation

This section defines compliance terms used in this document:

shall – This word indicates specific provisions that are to be followed strictly (no deviation is permitted).

shall not – This phrase indicates specific provisions that are absolutely prohibited.

should – This word indicates that a certain course of action is preferred but not necessarily required.

should not – This phrase means a certain possibility or course of action is undesirable but not prohibited.

3.2 Treatment of Syntactic Elements

This document contains symbolic references to syntactic elements used in its various subsystems. These references are typographically distinguished by the use of a different font (e.g., restricted), may contain the underscore character (e.g., sequence_end_code) and may consist of character strings that are not English words (e.g., dynrng).

3.2.1 Reserved Elements

One or more reserved bits, symbols, fields, or ranges of values (i.e., elements) may be present in this document. These elements are used primarily to enable adding new values to a syntactical structure without altering its syntax or causing a problem with backward compatibility, but they also can be used for other reasons.

Throughout this document, different bit settings (i.e., all "0's" or all "1's") are used as placeholders for reserved values, often to follow practices defined in other standards that are referenced in this document. Readers are cautioned that the values specified in each section containing a description of a structure having reserved elements are valid only for the reserved elements in that specific structure. This strategy may deviate from other ATSC practice with respect to such values for reserved bits.

3.3 Acronyms and Abbreviations

The following acronyms and abbreviations are used within this document.

A/V	Audio/Video
AAD	Authenticated Additional Data
AEA	Advanced Emergency Alert
AEAD	Authenticated Encryption with Additional Data
AEAT	Advanced Emergency Alert Table
AES	Advanced Encryption Standard
ALP	ATSC Link-layer Protocol
ALPTP	ALP Transport Protocol
APD	Associated Procedure Description
API	Application Programming Interface
ASN.1	Abstract Syntax Notation 1
ATSC	Advanced Television Systems Committee
BBP	Baseband Packet
BCD	Binary Coded Decimal
BCH	Bose, Chaudhuri, Hocquenghem
BICM	Bit-Interleaved Coded Modulation
BMFF	Base Media File Format
BPPS	Baseband Packetizer Packet Set
BRET	Bootstrap Reference Emission Time
BS	Bootstrap
BSID	Broadcast Stream Identifier
bslbf	bit string, left bit first
BSR	Baseband Sampling Rate
BXF	Broadcast eXchange Format
CAP	Common Alerting Protocol
CDT	Certification Data Table
CFB	Cipher Feedback
CRC	Cyclic Redundancy Check
CSRC	Contributing Source Identifier
CTI	Convolutional Time Interleaver
CTP	Common Tunneling Protocol
D/A	Digital to Analog
DASH	Dynamic Adaptive Streaming over HTTP
dB	decibel
DDE	Data Delivery Event
Demux	Demultiplexer
DER	Distinguished Encoding Rules

DSCP	Data Source Control Protocol
DSS	Data Source Signaling
DSTP	Data Source Transport Protocol
EA	Emergency Alert
ECC	Error Correction Coding
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FI	Frequency Interleaver
FIFO	First In First Out
Gbps	Gigabits per second
GCM	Galois/Counter Mode
GHASH	GCM-based Hash value
GI	Guard Interval
GMAC	Galois/Counter Mode-based MAC
GNSS	Global Navigation Satellite System
GoP	Group of Pictures
GPS	Global Positioning System
HSM	Hardware Security Module
HTI	Hybrid Time Interleaver
IDR	Instantaneous Decoding Refresh
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFFT	Inverse Fast Fourier Transform
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IS	Initialization Segment
ISO	International Organization for Standardization
IV	Initialization Vector
kB	kiloByte
kbps	kilobits per second
KDF	Key Derivation Function
KEK	Key-Encryption Key
LCT	Layered Coding Transport
LDM	Layered-Division Multiplexing
LDPC	Low Density Parity Check
LLS	Low-Level Signaling
LMT	Link Mapping Table
LSB	Least Significant Bit
MAC	Message Authentication Code
Mbps	Megabits per second

MDCoIP	Media Device Control over IP
MDE	Media Delivery Event
MHz	Megahertz
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
MITM	Man In The Middle
MMT	MPEG Media Transport
MMTP	MPEG Media Transport Protocol
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MSB	Most Significant Bit
MTU	Maximum Transmission Unit
Mux	Multiplexer
NAL	Network Abstraction Layer
NIST	National Institute of Standards and Technology
NoC	Number of Carriers
NRT	Non Real Time
OFDM	Orthogonal Frequency Division Multiplexing
OID	Object Identifier
OSN	On-Screen Notification
PAPR	Peak to Average Power Ratio
PGP	Pretty Good Privacy
PHY	Physical Layer
PLP	Physical Layer Pipe
PT	Payload Type
PTP	Precision Time Protocol
QP	Quality Point
RAP	Random Access Point
RDT	ROHC-U Description Table
RF	Radio Frequency
RFC	Request For Comments
ROHC-U	Robust Header Compression UDP
ROUTE	Real-time Object delivery over Unidirectional Transport
RRT	Region Rating Table
RTP	Real-time Transport Protocol
SAP	Segment Access Point
SBS	Subframe Boundary Symbol
SCT	Sender Current Time
SDPS	Security Data Packet Set
SDSP	Security Data Stream Packet
SFN	Single Frequency Network
SISO	Single Input Single Output

SLS	Service Layer Signaling
SLT	Service List Table
SMPTE	Society of Motion Picture and Television Engineers
SNR	Signal-to-Noise Ratio
S-RAP	System Random Access Point
SSM	Source-Specific Multicast
SSRC	Synchronization Source Identifier
STL	Studio–to-Transmitter Link
STLTP	Studio–to-Transmitter Link Transport Protocol
S-TSID	Service-based Transport Session Instance Description
T&M	Timing and Management
TAD	Transmitter-to-Antenna Delay
TAI	International Atomic Time
tcimsbf	two's complement integer, msb first
TI	Time Interleaver
T-RAP	Transport-Random Access Point
TxID	Transmitter Identification
UDP	User Datagram Protocol
uimsbf	unsigned integer, most significant bit first
USB	Universal Serial Bus
USBD	User Service Bundle Description
UTC	Universal Time Coordinated
XML	eXtensible Markup Language
XSD	XML Schema Definition
XOR	eXclusive OR

3.4 Terms

The following terms are used within this document.

a-millisecond – A time interval approximately equal to one millisecond derived from a binary count of nanoseconds and actually equaling 2^{20} nanoseconds, which represents 1,048,576 nanoseconds (i.e., having a Period of 1.048576 milliseconds).

Advanced Emergency Alert – Provides an emergency notification mechanism in ATSC 3.0 that is capable of forwarding a broad range of emergency data. See [4].

ALP Stream – A sequence of ATSC Link-layer Protocol (ALP) packets processed in the order in which they arrive at a Broadcast Gateway and passed over the STLTP to Exciter(s) for transmission in a particular PLP.

Analyzed Media Duration – The period of time on the media timeline utilized by a Scheduler to map input data Streams to a set of Physical Layer frames.

Authenticating Entity –Equipment or software that executes an Authentication Function.

Authentication – A process in which a hash of certain data is taken at the sending end of a link and the identical hashing process is applied again at the receiving end of the link, with a match between the results of the two hashing processes indicating a successful delivery and a mismatch between the two hashing processes indicating some sort of failure in the delivery.

Authentication Function – A process by which an Authentication Tag is calculated for a data packet and compared with an Authentication Tag calculated by and inserted into the header structure associated with the packet by a Signing Entity using a Signing Function.

Authentication Tag – A cryptographic checksum on data that is designed to reveal both accidental errors and the intentional modification of the data.

Baseband Packet – A set of $K_{payload}$ bits that form the input to an FEC encoding process. There is one Baseband Packet per FEC Frame.

Baseband Packetizer Packet Set – A collection of segmented packets representing a single Baseband Packet.

Block – In cryptographic processing, a unit of data that is processed together and to which a Block Cipher is applied. For any given Block Cipher, a bit string the length of which is the block size of the Block Cipher.

Block Cipher – A parameterized family of permutations on bit strings of a fixed length. The parameter that determines the permutation is a bit string called a Key.

Bonded Channels – Two or more broadcast signals on different channels that jointly transmit data or portions thereof representing particular content.

Bootstrap – A defined sequence of symbols that introduces each Physical Layer frame and provides a universal entry point into a digital Transmission signal. Each Bootstrap carries a value that serves as an indicator of the format of an immediately following Preamble symbol.

Bootstrap Reference Emission Time – A time value indicating the instant at which the leading edge of the first symbol of a Bootstrap is to be emitted from the transmitting antenna(s), absent any timing offsets of individual Transmitter(s) in a Network.

Broadcast Gateway – A Broadcast Gateway converts source file objects, for example media, system information, and other opaque files, into SFN baseband description for distribution to Transmitters.

Center Frequency – The point in the spectrum of a Physical Layer signal at which equal numbers of carriers are positioned both higher and lower in the spectrum.

Channel Bonding – A mode of broadcast transmission in which data representing particular content are transmitted on two or more RF channels by separate Transmitters to which portions or all of the data are distributed.

Cipher Block – A block of data equal in size to the fixed length of the bit strings to which the parameterized family of permutations of a Block Cipher is applied.

Common Tunneling Protocol – A notional transport protocol that describes a fundamental structure to tunnel multiple Tunneled Packet Streams within a CTP Stream for carriage from a Data Source to a data sink.

Control Plane – The part of a data network that controls how data is forwarded and provides other management and maintenance functions.

Creation Time – The time at which the value of a Public Key is determined through its derivation from a Private Key.

Cryptographic Token – A hardware device that is secure from intrusion on both physical and software levels, that executes cryptographic processes, and that provides secure storage for security Keys and other data. In this standard, it takes the form of a USB plug-in hardware security module (HSM) and uses the Cryptoki interface protocol.

Cryptoki – The name applied to the Cryptographic Token Interface Base Specification in [26] used to communicate with the USB Token HSMs,

CTP Stream – A notional construct that generalizes the several transport protocol Streams that are generated by the specific instantiations of the Common Tunneling Protocol (i.e., DSTP, ALPTP, and STLTP).

Data Consumer – A device that receives a formatted Stream of data and further processes and/or distributes it.

Data Item – One of a number of units of data stored within a Data Object introduced by related identifier and length values.

Data Object – A data structure that contains a set of Data Items in a defined sequence and that can be stored in a single location in the storage memory of a Cryptographic Token or equivalent software structure.

Data Producer – A device or process that generates a formatted Stream of data.

Data Source – An origination point for data to be transmitted as content by the Physical Layer.

Data Source Mapping Configuration – A set of routing instructions to a data switching function that defines the connection of data Streams from specific Data Sources to the inputs of specific ALP Encapsulators for inclusion in the ALP Streams they produce for transmission on their associated PLPs. Such configuration instructions typically are carried in the form of files.

Data Source Signaling –Information sent from a Data Source to downstream functions to identify specific characteristics of the content of certain packets within the data Stream to provide interlayer communications while avoiding layer violations in the system. Typically, Data Source Signaling information is carried in RTP headers in locations defined in this standard.

Data Source Tunnel – A Tunnel Stream that carries Tunneled Packets, according to the Data Source Transport Protocol (DSTP), from one or more Data Sources through data routing processes to the input(s) of ALP Encapsulator(s), according to routing instructions carried in a Data Source Mapping Configuration.

Earliest Time – A time value that accompanies data sent as input to a Broadcast Gateway to indicate the first instant, as determined using TAI, at which the first byte of related data, including all wrappers and encapsulating protocols, may start emission on the Physical Layer.

Emission Wakeup Field – The two Wakeup Bits in a Bootstrap when treated together as a field.

Exciter – An element of a Transmitter comprising data processing and signal processing functions including at least framing, error correction coding, waveform generation, modulation, and up-conversion to the output RF channel frequency.

Fingerprint – A hash value derived from a set of data for purposes of authenticating or validating the data set.

Fresh – For a newly generated Key, the property of being unequal to any previously used Key.

Header Extension – An optional section of data added to an RTP packet header to add information, space for which need not be allocated when it is not present. An example of the type of data carried in a Header Extension is the value of a GMAC Authentication Tag**.**

Information Header – A separate data structure that precedes a data packet in a Stream to carry information that might have been carried within the data packet but instead is carried separately to avoid having to access the contents of the packet as part of the communication process.

Key – The parameter of the Block Cipher that determines the selection of the forward cipher function from the family of permutations.

Key Identifier – A data value derived from the Fingerprint of a Public Key that can be used as a means to associate related data intended for use with the particular Key**.**

Latest Time – A time value that accompanies data sent as input to a Broadcast Gateway to indicate the last instant, as determined using TAI, by which the last byte of related data, including all wrappers and encapsulating protocols, must complete emission on the Physical Layer.

Majority Logic – A technique for improving the reliability of delivery of certain data through transmission of an odd number of redundant copies of the data and determination at the point of reception of the data whether more than half of the copies received were the same as one another, thereby validating the data received.

Man-In-The-Middle – A type of security attack in which an attacker substitutes content for that which originally was sent.

Media Segment – A portion of a data Stream that is treated as a unit by a Scheduler for purposes of analysis and Transmission.

Multicast – A set of data sent across, for example, an IPv4 network to many recipients simultaneously.

Multiplex – A group of services that are transmitted together over a Network.

Network – A group of Transmitters delivering the same Multiplex.

Nonce – A value that is used only one time under a particular set of conditions or circumstances.

Packet Set – A group of packets carrying segments of a large data structure that has been segmented for the purpose of carriage across a transport connection that is not configured to carry the large data structure.

Packetizer – A process that treats a collection of data (e.g., a portion of a data Stream) by breaking it into segments and wrapping the segments in a header structure, thereby creating packets for Transmission / delivery.

Period – A duration of time.

Physical Layer – A functional protocol that defines the framing, resource allocation, and waveforms of signals emitted for delivery of data and content to receivers.

Plain Channel Bonding – A transmission method in which an ALP Stream is demultiplexed into two or more Baseband Packet Streams and those BBP Streams are transmitted on separate RF channels.

Preamble – The portion of a Physical Layer frame that carries L1 signaling data (see [3]) for the frame.

Preamble Generator – A function within a Broadcast Gateway that accepts instructions from a Scheduler, creates and formats Preamble Packets according to those instructions, and releases the Preamble Packets in the form of a Preamble Stream that can be multiplexed with other data Streams for delivery to the Transmitter(s) under control of the Broadcast Gateway.

Preamble Packet – A packet of data that provides a complete set of information necessary for Transmission following the Bootstrap of a Physical Layer frame to instruct receivers regarding the necessary receiver data processing to permit recovery of the data contained within the frame. The packet also serves to instruct Transmitters with respect to the configuration of the Physical Layer frame that is to be emitted so that the Exciter data processing for the frame can be properly configured.

Preamble Parser – A function within an Exciter that receives Preamble Streams, extracts from them Preamble Payload data, and stores the Preamble Payload data until the time for its emission as part of a broadcast signal. The Preamble Parser then makes the Preamble Payload data available to the Exciter control system for use in configuring the data- and waveform-processing of the Exciter, and outputs individual Preamble Packets at the correct times for their inclusion in the emission of the Physical Layer frames the configurations of which they define.

Preamble Payload – The L1 data carried in a Physical Layer frame to define the structure of the frame and to specify the modulation, coding, and other parameters used in delivery of the frame.

Preamble Stream – A data Stream carrying Preamble Packets, comparable to the data Streams carrying data for PLPs and for Timing and Management functionality, that can be multiplexed with other Streams and delivered as a combined data Stream that is tunneled through the STLTP.

Private Key – A cryptographic Key that is paired with a Public Key, that is known only to a recipient of messages encrypted using the paired Public Key, and that can be used to decrypt messages encrypted with the paired Public Key. A Private Key also can be used to Sign a message so that it can be Authenticated by a corresponding Public Key.

Public Key – A cryptographic Key that can be obtained and used by anyone to encrypt messages intended for a particular recipient, such that the encrypted messages can be deciphered only by using a second Key (a Private Key) that is known only to the recipient of the messages. A Public Key also can be used to Authenticate a message Signed by a corresponding Private Key.

Public Key File – A text file in a defined format that contains a Public Key value, an associated Creation Time value, a Key Identifier, and a Fingerprint value based upon algorithmic processing of the Public Key value in combination with the Creation Time value, as well as a title and comments.

reserved – Set aside for future use by a Standard.

Scheduler – A studio-side function that allocates physical capacity to data Streams based on instructions from the System Manager combined with the capabilities of the specific system.

Security Data – A collection of data sent from a Cryptographic Token installed on a Signing Entity to Cryptographic Token(s) installed on Authenticating Entity(ies) or other Signing Entity(ies) to securely deliver Keys to be used in the Authentication of data and metadata carried in all of the other Tunneled Packet Streams within a CTP Stream or to securely deliver Keys between Signing Entity(ies) through TCP/IP connections between those Signing Entities.

Security Data Packet – A packet that carries data necessary for setting or updating security system parameters such as Keys of various types.

Security Data Stream – A data Stream carrying Security Data Packets that can be multiplexed with other Streams and delivered as part of a combined data Stream that is tunneled through a CTP Stream.

Security Data Stream Consumer – A process that receives and utilizes the Security Data Packets delivered in a Security Data Stream.

Security Data Stream Generator – A process that creates and formats a Security Data Stream containing Security Data Packets.

Signing – The process of generating an Authentication Tag for data formed as a bit string, attaching the Authentication Tag to the bit string data, and communicating the Authentication Tag together with the bit string data to enable confirmation of the Data Source and the accuracy of the bit string data at the point of reception.

Signing Entity – Equipment or software that executes a Signing Function.

Signing Function – A process by which an Authentication Tag is calculated for a data packet and inserted into the header structure associated with the packet.

Single Frequency Network – Multiple Transmitters in proximity to one another radiating the same waveform and sharing a frequency.

SNR Averaging Channel Bonding – A transmission method in which an ALP Stream is demultiplexed into two or more Broadband Packet Streams. The BBP Streams next are data-processed to prepare symbol Streams, data cells then are exchanged between the symbol Streams, and the resulting symbol Streams are transmitted on separate RF channels.

STL Interface – The origin point for the Studio-to-Transmitter Link Tunneling Protocol (STLTP).

Stream – A sequential set of packets carrying data from a Data Producer to one or more Data Consumers.

Studio Interface – The termination point for the ALP Transport Protocol (ALPTP) and/or the Data Source Transport Protocol (DSTP).

System Manager – A conceptual subsystem outside the Transport and Physical Layers that is responsible for coordinating the functions of at least those two layers and the static and quasi-static configurations of various system aspects, for example definition of PLPs or assignment of IP addresses and port numbers to Services. The System Manager does not manage real-time traffic directly.

System Random Access Point – A location within an emission at which entry points into multiple layers and components of a service align, enabling rapid and efficient acquisition of the service.

Timing and Management Data – A collection of data sent from a Timing and Management Generator in a Broadcast Gateway to the Transmitter(s) under control of the Broadcast Gateway for purposes of controlling the emission times of Physical Layer frames, establishing various Transmitter configurations, and setting various Transmitter parameters, independent of the configurations of the frames and waveforms of the emitted signal itself.

Timing and Management Generator – A function within a Broadcast Gateway that accepts instructions from a Scheduler, creates and formats Timing and Management Data according to those instructions, and releases the Timing and Management Data in the form of a Timing and Management Stream that can be multiplexed with other data Streams for delivery to the Transmitter(s) under control of the Broadcast Gateway.

Timing and Management Stream – A data Stream carrying Timing and Management Data, comparable to the data Streams carrying data for PLPs and for Preambles, that can be multiplexed with other Streams and delivered as a combined data Stream that is tunneled through the STLTP.

Token Identifier – A value stored as a concatenated Data Item within the Data Object of a Security Token HSM that uniquely identifies the Security Token HSM within the scope of the network within which the HSM will be utilized. The same Token Identifier value also is shown on the exterior of the Security Token HSM that the Token Identifier value identifies.

Transmission – The signal that is emitted by a Transmitter and the synchronized signal that is emitted by all Transmitters in a Network.

Transmitter – An individual emitter at a specific geographic location on a specific frequency.

Transport Layer – A functional protocol that defines the formatting of data that will be delivered to receivers after its recovery from the formatting required for its physical delivery by the Physical Layer. It provides logical communication between application processes running on different hosts within a layered architecture of protocols and other network components.

Tunnel Packet – A packet that carries in its payload the contents of one or more multiplexed packet Streams, including the corresponding headers and any other structural elements, i.e., a packet in the "outer" layer of a packet Tunneling system.

Tunnel Stream – A Stream of outer Tunnel Packets carrying one or more inner Tunneled Packet Streams and their Tunneled Packet Information Headers, if present.

Tunneled Packet – A packet within a multiplexed group of packets carried in a Tunnel Packet, i.e., a packet in the "inner" layer of a packet Tunneling system.

Tunneled Packet Information Header – A collection of metadata preceding each Tunneled Packet within the DSTP and ALPTP. Depending on the protocol, each Information Header provides length, routing, and priority information to allow recipients of the Tunneled Packets to process them without examining the Tunneled Packet contents.

Tunneled Packet Stream – A Stream of multiplexed inner Tunneled Packets carried within an outer Tunnel Packet, e.g., within an STLTP encapsulation for ECC processing.

Tunneling – A process by which a group of parallel and independent packet Streams is carried within a single packet Stream so that they can be processed, transported, and otherwise treated as a single Stream entity.

Tuple – The combination of Internet Protocol (IP) addresses and port numbers.

Unicast – A set of data sent across, for example, an IPv4 network individually to one recipient.

Wakeup Alert – The occurrence of an AEA message requesting the setting of a non-zero value for the Emission Wakeup Field.

Wakeup Bits – The bits in Bootstrap Symbols 1 and 2 that are used to indicate to receivers that there is high-priority emergency information included in the broadcast. See [4].

3.5 Extensibility

The protocols specified in the present standard are designed with features and mechanisms to support extensibility. In general, the mechanisms include:

- Use of "protocol version" fields
- Definition of fields and values reserved for future use
- Use of XML, which is inherently extensible by means of future addition of new attributes and elements, potentially associated with different namespaces

Receiving devices are expected to disregard reserved values, and unrecognized or unsupported descriptors, XML attributes and elements.

3.6 XML Schema and Namespace

A number of new XML elements are defined and used in this Standard. These elements provide various control elements and attributes defined in this standard. These new XML elements are defined in a namespace in a schema document that accompanies this standard. The namespace used in the schema referenced by this document is described in Section 7.1.1, Data Source Mapping Configuration Description. The sub-string part of the namespace between the right-most two '/' delimiters indicates major and minor versions of the schema. The schema defined in the present document shall have version '1.0', which indicates major version is 1 and minor version is 0.

The namespace designator, "`xs:`", and many terms in the "Data Type" column of tables is a shorthand for datatypes defined in W3C XML Schema [28] and shall be as defined therein.

In order to provide flexibility for future changes in the schema, decoders of an XML document with the namespace defined herein should ignore any elements or attributes they do not recognize, instead of treating them as errors.

All element groups and attribute groups are explicitly extensible with elements and attributes, respectively. Elements can only be extended from namespaces other than the target namespace. Attributes can be extended from both the target namespace and other namespaces. If the XML schema does not permit this for some element, that indicates an error in the schema.

XML schemas shall use `processContents="strict"` in order to reduce inadvertent typos in instance documents.

XML instance documents shall use UTF-8 encoding.

In the event of any discrepancy between the XML schema definitions implied by the tables that appear in this document and those that appear in the XML Schema Definition (XSD) file, those in the XSD file are authoritative and shall take precedence.

The XML schema document for the schema defined in this standard can be found at the ATSC website along with one or more example file(s).

4 SYSTEM OVERVIEW

4.1 Features

The STL subsystem exists between the Transport Layer, which creates ATSC Link-layer Protocol (ALP) packets, and the Physical Layer, which formats Streams of ALP packets for Transmission, in particular Physical Layer Pipes (PLPs), in an emission configuration specified continuously in detail by a Scheduler. Documents [4] and [5] define ALP and other transport layer protocols. Documents [2] and [3] define the Physical Layer protocols. Figure 4.1 shows a high-level overview of the system configuration with applicable document numbers for the adjoining subsystems not defined herein. As depicted in the figure, data to be transmitted enters a Broadcast Gateway using either ALPTP (defined herein) or DSTP (defined herein) in the lower left of the diagram. Other inputs to the Broadcast Gateway are instructions from a System Manager. A Scheduler internal to the Broadcast Gateway controls the pre-processing functions that occur before delivery of the data and various control information to the Transmitter(s). Delivery of the combined data and instructions to the Transmitter(s) occurs in the lower right of the figure using STLTP (defined herein) with optional ECC applied. At the Transmitter across the top of the figure, the data and instructions from the studio are separated, buffered, and used to control the Transmitter as well as to construct the waveform to be emitted to receivers.

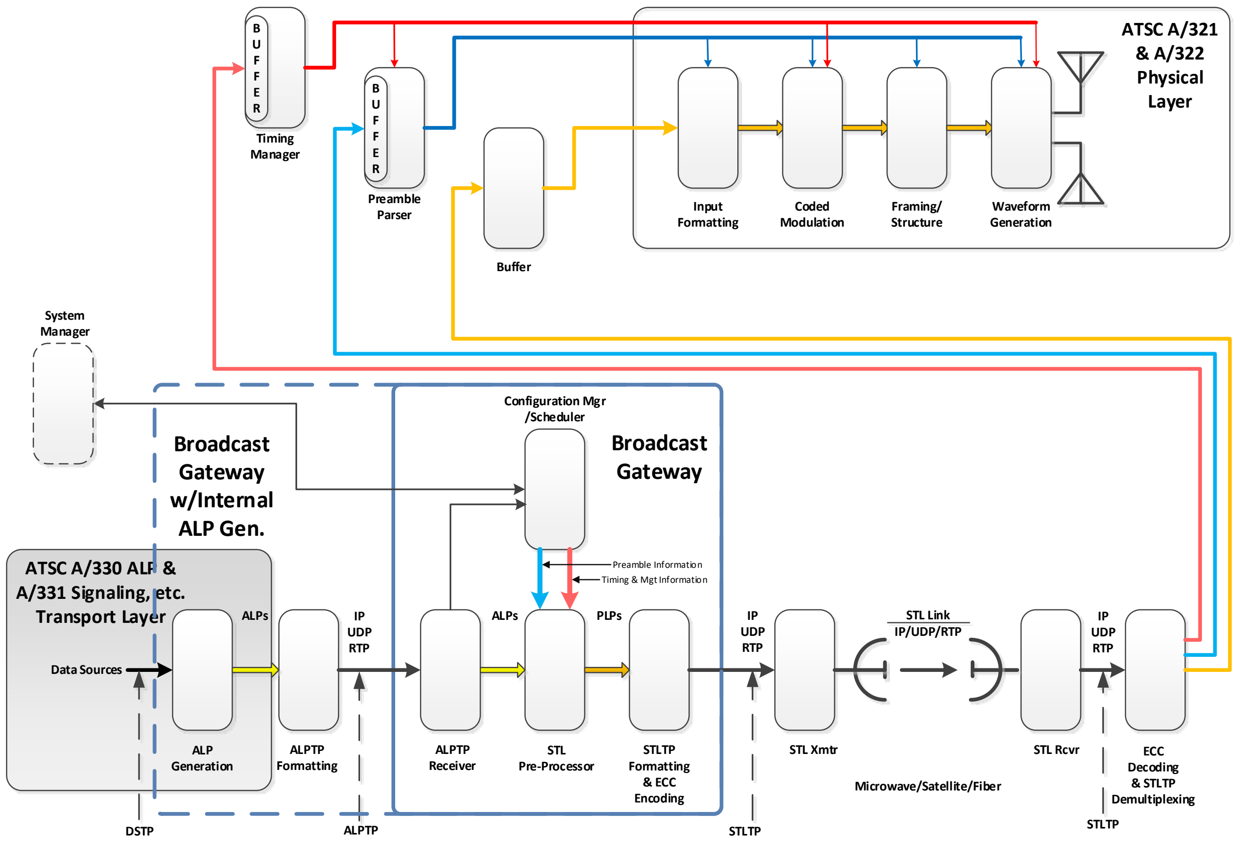


Figure 4.1 High-level overview of system configuration

There is a one-to-one correspondence between individual Streams of ALP packets and individual PLPs. To prepare ALP packets for Transmission, in the Broadcast Gateway, the ALP packets are encapsulated in Baseband Packets (BBPs), which have defined sizes that are determined by a parameter ($K_{payload}$) related to the specific characteristics of the particular PLP(s) in which they will be carried. The sizes of the BBPs in a given Stream are set to ensure that the assigned capacity of the related PLP in a frame is filled by the BBPs derived from an associated ALP packet Stream. ALP packets either are segmented or are concatenated so that they fill the allocated space in the BBPs carrying them as completely as possible without overflowing the available space.

To manage the flow of data through the system, several buffers are required to hold data for purposes of time alignment of data emission. Buffering also is required in certain instances to enable information to be obtained from a data Stream and used to control particular functionality of the system before the corresponding data is processed further. Two specific instances of such buffering exist in the system. The first buffer inserts at least one Physical Layer frame of delay in the STL Pre-Processor to enable sending Preamble information for a given Physical Layer frame to Transmitters prior to arrival of the data to fill that Physical Layer frame. The second buffer accommodates a delay of up to one second to enable synchronization of frame emission timing in the Physical Layer when the delivery delay to each of the Transmitters in a Network is different.

Maintaining the one-to-one correspondence between particular ALP packet Streams and their assigned PLPs through the system requires a method for identifying both the ALP and PLP data Streams. ALP packets are carried between ALP Generators and Schedulers using the ALPTP which is derived from the Common Tunneling Protocol (CTP). The binding of ALP packets to PLPs is accomplished using Tunneled Packet Information Headers that are carried for each ALP packet within the ALPTP. Thus, one ALPTP Stream can carry ALP packets destined for a single PLP or may multiplex multiple ALP packets for multiple PLPs. For STLTP, RTP/UDP/IP Multicast stacks are used with specific UDP port numbers assigned to particular PLPs. Thus, for example, a Baseband Packet Stream derived from an ALP Stream destined for PLP 07 will be carried within an STLTP Stream with a UDP port value also ending in 07. When the emission operates in Single-PLP mode, all the data to be transmitted will be carried in only a single ALP Stream, and all of that data will be transmitted with the same level of robustness. When multiple PLP Streams are used, each PLP can have a different tradeoff of data rate versus robustness, and data Streams can be assigned to appropriate combinations by the System Manager. If the ALP Stream(s) is (are) created within the same equipment that provides the Broadcast Gateway functionality, use of ALPTP may not be necessary.

Figure 4.1 shows a single path carrying the ALP packet Stream(s) and then the PLP packet Stream(s) on its (their) way(s) from the ALP Generator(s) to the Transmitter(s). In reality, if there are multiple Streams at any point in the system, the processing for each of the ALP packet Streams and/or Baseband Packet Streams is applied separately to each of the Streams destined for a different PLP. This separation of processes is diagrammed using parallel paths throughout the remainder of this document, starting with the detailed examination of Figure 4.2.

To manage the characteristics of the emission and to coordinate all of the elements of the Physical Layer subsystem with respect to their parameter settings and times of operation, a Scheduler function is included in the Broadcast Gateway. The Scheduler manages the operation of a buffer for each ALP Stream, controls the generation of BBPs destined for each PLP, and creates the signaling data transmitted in the Preamble as well as signaling data that controls creation of Bootstrap signals by the Transmitter(s) and the timing of their emission. To perform its functions, the Scheduler communicates with a System Manager to receive instructions and with the source(s) of the ALP packets both to receive necessary information and to control the rate(s) of their data delivery.

One form of data relationship that the Scheduler must establish is signaling, in the Preamble of any given Physical Layer frame, the presence of Low-Level Signaling (LLS) data in specific PLPs within that frame. To enable the Scheduler to meet that requirement, upstream ALP generators in turn are required to signal to the Scheduler the presence of LLS data in specific ALP packets. When ALPTP is used (i.e., when ALP generators and the Scheduler are in separate equipment units), such signaling takes place in the ALP Tunneled Packet Information Header that precedes each ALP Packet tunneled through ALPTP. This method avoids the layer violation that would occur if the Scheduler had to determine the presence of LLS by inspecting the content of the ALP packets and covers cases in which the content of the ALP packets is not IP packets.

One of the principal functions of the Scheduler is to generate Preamble data for the Transmitter(s) that it controls. Conceptually, as shown in Figure 4.2, the Preamble generation function is assigned to a Preamble Generator, which is part of the Broadcast Gateway. The Preamble Generator outputs the data to be transmitted to receivers to allow their configurations to match the processes and parameters that will be used in Transmission. As the Transmitter(s) process the Preamble data for emission to receivers, the Preamble data also will be used by the Transmitter(s) to set up the Input Formatting, Coded Modulation, Framing/Structure, and Waveform Generation so that the emitted waveform will match what receivers will be instructed by the Preamble to receive. The exact format for the Preamble data is specified in [3].

Similarly, the Scheduler must control the generation and emission of Bootstrap waveforms by the Transmitter(s). To accomplish this, a data structure, similar to the Preamble, is defined in this document to carry Timing and Management (T&M) data to the Transmitters. Conceptually, as shown in Figure 4.2, a Timing and Management Data Generator is included in the Broadcast Gateway and provides the function under control of the Scheduler.

BBP data is carried across the STL as an RTP/UDP/IP Multicast Stream for each PLP. These Streams are multiplexed into a single RTP/UDP/IP Stream for each broadcast emission to enable reliable delivery to the Transmitter(s) of correctly identified and ordered BBPs. Conceptually, the BBP data Streams, as well as the Preamble Stream, the Timing and Management Stream, and the Security Data Stream are encapsulated as inner Streams carried through the outer (or Tunneling) Stream formed by the STLTP. While the inner Streams are Multicast-only, the outer Stream can be Multicast or Unicast. (See Annex B for an example of Unicast use.) The inner Stream provides addressing of BBP Streams to their respective PLPs through use of UDP port numbers. The outer protocol, STLTP, provides maintenance of packet order through use of RTP header packet sequence numbering. The STLTP also enables use of (SMPTE ST 2022-1) ECC to maintain reliability of Stream delivery under conditions of imperfectly reliable STL Networks.

At the Transmitter(s), an input buffer is used for each PLP to hold BBP data until it is needed for Transmission. There also are FIFO buffers for the Preamble Stream and the Timing and Management Stream. The Preamble Stream processing includes a Preamble Parser that collects all of the configuration information for the next and possibly several upcoming Physical Layer frames to use in configuring the Transmitter data processing for those frames.

Preamble data is scheduled to arrive at the Transmitter input at least one full Physical Layer frame Period prior to the first byte of the associated payload to provide time for the Transmitter data processing to be configured properly. Preamble data also can be sent multiple times in advance to enable acquisition of the data with improved reliability. The same considerations also are applicable to the Timing and Management Data; i.e., it is scheduled to arrive at the Transmitter input at least one Physical Layer frame Period prior to the first byte of the associated payload (+processing delay) it describes, and it can be sent multiple times to enable improved reliability of its acquisition.

The ATSC 3.0 system supports in-order delivery of streaming media files, associated metadata, and generic file delivery to the IP layers of receivers. It is imperative that the systems comprising the ATSC 3.0 distribution path maintain in-order sequencing of the respective Data Source Streams through the several packet encapsulation and delivery processes so that their data is delivered to receivers in the correct order. The three protocols described in this document that operate within the emission portion of the system, namely, DSTP, ALPTP, and STLTP, utilize RTP to enable order recovery, should it be required. RTP transport need not and does not transit the ATSC 3.0 air interface.

4.2 System Architecture

The Studio to Transmitter Link (STL) interface is typically located between the Baseband Packetizer and the Forward Error Correction (FEC) block. There only needs to be one Scheduler and one Baseband Packetizer per RF emission. Multiplexing of multiple Services among stations sharing one RF emission can be accommodated on the input side of the Scheduler.

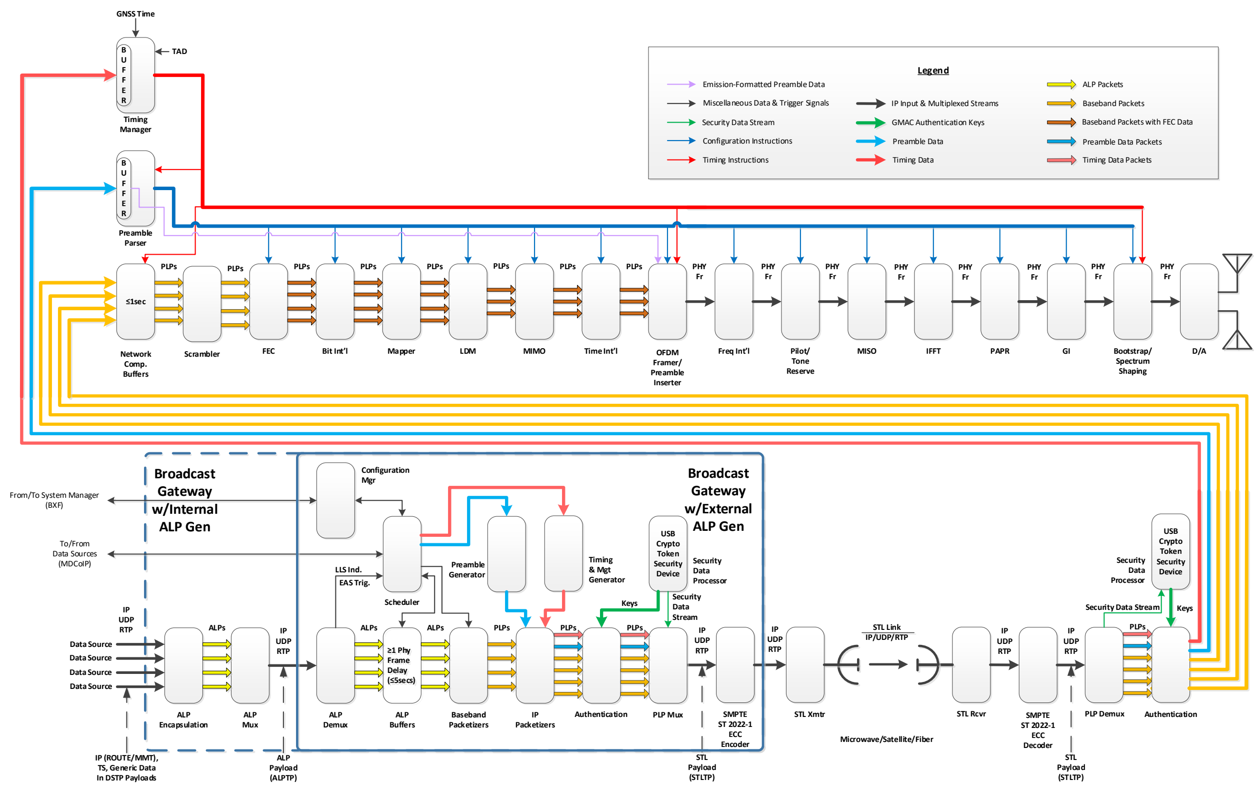


Figure 4.2 System architecture

Figure 4.2 shows a possible system architecture; other configurations are possible. When considering system configurations, data rates and interfaces between functional blocks must be taken into account in developing practical implementations.

4.2.1 System Manager

Configuration aspects of the overall system are controlled by a single entity called a System Manager, which is represented in Figure 4.2 only as a connection to the Configuration Manager in the Broadcast Gateway. A System Manager can be anything from a web-page based setup screen with manual data entry to a fully automated system; its scope is control of an overall facility or system. The System Manager provides high-level configuration parameters for numerous system functions. The System Manager controls static or quasi-static configurations of the Transmission chain. It controls the Physical Layer configuration with respect to how many PLPs operate and the configurations of the individual PLPs, the Services supplied on those PLPs, and the delivery sessions that support the Services that run in the PLPs. A System Manager can establish a pre-determined schedule for system re-configuration, sending instructions to various system devices and subsystems for delayed execution at specified times. All configuration parameters sent by a System Manager to a Configuration Manager are derived from the Preamble parameter set described in [3] Section 9 and listed in Table 5.1 herein. In response, the Configuration Manager provides feedback of Physical Layer capabilities or structure details of selected Physical Layer frame types. Instructions to the Configuration Manager for these changes in configuration must be sufficiently in advance of the time of emission change. The required minimum advance notice of configuration change time depends on total reconfiguration latency of the combination of the

Configuration Manager and the Scheduler. Further descriptions of Scheduler functionality appear in Sections 4.2.2 and 5.2.

Data sources feeding a Broadcast Gateway are identified to it by a System Manager using DSTP Mapping Configurations, as described in Section 7.1. These Data Sources also can have separate control IP addresses, in which case the IP addresses of the control connections are indicated to the Broadcast Gateway by the System Manager. Operation of the Data Source control interface and messages that cross it are described in overview in Section 4.9 and in detail in Section 5.5.

4.2.2 Broadcast Gateway

A Broadcast Gateway is an equipment item that incorporates a number of the conceptual functions necessary in processing data prior to its delivery over an STL to one or more Transmitters. Broadcast Gateways can be implemented in two primary ways, differing primarily in the location of one processing function and the type of interface used to deliver data Streams for Transmission by the Physical Layer subsystem. Both configurations of a Broadcast Gateway are shown in Figure 4.2. In the Broadcast Gateway outlined with a solid line, an external function, upstream in the system, generates ATSC Link-layer Protocol (ALP) packets [5] by encapsulating the data that is to be transmitted in specific PLPs. With external generation of ALP packets, those packets are delivered to the Broadcast Gateway using the ALP Transport Protocol (ALPTP) defined herein, and metadata associated with particular ALP packets and necessary for their Transmission is carried with the packets in their ALPTP headers. In the Broadcast Gateway outlined by the combination of the solid and dashed lines, a function within the Broadcast Gateway performs the ALP packet generation, and a slightly different Data Source Transport Protocol (DSTP) is used to carry both content data and related metadata, the latter of which again is carried in the headers of the DSTP packets.

4.2.2.1 Configuration Manager

The conceptual Configuration Manager within a Broadcast Gateway provides a configuration resource to the System Manager that can accept general waveform structure requirements and from them develop a complete description of the frame structure and waveform that is to be emitted by the Physical Layer. It can inform the System Manager of the capabilities of the system and can provide trial frame structure and waveform details to the System Manager for approval. It also can accept from the System Manager an operational schedule based on use of predefined configurations that the Configuration Manager and the System Manager both have accepted and stored for later use with a naming convention for reference to the predefined configurations. The Configuration Manager provides instructions to the Scheduler, at appropriate times, with respect to the detailed frame structure and waveform that are to be adopted starting at a particular instant and continuing until further instructions are given.

4.2.2.2 Scheduler

The Scheduler within a Broadcast Gateway receives an input of configuration instructions from the Configuration Manager and determines both the frame structure and timing and the waveform details for the next and subsequent Physical Layer frames. The Scheduler must look ahead in time in developing the details of a sequence of Physical Layer frames since instructions about the structures and emission timing of future frames must be sent to Transmitters in advance of arrival of the content data to be transmitted in those future frames. The Scheduler also manages a demultiplexer and a buffer that process data arriving in the form of ALP packets or DSTP packets and any associated metadata in the packet headers. DSTP-delivered data will be encapsulated into

ALP packets before reaching the Scheduler, and its associated metadata will be passed to the Scheduler in much the same way as metadata arriving in ALPTP packet headers. The metadata arriving with the content data is needed by the Scheduler for making a number of decisions. Included are decisions to send Wakeup Bits in the emitted Bootstrap of a Physical Layer frame and to signal the presence of various forms of table data (e.g., LLS) in a frame and in a particular PLP within that frame. The Scheduler also controls the Baseband Packetizer blocks, causing them to create, for each ALP Stream, Baseband Packets of one of several fixed sizes, the choice among which depends upon the value of the parameter $k_{payload}$, which is determined according to the characteristics of the specific Physical Layer Pipes (PLPs) into which their Baseband Packets will be sent [3].

Adjuncts to the Scheduler are a Preamble Generator and a Timing and Management Generator. The Scheduler determines the data to be sent to the Transmitter(s) for inclusion in the transmitted Preamble, and it similarly determines the data to be sent to the Transmitter(s) for control of their operation. The Preamble Generator uses instructions from the Scheduler to form and format a packet Stream to be sent in parallel with the PLP packet Streams to the Transmitter(s) for use by the Transmitters in configuring their operation and for emission to receivers to announce the signal configuration to be received and demodulated. The Timing and Management Generator similarly uses instructions from the Scheduler to form and format a packet Stream to be sent in parallel with the PLP packet Streams to the Transmitter(s) for control of their operations. Unlike the Preamble Stream, the Timing and Management Stream will not be broadcast but rather will be consumed by the Transmitter(s) as the Timing and Management Stream is used to control the timing of Transmitter emissions and other functions either of a group of Transmitters or individually by Transmitter.

4.2.3 Studio to Transmitter(s) Dataflow

The Studio-to-Transmitter Link (STL) may operate on any of fiber, satellite or microwave links. Reliability of all such links is likely not to be perfect, even when they apply their own error correction coding (ECC); consequently, additional ECC is applied in the STLTP path. In addition, Internet Protocol (IP) is supported on all link types. Because of its exposure on any of the delivery means to potential security breaches, it is important that the data traversing the STL be protected by a security mechanism. Therefore, a system for Authentication of the contents of the STLTP data Stream is provided.

4.2.4 STL Operation

Broadcasters have a need to send studio-generated data to their Transmitters. Usually those Transmitters are not co-located at the studio. An STL Interface from the studio to the Transmitter(s) is needed. Requirements for such an interface include:

1) Support for Real-Time Protocol / User Datagram Protocol / Internet Protocol (RTP/UDP/IP) IPv4 and addressing
2) Encapsulation of data for the link
3) Providing a synchronization method to TAI for both data and control
4) Providing signaling of the Transmitter time synchronization for data and control
5) Having measurable maximum latency to allow emission times to be correct
6) Applying Error Correction Coding to the data Stream carried
7) Applying Authentication of the content of the data Stream carried
8) Allowing for redundancy of the delivery channel

4.2.5 SFN Operation

Certain specifications in this standard enable Single Frequency Network (SFN) operations, and the protocol for data carriage from the studio to the Transmitter(s) has the capability to support delivery of certain control data individually to each Transmitter in an SFN. Specifically included are the following specifications and capabilities:

1) Inherent synchronization of the data processing functions of multiple Transmitters fed from the same Broadcast Gateway
2) Inherent time alignment of the emissions of multiple Transmitters fed from the same Broadcast Gateway
3) Specification of the carrier frequency tolerance of Transmitters in an SFN
4) Addressing each Transmitter individually and delivering specific data to each
5) Providing offsets from the Bootstrap Reference Emission Time individually to each Transmitter in a Network to facilitate Network service shaping

Timing considerations are satisfied by the Timing and Management Protocol described in Section 9.3.1, and carrier frequency accuracy is specified in Section 10.3.1.

4.3 Central Concepts: CTP

The Common Tunneling Protocol (CTP) forms the basis for the three main packet protocols described in this document. The CTP defines an SMPTE ST 2022-1 [8] RTP/UDP/IP *Tunnel* that carries a variety of packets and metadata according to each of the specific protocols. All Tunnel Packets comprising a CTP-based Stream may use either multicast or unicast addressing.

Real-time Transport Protocol (RTP) is used with its headers as redefined in Section 6.2.1. Segmentation and reassembly of large Tunneled payload packets and concatenation of small Tunneled payload packets within the Tunnel Packets is performed using RTP header signaling. A segment sequence number within an "outer" RTP header indicating position of a segment within a larger source packet supports segmentation and reassembly, and a value in an "outer" RTP header indicating the offset of the first "inner" packet segment within the payload of the associated "outer" packet supports concatenation. RTP also provides capabilities for ordering packets and determining if any packets have been lost.

If SMPTE ST 2022-1 [8] ECC is in use, any lost packets can be reconstituted. SMPTE ST 2022-1 defines the ECC function and is intended for data rates up to around 1 Gbps, which makes it appropriate for the protocols specified in this document. In addition to the Tunnel Packet Stream, one or two additional Streams of ECC overhead data having related UDP port numbers can be generated by the ECC subsystem and be carried in parallel with the Tunnel Packet Stream.

4.3.1 Content Security System

The Tunneled Packets within the CTP may be signed with a cryptographic signing method to allow detection of Man-in-the-Middle (MITM) attacks against the CTP. The security design is based on the Advanced Encryption Standard (AES) using Elliptic Curve Cryptography and the Galois/Counter Mode (GCM) of AES. The Signing Functions of GCM produce a Message Authentication Code (MAC) called GMAC (GCM-based Message Authentication Code). Tunneled Packets within CTP Streams can be signed with any of four random-number-based Key values or can be left unsigned. Keys for Stream-signing are generated at Signing Entities and communicated to Authenticating Entities over Public/Private-Key-encrypted and authenticated Security Data Streams also tunneled in CTP Streams. Keys used both for signing and for encrypted Key delivery are generated and retained in hardware Cryptographic Tokens. Private Keys are never exposed outside hardware Tokens. Public Keys are exchanged between and stored only in hardware Tokens. Keys used for signing similarly are exchanged between and stored only in hardware Tokens.

4.4 Central Concepts: DSTP

The Data Source Transport Protocol (DSTP) provides a solution for transferring content data, associated metadata, and signaling through a typical IP Network from Data Sources to a Broadcast Gateway as shown in Figure 4.2. While Data Source outputs are expected to be conducive to carriage on normal IP Networks, they do not provide the resources for carriage of metadata and signaling related to the content that must be communicated to Broadcast Gateways. Also, Data Source packets must be kept in order, and packet loss cannot be tolerated. To address the needs beyond normal UDP/IP functionality, the DSTP relies on the Common Tunneling Protocol (CTP) to wrap the various Data Source packets in a convenient RTP/UDP/IP Stream. A separate Tunneled Packet Information Header is prepended to each Data Source packet to carry required metadata and signaling that must be communicated from Data Sources to Broadcast Gateways. The DSTP Tunnel headers and Tunneled Packet Information Headers are discarded by the Broadcast Gateway once the metadata contained in them is processed.

The Tunneled Packets within the DSTP may be cryptographically signed using the Content Security System defined for the CTP (Section 4.3.1). The GMAC tag that results from the signing process is included in the Tunneled Packet Information Header for each signed packet. When signing is active, the Security Data Stream defined by the Content Security System is tunneled through the DSTP using truncated Tunneled Packet Information Headers. Refer to Section 7.2 for details.

4.5 Central Concepts: ALPTP

The ATSC Link-layer Protocol Transport Protocol (ALPTP) provides a solution for transferring ATSC Link-layer Protocol (ALP) packets [5] through a typical IP Network between two separated devices as shown in Figure 4.2. ALP packets are relatively simple constructs having only a minimal header plus packet length information sufficient just for an emission link layer. These headers are not sufficient for transferring ALP packets through a typical IP Network; consequently, ALPTP is defined to provide necessary additional networking information. Moreover, ALP packets must be kept in sequence for each PLP, and packet loss cannot be tolerated.

ALPTP is based on the Common Tunneling Protocol (CTP). The ALP packets are tunneled through the CTP preceded by a Tunneled Packet Information Header that describes the metadata associated with the ALP packet including its priority and PLP ID. There is a single ALP Stream associated with each PLP and the PLP ID in the ALPTP Tunneled Packet Information Headers allows these packets to be grouped. Note that there are no constraints on how many ALPTP Streams can be communicated between an ALP Generator and the Scheduler since metadata regarding binding of ALP packets to PLPs is contained in the Tunneled Packet Information Header accompanying each ALP packet in the ALPTP tunnel.

The Tunneled Packets within the ALPTP may be cryptographically signed using the Content Security System defined for the CTP (Section 4.3.1). The GMAC tag that results from the signing process is included in the Tunneled Packet Information Header for each signed packet. When signing is active, the Security Data Stream defined by the Content Security System is tunneled through the ALPTP using truncated Tunneled Packet Information Headers. Refer to Section 8.2 for details.

4.6 Central Concepts: STLTP

The Studio to Transmitter Link Transport Protocol (STLTP) provides a solution for transferring a potentially large number of parallel Baseband Packet (BBP) data Streams carrying content for a like number of PLPs plus Preamble data and Transmitter control signaling through a typical IP Network from a Broadcast Gateway output to one or more Transmitters using Tunneling, as shown in Figure 4.2. Each Tunneled BBP Stream feeding a PLP derives from a corresponding ALP Stream, and PLP IDs provided with the ALP Streams are used to establish the associations. Error Correction Coding (ECC) is specified to maintain reliable delivery of STL data over the links between sites, but the ECC method selected requires use of uniform size packets in its data processing. To accommodate this requirement, the multiple PLP, Preamble, and control Streams are multiplexed together on a packet-by-packet basis and then tunneled through a single resulting RTP/UDP/IP Common Tunneling Protocol data Stream. When the distribution link does not support multicast (e.g., a public IP network), an RTP/UDP/IP Unicast protocol stack also may be used to deliver the multiplexed PLP(s), Preamble, control, and security Streams.

4.6.1 STLTP Payload Data

The STLTP used to traverse STL IP links carries four distinct types of data:

1) Baseband Packet Streams, the data from each of which populates an individual PLP,
2) Preamble data packets derived from the scheduling process outcome and used both to populate emitted Preambles and to control Exciter configurations,
3) Timing and Management Data packets used to control Transmitter synchronization, to control Bootstrap emission timing, to provide Physical Layer frame identification, and for similar Transmitter management tasks, e.g., TxID control, and
4) Security packets used to provide security information allowing Exciters to authenticate the various Streams within the STLTP if signing is enabled.

The Tunneled Packets within the STLTP may be cryptographically signed using the Content Security System defined for the CTP (Section 4.3.1). The GMAC tag that results from the signing process is included as an RTP Header Extension for each signed Tunneled Packet. Refer to Section 8.2 for details.

4.7 System Time Domains

As can be seen in both Figure 4.1 and Figure 4.2, the Broadcast Gateway sits astride the boundary between studio and Transmitters, which also is the boundary between the Transport Layer (ALP packets) and the Physical Layer (BBPs). Due to the nature of the data formatting in each of those layers, they operate in different time domains. The Transport Layer and layers above it operate in the UTC [14] domain, matching the time on conventional wall clocks, accounting for leap seconds, and often expressing time in NTP [13]. The Physical Layer, on the other hand, operates in the International Atomic Time (TAI) [15] domain, which has second ticks at the same instants as UTC but does not account for leap seconds. This allows the Physical Layer to maintain a constant flow of data, including at leap second insertion instants, without interruption of the Transmission process. Reconciling the differences between the two time domains is a task that falls to the Broadcast Gateway for resolution.

Indication of the value of TAI time on the STL Interface (STLTP) is based on the Precision Time Protocol [11] using 32 bits to represent seconds and 32 bits to represent nsecs.

ATSC Physical Layer Time, which is indicated by the 32 LSBs of TAI seconds and BCD-coded msecs, µsecs, and nsecs, is delivered to receivers in the Preamble [3].

4.8 System Manager Configuration Interface

As described in Section 4.2.1, a System Manager is a conceptual entity that coordinates and controls all broadcaster facilities necessary to produce a specific desired station output configuration and emission. The configuration interface between the System Manager and the Configuration Manager in a Broadcast Gateway transfers information between the two subsystems to enable the system management process with respect to the Scheduler and its associated functions. The System Manager configuration interface may comprise a normal TCP/IP connection between the devices with control functionality structured according to the SMPTE Professional Media Over Managed IP Networks (ST 2110) standards suite [18] and carrying the information described in Section 5.4 herein. The messages are used to negotiate a detailed emission waveform configuration between the System and Configuration Managers and to permit the System Manager to provide instructions to the Configuration Manager with respect to emission requirements and schedules.

4.9 Real-Time Control Interface

As mentioned in Section 4.2.1, the System Manager directs the Broadcast Gateway through the Configuration Manager to control data delivery from various Data Sources by providing a DSTP Mapping Configuration for each Data Source Transport Protocol Stream feeding data to the Broadcast Gateway for delivery through various PLPs. The Broadcast Gateway communicates with the Data Source(s) at the address(es) and port number(s) assigned by the System Manager using a Real Time Control Interface that employs SMPTE Professional Media Over Managed IP Networks (ST 2110) methods [18] and carries the information described in Section 5.5 herein. Messages that can be communicated between the Broadcast Gateway and the various Data Sources are defined in Section 5.5. The Real-Time Control Interface provides such functions as discovery of the capabilities of a Data Source, setting a target bit rate for a Data Source, and managing the speed of data delivery from a Data Source in real time, i.e., updating rates of data delivery as necessary to maintain control of buffer fullness throughout the Physical Layer system.

4.10 Transmitter Requirements Overview

Operation of the ATSC 3.0 Physical Layer system depends on standards-compliant Transmitters meeting certain requirements. Among those requirements are: inclusion of Timing Manager functionality, inclusion of Preamble Parser functionality, conforming to certain buffer models and conditions, maintaining a specified frequency accuracy, supporting emission timing offsets from the Bootstrap Reference Emission Time, and providing for adoption of carrier frequency and emission time offsets as instructed by system Schedulers. These specifications and requirements are detailed in Section 10.

4.10.1 Carrier and Timing Offset for Co-Channel Interference Mitigation

When multiple, geographically neighboring stations are operated on the same RF channel using identical, or even relatively similar, frame and waveform configurations, interference can be caused between the co-channel stations. The interference arises when the pilot locations and characteristics are sufficiently similar in the several signals. Such interference is particularly troublesome when the various stations involved operate with highly robust emission characteristics having very low signal-to-noise ratios. The mechanism that leads to such interference is addition of pilot energy at each OFDM carrier in an RF channel at which pilots are positioned. Addition of the pilot energy at each pilot frequency will lead to incorrect channel estimation by receivers because the channel estimation will not be valid for either/any of the received signals in the overlap areas of the Transmitters' signals. The same effect will result from both single-Transmitter and SFN operation by the neighboring stations.

To counter the sort of interference described, the carrier frequencies of neighboring stations can be offset by a small amount to avoid pilot super-positioning, which would otherwise result in pilot level addition. In addition, super-positioning of Bootstrap emission times must be avoided to prevent failure of Bootstrap recovery from the individual neighboring stations in regions of signal overlap. Section 10.3.3 herein describes methods for avoidance of super-positioning of both pilots and Bootstraps by neighboring stations, thereby extending their interference-free service areas when they operate with low C/N threshold configurations.

5 SCHEDULER DESCRIPTION AND NORMATIVE REQUIREMENTS

Schedulers are the central subsystems in the Physical Layer that enable movement of data from Data Sources, translation of data formatting from Transport Layer forms to Physical Layer requirements, configuration of the transmitter(s) and the waveforms it/(they) emit, buffering of data to stage its delivery, timing of emission of Physical Layer frames, and all the other functions necessary to implement the ATSC 3.0 Physical Layer as embodied in [2] and [3]. A Scheduler accepts inputs from a System Manager that give the Scheduler general instructions as to how the emission is to be configured and scheduled, and the Scheduler outputs data and instructions to the transmitter(s), controlling precisely how the emissions actually are configured.

5.1 Relationship of Broadcast Gateway and Scheduler to the System

Figure 4.2 shows a conceptual block diagram of a Broadcast Gateway and its associated interfaces. A configuration interface allows provisioning of quasi-static aspects of Physical Layer configuration, such as PLP definitions. A Data Source Interface has two ports, one of which delivers content to the Broadcast Gateway along with Data Source Signaling that carries inter-layer information and the other of which is used for real-time control exchanges between the Broadcast Gateway and one or more Data Sources. The required message interchanges on the control interface are defined in this document. Their semantics and the protocol for their carriage are under development at the time of publication and will be documented in the future either as a revision to this standard or as a separate document. Output from the Broadcast Gateway and its Scheduler are via an STL Interface that communicates using STLTP, which carries a complete description of a Physical Layer instance on a frame-by-frame basis to one or more Transmitters.

5.2 Scheduler Functionality

The functional assets of a Scheduler are defined by data size(s), time(s), and levels of robustness on the Physical Layer. The Physical Layer can deliver defined quantities of data at certain discrete times with particular amounts of resilience to channel impairments. There are a large number of parameters, many having a wide range of settings, under control of the Scheduler, and a major part of its function is to make intelligent choices among settings to most efficiently accomplish the objectives communicated to it by the System Manager.

A Broadcast Gateway receives the following inputs and information:

1) Data for transmission, either in the form of ALP-encapsulated content data (via ALPTP) or in the form of content data Streams (via DSTP) that must be ALP-encapsulated within the Broadcast Gateway;

2) Inter-layer signaling sent from the Transport Layer to the Physical Layer in the form of Data Source Signaling to identify certain types of content requiring specific actions to be taken by the Scheduler to properly signal or treat the content in the broadcast emission;
3) Configuration and scheduling instructions from the System Manager; and
4) Capability and status information from the Data Source that feeds each of the ALP/PLP data Streams.

From these inputs, a Scheduler shall create an efficient allocation of the Physical Layer resources conforming to the scheduling and configuration instructions that it receives from its associated System Manager. The Scheduler shall manage buffer fullness throughout the Broadcast Gateway and Transmitter chain, based on current sizes of subframes and frames, maximum delay of the STL delivery Network, available STL channel bandwidth, and requirements for robustness and security across the STL. The Scheduler shall determine Bootstrap Reference Emission Times, create timed control instructions for all Transmitters (collectively and individually), and pass timing and management control messages to the Timing and Management Data Generator so that it can create Timing and Management Data Packets to be sent to the Transmitter(s). As part of managing robustness and security across the STL, the Scheduler shall instruct the Preamble Generator and the Timing and Management Generator how many repetitions of their respective packets shall be sent to the Transmitter(s) and at what times relative to delivery of the contents of the Physical Layer frames with which they are associated.

The Scheduler shall define all of the Physical Layer parameters, such as frame lengths, subframe sizes, PLP configurations, and modulation and coding type and value selections. It shall provide this information and all other details to be communicated to both the Transmitter(s) and off-air receivers to the Preamble Generator for formation into Preamble Packets. The Scheduler shall assign frame identifiers to all Baseband Packets using the Bootstrap Reference Emission Times of the frames in which they are to be emitted.

5.3 Preamble Construction

The Preamble defined in [3] is used to control the configurations of both Transmitter(s) and off-air receivers so that the signals emitted match the decoding processes in receivers, leading to successful data recovery. The Scheduler shall construct a single Preamble Packet for each Physical Layer frame, for this purpose, which packet shall be sent to the Transmitter(s), used by them for their own configurations, and transmitted as part of the Physical Layer frame with which the Preamble Packet is associated. Multiple copies of the Preamble packet for a particular Physical Layer frame may be sent to the Transmitter(s) over a period of time to improve robustness of the Preamble data delivery to the Transmitter(s), through use of majority logic or similar techniques, as described in Section 9.3.1.

A Preamble Packet consists of two fundamental categories of data: quasi-static and dynamic, with the former changing only at infrequent intervals and the latter changing potentially on every frame. The quasi-static information comes from the System Manager, while the dynamic information is developed within the Scheduler as part of its functionality. Both types of information end up in the Preamble. The various Preamble parameters are shown in Table 5.1, and their sources in either the System Manager or within the Scheduler processes are indicated by check marks in the two columns on the right side of the table.

Table 5.1 Preamble Parameters and Their Sources

	Parameters	Instructions from System Manager	Scheduler Generated
Per Frame Data	Channel bandwidth	✓	
	Sampling Rate	✓	
	Major and Minor version values	✓	
	Emergency Alert Wakeup		✓
	L1B Preamble Structure indicator		✓
	Minimum time to next Frame		✓
	Frame length		✓
	Number of subframes		✓
	Number of symbols in the Preamble		✓
	Frame alignment	✓	
	Frame PAPR	✓	
	BSID (L1D)	✓	
	CRC values (L1B)		✓
	CRC values (L1D)		✓
	Preamble NoC	✓	
	FEC Mode for L1-Detail	✓	
	Additional parity for next frame		✓
	Channel BSIDs involved in channel bonding	✓	
	Center frequency of channels involved in bonding	✓	
	MIMO scattered pilot encoding method	✓	
	Return Channel flag	✓	
	LLS flag		✓
	Time info flag		✓
Per Subframe Data	Subframe size	✓	
	Subframe PLP count	✓	
	Subframe MIMO/MISO/SISO	✓	
	Subframe FFT size	✓	
	Subframe NoC	✓	
	Subframe GI	✓	
	Subframe pilot pattern	✓	
	Subframe pilot boost	✓	
	Subframe boundary symbol flags	✓	
	Subframe FI mode	✓	
Per PLP Data	PLP ID	✓	
	PLP type		✓
	PLP size		✓
	PLP start position		✓
	PLP number of subslices		✓
	PLP subslice interval		✓
	PLP LLS flag		✓
	PLP scrambler type	✓	
	PLP FEC mode	✓	
	PLP position of first complete FEC Block		✓
	PLP code rate	✓	

	PLP modulation	✓	
	PLP time interleaver mode	✓	
	PLP CTI depth		✓
	PLP CTI start row		✓
	PLP CTI position of first complete FEC block		✓
	PLP HTI inter-subframe interleaving flag		✓
	PLP HTI number of TI blocks or subframes		✓
	PLP HTI max interleaving FEC blocks per interleaving frame		✓
	PLP HTI number of FEC blocks in the current interleaving frame		✓
	PLP HTI Cell Interleaver flag		✓
	PLP LDM layer	✓	
	PLP LDM injection level	✓	
	PLP Channel BSIDs involved in channel bonding	✓	
	PLP Center frequency of channels involved in bonding	✓	

5.4 Broadcast Gateway Management Protocol

The interface between the Broadcast Gateway and System Management functions may use SMPTE Broadcast eXchange Format (BXF) protocol as described in [9] or another appropriate syntax. Note that the Broadcast Gateway described here combines the Scheduler and internal ALP Generator functions as shown in Figure 4.2.

The parameters that use this protocol are those listed in Table 5.1 under the parameter column with a check mark in the "Instructions from System Manager" column. These parameters are quasi-static in nature and do not routinely change between Physical Layer frames. Scheduler function configurations and constraints are set with these parameters, which are allowed to change quasi-statically. An emission schedule for a set of parameters can be similar to a program schedule, in which the parameters can change over the course of a day. Detailed description of these parameters is provided in Annex A.

In addition to providing the quasi-static Preamble parameters, the System Manager is expected to identify for the Broadcast Gateway:

1) The Data Source Mapping Configuration, as defined in Section 7.1, which describes the mapping of Tunneled Data Source Packets from one or more incoming DSTP Streams into ALP Streams. This mapping is performed by a data switch positioned between the incoming DSTP Streams and the ALP Generator function of the Broadcast Gateway or a standalone ALP Generator when external ALP Generators are used.
2) The Destination Multicast IP address associated with each ALP Stream, where the port number used indicates the particular ALP / PLP pair.
3) Whether an ALPTP stream is to be generated and, if so, the ALPTP Destination Multicast IP address and port number and other configuration parameters to be used to generate the ALPTP output. When an external ALP Generation function is used, the same instructions and Data Source Mapping Configuration shall apply to it.

Messages between a System Manager and a Scheduler shall include:

- Capabilities inquiry to inform the System Manager of the assets and capabilities of the Scheduler.
- Configuration design inquiry to provide a detailed Physical Layer frame design from a Scheduler to a System Manager to meet a set of requirements from the System Manager.

- Current configuration inquiry to report to the System Manager the current state of operation of the Scheduler and any presets available for use.
- System Setup instructions to provide to the Scheduler the fundamental configuration settings for operation in the particular system.
- Scheduled configuration instructions to instruct the Scheduler when to place into operation one or a number of preset configurations stored in the Scheduler and available for use.

Scheduler feedback to the System Manager shall indicate the Physical Layer capabilities and the structure details for each requested frame type. Scheduler to System Manager messages include:

- Capabilities response
- Configuration design response (acknowledgement of preset instructions)
- Current configuration response
- System setup acknowledgement
- Scheduled configuration acknowledgement

5.5 Data Source Control Protocol (DSCP)

The interface between the Scheduler and the one or multiple Data Sources that provide content data for emission shall permit the Scheduler to control operation of the Data Sources so that they deliver the amount of data needed at the times at which it is needed to utilize the capacity of the various PLPs efficiently.

The Scheduler shall use the Data Source Control Protocol to direct Data Sources to provide data in specific target bit rate ranges and to throttle those Data Sources to efficiently fill PLP capacities. Via the DSCP, real-time control messages are sent to Data Source IP control addresses as specified by the System Manager. Data Sources can be multiplexers, encoders, servers, and the like.

Messages between Schedulers and Data Sources can include:

- Capabilities inquiries and responses
- Target bit range instructions from a Scheduler to a Data Source
- Throttling instructions from a Scheduler to a Data Source to make fine adjustments of data delivery

The control protocol for communication between the Scheduler and Data Sources shall use at least the fields shown in Table 5.2. The Scheduler configures the Physical Layer on a per PLP basis; therefore the DSCP has a link to a Data Source for each PLP. The DSCP can accommodate seamless changes that do not affect service.

Table 5.2 Real Time Control Field Definitions

Syntax	No. of Bits	Format
rt_control () {		
bit_rate_capacity	18	uimsbf
bit_rate_granularity	10	uimsbf
bit_rate_target	18	uimsbf
bit_rate_max	18	uimsbf
}		

bit_rate_capacity field shall indicate to the Scheduler the Data Source bit rate capacity (e.g., minimum and maximum bit rates) in units of kbps.

bit_rate_granularity field shall indicate to the Scheduler the Data Source bit rate step size in units of kbps. Rate of change in the PLP can be from a few kbps to tens of Mbps depending on input coding rates and PHY frame configurations. The fastest rate of change depends on the shortest allowed frame length, which is 50msec.

bit_rate_target field shall indicate the desired bit rate for Scheduler capability in handling PLP payloads. Range of targets shall be between 1 kbps and 157 Mbps with units of kbps.

bit_rate_max field shall indicate the maximum bit rate value not to be exceeded into the Scheduler in units of kbps.

6 COMMON TUNNELING PROTOCOL (CTP)

The Data Source Transport Protocol (DSTP), ALP Transport Protocol (ALPTP) and STL Transport Protocol (STLTP) all rely on the capability of encapsulating packets of some form specific to the protocol into an RTP/UDP/IP packet Stream. The encapsulated packets are similar enough that a Common Tunneling Protocol can be defined that is common to all three protocols. In each case, source packet Streams are combined into a single Tunnel Packet Stream. The Common Tunneling Protocol (CTP) hides the details of the various Tunneled Packets by treating them and any associated metadata as opaque payload data. The Common Tunnel consists of an RTP/UDP/IP unicast or multicast IPv4 packet Stream based on the SMPTE ST 2022-1 [8] standard whose payloads shall contain one or more Tunneled Packet Streams. The Tunneled Packets are also referred to as *inner* packets while the *outer* packets are referred to as Tunnel Packets.

Two of the protocols, DSTP and ALPTP, described in this document specify information headers that precede their respective Tunneled Packets, specifically, Data Source Tunneled Packets, which are UDP/IP packets and ALP Tunneled Packets, which are ALP Link Layer packets, respectively. For convenience, this section refers to these combined Information Header and corresponding packets as simply Tunneled Packets, with the Information Headers treated as part of the packet. In fact, this is appropriate since syntax and semantics of the internal packets should not be known to the tunnel except when framing to determine where each Tunneled Packet starts within the RTP/UDP/IP payload of the Tunnel Packet.

6.1 Overview

Tunnel Packets are distributed over UDP/IP multicast or unicast using a modified RTP [6] protocol. UDP/IP multicast is used successfully in many industries to deliver redundant video Streams over copper and fiber physical layers. Routing and redundancy can be accomplished using IGMPv3 Source-Specific Multicast (SSM) (RFC 4607) [17]. Examples of various network configuration topologies are described in the Network Configuration Examples found in Annex B.

Since UDP does not guarantee packet order or, in severe cases, delivery, RTP shall be used to maintain and allow reconstruction of the Data Source Tunnel Packet order as well as to detect loss in the network. All Tunnel Packets shall be carried with RTP/UDP/IP multicast IPv4 protocol using the RTP header modifications described in Section 6.2.1.

Since the multiple Tunnel Streams can be received through networks of varying reliability, the Common Tunneling Protocol allows SMPTE ST 2022-based ECC to be applied. If ECC is not applied, Tunnel Packets shall still be defined with a fixed packet size to simplify implementations. A depiction of the Tunnel encapsulation is shown in Figure 6.1.

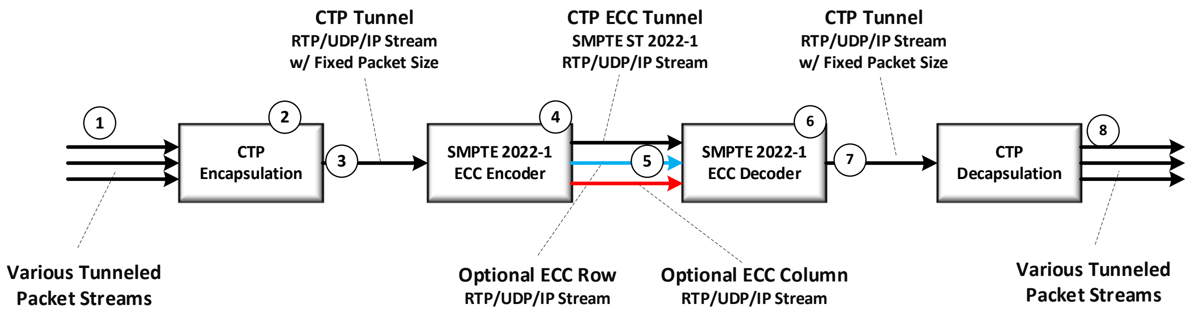


Figure 6.1 CTP Tunnel Encapsulation Process

The following paragraphs describe each of the callouts, (1) through (8), in Figure 6.1. Items (1) through (5) are further detailed in the ECC Encoding Process Section 6.3.1 below.

1) The multiple paths represent the Tunneled Packet Streams**,** which have a variety of forms depending on the actual transport protocol. For DSTP, the input is one or more UDP/IP Data Source Streams with metadata information headers. For ALPTP, the packets are ALP packets with metadata information headers. For STLTP, the Tunneled Packet Streams are RTP/UDP/IP Streams that are generated for each PLP as well as the Preamble, Timing and Management and Security Data Streams described in Sections 9.2.1, 9.3.1, and 6.4.5, respectively.
2) The CTP Encapsulation function is configured to accept packets from multiple packet Streams to be tunneled.
3) The Tunneled Packets are grouped into fixed-size payloads to accommodate the SMPTE ST 2022-1 ECC process [8]. The CTP RTP fields are defined to allow the Tunneled Packet Streams to be easily recovered and forwarded (refer to Section 6.3.2 below). The size of the fixed-size ST 2022-1 packets is not defined by this standard and is assumed to be configurable. It is expected that the packet size is within the typical 1500 MTU byte range limit. Note that, the larger the packets and the number of rows and columns defined, the longer the latency when performing ECC calculations. The resultant fixed-size RTP packets with the Tunneled Packet payloads are referred to as the Tunnel Packets.
4) The process described in Section 6.3.2 below buffers multiple fixed-size payloads and performs XOR operations on groups of Tunnel Packets producing additional ECC packets. The level of error correction can be configured, including the number of columns ('L') and the number of rows ('D'). Both Level A and Level B configurations, described in [8], shall be supported, with a preference for use of the 2D scheme (Level B). The minimum packet array size supported shall be 256 packets, which shall be configurable into any valid arrangement according to ST 2022-1 [8].
5) The Tunnel Stream and up to two ECC packet Streams are sent to the SMPTE ST 2022-1 ECC Decoder using up to three RTP/UDP/IP Streams on three separate ports. If the Tunnel Stream is sent on IP address 239.xxx.xxx.xxx, with port N, then the first ECC Stream is sent on the same address 239.xxx.xxx.xxx, with port N+2, and the second ECC Stream is also sent on the same address 239.xxx.xxx.xxx, with port N+4. This is in accordance with SMPTE ST 2022-1 ([8] Section 8.1, Paragraph 6).
6) The Tunnel Stream and the associated ECC Streams are processed by the SMPTE ST 2022-1 ECC Decoder. Note that the decoder detects missing packets from the RTP sequence numbers on the Tunnel Stream and reconstitutes them from the ECC Stream(s). The

decoder also reorders any packets as they are being received within the constraints of its internal buffering.

7) The resultant Tunnel Packet Stream is an in-order collection of fixed-size packets identical to the input Stream described in step (3) above.
8) The CTP Decapsulation function extracts each Tunneled Packet Stream from the payloads of the Tunnel Packets. The Tunnel Packet RTP headers contain information allowing the Tunneled Packet Streams to be reframed.
9) The reconstituted IP and associated metadata packets, if applicable, then are forwarded to the next stage of the Transmission system.

6.2 RTP/UDP/IP Multicast Considerations

In addition to the context described above, RTP/UDP/IP Multicast with SSM may be used when moving multicast traffic between Data Producers and Data Consumers. Here a 'Data Producer' is defined as a device or process generating an IP multicast, and a 'Data Consumer' is a device or process receiving the IP multicast. Since RTP defines a 'sequence number' field, the various packets can be recovered and re-sequenced within each Data Consumer. This allows packets to be reordered, duplicate packets to be ignored, and packet loss to be detected, possibly avoiding the problems sometimes encountered by UDP datagram delivery.

In redundant systems, the Data Consumer must be aware of the several sources of contribution multicast Streams. For each contribution Stream, an IGMPv3 SSM 'join' is issued by the Data Consumer for a specific multicast destination IP address and IP source address. The router only sends UDP packets matching the specified source and destination addresses requested in the IGMP join to the Data Consumer's physical IP connection. If the Data Consumer detects loss or problems with the current packet Stream, it can release that Stream and join another using the same IP destination address with a different IP source address. The Producers, in this case, operate independently and with no knowledge of the redundant topology.

6.2.1 Multicast Addressing

To use multicast addressing, a CTP Stream shall be broadcast using a multicast address in the range 239.0.0.0 through 239.255.255.255, which is within the Organization Local Scope range defined by IETF. The actual addresses used are implementation-dependent and are defined by configuration.

6.3 Common Tunneling Protocol Design

The three protocols described in this document transport the various content, control and metadata that make up an ATSC 3.0 broadcast. As indicated in the introduction to Section 6, each of the protocols relies on the Common Tunneling Protocol (CTP) to hide various aspects of the individual protocol so that they may all be treated in a similar fashion when routing through a broadcast studio. The CTP shall encapsulate these various tunneled Streams into an RTP/UDP/IP Tunnel Stream based on SMPTE ST 2022-1 [8]. SMPTE ST 2022-1, in turn, extends the RTP [6] protocol with Generic ECC Payload Format. RTCP and other control Streams are explicitly excluded by SMPTE ST 2022-1 ([8] Section 7.1, Paragraph 2).

In addition to the constraints defined by [8], this document changes the definition of some of the standard RTP header fields to enable internal payload framing. The **marker (M)** bit, along with the repurposed **SSRC_ID** field, are defined herein to provided various signaling functions, including the **packet_offset**. These fields and the payload framing mechanism for the Common Tunneling Protocol RTP header are described in Table 6.1.

Table 6.1 RTP Header Field Definitions for Common Tunneling Protocol

Syntax	No. of Bits	Format
RTP_Fixed_Header () {		
version (V)	2	'10'
padding (P)	1	bslbf
extension (X)	1	bslbf
CSRC_count (CC)	4	'0000'
marker (M)	1	bslbf
payload_type (PT)	7	uimsbf (Table 6.2)
sequence_number	16	uimsbf

if (PT == DSTP ||
  PT == ALPTP) {		

timestamp_min ()	32	Table 6.3
}		
else if (PT == STLTP) {		
timestamp ()	32	Table 9.2
} else {		
reserved	32	for (i=0; i<32; i++) '0'
}		
protocol_version	2	uimsbf
if (PT == STLTP) {		
redundancy	2	uimsbf
number_of_channels	2	uimsbf
reserved	10	for (i=0; i<10; i++) '0'
}		
else {		
reserved	14	for (i=0; i<14; i++) '0'
}		
packet_offset	16	uimsbf
}		

Please refer to [6] for base definitions of the fields described in Table 6.1 and refer to [8] Section 7.1 for constraints on those base definitions.

version – Indicates the version number of the RTP protocol. '2' is currently defined by [6].

padding – Indicates, if set to '1', that padding may be included in the payload. No padding is included if the value is '0'. Padding shall be supported as specified in [6].

extension – Indicates, if set to '1', that an RTP extension follows the header. If set to '0', no Header Extension is included. If a Header Extension is included, all CTP packets must have the same length extension as dictated by Section 7.1 of [8].

CSRC_count – Provides the count of additional contributing source identifier fields. ST 2022-1 [8] requires this field to be set to '0'.

marker – Indicates, if '1', that a Tunneled Packet starts within the payload of this RTP Tunnel Packet. When the **marker** bit is set to '1', the first byte of the first Tunneled Packet starting within the payload shall be indicated by the value of the **packet_offset** field. Subsequent Tunneled Packets can be found by using the length fields within each Tunneled Packet, depending on the protocol, DSTP, ALPTP or STLTP. Typically, bytes preceding the first Tunneled Packet start will be a continuation of the last Tunneled Packet of the previous Tunnel Packet.

payload_type – The values used for the various Tunnel Streams defined in this standard shall be as specified in Table 6.2.

Table 6.2 Code Values for **payload_type**

payload_type	Abbreviation	Meaning
81 ('1010001')	DSTP	Data Source Transport Protocol Tunnel Packets and Information Headers
82 ('1010010')	ALPTP	ALP Transport Protocol Tunnel Packets and Information Headers
97 ('1100001')	STLTP	STL Transport Protocol Tunnel Packets

sequence_number – As per [6], the **sequence_number** shall increment by one, modulo 2^{16} or 65536, for each packet with the same IP address and port tuple. The initial **sequence_number** should be randomized, although this is not important in this setting since the Stream will not be restarted frequently nor is it expected to be on a public network. Since UDP/IP does not guarantee ordered delivery and may even duplicate packets, the **sequence_number** can be used to reconstitute any Stream as produced and to detect loss, allowing a backup Stream to be selected or ECC to be applied if enabled.

timestamp_min () – When the **payload_type (PT)** value is either '81' (DSTP) or '82' (ALPTP), the **timestamp_min ()** value shall contain the earliest time specified in any of the Tunneled Packet Information Headers (see Section 6.4 and Section 8). If no Tunneled Packet Information Headers are present, then this value shall be set to '0'. Providing the earliest time allows the scheduling system to prioritize packets from multiple DSTP input Streams. The **timestamp_min ()** value shall be formatted according to the short-form of NTP specified in RFC 5905 [13] as shown in Table 6.3. Note that this field replaces the **SSRC_ID** field specified in RFC 3550 [6]. The semantic definitions of the fields may be found in paragraphs following the table.

Table 6.3 Timestamp Field Definitions for DSTP and ALPTP

Syntax	No. of Bits	Format
timestamp_min () {		
seconds	16	uimsbf
fraction	16	uimsbf
}		

seconds shall carry a value equal to the 16 least significant bits (LSBs) of the seconds portion of the UTC time value [14] of the targeted Bootstrap Reference Emission Time.

fraction shall carry a 16-bit fractional seconds value of the UTC time value [14] of the targeted Bootstrap Reference Emission Time – allowing a resolution of approximately 15 microseconds.

The **timestamp_min ()** field within the RTP header allows a Data Source or ALP encapsulation system to specify schedule constraints on delivery of packets. Methods for determining the timestamp values are outside the scope of this standard, but provision is made herein to enable respecting the constraints without requiring inspection of the contents of the packets coming from Data Sources and being transported to the ALP layer, as well as ALP packets being carried to the Scheduling system.

Please note that NTP is based on UTC [14] and, thus, is adjusted for leap seconds. There is no adjustment for leap seconds in emission timing, which is based purely on TAI seconds and fractional seconds [15]. See Section 9.3.2 for more information on Bootstrap emission timing.

timestamp () – If the **payload_type** is '97' indicating an STLTP Tunnel Packet, the **timestamp ()** field shall be defined according to Table 9.2 and its subsequent semantic definitions. This value shall indicate the latest time at which the start of the payload may be delivered. A value of '0' shall denote that the packet may be delivered with best effort. Note that this field replaces the **SSRC_ID** field specified in RFC 3550 [6].

protocol_version – shall indicate the version number of the outer Tunnel Packet protocol. The current version is '01'.

redundancy – When the **payload_type** is set to '97' indicating an STLTP packet, this field shall indicate the number of redundant Streams of outer Tunnel Packets sent over the STL. The value '0' shall indicate that there is no redundancy of outer Tunnel Packets, and values greater than '0' shall indicate that there is redundancy of outer Tunnel Packet Streams over the STL and the number of such redundant Streams.

number_of_channels – When the **payload_type** is set to '97' indicating an STLTP packet, the **number_of_channels** field shall indicate the number of broadcast channels minus one transported by the outer Tunnel Packet Stream. When the value is set to '0', an outer Tunneling Stream shall contain one group of inner Tunneled Packet Streams, using the port range 30000 – 30066. Note that a group of inner Tunneled Packet Streams is composed of the Preamble, Timing and Management, Baseband Packet, and Security Data Streams required for transmission of one RF channel. When the value is set to '1', an outer Tunnel Data Stream shall contain two groups of inner Tunneled Packet Streams. When the value is set to '1', the port range of one group of inner Tunneled Packet Streams (corresponding to one RF channel) shall be 30000 – 30066, and the port range of the other group of inner Tunneled Packet Streams (corresponding to the other RF channel) shall be 30100 – 30166. See Section 9.5 for an example of use of this field.

packet_offset – If the **marker (M)** bit equals '1', this field shall define the payload offset of the first Tunneled Packet. The field shall indicate the number of bytes *after the RTP header and extension* before the first Tunneled Packet starts within the tunnel payload. If the Tunneled Packet starts immediately after the header and extension (if any), then **packet_offset** shall be '0'. Figure 6.2 provides a graphical example of how Tunneled Packets are encapsulated within the fixed-size SMPTE ST 2022-1 CTP RTP Tunnel Packets. Note that only the position of the first packet within the tunnel payload is required since all Tunneled Packet types contain a length field that can be used to calculate the offset of subsequent packets.

6.3.1 CTP RTP Encapsulation Example

Following is an example that shows how encapsulation is done using RTP to create a complete Tunneled Data Stream for SMPTE ST 2022-1 ECC processing. It examines the case of a single Stream. Note that the protocols derived from this Common Tunneling Protocol may request multiple, simultaneous Tunnel Streams. The example is broken into two diagrams that show the overall structure of a CTP Stream (in Figure 6.2) and the details of the packing process, including the locations of the several packets of the different layered protocols within the Stream (in Figure 6.3).

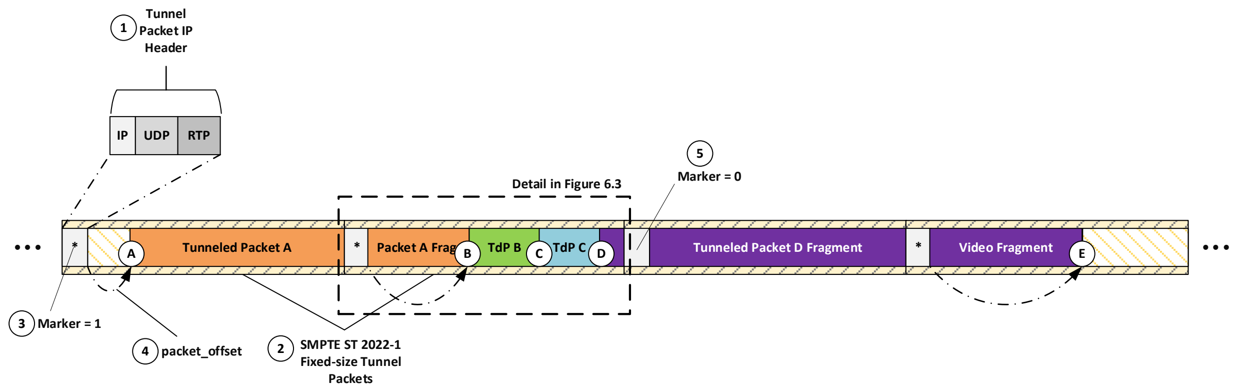


Figure 6.2 Common Tunneled Packet Stream

The following paragraphs describe each of the callouts in Figure 6.2. Note that this is an example of a potential implementation to illustrate how the fixed-size Tunnel Packets are constructed according to this specification. Actual implementations could differ substantially; the resulting packet Stream, however, would have the same semantic organization.

1) A standard RTP/UDP/IP header precedes each of the CTP fixed-size Tunnel Packets that contain the content to be protected by the ST 2022-1 ECC process. All Tunnel Packet headers are fixed size even if they include an extension header as required by ST 2022-1.
2) The example in Figure 6.2 is presented from the decoding standpoint, showing how the Tunnel Packets would be perceived at the consumer of the Tunnel Stream, assuming that the consumer started with an arbitrary packet in a continuous Stream of Tunnel Packets. Each Tunnel Packet in the Stream (2) is of a fixed size and contains a header and payload that may or may not contain the start of a Tunneled Packet.
3) If the **marker (M)** bit (3) of a Tunnel Packet is set to '1', signified in the diagram by an asterisk in the RTP header box, then **packet_offset** (4) is used to indicate the byte where the first Tunneled Packet starts within the payload of the Tunnel Packet.
4) The **packet_offset** is the number of bytes after the RTP header and possible extension where the first byte of the first Tunneled Packet resides within the Tunnel Packet payload (A). This field is used only if the **marker (M)** bit is set to '1'. Note that when more than one packet starts in the Tunnel Packet payload, only the first is indicated by the **packet_offset.** This is shown in Figure 6.2 at Tunneled Packets (B) and (C). The start of Tunneled Packet (B) is indicated by the **packet_offset** as being immediately after the end of the remainder of Tunneled Packet (A). The start of Tunneled Packet (C) can be determined using the length of Tunneled Packet (B) found within Tunneled Packet (B). Note that every protocol derived from the CTP has a length field defined somewhere in each Tunneled Packet header. Of

course, the start of Tunneled Packet (B) also could be determined by using the total length of Tunneled Packet (A), which started in the previous ST 2022-1 fixed-size Tunnel Packet. Robust decoders likely would use both the length of the previous packet and the **packet_offset** to verify the encapsulation process and that the payload packets were correctly formed. This mechanism of framing the Tunneled Stream Packets allows the consumer to recover lost connections quickly since only one or two fixed-size Tunnel Packets are required to begin extracting the Tunneled Packets.

5) The Tunneled Packet, the start of which is labeled (D) in the figure, shows a large packet spanning three ST 2022-1 fixed-size Tunnel Packets. In this case, the middle ST 2022-1 Tunnel Packet does not have the **marker (M)** bit set to '1' and the **packet_offset** field is set to '0' and is ignored.

Figure 6.3 shows the details of the encapsulated Tunnel Packet outlined by the dashed line in Figure 6.2. In the top row of Figure 6.3 are packets from three separate Tunneled Packet Streams. In the center row of Figure 6.3 are the details of the construction of the Tunnel Packet that contains the Tunneled Packets or segments of Tunneled Packets from the top row. As shown by the bottom row, the middle row is a more detailed version of the Tunnel Packet structure shown within the dashed line in Figure 6.2.

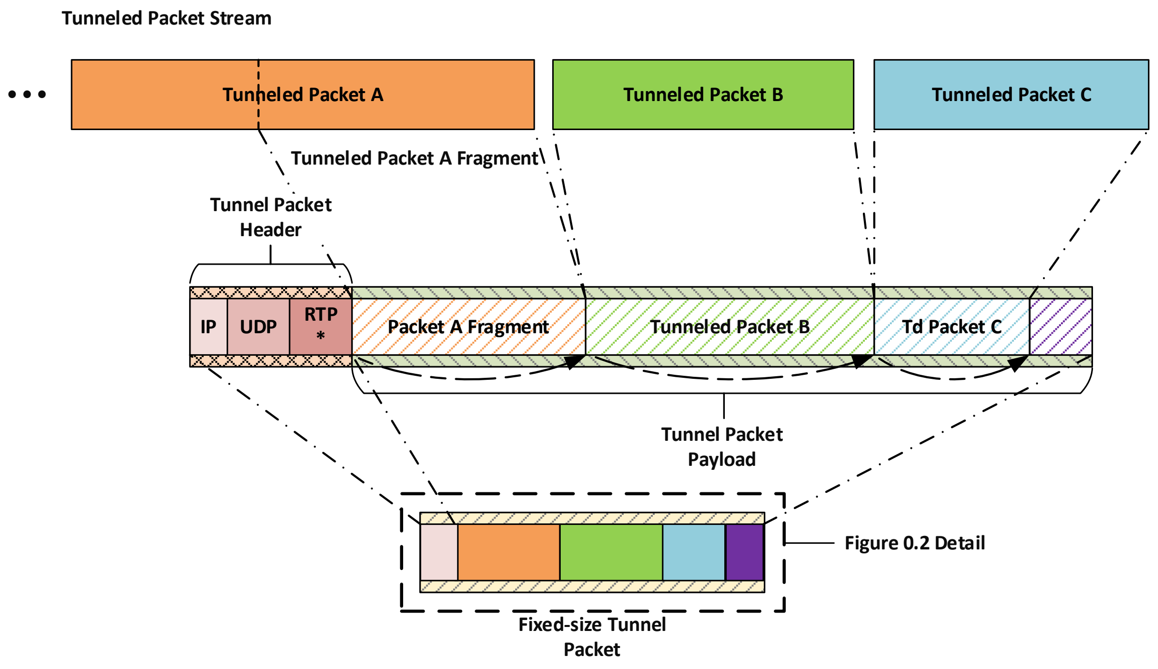


Figure 6.3 Data Source Tunneled Packet to Tunnel Packet Relationship

Starting from the top row of Figure 6.3, each of the three Tunneled Packet Streams has its own procession of packets, the details of which are defined by the particular protocol type. Either complete packets or segments of packets from the Tunneled Packet Streams are multiplexed together on the second row to form the payload of a Tunnel Packet, and another header is prepended to the payload to form a new RTP/UDP/IP packet. As shown in the bottom row, that packet is the one in dashed lines in Figure 6.2.

Looking at the middle row of Figure 6.3 in more detail, there is one RTP/UDP/IP header for the Tunnel Packet, two complete Tunneled Packets and two Tunneled Packet fragments. The first fragment, on the left, is the remainder of Tunneled Packet (A) left over from the previous Tunnel Packet. The second and third blocks represent complete Tunneled Packets (B) and (C) that are fully encapsulated within the Tunnel Packet payload. Finally, the beginning of another Tunneled Packet starts immediately after Tunneled Packet (C) and does not complete so must be continued in the next Tunnel Packet (not shown).

In the sequence of payload packets and segments described within the payload of the Tunnel Packet, there is a pointer from the Tunnel Packet RTP header to the start of the first bit of the Tunneled Packet (B). The Tunnel Packet RTP header **packet_offset** points there and the **marker (M)** bit is set to '1'. Since the RTP header pointer did not point to the first bit following itself, the implication is that the first data segment within the Tunnel Packet payload is a segment of a Tunneled Packet that was not completed in an earlier Tunnel Packet. When examining the Tunnel Packet RTP header, the consumer can determine both that the first data within the Tunnel Packet payload is a segment of a previously incomplete packet and where the next Tunneled Packet begins within the Tunnel Packet. By examining the length value contained within the next Tunneled Packet, the consumer can locate the next header within the Tunnel Packet payload. This process can continue until the consumer finds a Tunneled Packet header that indicates a length longer than the remaining space available in the Tunnel Packet payload, which implies that there will be at least a segment of the final Tunneled Packet contained within the next Tunnel Packet having the same port and address as the current Tunnel Packet.

6.3.2 Example SMPTE ST 2022-1 ECC Encoding Process

Figure 6.4 provides a detailed example diagram of how the various Tunneled Packet Streams are encoded into CTP Tunnel Packets with a two-dimensional arrangement of ST 2022-1 ECC packets [8]. This exemplifies some of the steps shown in Figure 6.1. The process shown in Figure 6.4 implements steps 3, 4, and 5 of Figure 6.1. This example shows a combination of the CTP Encapsulation function and the ECC Encoder function operating on a shared memory area. A similar decode process occurs where these packets are gathered into memory, ECC is used to reconstitute missing packets, and the Tunneled Packets are extracted directly from the in-memory buffers. Note that SMPTE ST 2022-1 may also be used in a one-dimensional arrangement, where the ECC encoding operation is applied to a single row of packets.

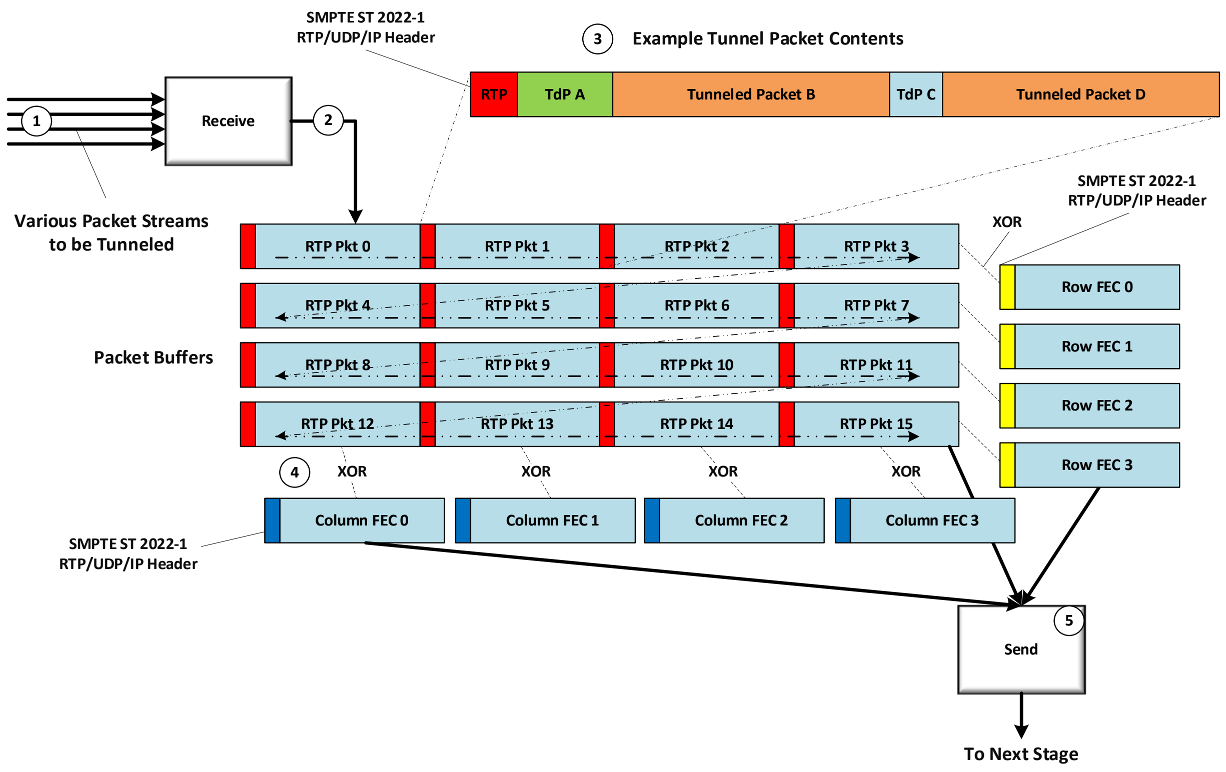


Figure 6.4 Example ECC encoding process diagram

The following paragraphs describe each of the callouts from Figure 6.4.

1) The multiple paths represent the RTP/UDP/IP Tunneled Packet Streams. The content of these Streams depend on which protocol is being processed: DSTP, ALPTP or STLTP.
2) As Tunneled Packets are received from the input interface, they are placed in memory as they are received. A conceptual memory buffer is preconfigured to contain as many Tunnel Packets as required to meet the ECC packet columns ('L') and rows ('D') configuration with the Tunnel Packet payloads and placeholder RTP headers. The appropriate **marker (M)** bit and **payload_offset** fields will be updated in these headers as the Tunneled Packets are copied into the buffers.
3) The "Example Tunneled Packet Contents" diagram a conceptual model of how Tunneled Packets are placed into individual Tunnel Packet buffers. Tunneled Packets are placed into the buffer sequentially as received. If a Tunneled Packet exceeds the length of the current Tunnel Packet buffer, the portion of the Tunneled Packet that fits is placed into the buffer and the remainder is placed into the next buffer. The **payload_offset** field of the next Tunnel Packet buffer RTP header is set to indicate the byte of the next Tunneled Packet starts within the Tunnel Packet payload and the **marker (M)** bit is set to indicate that a payload (i.e., Tunneled Packet) start is present within the particular buffer. An example of this packing process is described in detail in Section 6.3.1 above.
4) When a column or row is completed, the ECC function (XOR per [8]) is performed creating an SMPTE ST 2022-1 ECC packet for the column or row, respectively. The headers and content of these packets are described in [8].
5) The ECC packets along with the Tunnel Packets are sent to the Send function per the guidelines defined in [8]. There are several packet interleaving options defined in [8] that

can be configured based on the number of columns and rows defined and the system latency desired.

6.4 Tunneled Packet Security Protocol

The Tunneled Packet Security Protocol has two primary components: A Signing Function that can authenticate the structural data and content of any and all Tunneled Packet Streams other than the Security Data Stream, and a Key Generation and Distribution function that securely generates and delivers the Keys used by the Signing Function. The Signing Function is based on a Symmetric Key and process called the Galois/Counter Mode (GCM) [19] of the Advanced Encryption Standard (AES) [20]. When GCM is used exclusively for Authentication and not for encryption, as here, the signature data is called Galois/Counter-Mode-based Message Authentication Code (GMAC) [19]. The Key Generation and Distribution function is secured using a Symmetric Key wrapped with an Asymmetric Public/Private Key and processed in a scheme called OpenPGP [22]. OpenPGP is used to deliver both symmetric Keys for the GMAC Authentication of Tunneled Packet Streams and Public Keys to be used in the OpenPGP processes by new Signing Entities added to the network.

The following subsections treat each of the elements required to implement the entire Tunneled Packet Security Protocol. They cover the signing procedures for Tunneled Packets, the generation of Keys for the signing procedures, the delivery of Keys used for the signing procedures, the generation and distribution of Public/Private Key pairs used for securely delivering the Keys for the signing and Key-delivery processes, the protocol of the Security Data Stream used to deliver the several Keys, and implementation considerations for the Security System.

6.4.1 Tunneled Packet Signing Procedures

The individual packets in Tunneled Packet Streams can be signed for purposes of Authentication of the source of the packets. Packets in a Security Data Stream are treated by separate encryption plus Authentication methods described in Sections 6.4.3, 6.4.3.1, and 6.4.3.2. Tunneled Packets to be authenticated shall be signed using the Galois/Counter Mode (GCM) [19] of the Advanced Encryption Standard (AES) [20] (i.e., GMAC). Although the GCM is capable of Authenticated Encryption with Additional Data (AEAD), no encryption and only the Authentication of Additional Authenticated Data (AAD) shall be applied. See Figure 6.5 for a diagram showing an overview of the GCM processes in a Signing Entity and an Authenticating Entity. Keys of length 256 bits shall be applied in the GCM/AES process, and AES processing Blocks of 128 bits shall be used. The GMAC Tag value shall be 128 bits in length.

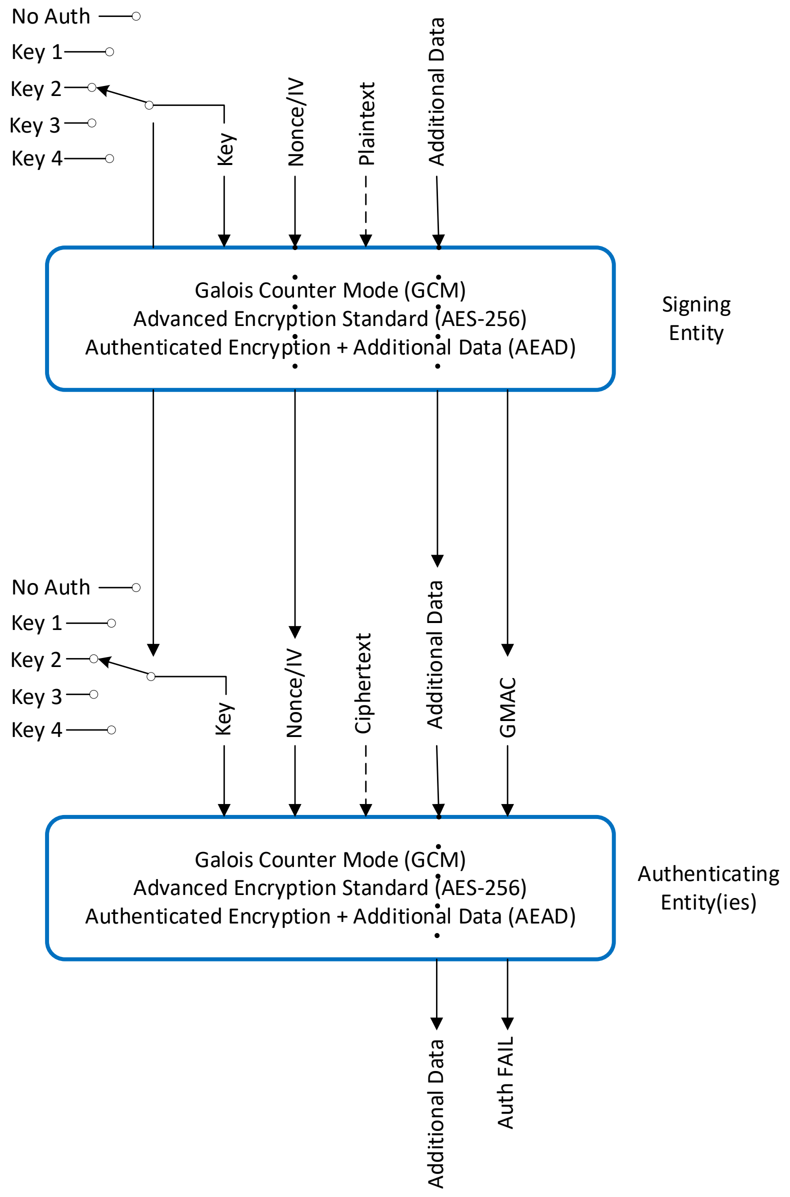

No Auth
Key 1
Key 2
Key 3
Key 4
Key
Nonce/IV
Plaintext
Additional Data
Galois Counter Mode (GCM)
Advanced Encryption Standard (AES-256)
Authenticated Encryption + Additional Data (AEAD)
Signing Entity
No Auth
Key 1
Key 2
Key 3
Key 4
Key
Nonce/IV
Ciphertext
Additional Data
GMAC
Galois Counter Mode (GCM)
Advanced Encryption Standard (AES-256)
Authenticated Encryption + Additional Data (AEAD)
Authenticating Entity(ies)
Additional Data
Auth FAIL

Figure 6.5 GCM AEAD and GMAC Authentication of AAD

The GCM process shall be applied to Tunneled Packets at a Signing Entity and GMAC Tags calculated for all such packets. The GMAC Tags shall be communicated to Authenticating Entity(ies) in RTP Header Extensions along with their respective Tunneled Packets**.** At the Authenticating Entity(ies), the GCM process shall be applied again to the Tunneled Packets and new GMAC Tags calculated for those packets. The GMAC Tags calculated at the Authenticating Entities shall be compared with those communicated from the Signing Entity. Any difference between the two GMAC Tags calculated for a specific packet shall indicate either one or more errors in the delivery of the packet or tampering with the packet.

In addition to the secret GCM Keys that must be known to both a Signing Entity and its associated Authenticating Entity(ies), a separate Initialization Vector (IV) also must be known to both types of units. The IV must be a form of Nonce but is not required to be secret. The most important characteristic of the IV is that it is not repeated in conjunction with any particular GCM Key value. The required conditions with respect to the IV can be met by concatenating the bit patterns of several values carried in the headers of the packets to be authenticated or nearby in the data Stream. The IVs shall be 96 bits in length, formatted as depicted in Table 6.4 and as described immediately thereafter.

Table 6.4 Initialization Vector (IV) Organization

Syntax	No. of Bits	Format
Initialization_Value () {		
if (PT == STLTP) {		
bootstrap_time_msb	10	uimsbf
} else {		
time_msecs	10	Table 6.5
}		
signing_src_addr_lsb	22	uimsbf
packet_dest_port	16	uimsbf
packet_seq_num	16	uimsbf
if (PT == STLTP) {		
timestamp ()	32	Table 9.2
}		
else {		
time_secs	32	Table 6.5
}		
}		

PT refers to the value of **payload_type** as described in the text following Table 6.1.

bootstrap_time_msb shall contain the 10 most significant bits of the Bootstrap Reference Emission Time of the Physical Layer frame with which the authenticated Tunneled Packet will be associated. It applies only to applications of STLTP.

time_msecs shall be the value described in text following Table 6.5. It applies only to applications of DSTP and ALPTP.

signing_src_addr_lsb shall contain the 22 least significant bits of the IP Source Address of the Signing Entity sending the authenticated Tunneled Packet.

packet_dest_port shall contain the UDP port number of the authenticated Tunneled Packet.

packet_seq_num shall contain the RTP sequence number of the authenticated Tunneled Packet.

timestamp () shall contain the 32 bits of the **timestamp ()** structure defined in Table 9.2 and in the paragraphs defining the individual **timestamp ()** fields that follow Table 9.2, as carried in the RTP header of the authenticated Tunneled Packet.

time_secs shall be the value described in text following Table 6.5. It applies only to applications of DSTP and ALPTP.

Four secret GCM Keys shall be stored in Cryptographic Token**s** connected to each Signing Entity and to each Authenticating Entity, as described in Section 6.4.4. The Keys shall be communicated from a Signing Entity to all Authenticating Entities in a network through use of the

Security Data Stream protocol described in Section 6.4.5. The use of a particular one of the four Keys for the signing of a specific Tunneled Packet shall be signaled in a Header Extension of the packet as described in the next paragraph. Similarly, the GMAC Tag resulting from GHASH processing of the packet shall be communicated in the same Header Extension.

To accommodate indication of the Key used for signing of a packet and to support carriage of the GMAC Tag resulting from GHASH processing of the packet, a Header Extension shall be applied to each signed Tunneled Packet, as described in Sections 7.2.1, 8.2.1, 9.2.1, 9.3.1, and 9.3.4. The Header Extension shall be constructed according to the format defined in Table 6.5 and the following paragraphs. Note that for DSTP and ALPTP, the Header Extension is added to the Tunneled Packet Information Header, while for STLTP, the Header Extension is added to each Tunneled Packet RTP header.

Table 6.5 GMAC Header Extension

Syntax	No. of Bits	Format
GMAC_Header_Extension () {		
if (PT == DSTP || PT == ALPTP)		
{		
time_msecs	10	uimsbf
reserved	3	0x7
else		
reserved	13	0x1FFF
key_num	3	uimsbf
length	16	0x0004
GMAC_tag	128	uimsbf
if (PT == DSTP || PT == ALPTP)		
{		
time_secs	32	uimsbf
}		
}		

PT refers to the value of **payload_type** as described in the text following Table 6.1.

time_msecs shall indicate the value of the milliseconds portion of UTC time (or TAI time) at the instant when the associated Information Header of a DSTP or ALPTP payload packet is created. **time_msecs** is not used when the GMAC Header is employed in authentication of STLTP Tunnel Packets. Precision of the **time_msecs** value is immaterial, but it must be a monotonically increasing value to serve its purpose of ensuring that the Initialization Vector for the GMAC authentication process is a Nonce.

key_num shall indicate the index number of the GCM Key used in Authentication of the packet. It shall have a value in the range from 0x0 through 0x4. A value of 0x0 shall indicate that no Authentication processing is applied. Values from 0x1 through 0x4 shall indicate that Authentication processing is applied using the Key having the index number given. Values 0x5 through 0x7 shall be reserved.

length shall indicate the length of the RTP Header Extension following the last bit of the **length** field. Its value shall be the count of the number of 32-bit words included in the Header Extension, excluding the four-byte extension header; thus, zero shall be a valid length. In the instance of packet signing exclusively, **length** shall have a value 0x0004, indicating a length of the Header Extension following the **length** field of 128 bits.

GMAC_tag shall be the 128-bit GMAC Tag calculated for the packet using a combination of the Key value indicated by the **key_num** field, the Initialization Value determined as described above in this section, and the GHASH function applied to the structural elements plus the payload of the packet.

time_secs shall indicate the value of the seconds portion of UTC time (or TAI time) at the instant when the associated Information Header of a DSTP or ALPTP payload packet is created. **time_secs** is not used when the GMAC Header is employed in authentication of STLTP Tunnel Packets. Precision of the **time_secs** value is immaterial, but it must be a monotonically increasing value to serve its purpose of ensuring that the Initialization Vector for the GMAC authentication process is a Nonce.

6.4.1.1 Procedure for Calculating GMAC Tag

The following conceptual steps are applied when calculating the GMAC Tag for an individual Tunneled Packet.

Signing Entity Procedure – Calculate Signature

1) Create a Tunneled Packet including a Header Extension either in the RTP Header portion of the packet (STLTP) or the appropriate Tunneled Packet Information Header.
2) Fill the **GMAC_tag** field with 128 '0' (zero) bits.
3) Calculate the GMAC Tag value by applying the GHASH function to the entirety of the Tunneled Packet extending from the first bit of its IP Header through the last bit of its payload. Encrypt the GMAC Tag with the Authentication Key selected for use with the Tunneled Packet.
4) Substitute the resulting GMAC Tag value into the 128-bit **GMAC_tag** field.
5) Transmit the packet through the CTP Stream.

Authenticating Entity Procedure – Check Signing

6) On reception of the Tunneled Packet, harvest the **GMAC_tag** value from the packet for later comparison.
7) Fill the **GMAC_tag** field with 128 '0' (zero) bits.
8) Calculate the GMAC Tag value by applying the GHASH function to the entirety of the Tunneled Packet extending from the first bit of its IP Header through the last bit of its payload.
9) Decrypt the **GMAC_tag** value harvested from the Tunneled Packet using the Authentication Key indicated for use with the packet. Compare the GMAC Tag value calculated at the Authenticating Entity with the **GMAC_tag** value harvested from the RTP Header Extension upon reception of the packet.
10) Any difference between the two renditions of the **GMAC_tag** value shall indicate either errors in transmission of the Tunneled Packet or tampering with its delivery.

6.4.2 Authentication Key Generation and Utilization

Given the parameters selected for this standard, the GCM/GMAC signing scheme requires as one of its inputs a random and Fresh 256-bit Key. As described in [19], "The Key shall be generated uniformly at random, or close to uniformly at random, i.e., so that each possible Key is (nearly) equally likely to be generated. Consequently, the Key will be Fresh, i.e., unequal to any previous Key, with high probability. The Key shall be secret and shall be used exclusively for GCM with the chosen Block Cipher."

As described in Section 6.4.3, security Keys for Signing purposes generally shall be generated using a random number generator in a Cryptographic Token installed in a Signing Entity. Four such Keys shall be generated initially, and they may be applied in any order, as also described in Section 6.4.6. The strength of the security provided by the Keys used in the GCM/GMAC Authentication scheme has a limited lifetime; therefore, the Authentication Keys shall be replaced on a routine basis at appropriate times that depend on the patterns of their use. The fundamental limitations on Key longevity are the number of times the Keys are invoked and the number of Cipher Blocks that they process. The number of invocations of a Key shall be limited to 2^{32}. The number of Cipher Blocks processed shall be limited to 2^{64}. Given the largest possible size of the data set to which one invocation of a Key could apply, observing the limit on the number of Key invocations effectively would observe the limit on the number of Cipher Blocks processed with that Key.

In lieu of counting the number of invocations, a conservative method for limiting Key utilization is to limit the time period during which a given Key can be active. The time period limit can be derived from a calculation of the maximum number of packets that possibly can be transmitted over the CTP per time unit with all parameters set to maximize the packet rate, followed by division of the number of packets per time unit into 2^{32}, yielding the smallest number of time units (i.e., seconds) before the earliest point at which the security capacity of a Key could be fully expended. The time period obtained by such a calculation then could be adjusted using factors relating actual operating parameters with those parameters used to determine the maximum possible packet rate for the system. For example, for the STLTP design described in this standard, the maximum packet rate is approximately 480,000 packets per second, and the maximum safe continuous utilization time for a Key becomes just under 8,950 seconds or about 2½ hours. That value is based on a parameter set including an 8 MHz channel bandwidth, 64 PLPs, and use of MIMO. Adjustment of the time limit for a system with a parameter set including a 6 MHz channel bandwidth, 4 PLPs, and no use of MIMO would result in a time limit per Key of just under 381,900 seconds or about 106 hours (almost 4½ days). If the actual packet rate, based on the Physical Layer parameters in use, were used in the calculations instead of assumed worst-case values, it is likely that the packet rate, and hence the Authentication Key invocation rate, would be substantially lower and the corresponding time before exhaustion of the Authentication Key security capacity would be proportionally greater. It should be noted that, if Key rotation were used instead of application of a single Key for an extended period, it would be necessary either to track the utilization times of the individual Keys or to apply a further factor to determine Key longevity based on the utilization percentage of each Key.

6.4.3 Authentication Key Delivery

It is a requirement that the Keys used in Signing of all Tunneled Packet Streams other than the Security Data Stream remain secret. Nevertheless, the Keys must be updated from time to time. To facilitate such updates, the Security Data Stream is designed to carry secret data through use of authenticated encryption based on the OpenPGP protocol [22], extended through use of Elliptic Curve Cryptography as described in [23]. OpenPGP uses a combination of symmetric and asymmetric Keys plus standardized hashing algorithms and cryptographic processes. IETF RFC 6637 [23], which describes the addition of Elliptic Curve Cryptography to OpenPGP, applies the latest cryptographic technology and also narrows the choices among many parameters available in OpenPGP. In the next several paragraphs, this standard narrows the choices further to a single interoperable set of processes and parameters.

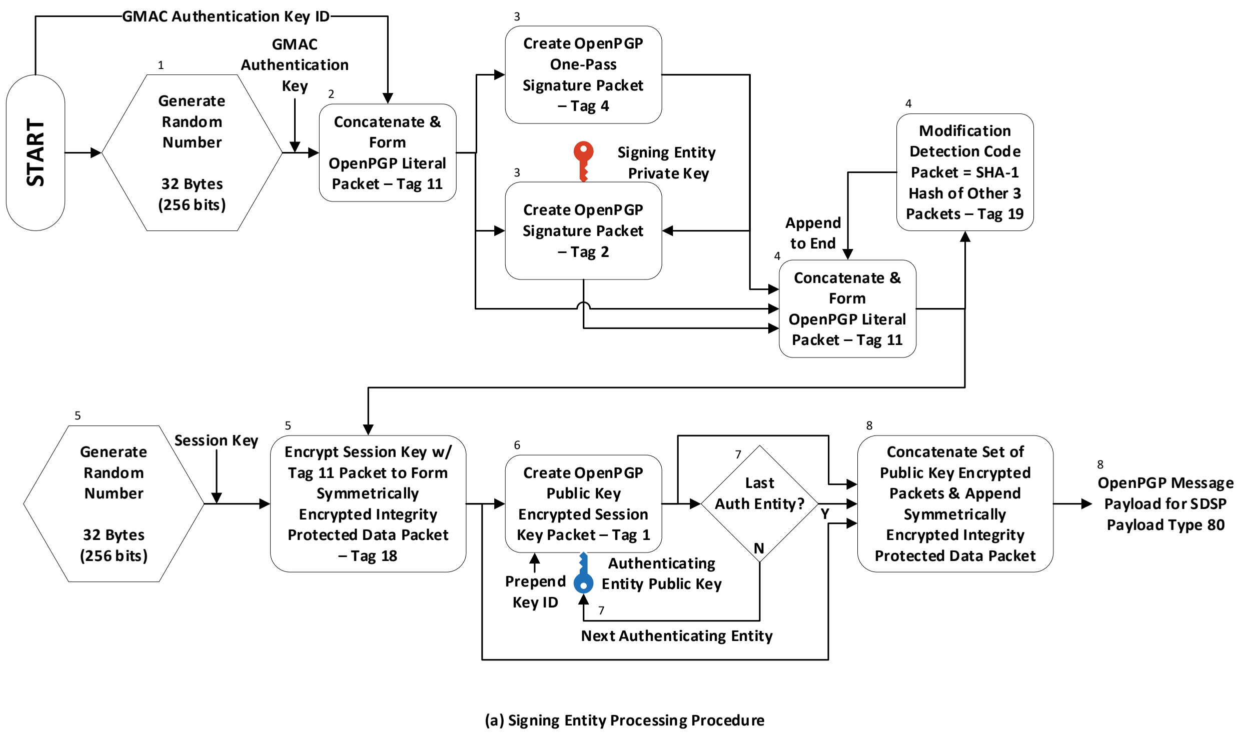
Delivery of Authentication Keys between Signing Entities and from a Signing Entity to one or more Authenticating Entities shall employ OpenPGP [22], extended for use of Elliptic Curve Cryptography [23]. Each GMAC Authentication Key shall be a 256-bit random number, which shall be carried as the payload of an OpenPGP Session. Each Session Key, as specified in [23], shall be derived from a random number and shall use the prime field curve defined in [24] as "Curve P-384". The Public Key Algorithm used shall be Elliptic Curve Diffie-Hellman (ECDH), and the Key Derivation Function (KDF) for ECDH shall be that described in [23]. The associated Hash function shall be SHA2-384, and the Key wrapping method shall be that specified in [25] with AES-256 as the Key-Encryption Key algorithm (KEK).

The Key delivery process makes use of the relationships between Public and Private Keys in a pair for purposes of Authentication and Encryption/Decryption. The relationships are:

- Data Encrypted with a Public Key can be Decrypted with a Private Key
- Message Authentication Codes (MACs) signed with a Private Key can be verified with a Public Key

The delivery method shall proceed according to the following steps, as diagrammed in Figure 6.6. The procedure for Signing Entity Processing is shown in Figure 6.6(a), and the procedure for Authenticating Entity Processing is shown in Figure 6.6(b). In the figures, Keys shown in red are Private Keys; Keys shown in other colors are Public Keys. Keys are labeled as to their types. Small numbers alongside the various blocks and text items tie them to the steps given in Sections 6.4.3.1 for Figure 6.6(a) and 6.4.3.2 for Figure 6.6(b).

In the Security Data Stream, packets carrying GMAC Authentication Keys, as described in this subsection, shall be identified by Payload Type value 80 (decimal), equivalent to 0x50 (hex).


START
GMAC Authentication Key ID
1
Generate Random Number
32 Bytes (256 bits)
GMAC Authentication Key
2
Concatenate & Form OpenPGP Literal Packet – Tag 11
3
Create OpenPGP One-Pass Signature Packet – Tag 4
Signing Entity Private Key
3
Create OpenPGP Signature Packet – Tag 2
4
Modification Detection Code Packet = SHA-1 Hash of Other 3 Packets – Tag 19
Append to End
4
Concatenate & Form OpenPGP Literal Packet – Tag 11
5
Generate Random Number
32 Bytes (256 bits)
Session Key
5
Encrypt Session Key w/ Tag 11 Packet to Form Symmetrically Encrypted Integrity Protected Data Packet – Tag 18
6
Create OpenPGP Public Key Encrypted Session Key Packet – Tag 1
Prepend Key ID
Authenticating Entity Public Key
7
Last Auth Entity?
Y
N
7
Next Authenticating Entity
8
Concatenate Set of Public Key Encrypted Packets & Append Symmetrically Encrypted Integrity Protected Data Packet
8
OpenPGP Message Payload for SDSP Payload Type 80
(a) Signing Entity Processing Procedure

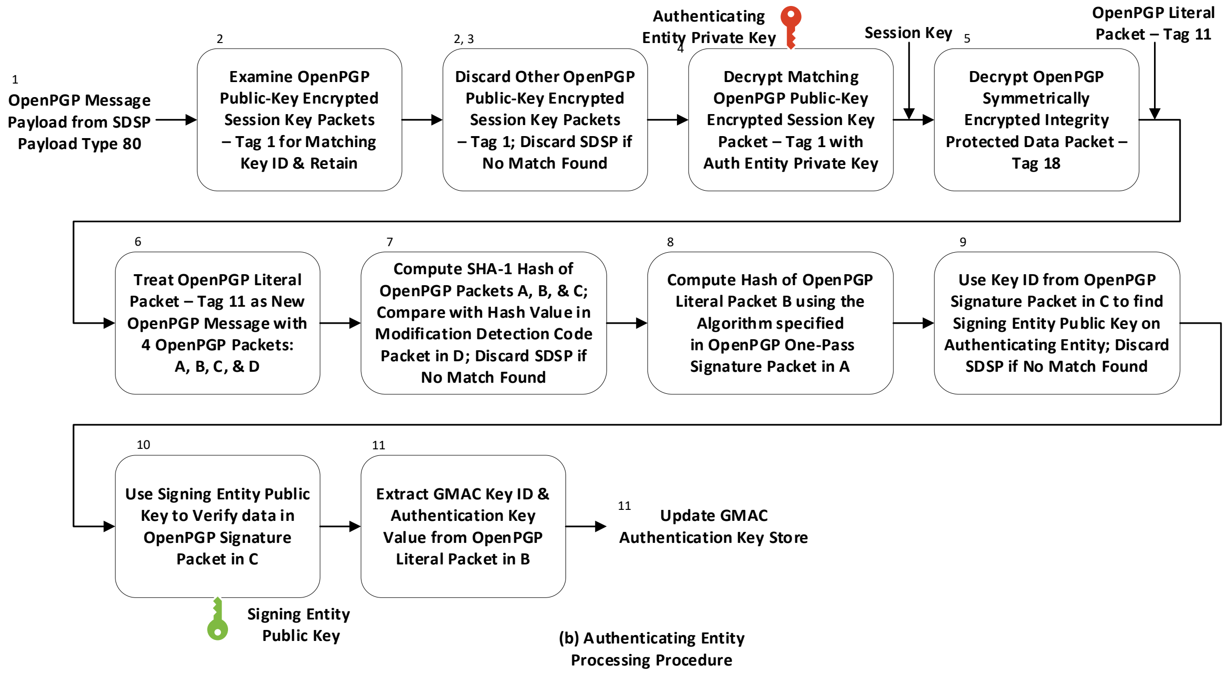

1
OpenPGP Message Payload from SDSP Payload Type 80
2
Examine OpenPGP Public-Key Encrypted Session Key Packets – Tag 1 for Matching Key ID & Retain
2, 3
Discard Other OpenPGP Public-Key Encrypted Session Key Packets – Tag 1; Discard SDSP if No Match Found
Authenticating Entity Private Key
4
Decrypt Matching OpenPGP Public-Key Encrypted Session Key Packet – Tag 1 with Auth Entity Private Key
Session Key
5
Decrypt OpenPGP Symmetrically Encrypted Integrity Protected Data Packet – Tag 18
OpenPGP Literal Packet – Tag 11
6
Treat OpenPGP Literal Packet – Tag 11 as New OpenPGP Message with 4 OpenPGP Packets: A, B, C, & D
7
Compute SHA-1 Hash of OpenPGP Packets A, B, & C; Compare with Hash Value in Modification Detection Code Packet in D; Discard SDSP if No Match Found
8
Compute Hash of OpenPGP Literal Packet B using the Algorithm specified in OpenPGP One-Pass Signature Packet in A
9
Use Key ID from OpenPGP Signature Packet in C to find Signing Entity Public Key on Authenticating Entity; Discard SDSP if No Match Found
10
Use Signing Entity Public Key to Verify data in OpenPGP Signature Packet in C
Signing Entity Public Key
11
Extract GMAC Key ID & Authentication Key Value from OpenPGP Literal Packet in B
11
Update GMAC Authentication Key Store
(b) Authenticating Entity Processing Procedure

Figure 6.6 Tunneled Packet Security System Authentication Key generation and distribution: (*a*, top) Signing Entity Processing Procedure; (*b*, bottom) Authenticating Entity Processing Procedure

6.4.3.1 Signing Entity Processing Procedure

1) Generate a random number to be used as a Symmetric GMAC Authentication Key and associate it with one of the four Key Identifiers used for GMAC Authentication Keys.
2) Create an OpenPGP Literal Packet (RFC 4880 [22] Tag 11) containing the concatenation of the GMAC Authentication Key Identifier and the GMAC Authentication Key from step (1).
3) Select the Signing Entity Key Pair with which to sign the OpenPGP Literal Packet from step (2). Create an OpenPGP One-Pass Signature Packet (RFC 4880 [22] Tag 4). Also create an OpenPGP Signature Packet (RFC 4880 [22] Tag 2) containing the concatenation of information prescribed for the Version 4 Signature Packet Format in RFC 4880 [22], which includes a signature built from a hash of the OpenPGP Literal Packet from step (2) plus some of the previous fields of the OpenPGP Signature Packet as specified in RFC 4880 [22].
4) Create a second OpenPGP Literal Packet (RFC 4880 [22] Tag 11) containing the concatenation of the following four elements in the order given: (i) the One-Pass Signature Packet, (ii) the OpenPGP Literal Packet from step (2), (iii) the Signature Packet from step (3), and (iv) a Modification Detection Code Packet (RFC 4880 [22] Tag 19), containing the SHA-1 hash of the preceding elements (i) – (iii) in the concatenation.
5) Generate a second random number to be used as the Session Key and encrypt it with the second OpenPGP Literal Packet from step (4) to form a Symmetrically Encrypted Integrity Protected Data Packet (RFC 4880 [22] Tag 18).

   Note: Per their description in RFC 4880 [22], Tag 18 packets require use of Cipher Feedback (CFB) mode encryption.
6) Select a target Authenticating Entity Public Key and create an OpenPGP Public-Key Encrypted Session Key Packet (RFC 4880 [22] Tag 1) containing the encrypted Session Key generated in step (5), prepended with the Key ID of the selected Authenticating Entity Public Key. The Key ID shall be the final 8 bytes of the Fingerprint of the Authenticating Entity Public Key, which Fingerprint shall be constructed according to RFC 4880 [22] instructions for Fingerprints for V4 Keys. The timestamp of Key creation shall be the 32 bits representing the seconds count of NTP Time per RFC 5905 [13] at the time of creation of the Public Key. The Algorithm-Specific Fields shall be those specified for ECDH Keys in RFC 6637 [23].
7) Repeat step (6) for each target Authenticating Entity.
8) Concatenate the set of Public-Key Encrypted Session Key Packets generated in steps (6) and (7) (in any order) and then append the Symmetrically Encrypted Integrity Protected Data Packet created in step (5). This produces an OpenPGP Message Payload that will be distributed to the Authenticating Entities in an A/324 Security Data Stream Packet with Payload Type 80 (0x50).

6.4.3.2 Authenticating Entity Processing Procedure

1) On receipt of a Security Data Stream Packet with Payload Type 80 (0x50), extract the OpenPGP Message Payload and process it as follows.
2) Locate each of the OpenPGP Public-Key Encrypted Session Key Packets (RFC 4880 [22] Tag 1) at the beginning of the OpenPGP message and locate a packet that contains an Authenticating Entity Key Identifier that matches a Private Key installed on the Authenticating Entity. Discard any OpenPGP Public Key Encrypted Session Key Packets

other than the first one that contains an Authenticating Entity Key Identifier matching a Private Key installed on the recipient Authenticating Entity.

3) If no matching Authenticating Entity Key Identifier is found, discard the Security Data Stream Packet.
4) Decrypt the OpenPGP Public-Key Encrypted Session Key Packet (RFC 4880 [22] Tag 1) matching the Authenticating Entity Key Identifier using the corresponding Authenticating Entity Public Key to access the Session Key.
5) Locate the OpenPGP Symmetrically Encrypted Integrity Protected Data Packet (RFC 4880 [22] Tag 18) and decrypt the contents of the packet with the Session Key to obtain an OpenPGP Literal Packet (RFC 4880 [22] Tag 11).

   Note: Per their description in RFC 4880 [22], Tag 18 packets require use of Cipher Feedback (CFB) mode encryption and decryption.
6) Treat the contents of the OpenPGP Literal Packet from step (5) as a new OpenPGP Message that contains the following four OpenPGP packets in order:

A. An OpenPGP One-Pass Signature Packet
B. An OpenPGP Literal Data Packet, and
C. An OpenPGP Signature Packet
D. An OpenPGP Modification Detection Code Packet

7) Verify the contents of the OpenPGP Modification Detection Code packet (6)(D) against the SHA-1 hash of the three packets in (6)(A), (6)(B), and (6)(C). If the integrity protection checks fail, discard the Security Data Stream Packet.
8) Calculate the hash of the OpenPGP Literal Packet in (6)(B) using the algorithms specified in the OpenPGP One-Pass Signature Packet in (6)(A).
9) Use the Key Identifier from the OpenPGP Signature Packet in (6)(C) to locate the Signing Entity Public Key stored on the Authenticating Entity. If the Key Identifier does not correspond with a Signing Entity Public Key known to the Authenticating Entity, discard the Security Data Stream Packet.
10) Using the Signing Entity Public Key from step (9), verify the signature data contained in the OpenPGP Signature Packet (6)(C) against the hash value calculated in step (8). If the signature does not verify, discard the Security Data Stream Packet.
11) Extract the GMAC Key Identifier and the GMAC Authentication Key from the OpenPGP Literal Packet obtained in (6)(B) and update the Authenticating Entity GMAC Authentication Key Store.

6.4.4 Public/Private Key Generation and Distribution

To enable the interchange of secured secrets such as Public and Private Key values, Authentication Keys, and the like, it is necessary for a strong set of Keys to be distributed among Signing Entities and Authenticating Entities in such a way that all needed communication channels can be protected. The matter is complicated by the fact that Signing Entities and Authenticating Entities can be widely separated geographically, and there can be many of them in a network. It also is necessary to support redundant operation of equipment, to allow for growth of networks, and to provide for spare parts and repairs of equipment and components. To meet these needs, resilient security elements must be deployed in semi-permanent installations, and appropriately secure processes deployed to maintain the privacy of the secrets that secure communications between network entities.

All of the foregoing considerations are met in the Tunneled Packet Security System design through use of a combination of long-lasting, asymmetric, Public/Private Key Infrastructure with symmetric Key processes that use either replaceable Keys that can be periodically updated or ephemeral Keys that are used only once. The foundation for the system, though, is the Public/Private Key scheme that is installed in all units on the network. To make that approach work, each unit must have its own Private Key that is well-protected, remains secure, and never is exposed to other entities. As described in Section 6.4.6, Cryptographic Tokens with USB interfaces are employed to process and store the various secrets. As a network is installed, the several Keys needed must be initialized on the Tokens and then installed in the equipment.

To preserve the secrecy of Private Keys, in particular, it is imperative that they be generated on the devices where they will be stored and never exposed to communications between devices or systems where they possibly could be intercepted. Consequently, Public/Private Key pairs shall be generated on Cryptographic Tokens, and the Private Keys shall be retained there, without access from the outside of the devices on which they are generated. Public Keys shall be shared between security devices on the network, but they shall not be distributed to devices outside the network except to back up their values to facilitate generation of replacement or expansion Keys at a later time. As shown in Figure 6.7, each Cryptographic Token shall hold its own Private Key (shown in red) and also its Public Key, shown in different colors for the different applications of the Tokens. For sake of example, three Signing Entities and five Authenticating Entities are shown in Figure 6.7; the number of each type can be a value of one or greater. Upon initialization, each Token intended for use in a Signing Entity, in addition to its own Key pair, shall hold copies of the Public Keys of each of the other Signing Entities and of each of the Authenticating Entities in the network. Each Token intended for use in an Authenticating Entity, in addition to its own Key pair, shall hold copies of the Public Keys of each of the Signing Entities in the network.

In addition to the Public and Private Keys stored on a Token, each one shall be given a number unique within the scope of the network, so that it can be addressed directly for delivery of its unique Session Key encrypted with its Public Key by a Token in a Signing Entity. The number associated with each Token and the Public Key value of the Token having that number shall be stored securely in the Signing Entity(ies) to permit it/them to properly prepare security data for distribution to the other entities on the network, including both Authenticating Entities and other Signing Entities.

When preparing Cryptographic Tokens for a network, it is advisable to populate additional Tokens beyond those necessary just to equip the units that will be installed from the start. Doing so can provide both backup units and units to enable expansion of equipment on the network. When redundant equipment is installed in a network, the same considerations shall apply as to single units, i.e., Tokens for additional Signing Entities shall hold their own Public/Private Key pairs plus copies of the Public Keys of all other Tokens for both Signing Entity and Authenticating Entity use, and Tokens for Authenticating Entity use shall hold their own Public/Private Key pairs plus copies of the Public Keys of all Tokens for Signing Entity use. When redundant Authenticating Entities are used in a Transmitter, each shall be treated as if it were a standalone unit and shall have installed a Token that is identical in the types of Keys it contains to all the other Tokens intended for Authenticating Entity use. Since the data to be delivered to all Authenticating Entities through the Security Data Stream is identical, any Token set up for Authenticating Entity use can be installed in any Authenticating Entity; the only requirement is that its number and Public Key be included in the database retained in the Signing Entity(ies). Similarly, any Token set up for use in a Signing Entity should be usable in any Signing Entity on the same network, so long as the Signing Entity also contains a copy of the database of unit numbers assigned to Tokens and copies of the Public Keys associated with those numbers.

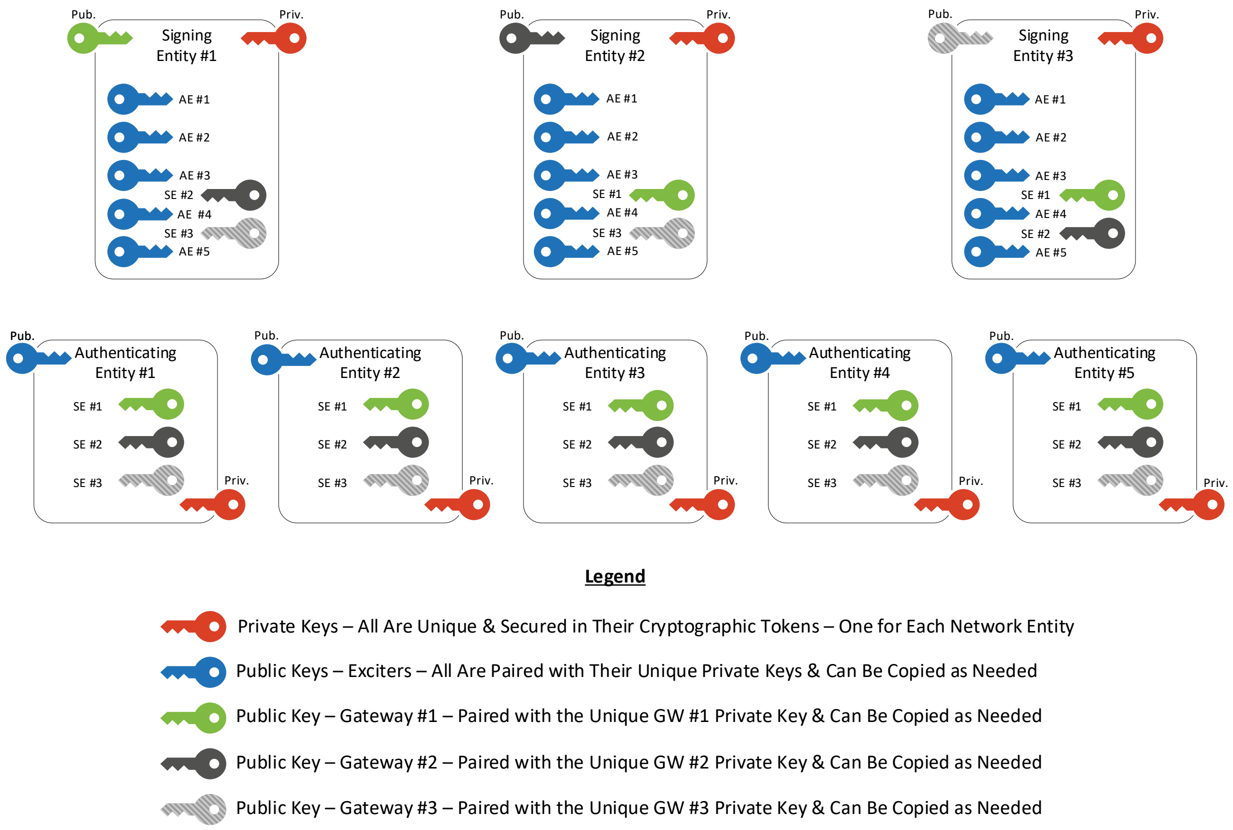

Pub.
Priv.
Signing Entity #1
AE #1
AE #2
AE #3
SE #2
AE #4
SE #3
AE #5
Signing Entity #2
SE #1
Signing Entity #3
Authenticating Entity #1
Authenticating Entity #2
Authenticating Entity #3
Authenticating Entity #4
Authenticating Entity #5
SE #1
SE #2
SE #3
Legend
Private Keys – All Are Unique & Secured in Their Cryptographic Tokens – One for Each Network Entity
Public Keys – Exciters – All Are Paired with Their Unique Private Keys & Can Be Copied as Needed
Public Key – Gateway #1 – Paired with the Unique GW #1 Private Key & Can Be Copied as Needed
Public Key – Gateway #2 – Paired with the Unique GW #2 Private Key & Can Be Copied as Needed
Public Key – Gateway #3 – Paired with the Unique GW #3 Private Key & Can Be Copied as Needed

Figure 6.7 Tunneled Packet Security System Public/Private Key locations after system initialization

Despite possibly having provided spare Tokens for Signing Entities and/or Authenticating Entities, at some point in the life of a network, it may become necessary to prepare additional Cryptographic Tokens and add them to the system. There are two cases that must be considered: addition of Tokens for use in Signing Entities and addition of Tokens for use in Authenticating Entities. As described in Section 6.4.1, it is expected that Token initialization will be carried out using functionality of a Signing Entity or supplemental software. Given such an arrangement, a new Token for Authenticating Entity use can be loaded with the Public Keys of the Signing Entity(ies), and, conversely, the Public Key of the new Token can be captured and distributed among Signing Entities prior to the new Token being put into service. When a new Token is for use in a Signing Entity, however, it would be inconvenient, perhaps impossible, to bring all the Tokens from perhaps many Authenticating Entities to the Signing Entity to load the new Signing Entity Public Key onto them. For this purpose, provision is made to transfer Public Keys from new Tokens initialized for Signing Entity use to all Authenticating Entities on the network in the same way that GMAC Authentication Keys are distributed.

In the Security Data Stream, packets carrying Signing Entity Public Keys, as described in this subsection, shall be identified by Payload Type value 81 (decimal), equivalent to 0x51 (hex).

6.4.5 Security Data Stream Protocol

The Security Data Stream shall be delivered in an RTP/UDP/IP multicast Stream conforming to RFC 3550 [6] with the constraints defined below. The payloads described in the previous sections can exceed typical MTU sizes, so a mechanism is defined to allow segmentation of the Security Data Stream across multiple RTP/UDP/IP packets. Note that such segmentation is required only to conform with typical MTU sizes. If a local network allows larger multicast packets, such segmentation may not be needed. All packets carrying segments of a single payload shall be sent before starting to send a new payload. In other words, packets from a segmented Security Data payload shall not be interleaved with other packets comprising the Security Data Stream.

The resultant Stream of Security Data packets shall have destination address 239.0.51.48 and destination port 30066.

The RTP header fields of a segmented set of Security Data Packets shall be configured as described below, with the **marker (M)** bit of the packet containing the beginning of a Security Data payload set to '1'. The **marker (M)** bits of remaining packets of the set shall be set to '0'. This allows the Transmission system on the Data Consumer end of the CTP to reconstruct the Security Data payload after any resequencing takes place.

On reception, a Security Data Packet Set shall be re-sequenced and extracted into an SDPS Encrypted Data payload as described in Section 6.4.5.1. This payload shall be processed as described in Section 6.4.5.3 to produce either an Authentication Key payload as described in Section 6.4.5.4 or a Public Key Distribution payload as described in Section 6.4.5.5. A Data Consumer can accumulate RTP packets until it has received a syntactically complete data structure. If a packet is missed, as determined by a missing sequence number or premature reception of a packet with the **marker (M)** bit set to '1', indicating the start of the next Security Data Packet Set, then one or more packets have been lost, and the entire Security Data set has been lost. Any accumulated Security Data shall be discarded. Note that, because packets from different payloads within a Security Data Stream cannot be interleaved, a single segmented payload will appear in consecutive Tunneled Packets.

Security Data Stream RTP header fields shall follow the syntax defined in RFC 3550 [6], with the following additional constraints:

The **Padding (P)** bit shall conform to the RFC 3550 [6] specification.

The **Extension (X)** bit shall be set to '0' to indicate that the header contains no extension. The Security Data Stream is secured using the OpenPGP method [22], described in previous sections, instead of the GMAC Tag signing technique [19] applied to all other types of Tunneled Packets, as described in Section 6.4.1.

The **CSRC Count (CC)** shall be set to '0', as no CSRC fields are necessary.

The **marker (M)** bit shall be set to '1' to indicate that the first byte of the payload is the start of a Security Data Packet Set. A **marker (M)** bit value of '0' shall indicate that the payload is a continuation of the Security Data Packet Set from the previous packet.

The **Payload Type (PT)** shall be set either to 79 (0x4f), indicating the Authentication Key payload type or to 80 (0x50), indicating a Signing Entity Public Key payload type.

The **Sequence Number** shall conform to the RFC 3550 [6] specification.

The **Timestamp** shall be defined as in Table 9.2. The timestamp shall be set to the same value for all packets within a Security Data Packet Set.

The **Synchronization Source (SSRC) Identifier** shall be set to '0'. There shall be no other sources of Security Data carried within an STLTP Stream. Any redundant sources can be managed using IGMP Source-Specific Multicast (SSM) mechanisms.

Table 6.6 Security Data Stream Packet Payload

Syntax	No. of Bits	Format
Security_Data_Stream_Packet (SDS) () {		
Structure_Data () {		
length	16	uimsbf
version_major	4	uimsbf
version_minor	4	uimsbf
num_tokens_minus_1	8	uimsbf
for (i=0; i<length–4; i++)		
SDPS_Encrypted_Data_payload_byte	8	bslbf
}		
}		
}		

length shall indicate the number of bytes in the Security Data Stream Packet following the RTP/UDP/IP header structure. Up to 65,535 Bytes can be indicated.

version_major, in conjunction with **version_minor**, shall indicate the version of the protocol used to construct the Security Data Stream Packet. Increments in the value of **version_major** are intended to indicate changes in the structure that are not fully compatible with lower-ordered **version_major** values. The value of **version_major** can range from 0 through 15. Security Data Stream Packets constructed according to this version of this standard shall have the value of **version_major** set to 0.

version_minor, in conjunction with **version_major**, shall indicate the version of the protocol used to construct the Security Data Stream Packet. Increments in the value of **version_minor** are intended to indicate changes in the structure that are fully backward compatible with lower-ordered **version_minor** values having the same **version_major** value. Security Data Stream Packets constructed according to this version of this standard shall have the value of **version_minor** set to 0.

num_tokens_minus_1 shall indicate the number of Cryptographic Tokens for which encrypted Session Keys are included in the Security Data Stream Packet. Its value shall range from 0 through 255 to represent Cryptographic Token counts from 1 through 256, respectively.

SDS_Encrypted_Data_payload_byte shall contain the sequence of bytes that form an OpenPGP message containing the OpenPGP packets described in Sections 6.4.5.1 through 6.4.5.5.

6.4.5.1 SDPS Encrypted Data Processing

The SDS Encrypted Data payload shall contain an OpenPGP message with the following components, in the order specified:

(1) One OpenPGP Public Key Encrypted Packet for each of the Tokens. The number of these packets shall match **num_tokens_minus_1** in the Security Data Stream Packet plus 1. The relative order of these packets is unspecified.

The content of each of these packets shall be as follows:

Field Name	Size	Value	Description
Packet Header	one to five bytes	Packet Tag = 1	The format of the Packet Header shall be as described in RFC 4880 [22] Section 4.2 for New Format Headers
Version	one byte	3	
Key Identifier	eight bytes		The identifier of the Public Key used to encrypt this packet
Encryption Algorithm	one byte	18	Elliptic Curve Diffie-Hellman
Encrypted Data	remainder of packet length		The ECDH algorithm-specific fields described in RFC 6637 [23] Section 10 for Packet Tag 1. The last of these algorithm-specific fields shall contain the encrypted Session Key. See Section 6.4.5.2 for details on encapsulation of the Session Key.

(2) One OpenPGP Symmetrically Encrypted Integrity Protected Data Packet. The content of this packet is as follows:

Field Name	Size	Value	Description
Packet Header	one to five bytes	Packet Tag = 18	The format of the Packet Header is described in RFC 4880 [22] Section 4.2 for New Format Headers
Version	one byte	1	
Encrypted Data	remainder of packet length		

The recipient of the SDS Encrypted Data Payload shall perform the following steps:

1) Locate each of the OpenPGP Public Key Encrypted packets and select the first packet that contains a Key Identifier that matches a Private Key installed on the Authenticating Entity. Discard any OpenPGP Public Key Encrypted packets other than the first one that contains a Key Identifier for the recipient Authenticating Entity.
2) If no matching Key Identifier is found, discard the Security Data Stream Packet.
3) Decrypt the OpenPGP Public Key Encrypted packet matching this Authenticating Entity Key Identifier with the corresponding Authenticating Entity Private Key to access the Session Key.
4) Locate the OpenPGP Symmetrically Encrypted Integrity Protected Data Packet and decrypt the contents of this packet with the Session Key to obtain an OpenPGP Literal Packet which contains an OpenPGP message with the SDS Authenticated Key.

The SDS Authenticated Key shall be processed as described in Section 6.4.5.3.

6.4.5.2 Session Key Encapsulation

The Session Key shall be a randomly generated 256-bit (32-byte) value that is encapsulated in the manner specified in RFC 6637 [23] Section 8 to form a 40-byte value that is input to the Key wrapping method. The content of the encapsulated Session Key shall be as follows:

Field Name	Size	Value	Description
Algorithm Identifier	one byte	AES-256= 09	The AES algorithm used for the Session Key
Session Key	32 bytes		A randomly generated 256-bit number
Checksum	2 bytes		Sum of the 32 Session Key bytes modulo 65536. (Note: The Algorithm Identifier byte is not included in this calculation.)
Padding	5 bytes	each byte = 05	

6.4.5.3 SDS Authenticated Key Processing

The SDS Authenticated Key is an OpenPGP formatted message with the following four components:

(1) An OpenPGP One-Pass Signature Packet
The content of this packet shall be as follows:

Field Name	Size	Value	Description
Packet Header	one to five bytes	Packet Tag = 4	The format of the Packet Header shall be as described in RFC 4880 [22] Section 4.2 for New Format Headers
Version	one byte	3	
Signature Type	one byte	0	Signature of a binary document
Hash Algorithm	one byte	9	SHA-384
Public Key Algorithm	one byte	19	ECDSA
Key Identifier	eight bytes		The identifier of the Signing Entity Public Key used to authenticate this packet
Flag	one byte	1	Indicator that this is the last (only) One-Pass Signature Packet

(2) An OpenPGP Literal Packet
This packet may contain either an Authentication Key Payload (in the case that the Security Data Stream RTP header is type 80 (0x50)) as described in Section 6.4.5.4 below or a Public Key Distribution Payload (in the case that the Security Data Stream RTP header is type 81 (0x51)) as described in Section 6.4.5.5 below.

(3) An OpenPGP Signature Packet
The content of this packet shall be as follows:

Field Name	Size	Value	Description
Packet Header	one to five bytes	Packet Tag = 2	The format of the Packet Header is described in RFC 4880 [22] Section 4.2 for New Format Headers
Version	one byte	4	
Signature Type	one byte	0	Signature of a binary document
Public Key Algorithm	one byte	19	ECDSA
Hash Algorithm	one byte	9	SHA-384
Hashed Data Length	two bytes	16	
Creation Time Subpacket Length	one byte	5	

Creation Time Subpacket Type	one byte	2	
Signature Creation Time	four bytes		Number of seconds since midnight 1 January 1970 UTC.
Issuer Subpacket Length	one byte	9	
Issuer Subpacket Type	one byte	16	
Issuer	eight bytes		The identifier of the Signing Entity Public Key used to authenticate this packet
Unhashed Data Length	two bytes	0	
Truncated Hash	two bytes		The leftmost 16 bits of the signed hash value
Signature Data	remainder of packet length		The value of the ECDSA signature as specified in RFC 6637 [23] Section 10.

(4) An OpenPGP Modification Detection Code Packet
The content of this packet shall be as follows:

Field Name	**Size**	**Value**	**Description**
Packet Header	two bytes	Packet Tag = 19	The format of the Packet Header shall be as described in RFC 4880 [22] Section 4.2 for New Format Headers. For this packet the packet length is always encoded in a single byte.
SHA-1 Hash	twenty bytes		The SHA-1 hash of the preceding three packets – One-Pass Signature Packet, Literal Packet, and Signature Packet – in that order

The recipient of the SDS Authenticated Key shall perform the following processing steps:

1) Verify the contents of the OpenPGP Modification Detection Code packet against the SHA-1 hash of the three preceding packets. If the integrity protection checks fail, discard the Security Data Stream Packet.
2) Calculate the hash of the OpenPGP Literal Packet using the algorithms specified in the OpenPGP One-Pass Signature Packet.
3) Use the Key Identifier from the OpenPGP Signature Packet to locate the Signing Entity Public Key stored on the Authenticating Entity. If this Key Identifier does not correspond with a Signing Entity Public Key known to the Authenticating Entity, discard the Security Data Stream Packet.
4) Using the Signing Entity Public Key from (3), verify the signature data contained in the OpenPGP Signature Packet against the hash value calculated in (2). If the signature does not verify, discard the Security Data Stream Packet.
5) Process the contents of the OpenPGP Literal Packet as specified in Section 6.4.3 (for Authentication Key Payloads) or Section 6.4.4 (for Public Key Distribution Payloads).

6.4.5.4 Authentication Key Payload

The Authentication Key Payload shall be formatted as an OpenPGP Literal Packet with the following content:

Field Name	Size	Value	Description
Packet Header	one to five bytes	Packet Tag = 11	The format of the Packet Header shall be as described in RFC 4880 [22] Section 4.2 for New Format Headers
Data Format	one byte	'b' (0x62)	Binary Data
File Name string length	one byte	1 (0x01)	Length of following "file name" field
File Name (= Authentication Key Number)	one byte	'1' to '4' (0x31 to 0x34)	The index number to which the following Authentication Key Value is assigned.
Date/Time	four bytes	= RTP Timestamp	Carries RTP Timestamp format from Table 9.2 indicating time of first use of Key
Authentication Key Value	32 bytes		The value of the Authentication Key

A recipient of the Authentication Key Payload shall update the HSM Token containing the Authentication Keys with the new Authentication Key Value for the specified Authentication Key Number.

6.4.5.5 Public Key Distribution Payload

The Public Key Distribution Payload shall be formatted as an OpenPGP Literal Packet with the following content:

Field Name	Size	Value	Description
Packet Header	one to five bytes	Packet Tag = 11	The format of the Packet Header shall be as described in RFC 4880 [22] Section 4.2 for New Format Headers
Data Format	one byte	'b' (0x62)	Binary Data
File Name string length	one byte	0 (0x00)	Length of following "file name" field
Date/Time	four bytes	= RTP Timestamp	Carries RTP Timestamp format from Table 9.2 indicating time of creation of the Key
Public Key Identifier	eight bytes		The value of the Network-unique Key Identifier used to access this Public Key
Public Key File Data	Remainder of the packet length		The DER [30] encoded value of the Public Key File

The recipient of the Public Key Distribution Payload shall update the HSM Token containing the Signing Entity Public Keys with the new Public Key Data for the specified Public Key Identifier.

6.4.6 Security System Implementation

Certain aspects of Security System implementation involve physical security and interoperability in addition to interoperability of data interchange protocols. In particular, it is critical that certain Security System secrets not be exposed to potential interception by an adversary attempting to carry out a Man-In-The-Middle (MITM) attack. Primary among such secrets are the Private Keys of all the equipment on the network. They should not be exposed through communicating them anywhere, and even exposing them within the equipment that must use them can lead to a security breach. For these reasons, protection of the Private Keys of all equipment as well as of other secrets such as Authentication Keys is entrusted to hardware-based Cryptographic Token**s**. The Tokens operate using a common interface defined in [26] and related documents referenced in the cited work. They have both internal processing capability and non-volatile storage within their hardened physical instantiations. The Tokens connect to equipment with which they work through a USB interface. Physical security for the Tokens themselves also is a consideration in overall network security.

6.4.6.1 Token Initialization

To establish a secure network, it is necessary to initialize a Cryptographic Token for each equipment unit on the network. It is anticipated that Signing Entities will include the necessary facilities for managing the initialization process. It is assumed that the Signing Entity used for initialization will have at least two USB ports for connecting Tokens. The initialization process will include the following steps, which assume redundant Signing Entities and multiple Authenticating Entities:

1) Install a Token into the 1st USB port of the Signing Entity to be used for initialization purposes
2) Generate a Public/Private Key pair within the Token of step 1.
3) Install a Token to be used in another Signing Entity into the 2nd USB port of the Signing Entity of step 1 (if none needed, continue with step 7)
4) Generate a Public/Private Key pair within the Token of step 3
5) Copy the Public Key from the Token of step 3 to the Token of step 1
6) If another Token for use in a Signing Entity is needed, return to step 3; otherwise, continue with step 7
7) Install a Token to be used in an Authenticating Entity into the 2nd USB port of the Signing Entity of step 1
8) Generate a Public/Private Key pair within the Token of step 7
9) Copy the Public Key from the Token of step 7 to the Token of step 1
10) Copy the set of Public Keys from all the Tokens of steps 3 – 5 for use in Signing Entities from the Token of step 1 to the Token of step 7
11) If another Token for use in an Authenticating Entity is needed, return to step 7; otherwise, continue with step 12
12) Reinstall a Token from steps 3 – 5 for use in a Signing Entity into the 2nd USB port of the Signing Entity of step 1
13) Copy the Public Keys of all other Tokens of steps 3 – 5 for use in Signing Entities from the Token of step 1 to the Token of step 12
14) Copy the Public Keys of all Tokens of steps 7 – 11 for use in Authenticating Entities from the Token of step 1 to the Token of step 12
15) If another Token for use in a Signing Entity remains to be completed, return to step 12; otherwise continue to step 16
16) Remove the last Token for use in a Signing Entity from the 2nd USB port of the Signing Entity of step 1
17) Install a Token processed in steps 3 – 5 for use in a Signing Entity into another Signing Entity and repeat until all Signing Entities have Tokens installed
18) Install a Token processed in steps 7 – 10 for use in an Authenticating Entity into an Authenticating Entity and repeat until all Authenticating Entities have Tokens installed
19) Save initialized but unused Tokens as spares

6.4.6.2 Token Data Storage

To accommodate commonly applied open-source development tools that have limitations on the number of Data Objects they can create,[1] only one Data Object shall be used for storage in Cryptographic Tokens formatted according to PKCS #11 [26]. To accommodate this limitation, except for Private Keys, all of the security information that is stored on each Cryptographic Token shall be concatenated into a single Data Object. A concatenated Data Object comprises a series of Data Items, each beginning with an identifier of the information element included in the Data Item, followed by a value indicating the length of the Data Item, and ending with the value of the information element itself.

6.4.6.2.1 Token ID Storage and Marking

Each Cryptographic Token HSM shall have an associated Token Identifier that is unique within the context of the network within which the USB Token is used. The Token Identifier shall be included in a Data Item within the Data Object stored on the HSM, as described in Table 6.7 and the text following that table in Section 6.4.6.5 below. The Token Identifier value also shall be shown on the exterior of the Cryptographic Token HSM to which it applies.

6.4.6.2.2 Private Key Storage

The Private Key associated with each Cryptographic Token shall be stored in a separate Data Object dedicated to the purpose of storage of Private Keys, as defined in [26], in the Cryptographic Token HSM to which it applies.

6.4.6.2.3 Public Key Storage

Public Keys of two types shall be stored on Cryptographic Token HSMs – the Public Key derived from the Private Key associated with the Cryptographic Token (called the Local Public Key below) and the Public Keys associated with other Cryptographic Tokens in the same network (called Remote Public Keys below). Both types of Public Keys shall be stored as Data Items within the Data Object of the Cryptographic Token HSM in which they are stored as described in Table 6.7 and the text following that table in Section 6.4.6.5 below.

6.4.6.2.4 Creation Time Value Storage

The Creation Time value shall be stored on the Cryptographic Token HSM as a Data Item value representing the time at which the associated Local Public Key was derived from the Private Key of the Cryptographic Token, as described in Table 6.7 and the text following that table in Section 6.4.6.5 below.

6.4.6.2.5 Fingerprint Value Storage

The Fingerprint value shall be stored on the Cryptographic Token HSM as a Data Item value containing the Fingerprint of the Local Public Key that was derived from the Private Key of the Cryptographic Token, as described in Table 6.7 and the text following that table in Sections 6.4.6.5 and 6.4.6.6 below.

6.4.6.2.6 Public Key File Storage

The components of the Public Key File associated with the Cryptographic Token HSM are stored as separate Data Items within the Data Object of the Cryptographic Token, as described in Table 6.7 and the text following that table in Section 6.4.6.5 below, but they are turned into a formatted

[1] E.g., OpenSC.

Public Key File only as needed on export from the Cryptographic Token or the system to which it is connected.

6.4.6.2.7 Authentication Keys Storage

The four Authentication Keys shall be stored as individual Data Items within the Data Object of the Cryptographic Token HSM. If the Cryptographic Token is capable of fast enough data recall to permit outputting an Authentication Key each time a data packet arrives at the Authentication process, then Authentication Keys should not be stored on the system platform to which the Cryptographic Token HSM is connected but should be requested by the system platform each time an Authentication Key is needed. If the Cryptographic Token is not capable of sufficiently fast operation to permit outputting an Authentication Key each time a data packet arrives at the Authentication process, then it will be necessary to store Authentication Keys on the system platform to which the Cryptographic Token HSM is connected, but such Keys should be stored there in volatile memory so that they will not be held in memory when the system is powered down or restarted.

6.4.6.2.8 Remote Public Key Storage

In terms of the protocols defined in this document, the number of Public Keys that can be present in a single network is limited only by the size of the number space allocated for identifying individual entities in that network so that they can be separately addressed as necessary for the functions they perform. In the case of the STLTP, for example, that allocation is limited to 8192 addressable entities due to the address space provided in the Timing and Management Data Stream. When larger numbers of Public Keys must be made available to certain entities on a network, due either to limitations in address space or limitations in available memory in a Cryptographic Token in use, since Public Key values do not require secure storage, it is acceptable for them to be stored in more conventional storage devices, e.g., USB thumb drives, separate from the Cryptographic Token. As a matter of convenience in transferring Public Key values into or between network entities, however, it is desirable to minimize the number of storage devices that must be installed or moved when adding equipment or replacing either equipment or storage devices. These considerations point to the desirability of equipment providing multiple USB connections that can support storage devices containing Public Keys and that Cryptographic Tokens with included additional memory, operating over conventional USB interfaces shared with the Cryptographic Token functionality, be used when more memory is needed than provided with a conventional Cryptographic Token. Such multifunction storage devices are known to exist in the marketplace.

6.4.6.3 Security in a Redundant Environment

Broadcast transmission systems often are constructed with redundant equipment to improve the reliability of those systems. Redundant operation must be considered when configuring a Tunneled Packet Security System. Fortunately, most of the considerations have been taken into account in the design of the Security System itself.

In Transmitters, multiple Authenticating Entities frequently are installed. In the Tunneled Packet Security System, Authenticating Entities are treated independently, whether they are installed in separate Transmitters or installed together in a single Transmitter. Thus, nothing needs to be done to facilitate redundant Authenticating Entities in a Transmitter other than installation of an initialized Cryptographic Token in each Authenticating Entity.

With respect to redundant Signing Entities, somewhat more is required. Because they serve as the sources of the content and of the security data and Keys, they generally will operate in a switched configuration in which one unit will be on line at a time, with additional units performing the same functions in a synchronized way and ready to be switched on line at any time when a downstream monitor senses a problem in the unit currently on line. When no security functionality is applied, the redundant operation adds only a relatively small increment to the complexity of operation. When security functionality is active, and especially if a failure occurs at a critical time, considerably more complexity in the security system is required to assure a smooth and reliable transition between Signing Entities.

Two specific, potential conditions must be considered with respect to switching between redundant Signing Entities. First, since monitors typically will examine the outer, Tunnel Packet Stream and switch on a packet boundary in that Stream, and since Signing Entities can be set to rotate or randomize their use of the different Authentication Keys that are available on the network, it becomes necessary to synchronize the selection of the particular Key in use for each Tunneled Packet and to assure that the same one is used by all Signing Entities in the redundant set. A further implication is that all Signing Entities must have available the same set of Keys at all times, requiring additional synchronization in the Key update process. Another complication can occur when the Signing Entity that is online is in the process of updating Keys on the Authenticating Entity(ies) when it fails. In that case, the redundant Signing Entity that goes online next must continue the process of downloading the same Key to the Authenticating Entity(ies) until the new Key is fully installed. At the current time, the requirement for such synchronization between Signing Entities points to the need to obtain them from a common source for any particular redundant installation.

6.4.6.4 Data Processing and Storage

Cryptographic Tokens have the capability to provide both data processing and non-volatile data storage. For interoperability between Tokens and the host systems in which they operate, it is necessary that Tokens selected for use in the Tunneled Packet Security System have certain characteristics and support certain algorithmic functions. Functions needed by the Security System but not provided by Cryptographic Tokens must be provided by the host systems. To minimize and normalize the security workload on the host systems, Tokens shall have the following features:

- Support for USB 2.0 or greater using a USB Type A connector
- Non-Volatile Memory of at least 80 kB
- Support for OASIS PKCS #11 Cryptographic Token Interface API Version 2.40 [26]
- Data processing for NIST AES-256 [20]
- Data processing for SHA2-384 Hash [27]
- Data processing for the OpenPGP Message Format [22]
- Data processing for Elliptic Curve Cryptography in OpenPGP [23]
- Data processing for the Elliptic Curve Diffie-Hellman KDF [23]
- Data processing for the Elliptic Curve Digital Signature Algorithm [24]
- Support for NIST "Curve P-384" [24]

Cryptographic Tokens found in the marketplace as of the date of this standard do not directly support GCM/GMAC data processing. Consequently, for the GMAC Authentication Function described herein, the data processing shall be divided between the Token and the host system. Since GCM/GMAC is a mode of operation of the Advanced Encryption Standard (AES) and since data processing for AES is included in available Cryptographic Tokens meeting the specifications above, application of the underlying AES block cipher process shall take place in a Token, while the remaining processes involved in forming and applying GMAC shall take place in the host.

Easing the processing burden and speed requirement on the Token is the factor that only one application of the AES-256 cipher process is required for each invocation of GHASH and GCTR, which comprise GMAC Authentication and are applied once per authenticated packet. Additional processing is required of the host, however, as it performs the remainder of the GHASH and GCTR function processing.

Data stored within a Cryptographic Token can take one of two forms: Secrets and related data generated within the Token (i.e., the Public and Private Keys and any Authentication Keys generated by the Token), and secrets and related data generated by other Tokens in the network and delivered to the Token as part of one of the security or Authentication processes (i.e., Public Keys and Authentication Keys generated by other Tokens). All such Keys and related data shall be stored in the Cryptographic Token installed on each host and shall not be stored on the host itself. Rather, when a host has need of any data stored on a Token or a processed version of such data, the host shall withdraw a copy of the data or processed data from the Token for one use only, and the host shall withdraw another copy of the data or processed data when it is needed again. One use shall constitute the processing of one Tunneled Packet of a CTP Inner Stream at both Signing Entities and Authenticating Entities; the data withdrawn shall be a Hash subkey for use in the Authentication processing of one Tunneled Packet of a CTP Inner Stream. All other Tunneled Packet Security System processing shall take place within the Tokens.

It should be noted that, through the storage of all the security data and Keys on Tokens, it becomes relatively easy to replace equipment while retaining the information necessary to resume operation immediately without requiring any refresh process. Further, since all of the security data is tied to a number identifying a Token and not the equipment in which the Token is installed, it also is possible to freely replace a Token, if necessary, with another Token, so long as the Token has been initialized for the correct type of application, i.e., for a Signing Entity or for an Authenticating Entity. In the case of replacing a Token with one that was initialized at an earlier time, it may be necessary to update Authentication Keys and possibly Public Keys in the replacement prior to its installation if any such values have been updated in the system while the replacement was not in service.

6.4.6.5 Cryptographic Token Data Object

Most data within Cryptographic Token**s** formatted for use in the Tunneled Packet Security Protocol shall be stored within a single Data Object that is accessible using the PKCS #11 [26] Cryptoki interface. The only data related to the Security Protocol not stored within the single Data Object shall be the Private Key of each Cryptographic Token.

The data within the single Data Object shall be stored with all the stored values in sequence from the beginning to the end of the Data Object, using an Identifier value for each Data Item within the Data Object, with each Identifier followed by a Length value for the Data Item. The Length associated with a Data Item shall indicate the number of Bytes of the Data Item payload that immediately follow the last Byte of the Length and immediately precede the next Data Item Identifier, if there is one. The end of the Data Object is found using the value carried in the Data Object Length Data Item.

The Data Items, their Identifiers, and their Lengths are defined in **Table 6.7**, and the semantics of the various Data Item payloads are described in the paragraphs following the table.

Table 6.7 Cryptographic Token Data Object Structure

Identifier	Length (Bytes)	Mandatory/Optional	Description
0x0000	1	M	Data Object Format Version Number
0x0001	32	M	Authentication Key #1
0x0002	32	M	Authentication Key #2
0x0003	32	M	Authentication Key #3
0x0004	32	M	Authentication Key #4
0x0005	4	M	USB Token Identifier
0x0006	4	M	Data Object Length
0x0007	160	M	Local Public Key Value
0x0008	10	M	Local Public Key Creation Time
0x0009	20	M	Local Public Key Fingerprint
0x000A	8	M	Local Public Key Identifier
0x000B	1 – 255	O	Local Public Key File Title
0s000C	1 – 255	O	Local Public Key File Comment
0x000D – 0x00FF	–	–	Reserved
0x100 – 0xFFFF	171	O	Remote Public Keys with Key Identifiers

Data Object Format Version Number shall indicate the version number of the format described in this section of this document. The value for instances of the Data Object structured according to this description shall be 0x00. Its value shall be incremented by one for each successive revision.

Authentication Key **#1** shall be the 32-Byte (256-bit) value currently assigned to the Authentication Key **#1** function in the Authentication processing of each authenticated Tunneled Packet. The Authentication Key **#1** value will be applied to Authentication of a packet when the Header Extension of the packet indicates use of Authentication Key **#1**.

Authentication Key **#2** shall be the 32-Byte (256-bit) value currently assigned to the Authentication Key **#2** function in the Authentication processing of each authenticated Tunneled Packet. The Authentication Key **#2** value will be applied to Authentication of a packet when the Header Extension of the packet indicates use of Authentication Key **#2**.

Authentication Key **#3** shall be the 32-Byte (256-bit) value currently assigned to the Authentication Key **#3** function in the Authentication processing of each authenticated Tunneled Packet. The Authentication Key **#3** value will be applied to Authentication of a packet when the Header Extension of the packet indicates use of Authentication Key **#3**.

Authentication Key **#4** shall be the 32-Byte (256-bit) value currently assigned to the Authentication Key **#4** function in the Authentication processing of each authenticated Tunneled Packet. The Authentication Key **#4** value will be applied to Authentication of a packet when the Header Extension of the packet indicates use of Authentication Key **#4**.

USB Token Identifier shall be a serial number for the USB Token with which it is associated, which, in turn, shall be formatted according to the Cryptoki standard [26]. The USB Token Identifier shall be unique within the universe of USB Tokens used in a secure network. Since each USB Token in a secure network will contain different data, the USB Token Identifier provides a relationship between the data set contained within the USB Token Data Object and the physical USB Token itself. The same value contained within the USB Token Identifier Data Item should be displayed on the exterior of the physical USB Token on which it is stored in the Data Object. For any physical display of the Token Identifier value, its hexadecimal value shall be shown.

Data Object Length shall indicate the length in Bytes of the total Data Object from the start of the first Byte of the Data Item Identifier of the USB Token Identifier (Data Item Identifier value 0x0000) to the end of last Byte of the payload of the last Remote Public Key Data Item.

Local Public Key Value shall be the value of the Public Key derived from the Local Private Key stored on the same USB Token as the Data Object in which the corresponding Local Public Key Value Data Item is stored. The Local Public Key Value Data Item shall be formatted in 160 Bytes of Base64-coded text, including two Object Identifier (OID) values, as described in detail in Section 6.4.6.6 below.

Local Public Key Creation Time shall be the UTC time at which the Local Public Key Value Data Item was derived from the Local Private Key, formatted as described in detail in Section 6.4.6.6 below.

Local Public Key Fingerprint shall be the Fingerprint value derived and coded as described in detail in Section 6.4.6.6 below.

Local Public Key Identifier shall be the identifier value derived from the Local Public Key Fingerprint as described in detail in Section 6.4.6.6 below.

Local Public Key File Title shall be the text contained on the Title line of the Public Key File described in detail in Section 6.4.6.6 below.

Local Public Key File Comment shall be the text contained on the Title line of the Public Key File described in detail in Section 6.4.6.6 below.

Reserved values shall not be used except as defined in later versions of this standard or in another ATSC standard. Implementers are advised that Reserved values may be applied in the future and that equipment/software built to this standard should ignore such values if it does not understand their meaning.

Remote Public Key shall be a combination of the Key Identifier and the Public Key data stored on another Cryptographic Token in the set of such tokens in use within a single network. Both the Key Identifier and the Public Key data shall be independently coded in Base64 form, as described in Section 6.4.6.6 below, the first 11 characters carrying the Key Identifier data derived from the Fingerprint of the Public Key, and the last 160 characters carrying the Public Key value.

6.4.6.6 Public Key File Format

To support the Key delivery methods described in Section 6.4.3 and to enable interchange and storage of Public Keys for system initialization and Key value retention, a format is specified for placing required information into text files for these purposes. The text files shall contain copies of the Public Key data itself, of the Fingerprint of the Public Key data, and of the Creation Time of the Public Key. Only one Public Key and its associated data shall be stored in each text file of the type described in this section. Note the Public Keys referenced in this section are Public Keys derived using the Public Key Algorithm described in Section 6.4.3.

The text files shall be formatted with ASCII headers and footers as well as metadata descriptions, as described below. The text files shall contain only ASCII text and data expressed in ASCII characters.

- The first line shall begin with a `"Title:"` description followed by title information, which shall have a maximum length of 255 characters.
- The second line shall begin with a `"KeyID:"` description followed by a representation of the Public Key Identifier derived from the Fingerprint of the Public Key carried in the file. The Public Key Identifier shall be a 16-hexadecimal-character value, representing an 8-byte binary number, expressed as 16 ASCII characters, that identifies the Private/Public

Key pair from which the data in the Public Key File is derived. The KeyID value shall be the low-order 16-hexadecimal-character value of the Fingerprint.

- The third line shall begin with a `"Comment:"` description, followed by any comment information, which shall have a maximum length of 255 characters.
- The fourth line shall begin with a `"Creation Time:"` description, followed by a 10-character ASCII representation of the decimal UTC seconds count from 1 January 1970 at 00:00:00 corresponding to the time when the Public Key was created.
- The fifth line shall be blank.
- The sixth line shall contain a fixed header value of `"-----BEGIN PUBLIC KEY-----"`. Note that the header line begins with five dashes before the text and ends with five dashes immediately following the text.
- After the header row, the next line shall begin the Public Key Identifiers and data, which shall be encoded according to the ASN.1 Distinguished Encoding Rules (DER) [30] and expressed in the Base64 format, which uses only ASCII characters.
- The Base64-formatted text shall begin with a sequence of two Object Identifier (OID) values, one that identifies an Elliptic Curve Public Key (`OID = 1.2.840.10045.2.1`), and one that identifies the P-384 elliptic curve used in the STLTP Security Data scheme (`OID = 1.3.132.0.34`). The OIDs shall be coded using the ASN.1 DER [30]. The DER-coded OIDs shall constitute 72 and 56 bits, respectively, inclusive of the Object Identifier and Length Tags of the encoded Object Identifiers and after padding to integer numbers of octets (bytes). The OIDs shall be followed by the Public Key stored in the file. The Public Key shall constitute 776 bits of data after padding to an integer number of octets. When correctly encoded into Base64, the data representing the OIDs and Public Key value will occupy 160 characters.
- Following the end of the Base64-formatted text, the next line shall contain a fixed footer value of `"-----END PUBLIC KEY-----"`. Note that the fixed footer line begins with five dashes before the text and ends with five dashes immediately following the text.
- The fixed footer shall be followed by a blank line.
- After the blank line following the fixed footer, the next line shall begin with `"Fingerprint:"`, which shall be followed by the Fingerprint data expressed in hexadecimal values represented in ASCII characters, with two characters in the range from '0' – '9' and 'a' – 'f' (or 'A' – 'F') per byte of Fingerprint data. The Fingerprint value shall be computed according to the procedure and using the component data elements specified for V4 Fingerprints in Section 12.2 on "Key IDs and Fingerprints" in [22]. The Creation Time value used in calculating the Fingerprint shall be the value included on the third line of the file content.

An example of the text file format follows:

```
Title: This is a title.
KeyID: This is a unique ID for the Security Token - 64 bits / 8 Bytes (Add
Scope language)
Comment: This is a comment.
Creation Time: 1570618441 [Express as UTC time = 10 Bytes carrying ASCII
representation of decimal count of number of seconds from 1/1/1970  at
00:00:00.0,]
```

```
-----BEGIN PUBLIC KEY-----
MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEWN6ZQhII4jbS7Kph0x4QoX4kXn0+/k5318At30WaO3u
mME3hMZRoMqklzA8auRoxeVj1OWdnjGMIzrrkXfQDp8BEJJM6MdaTiwqtIDNbDBcvEMcuY6cVFT
R2Rgi0H6pb
-----END PUBLIC KEY-----

Fingerprint: 5587FDC6F0792985AF59DA4AB2262BD4B32B0CB6
```

It is recommended that File Names of Public Key Files incorporate the associated Public Key Identifiers within their respective file names. It should be noted that the derivation of Public Key Identifiers from Fingerprints, as described above, can lead to duplicate Public Key Identifier values within a network, i.e., they are not guaranteed to be unique. Therefore, if duplicate Public Key Identifiers are derived from Fingerprints, one Private/Public Key pair should be regenerated to avoid possible collisions in the decryption process incorporated in the Security Data Stream secure delivery process.

6.4.6.7 Physical Security

For full security of CTP delivery, the Security System itself must be secured. While the components used are secure electronically, they also must be secured physically. Physical security, in this instance, involves protecting the Cryptographic Token**s** from removal, substitution, or reprogramming. Physical designs of Signing Entities and Authenticating Entities that permit installing the Tokens inside the equipment or behind lockable doors are among the ways in which physical security of the Tokens can be obtained.

7 DATA SOURCE TRANSPORT PROTOCOL

The Data Source delivery structure uses the Data Source Transport Protocol (DSTP), which applies between the various Data Sources and an ALP encapsulation function. There may be multiple Data Sources for each PLP, each carried on a separate packet Stream. The packets produced by the Data Sources are UDP/IP multicast packets intended to be consumed by receivers. Thus, their construction follows rules defined by A/331 [4] including their IP addressing which is not conducive to efficient broadcast studio routing.

The Data Source Transport Protocol hides the details of the Data Source packets by encapsulating them in a Data Source Packet Tunnel derived from the Common Tunneling Protocol (see Section 6). The Data Source Packet Tunnel consists of an RTP/UDP/IP multicast or unicast IPv4 packet Stream whose payloads shall contain one or more Data Source packet Streams. In addition, the DSTP includes a Tunneled Packet Information Header prior to each Data Source Tunneled Packet that contains information for how to route and schedule each packet without the need to examine the actual packet itself. The Data Source packets along with their associated Information Headers are referred to as the Data Source Tunneled Packets or *inner* packets while the *outer* packets are referred to as Data Source *Tunnel Packets*. The details of encapsulation of the Data Source Packets into the Packet Tunnel are described in the Common Tunneling Protocol in Section 6.

7.1 Overview

The system diagram in Figure 4.2 provides the functional context of the Data Source Transport Protocol. It is possible that an actual implementation may be much more complicated, requiring multiple sources of Data Source packets. Such sources could be for redundancy or could be external to the studio. To flexibly accommodate various network topologies, there may be multiple Data Source Tunnel Streams sent to an ALP Generator function. The Data Source Tunnel Streams act as transports for carrying one or more Data Source Tunneled Streams from various Data Source generation devices. The ALP Generator function shall accept one or more DSTP-formatted data Streams and provide mapping of the tunneled Data Source packets into the appropriate PLP data Streams. Figure 7.1 provides a depiction of three Data Source generators supplying multiple Data Source Streams through three DSTP Packet Tunnel Streams into an ALP Generator. The ALP Generator's Data Source Mapping configuration, supplied by the System Manager, and the associated Tunneled Packet information headers dictate how the Tunneled Packets are mapped to the appropriate PLP as shown by the dashed lines within the ALP Generator in the figure. Note that this standard does not constrain the number of DSTP-formatted input Streams defined for receipt by the ALP generator. The DSTP descriptions below apply to a single Data Source Tunnel Packet Stream.

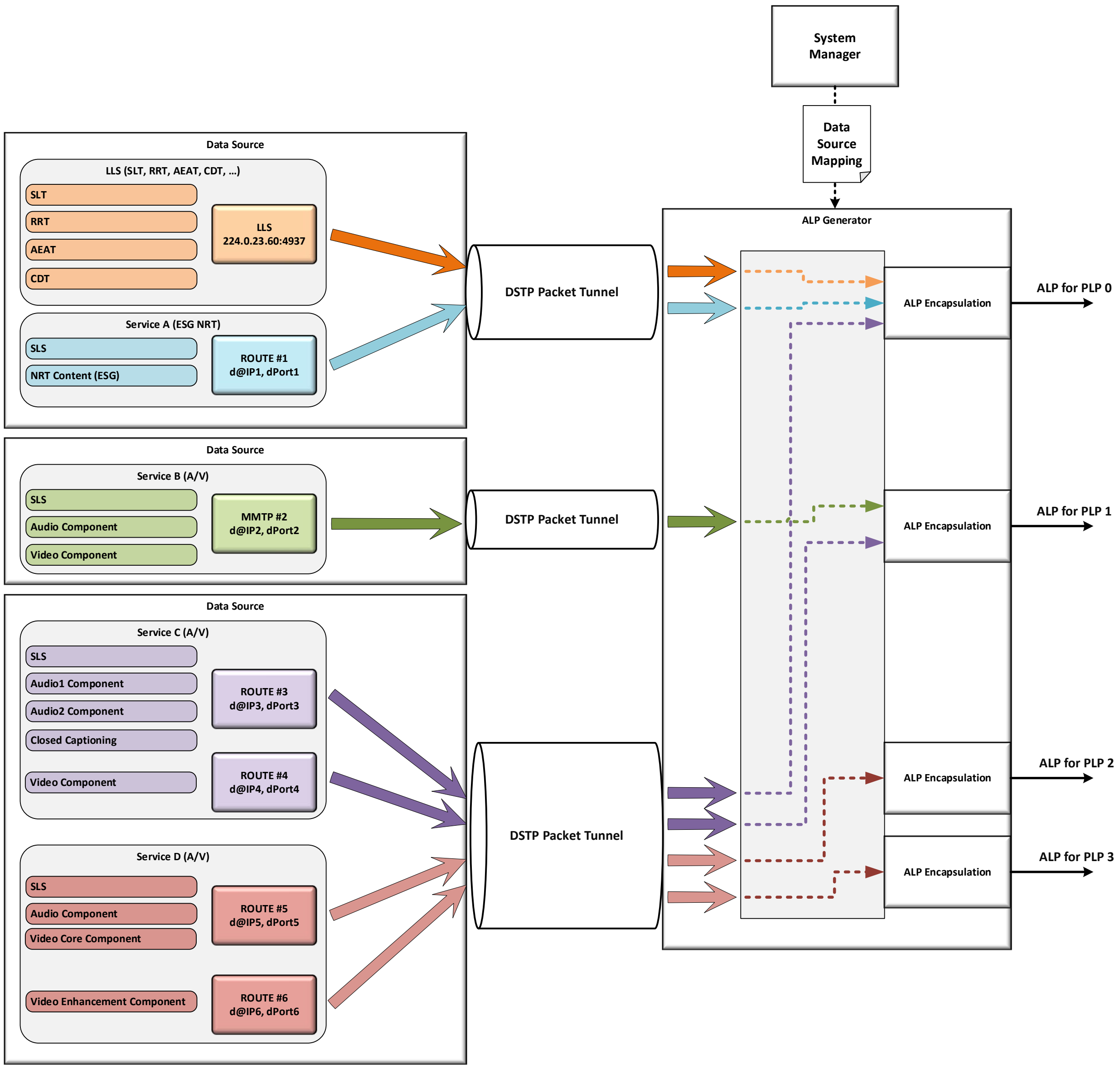


Figure 7.1 Multiple Data Source Packet Tunnels Concept

7.1.1 Data Source Mapping Configuration Description

The Data Source Mapping Configuration is provided by the System Manager to the Broadcast Gateway ALP Generator function through protocols that are out of scope of the present document as described in Section 4.2.1. Two syntax definitions are provided herein, XML and JSON, to normatively define the semantics and syntax of the configuration regardless of the communication protocol that is used. Note that this configuration information is intended for carriage on a Control Plane and does not transit the normal broadcast data paths (i.e., via DSTP, ALPTP or STLTP).

The Data Source Mapping Configuration provided to the ALP Generator may be represented as an XML document containing a **`DSMapping`** root element that conforms to the definitions in the XML Schema Definition (XSD) file that has namespace:

`tag:atsc.org,2021:XMLSchemas/ATSC3/Delivery/DS_MAPPING/1.0/`

The definition of this schema is contained in an XML Schema Definition (XSD) file, DSMapping-1.0-20210512.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be "`dsm`".

Alternatively, the Data Source Mapping Configuration may be provided to the ALP Generator in the form of a JSON schema file. The definition of this JSON schema can be found in the JSON schema file, dsmapping-20210512.json, accompanying this standard.

The semantics provided in Table 7.1 shall be used to capture the Data Source Mapping Configuration information. While the indicated XSD or JSON schema files specify the normative syntax of the **`DSMapping`** element, informative Table 7.1 below describes the structure of the **`DSMapping`** element, expressed as both XML and JSON data types, in a more illustrative way. The specifications following the table give the semantics of the elements and attributes applicable to both forms of representation.

Table 7.1 Data Source Mapping Configuration Semantics

Element or Attribute Name	Use	XML Data Type (JSON Data Type)	Short Description
DSMapping			Root element of the Data Source Mapping configuration.
DSTunnel	1..N		Defines one or more incoming Data Source Tunnel Streams.
@destAddr	1	IPv4address (string / ipv4)	The destination IPv4 address of the incoming DSTP Tunnel Stream.
@destPort	1	unsignedShort (integer)	The destination IP port of the incoming DSTP Tunnel Stream.
@srcAddr	0..1	IPv4address (string / ipv4)	The optional source IPv4 address of the incoming DSTP Tunnel Stream.
@igmpVersion	0..1	unsignedByte 2..3 (integer 2..3)	The optional IGMP version number. Absence of this attribute indicates IGMP is not in use.
@defaultPLP	0..1	unsignedShort (integer)	Optionally, supplies the PLP where any unspecified Tunneled Packet within the Data Source Tunnel should be routed.
DSTBackup	0..N		An optional collection of backup Data Source Tunnel Streams.
@srcAddr	1	IPv4address (string / ipv4)	The source IPv4 address of an optional incoming DSTP backup Tunnel Stream.

Element or Attribute Name	Use	XML Data Type (JSON Data Type)	Short Description
TPS	0..N		Element defining each Tunneled Packet Stream.
@destAddr	1	IPv4address (string / ipv4)	Defines the IPv4 destination address of a Tunneled Packet Stream within the DSTP Tunnel Stream. This address is matched with the value found in the Tunneled Packet Information Header (see Section 7.2.1).
@destPort	1	unsignedShort (integer)	Defines the IPv4 destination port of a Tunneled Packet Stream within the DSTP Tunnel Stream. This port is matched with the value found in the Tunneled Packet Information Header (see Section 7.2.1).
@plp	1	unsignedShort (integer)	Supplies the PLP where the associated Tunneled Packet Stream should be routed.

The following text specifies the semantics of the elements and attributes in a Data Source Mapping Configuration.

`DSMapping` – The root element of a Data Source Mapping Configuration.

`DSTunnel` – The `DSTunnel` element shall contain the configuration of each of the Data Source Tunnel Streams to be received and processed by an ALP Generator. Each `DSTunnel` element contains the address of the incoming DSTP packet Stream with mapping information for each Stream of Tunneled Packets within it.

`@dstAddr` – The required destination address of the DSTP Tunnel Stream. This may be a multicast or unicast address.

`@dstPort` – The required destination port of the DSTP Tunnel Stream.

Note that according to A/331 Section 6.1 [4], all destination addresses and port combinations must be unique across the broadcast. Therefore, each Data Source packet Stream received by an ALP Generator must have a unique address/port and shall be routed only to a single PLP.

`@srcAddr` – An optional source address for the DSTP Tunnel Stream. This attribute is required if IGMP version 3 is being used by the routing system to identify the desired source of the DSTP packet Stream.

`@igmpVersion` – The optional `@igmpVersion` attribute allows the version of IGMP to be specified. If '2' is specified, then IGMP version 2 is being used to route the DSTP Tunnel Stream. A value of '3' indicates that IGMP version 3 is being used and Source-Specific Multicast (SSM) is available. If this attribute is provided, has value '3', and Source-Specific Multicast (SSM) is in use, then `@srcAddr` shall be provided. If this attribute is not provided, then IGMP is not being used.

`@defaultPLP` – The optional `@defaultPLP` attribute provides the PLP ID to which any unspecified Tunneled Packet Streams shall be routed. An unspecified Tunneled Packet Stream is any set of Tunneled Packets that have a Tunneled Packet Information Header that has `dest_address` and `port_number` values that do not correspond to any of those identified by `TPS` elements within the `DSTunnel` element. Packets from each unspecified Tunneled Packet Stream shall be routed to the PLP specified by this attribute. If this attribute is not provided, the default PLP shall be PLP 0.

`DSTBackup` – The optional `DSTBackup` element defines a backup Data Source Tunnel Stream that may be sourced from an alternate Data Source. The single, required `@srcAddr` attribute specifies the IP source address of the backup Data Source Tunnel.

`@srcAddr` – The required source address of the backup Data Source Tunnel Stream. This attribute is used in IGMP version 3 to join the backup Data Source Tunnel Stream if the primary Tunnel Stream is not available.

`TPS` – The `TPS` element contains three attributes that specify the Tunneled Packet Stream and where that Stream shall be routed to. Note that if the TPS element is missing then all packets within the tunnel are routed to the PLP specified in the `@defaultPLP` attribute.

`@dstAddr` – The required IPv4 destination address of the Tunneled Packet Stream within the DSTP Tunnel Stream. This address is matched with the value found in the Tunneled Packet Information Header (see Section 7.2.1) to determine into which PLP packets from this Stream should be placed.

`@dstPort` – The required destination port of the Tunneled Packet Stream within the DSTP Tunnel Stream. This port is matched with the value found in the Tunneled Packet Information Header (see Section 7.2.1) to determine into which PLP packets from this Stream should be placed.

`@plp` – The required `@plp` attribute shall provide the PLP ID to which the Tunneled Packet Streams should be routed. Packets from the Tunneled Packet Stream shall be routed to the PLP specified by this attribute.

7.2 DSTP Design

The Data Source packet Streams transport the various signaling, audio, video, captioning and data that make up an ATSC 3.0 broadcast [4]. As indicated in the introduction to Section 6.4, the Data Sources create packets with the destination addresses and ports intended to be seen by receivers. To avoid conflicts and readdressing issues, the DSTP encapsulates these Data Source Packet Streams in one or more Tunnel Streams based on the Common Tunneling Protocol described in Section 6.

The ALP Generator may receive one or more DSTP Streams and differentiates various Data Sources from within the DSTP Tunnel Packet Stream by a mapping supplied through configuration. To avoid the ALP Generator being required to understand the contents of various Data Source Tunneled Packets, additional delivery metadata is included as a Tunneled Packet Information Header. Placing delivery metadata in a separate header for each packet within the tunnel allows the Data Source packets to remain opaque to the transmission system. The ALP encapsulation function extracts the Data Source Tunneled Packets and their associated Information Headers from each DSTP tunnel before placing them in the respective ALP Streams. Note that the Information Headers are used as part of the mapping and ALP Stream creation but are discarded by the ALP Generation function.

7.2.1 DSTP Tunneled Packet Information Header Definition

The DSTP defines a Tunneled Packet Information Header to avoid modification to the Data Source Tunneled Packets. This header provides information to help the ALP Generator route the various packets to the appropriate PLP as well as to improve the scheduling performance of the Broadcast Gateway scheduling system when creating Physical Layer frames. A Tunneled Packet Information Header shall precede every Tunneled Packet that starts within the DSTP Tunnel Packet.

Table 7.2 provides the field definitions of the DSTP Tunneled Packet Information Header structure. The paragraphs following the table describe the fields within the table.

Table 7.2 DSTP Tunneled Packet Information Header

Syntax	No. of Bits	Format
information_header () {		
dest_address	32	uimsbf
if (dest_address ≠ 0) {		
port_number	16	uimsbf
length	16	uimsbf
group	16	uimsbf
type	8	uimsbf
random_access_point	1	bslbf
time_limit_flag	1	bslbf
if (type >= 1 && type <= 5) {		
wakeup_control()	2	Section 7.2.2
}		
else {		
reserved	2	'00'
}		
signed_flag	1	bslbf
reserved	3	'000'
if (time_limit_flag == 1) {		
timestamp_min()	32	Table 6.3
}		
if (signed_flag == 1) {		
GMAC_Header_Extension()	var.	Table 6.5
}		
}		
}		

dest_address – The unsigned, 32-bit **dest_address** shall specify the destination IP address of the Tunneled Packet following the information header. This value, the **port_number,** and the **group** are used by the ALP Generator to map incoming Tunneled Packets to outgoing ALP encapsulated Streams. Note that, in practice, this field is a copy of the destination address within the Data Source Tunneled Packet. It is provided here to avoid examining the Tunneled Packet in any way.

A value of zero (0x00000000) for the **dest_address** shall indicate that the Tunneled Packet Information Header is truncated and that a security data Stream packet begins immediately after the **dest_address**. The security data Stream is consumed by the system receiving the DSTP and is an RTP/UDP/IP packet containing the length so none of the information in the remainder of the header is needed.

port_number – This unsigned, 16-bit value shall define the destination port number of the Tunneled Packet immediately following the information header within the Tunnel Packet payload. This value, the **dest_address**, and the **group** are used by the ALP generator to map incoming Tunneled Packets to outgoing ALP encapsulated Streams. Note that, in practice, this field is a copy of the destination port within the Data Source Tunneled Packet. It is provided here to avoid examining the Tunneled Packet in any way.

length – This unsigned, 16-bit value shall specify the number of bytes occupied by the Tunneled Packet immediately following the information header. The length is the total size of the

Tunneled Packet, even if it is not fully contained within the current Tunnel Packet payload. Note that the length here is different than the IP header length contained within the Tunneled Packet since it also includes the length of any headers contained within the Tunneled Packet.

group – If the **type** is LLS, this unsigned, 16-bit value shall provide the `groupID` as defined in [4]. Otherwise, the field shall provide the `serviceID` associated with the Tunneled Packet Stream. If the `serviceID` is provided, it shall be taken directly from the service defined within the associated SLT as specified in A/331 [4]. The `groupID` or `serviceID` is provided to allow the scheduling functions within the Broadcast Gateway to optimize delivery of various services.

type – This unsigned, 8-bit value shall define the type of packet immediately following the information header within the Tunnel Packet payload. Table 7.3 defines the types currently supported by the Data Source Transport Protocol.

Table 7.3 type Field Available Values

Type	Title	Description
0	Reserved	
1	LLS SLT	
2	LLS RRT	
3	LLS System Time	
4	LLS AEAT	
5	LLS OSN	
6	LLS CDT	
7-13	Reserved	Reserved for future types
14	LLS Signed Multi-Table	
15	LLS User Defined	
16	SLS	
17-63	Reserved	Reserved for future types
64	Primary Video	
65	Auxiliary Video	
66	Video Enhancement	
67-127	Reserved	Reserved for future types
128	Primary Audio	
129	Auxiliary Audio	
130-191	Reserved	Reserved for future types
192	Primary Application Data	
193	Guide Data	
194	Application Data	
195	AEAT-Referenced Data	Associated NRT Data
196-254	Reserved	Reserved for future types
255	Unspecified	Indicates that the associated packet Stream type is not specified.

random_access_point – If set to 1, indicates that the associated inner packet contains some or all the payload containing a random-access point (RAP). If set to 0, no random-access point is present within the packet. See Annex C for a discussion of random-access points and their meaning within the system. The **random_access_point** bit shall remain 1 for all of the packets containing the portion of the Stream comprising the random-access point. For example, for packets carrying a video Stream, all of the packets containing the segment defined as the RAP would have the **random_access_point** bit set to 1.

time_limit_flag – This flag shall be set to 1 to indicate that a **timestamp_min ()** field is present. A value of 0 shall indicate that no **timestamp_min ()** field is present and that the packet may be delivered with best effort.

wakeup_control () – A 2-bit field that communicates information needed to control the emission wakeup field. See Section 7.2.2 for a definition of this field. The **wakeup_control ()** field shall only be interpreted if the Tunneled Packet type is one of the main LLS types with type values between 1 and 5.

signed_flag – This flag shall be set to 1 to indicate that the following Tunneled Packet has been signed and a **GMAC_Header_Extension ()** field that resulted from that signing process is present. A value of 0 shall indicate that the packet was not signed and that no **GMAC_Header_Extension ()** field is present.

timestamp_min () – This field shall be defined according to Table 6.3 and its subsequent semantic definitions. This time value shall indicate the latest time at which the start of the payload may be delivered.

GMAC_Header_Extension () – This field shall be defined according to Table 6.5 and its subsequent semantic definitions.

7.2.2 Emergency Alert Wakeup Controls

Emergency Alert Wakeup fields in the DSTP Tunneled Packet Information Header provide signaling information that indicates to the Scheduler how to manage the **ea_wake_up_1** and **ea_wake_up_2** bits in the Bootstrap. In the text that follows, these two bits are treated together as the "Emission Wakeup Field". There are two flags provided in the **wakeup_control ()** field. The first, **wakeup_active**, indicates that a currently active Advanced Emergency Alert (AEA) message has the @wakeup attribute set to 'true'. The flag shall reflect this condition on the DSTP Tunneled Packet Information Headers of all LLS packets generated by a given source, not just on packets containing an Advanced Emergency Alert Table (AEAT). In contrast, the **AEAT_wakeup_alert** flag shall reflect specific values within the current packet and shall apply only to packets containing an AEAT.

Table 7.4 RTP **payload_type** Wakeup Control Field Definition

Syntax	No. of Bits	Format
wakeup_control () {		
wakeup_active	1	bslbf
AEAT_wakeup_alert	1	bslbf
}		

wakeup_active – A 1-bit flag that, when set to '1', shall indicate that the source providing the LLS data is requesting that the Emission Wakeup Field be non-zero. When set to '0', this flag shall indicate that the source providing the LLS data is requesting that the Emission Wakeup Field be set to a value of zero if no other source is requesting a non-zero value.

AEAT_wakeup_alert – A 1-bit flag that, when set to '1', shall indicate that the packet contains a new or updated AEAT that carries a new Wakeup Alert. When set to '0', this flag shall indicate that the packet does not contain an AEAT with a new Wakeup Alert. This flag shall only be '1' when the **LLS_table_version** of the AEAT LLS_table () has been incremented and the AEAT contains an AEA element with the @wakeup attribute newly set to 'true'. In other words, if a new AEA is added with @wakeup set to 'true' or a previous AEA has been updated with @wakeup set to 'true', then the **LLS_table_version** will be incremented and the **AEAT_wakeup_alert** shall be set to '1'.

The **wakeup_active** field controls whether the Emission Wakeup Field is 'off' (zero) or 'on' (non-zero). **AEAT_wakeup_alert** controls when the Emission Wakeup Field toggles through the 'on' (non-zero) states, i.e., 1, 2, and 3. (See Table 7.5 for an example of **wakeup_active** and **AEAT_wakeup_alert** usage.) It is expected that LLS tables, such as the SLT or SystemTime tables, will be updated relatively frequently, allowing timely control. There is no restriction, however, on providing extra LLS signaling to accommodate updating the Emission Wakeup Field. For example, the Data Source could emit an SLT specifically to return the Emission Wakeup Field value to zero.

Table 7.5 Example Wakeup Bit Controls

Time	LLS Table	Wakeup State	wakeup_active	AEAT_wakeup_alert	Emission Wakeup Field
t0	SLT	Off	0	0	00
t1	SystemTime	Off	0	0	00
t2	New AEA @wakeup=true	On	1	1	01
t3	SLT	On	1	0	01
t4	Same AEA	On	1	0	01
t5	SystemTime	On	1	0	01
t6	Updated AEA @wakeup=true	On	1	1	10
t7	SLT	On	1	0	10
t8	SLT	Off	0	0	00
t9	New AEA @wakeup=false	Off	0	0	00

Note that this example is for a single source of LLS. If there is more than one source of LLS, the Scheduler is responsible for combining those controls and setting the Emission Wakeup Field appropriately.

8 ALP TRANSPORT PROTOCOL (ALPTP)

The ALP delivery structure uses the ALP Transport Protocol (ALPTP), which applies between ALP sources and Broadcast Gateway inputs when the ALP encapsulation function is external to the Broadcast Gateway. There may be multiple Data Sources for each PLP, but individual ALP packet Streams carry only a single source data Stream for each respective PLP. Thus, any source multiplexing is done upstream of ALP encapsulation and header compression functions (see Section 7.1).

The ALP Transport Protocol hides the details of the ALP packets by encapsulating them in an ALP Packet Tunnel derived from the Common Tunneling Protocol (Section 6). The ALP Packet Tunnel consists of an RTP/UDP/IP multicast IPv4 packet Stream whose payloads shall contain one or more ALP packet Streams. In addition, the ALPTP includes a Tunneled Packet Information Header prior to each ALP Tunneled Packet that contains metadata for how to schedule the various packets into Baseband Packets for the transmission as well as the PLP ID to which the ALP packet is bound. The ALP packets along with their associated Information Headers are referred to as the ALP Tunneled Packets or *inner* packets while the *outer* packets are referred to as ALP *Tunnel Packets*. The details of encapsulation of the ALP Packets into the Packet Tunnel are described in the Common Tunneling Protocol Section 6.

8.1 Overview

The system diagram in Figure 4.2 provides the functional context of the ALP Transport Protocol (ALPTP). It is possible that an actual implementation may be much more complicated, requiring multiple sources of ALP packets. Such sources could be for redundancy or could be external to the studio.

For example, consider a Transmitter-sharing agreement where two Physical Layer Pipes (PLPs) are dedicated to two separate stations. The individual studios could produce separate ALP packet Streams that are then delivered for injection into the shared Transmitter. If one of the studios were separated geographically, some sort of robust connection would be required to deliver the ALP packet Stream to the Scheduler / framing subsystem.

The functionality of the ALP standard [5] is sufficient for an emission link-layer protocol but should not be burdened with additional functionality to support intra- and inter-studio routing and distribution.

For these reasons, the ALPTP is based on the Common Tunneling Protocol carrying ALP packets along with the associated Tunneled Packet Information Headers. The use of the Tunneled Packet Information Header to bind each ALP packet with a PLP means that any number of ALPTP Tunnel Streams may be used between the ALP Generator and the Scheduler. However, to avoid issues with timing and scheduling, all ALP packets destined for a single PLP shall be carried in the same ALPTP Tunnel Stream. Furthermore, there is no limit on the number of PLPs that can be supported by a single ALPTP Tunnel Stream. The ALPTP Streams may be unicast or multicast consistent with CTP.

8.2 ALPTP Design

ALP describes the link-layer encoding for emitting various types of packets of a broadcast [5]. As indicated in the introduction to Section 8 above, the ALP protocol should not be burdened with routing requirements within the broadcast facility since this routing information is irrelevant to broadcast receivers. To provide ordering, timing and framing information required to route ALP Packet Streams, the ALPTP encapsulates these ALP Packet Streams in a Tunnel Stream for each PLP based on the Common Tunneling Protocol described in Section 6.

8.2.1 ALPTP Tunneled Packet Information Header Definition

The ALPTP defines a Tunneled Packet Information Header to avoid modification or analysis of the ALP Tunneled Packets. This header provides information to help the Scheduler associate the ALP packets with the targeted PLP as well as improve the scheduling performance of the Broadcast Gateway scheduling system when creating Physical Layer frames. An ALPTP Tunneled Packet Information Header shall precede every Tunneled Packet that starts within each ALPTP Tunnel Packet.

Table 8.1 provides the field definitions of the ALPTP Tunneled Packet Information Header structure. The paragraphs following the table describe the fields within the table.

Table 8.1 ALPTP Tunneled Packet Information Header

Syntax	No. of Bits	Format
alptp_information_header() {		
length	16	uimsbf
if (length == 0) {		
alp_sid	8	uimsbf
plp_id	6	uimsbf
lls_flag	1	bslbf
lmt_rdt_flag	1	bslbf
random_access_point	1	bslbf
time_limit_flag	1	bslbf
if (lls_flag == 1) {		
wakeup_control()	2	Section 7.2.2
} else {		
reserved	2	'00'
}		
signed_flag	1	bslbf
reserved	27	uimsbf
if (time_limit_flag == 1) {		
timestamp_min()	32	Table 6.3
}		
if (signed_flag == 1) {		
GMAC_Header_Extension()	var.	Table 6.5
}		
} else {		
reserved	16	'0x0000'
}		
}		

length – This unsigned, 16-bit value shall specify the number of bytes occupied by the Tunneled Packet immediately following the information header. The length is the total size of the Tunneled Packet, even if it is not fully contained within the current tunnel packet payload. Note that the length here is different than the IP header length contained within the Tunneled Packet since it also includes the length of any headers contained within the Tunneled Packet.

A value of zero (0) for the **length** shall indicate that the Tunneled Packet Information Header is truncated and that a Security Data Stream packet begins immediately after the **length** and the 16-bit **reserved** field filled with zeros. The Security Data Stream is consumed by the system receiving the ALPTP and is an RTP/UDP/IP packet containing the length so none of the information in the remainder of the header is needed.

alp_sid – This 8-bit unsigned integer field shall indicate a sub-stream identifier for the ALP packets carrying the multicast identified by the above four fields, **src_ip_add**, **dst_ip_add**, **src_udp_port** and **dst_udp_port**.

plp_id – This 6-bit unsigned integer field shall be used by the Scheduler to map the incoming Tunneled Packet to the corresponding PLP (defined by **L1D_plp_id** in A/322 [3]).

lls_flag – A 1-bit flag that shall indicate that the ALP packet payload contains at least one instance of a Low-Level Signaling table in compliance with [4]. A value of 1 shall indicate that the packet contains a Low-Level Signaling table. A value of 0 shall indicate that no LLS is present in the ALP payload.

lmt_rdt_flag – A 1-bit flag that shall indicate that the ALP packet payload contains at least one instance of a Link Mapping Table or a RoHC Description Table. A value of 1 shall indicate that the packet contains an LMT or RDT. A value of 0 shall indicate that no LMT or RDT is present in the ALP payload.

random_access_point – If set to 1, shall indicate that the associated inner packet contains some or all of the payload containing a random-access point (RAP). If set to 0, shall indicate that no random-access point is present within the packet. See Annex C for a discussion of random-access points and their meaning within the system. The **random_access_point** bit shall remain 1 for all of the packets containing the portion of the Stream comprising the random-access point. For example, for packets carrying a video Stream, all the packets containing the segment defined as the RAP shall have the **random_access_point** bit set to 1.

time_limit_flag – This flag shall be set to 1 to indicate that a **timestamp_min ()** field is present. A value of 0 shall indicate that no **timestamp_min ()** field is present and that the packet may be delivered with best effort.

wakeup_control () – A 2-bit field that communicates information needed to control the emission wakeup field. See Section 7.2.2 for a definition of this field. The **wakeup_control ()** field shall only be interpreted if the **lls_flag** is set to 1.

signed_flag – This flag shall be set to 1 to indicate that the following Tunneled Packet has been signed and a **GMAC_Header_Extension ()** field that resulted from that signing process is present. A value of 0 shall indicate that the packet was not signed and that no **GMAC_Header_Extension ()** field is present.

timestamp_min () – Defined in Table 6.3, this value shall indicate the latest time at which the start of the payload may be delivered.

GMAC_Header_Extension () – This field shall be defined according to Table 6.5 and its subsequent semantic definitions.

9 STL TRANSPORT PROTOCOL

The STL Transport Protocol (STLTP) shall extend the Common Tunneling Protocol (CTP) described in Section 6. The inner, Tunneled Data Streams shall carry sequences of packets comprising an RTP/UDP/IP header plus payload data. All packets delivered over the STL shall use an RTP/UDP/IP IPv4 protocol. The inner layer shall use Multicast Streams only, while the outer layer can use Multicast or Unicast consistent with CTP. See Annex D for more details of the Unicast use case.

9.1.1 Address Assignments

IPv4 packet format and addressing shall be used exclusively on the STL. The Multicast (destination) address range is 224.0.0.0 – 239.255.255.255. Of that range, 239.0.0.0 – 239.255.255.255 are for private addresses and shall be used for both inner Tunneled and outer Tunnel Packets when multicast is used.

When Unicast is used for outer Tunnel Data Packets, assignment of destination addresses and port numbers will depend on Internet Service Providers.

9.1.2 Port Assignments

Each Multicast destination address has usable port numbers from 1 through 65535. Values of 30000 – 30366 inclusive shall be used for this standard. Ports 30000 through 30063 shall be used to identify inner Tunneled Streams having destinations of PLPs 0 through 63, respectively; port 30064 shall be used to identify the inner Tunneled Stream carrying the Preamble data; port 30065 shall be used to identify the inner Tunneled Stream carrying the Timing and Management Data; and port 30066 shall be used to identify the inner Tunneled Stream carrying the Security Data Stream. All the listed Tunneled Packet Streams (i.e., Preamble, Timing and Management, Baseband Packets, and Security Data) required to populate an RF channel comprise a group of inner Tunneled Packet Streams.

The arrangement of UDP port numbers described above for an inner group of Tunneled Packet Streams can be offset in steps of 100 to permit addressing similar sets of Tunneled Packet groups intended for delivery to other Transmitters over the same STLTP. In such cases, an offset value for the port numbers for Transmitter P (where P is a value from 0 – 3) can be obtained from Offset [P] = P × 100. Alternatively, the number P can be used as the middle digit of a 5-digit number, with the first two digits being '30' and the last two digits ranging from '00' through '66'. (See Table 6.1 for the semantics of the **number_of_channels** field.) For example, in a case in which a Broadcast Gateway processes two channels, respectively A and B, the group of inner Tunneled Packet Streams of channel A would use the UDP port range from 30000 to 30066 (offset 0), while the group of inner Tunneled Packet Streams of channel B would use the UDP range 30100 to 30166 (offset 100). All these inner Tunneled Packet Streams then would be carried by the same Stream of Tunnel Packets. See Annex E for descriptions of the use of multichannel STLTP in Channel Bonding cases.

9.1.3 Majority Logic

Majority Logic is a technique for increasing the reliability of delivery of data to which it is applied. Delivery of the Preamble and of the Timing and Management data across the STL are required to maintain synchronization of Transmitters in a Network and possibly to avoid muting of individual Transmitters when there are interruptions of the delivery channels from a Broadcast Gateway to one or more Exciters. By permitting spreading out delivery of the critical data over time, Majority Logic can be useful in improving delivery of Preamble and of Timing and Management data when an STL experiences longer outages than can be corrected by the other reliability-improvement methods included in the STLTP. The amount of improvement that can be obtained with respect to relatively long outages increases with larger numbers of repetitions of the data and with longer time intervals between the repetitions of the data. The number of repetitions and the time between repetitions are separately configurable for the Preamble and the Timing and Management data.

When majority logic is applied to Preamble and/or T&M data, the objective is to spread the repeated information over the longest reasonable time so that disturbances in the STL delivery channel will be overcome through the redundant transmission of the primary synchronization information of the transmission system. The time spread of the repeated packets must be balanced against the latency added to the emission process by a greater spreading time. Therefore, successive redundant packets of Preamble or T&M Data, at a minimum, shall be carried both in separate inner Tunneled Packets and separate outer Tunnel Packets. Figure 9.1 shows an example of how multiple copies of Preamble and T&M can be sent to the Transmitter(s). As depicted in Figure 9.1, the repeated packets of Preamble and T&M may be sent over a duration of multiple Physical Layer frames prior to delivery of the Baseband Packets to be carried in the Physical Layer frames that they control.

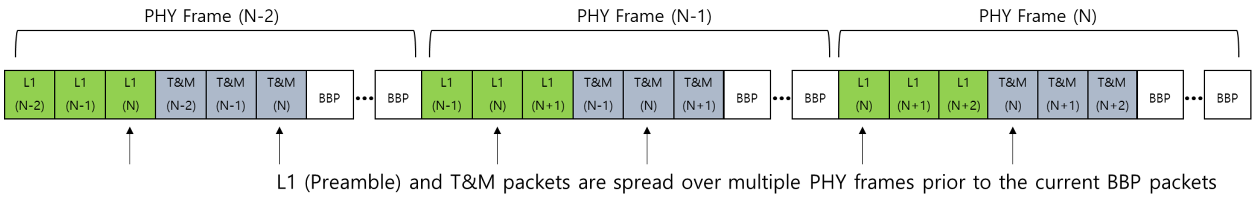


Figure 9.1 Example use of Majority Logic for Preamble (denoted as L1) and T&M: (when 3 repetitions of each are applied)

9.2 Preamble Data Generator

Preamble information is constructed in the Preamble Generator according to instructions from the Scheduler. It is output by the Preamble Generator in the form of RTP/UDP/IP Multicast packets, similar to those used to carry Baseband Packets (BBPs) in PLP Streams, so that they form a Stream that can be multiplexed together with the PLP Streams in the STLTP. The Preamble Stream carries a description of the configuration of the Transmitter processing functions and the resulting emitted waveform that is identical to the Preamble data structure sent to receivers.

To set up Transmitter configurations, the Preamble must be sent from the Scheduler to arrive at the Transmitter(s) at least one Physical Layer frame in advance of the start of construction by the Transmitter(s) of the frame that the Preamble describes.

To enable receivers to decode transmitted data, the same Preamble data used to configure the Transmitter(s) for a particular frame shall be carried in the emitted Preamble waveform immediately following the Bootstrap of that frame.

9.2.1 Preamble Data Stream Protocol

The Preamble data shall be delivered in an RTP/UDP/IP multicast Stream conforming to RFC 3550 [6] with the constraints defined below. The maximum Preamble data structure size can exceed the typical 1500-byte MTU, so a mechanism is defined herein to allow segmentation of the Preamble data across multiple RTP/UDP/IP packets. Note that such segmentation is required only to conform with typical MTU sizes of 1500 bytes. If the local Network allows larger multicast packets, this segmentation may not be needed.

The payload data for each Preamble Stream RTP/UDP/IP packet shall be a fragment of the Preamble Payload data structure described in Table 9.1. To provide validation that the **L1_Basic_signaling ()** and **L1_Detail_signaling ()** structures are delivered correctly over the STL, a 16-bit cyclic redundancy check (CRC) is provided. The CRC is applied to the combined contents of the **length** field, **L1_Basic_signaling()** and **L1_Detail_signaling ()** and is appended as the last 16 bits of the payload data. The resultant Stream of Preamble Payload packets shall have destination address 239.0.51.48 and destination port 30064 before application of channel number offset of the port number in the case of multichannel carriage within a single STL Tunnel Packet Stream.

Table 9.1 Preamble Payload

Syntax	No. of Bits	Format
Preamble_Payload () {		
length	16	uimsbf
L1_Basic_signaling ()	200	Table 9.2 of [3]
L1_Detail_signaling ()	var	Table 9.12 of [3]
crc16	16	uimsbf
}		

The following paragraphs describe the fields shown in Table 9.1.

The **length** field shall contain the number of bytes in the Preamble Payload data structure following the **length** field excluding the **crc16** bytes. The **length** field allows Data Consumers to avoid knowing the detailed syntax of Preamble data structures and still to reconstruct the overall Preamble Payload data structure.

The **crc16** field shall be the value resulting from application of the 16-bit cyclic redundancy check defined in [10], applied to the combined **length**, **L1_Basic_signaling ()** and **L1_Detail_signaling ()** structures in the Preamble Payload immediately preceding the **crc16** field.

The Preamble Data Generator shall form the necessary Preamble Payload data, as detailed in Table 9.1, from the Scheduler configuration and calculated information. Once the data structure has been populated, it shall be partitioned, if necessary, into multiple RTP/UDP/IP packets, each conforming, with the necessary headers, to the local Network MTU size. This process results in creation of a Preamble Payload Packet Set that typically consists of multiple packets of the same size followed by a smaller remainder packet. Constructing the packets in this way, however, is not normative.

The RTP header fields of the Preamble Payload Packet Set shall be as described below, configured with the **marker (M)** bit of the packet containing the beginning of a Preamble Payload data structure set to '1'. The **marker (M)** bits of remaining packets shall be set to '0'. This allows the Transmission system on the Data Consumer end of the STL to reconstruct the Preamble Payload data structure after any resequencing takes place. The timestamps of the packets of a given Preamble Payload Packet Set shall have the same values. The timestamp values are derived from a subset of the "**Bootstrap_Timing_Data ()**" appearing in Table 9.3, providing a mechanism to uniquely associate each of the Preamble Payload packets with a specific Physical Layer frame. The format of the timestamp field is described in Table 9.2.

Table 9.2 RTP Header Timestamp Field Definitions

Syntax	No. of Bits	Format
timestamp () {		
seconds_pre	22	uimsbf
a-milliseconds_pre	10	uimsbf
{		

seconds_pre shall carry a value equal to the 22 least significant bits (LSBs) of the **seconds** field of the **Bootstrap_Timing_Data ()**, described in Table 9.3.

a-milliseconds_pre shall carry a 10-bit value identical to the value contained in the 3rd through 12th MSBs of the **nanoseconds** field of the **Bootstrap_Timing_Data ()**, described in Table 9.3. Note that the **a-millisecond_pre** value is used in the RTP Header Timestamp only as an identifier of the Bootstrap Reference Emission Time of the Frame in which its contents belong; consequently, the somewhat longer Period of an **a-millisecond_pre** relative to precisely one millisecond is immaterial for this use.

On reception, the Preamble Payload Packet Set shall be resequenced, when necessary, and extracted into the Preamble Payload data structure as described in Table 9.1. The Data Consumer can accumulate RTP packets until it has received all the bytes defined by the length field in the first packet. If a packet is missed, as determined by a missing sequence number, or if a packet with the **marker (M)** bit set to '1' is received prematurely, indicating the start of the next Preamble Payload

Packet Set, then one or more packets have been lost and the entire Preamble Payload data set has been lost. Any accumulated data shall be discarded.

The RTP header fields shall follow the syntax defined in RFC 3550 [6], with the following additional constraints:

The **Padding (P)** bit shall conform to the RFC 3550 [6] specification.

The **Extension (X)** bit shall be set to '0' to indicate that the header contains no extension and that the packet has not been signed. The **Extension (X)** bit shall be set to '1' to indicate that a signing (or other) extension is present. Refer to Section 6.4 for details regarding packet signing and the definition of the STLTP signing extension.

The **CSRC Count (CC)** shall be set to '0', as no CSRC fields are necessary.

The **marker (M)** bit shall be set to '1' to indicate that the first byte of the payload is the start of the Preamble Payload data. A '0' value shall indicate that the payload is a continuation of the Preamble Payload data from the previous packet.

The **Payload Type (PT)** shall be set to 77 (0x4d), indicating the Preamble Payload type.

The **Sequence Number** shall conform to the RFC 3550 [6] specification.

The **Timestamp** shall be defined as in Table 9.2. The timestamp shall be set to the same value for all the Preamble Payload packet set.

The **Synchronization Source (SSRC) Identifier** shall be set to '0'. There shall be no other sources of Preamble Payload data carried within an STLTP Stream. Any redundant sources can be managed using IGMP Source-Specific Multicast (SSM) mechanisms.

If the Preamble Payload data packet is signed, the Header Extension of the Tunneled Packet (see Section 6.4.1) shall be placed immediately following the **SSRC** field and the **Extension (X)** bit shall be set to '1'.

9.3 Timing and Management Generator

Timing and Management (T&M) information is constructed in the Timing and Management Generator according to instructions from the Scheduler. It is output by the Timing and Management Generator in the form of RTP/UDP/IP multicast packets, similar to those used to carry Baseband Packets (BBPs) in PLP Streams, so that they form a Stream that can be multiplexed together with the PLP Streams in the STLTP. These Timing and Management packets are not transmitted over the air. The resulting Timing and Management Stream carries a set of instructions for controlling the emission of Physical Layer frames comprising a Bootstrap, Preamble, and Baseband Packets. Configurations of Bootstraps and certain other components of the Physical Layer frames are carried in the Timing and Management Stream. Also included in the Timing and Management Stream are the emission time of each Bootstrap and, hence, the start of each Physical Layer frame, the offset times of each Transmitter in an SFN from the Bootstrap Reference Emission Times for the Network, and other information used to control the Transmitter(s).

To set up Transmitter configurations, the Timing and Management Data for a Physical Layer frame must be sent from the Scheduler to arrive at the Transmitter(s) at least one Physical Layer frame in advance of the start of construction by the Transmitter(s) of the Physical Layer frame that it describes.

9.3.1 Timing and Management Data Stream Protocol

The Timing and Management Data shall be delivered in an RTP/UDP/IP multicast Stream conforming to RFC 3550 [6] with the constraints defined below. The maximum T&M data structure size may exceed the typical 1500-byte MTU, so a mechanism is defined herein to allow segmentation of the T&M data across multiple RTP/UDP/IP packets. Note that such segmentation is required only to conform with typical MTU sizes of 1500 bytes. If the local network allows larger multicast packets, this segmentation may not be needed.

The payload data for each T&M Stream RTP/UDP/IP packet shall be a fragment of the **TMP ()** data structure described in Table 9.3. To provide validation that the **TMP ()** structure is delivered correctly over the STL, a 16-bit cyclic redundancy check is provided as part of the **TMP ()** data. The resultant Stream of **TMP ()** packets shall have IP destination address 239.0.51.48 and destination port 30065, before application of channel number offset of the port number in the case of multichannel carriage within a single STL Tunnel Packet Stream.

The T&M Data Generator shall form the necessary **TMP ()** data structure, as detailed in Table 9.3, from the Scheduler configuration and calculated information. Once the data structure has been populated, it shall be partitioned, if necessary, into multiple RTP/UDP/IP packets, each conforming, with the necessary headers, to the local network MTU size. This process results in creation of a **TMP ()** packet set that typically consists of multiple packets of the same size followed by a smaller remainder packet. Constructing the packets in this way, however, is not normative.

The RTP header fields of the TMP packet set shall be as described below, configured with the **marker (M)** bit of the packet containing the beginning of a **TMP ()** data structure set to '1'. The **marker (M)** bits of the remaining packets shall be set to '0'. This allows the transmission system on the consumer end of the STL to reconstruct the **TMP ()** data structure after any resequencing takes place. The timestamps of the packets of a given **TMP ()** packet set shall have the same values. The timestamp values are derived from a subset of the **Bootstrap_Timing_Data ()**, providing a mechanism to uniquely associate each of the **TMP ()** packets with a specific Physical Layer frame.

The RTP header fields shall follow the syntax defined in RFC 3550 [6] with the following additional constraints:

The **Padding (P)** bit shall be set to '0', indicating no padding is present in the Timing and Management Data packet.

The **Extension (X)** bit shall be set to '0' to indicate that the header contains no extension and that the packet has not been signed. The **Extension (X)** bit shall be set to '1' to indicate that a signing (or other) extension is present. Refer to Section 6.4 for details regarding packet signing and the definition of the STLTP signing extension.

The **CSRC Count (CC)** shall be set to '0', as no **CSRC** fields are necessary.

The **marker (M)** bit shall be set to '1' to indicate that the first byte of the payload is the start of the TMP data. A '0' value shall indicate that the payload is a continuation of the TMP data from the previous packet.

The **Payload Type (PT)** shall be set to 76 (0x4c) indicating the Timing and Management Data payload type.

The **Sequence Number** shall conform to the RFC 3550 [6] specification.

The **Timestamp** shall be defined as in Table 9.2.

The **Synchronization Source (SSRC) Identifier** shall be set to '0'. There should be no other sources of Timing and Management Data carried by the STLTP. Any redundant sources can be managed using IGMP Source-Specific Multicast (SSM) mechanisms.

If the TMP packet is signed, the Header Extension of the Tunneled Packet (see Section 6.4.1) shall be placed immediately following the **SSRC** field and the **Extension (X)** bit shall be set to '1'.

Table 9.3 Timing and Management Stream Packet Payload

Syntax	No. of Bits	Format
Timing and Management_Packet (TMP) () {		
Structure_Data () {		
length	16	uimsbf
version_major	4	uimsbf
version_minor	4	uimsbf
maj_log_rep_cnt_pre	4	uimsbf
maj_log_rep_cnt_tim	4	uimsbf
bootstrap_major	4	uimsbf
bootstrap_minor	4	uimsbf
min_time_to_next	5	uimsbf
system_bandwidth	2	uimsbf
bsr_coefficient	7	uimsbf
preamble_structure	8	uimsbf
ea_wakeup	2	bslbf
num_emission_tim	6	uimsbf
num_xmtrs_in_group_minus_1	6	uimsbf
xmtr_group_num	7	uimsbf
maj_log_override	3	bslbf
num_miso_filt_codes	2	bslbf
tx_carrier_offset	2	tcimsbf
reserved	6	for (i=0; i<6; i++) '1'
}		
Bootstrap_Timing_Data () {		
for (i=0; i<=num_emission_tim; i++)		
seconds	32	uimsbf
nanoseconds	32	uimsbf
}		
}		
Per_Transmitter_Data () {		
for (i=0; i<=num_xmtrs_in_group_minus_1; i++) {		
xmtr_id	13	uimsbf
tx_time_offset	16	tcimsbf
txid_injection_lvl	4	uimsbf
miso_filt_code_index	2	bslbf
reserved	29	for (i=0; i<29; i++) '1'
}		
}		
Packet_Release_Time () {		
pkt_rls_seconds	4	uimsbf
pkt_rls_a-milliseconds	10	uimsbf
reserved	2	'11'
}		
Error_Check_Data () {		
crc16	16	uimsbf
}		
}		

length shall indicate the number of bytes in the Timing and Management Data packet following the RTP/UDP/IP header structure. Up to 65,535 Bytes can be indicated.

version_major, in conjunction with **version_minor**, shall indicate the version of the protocol used to construct the Timing and Management Data packet. Increments in the value of **version_major** are intended to indicate changes in the structure that are not fully compatible with lower-ordered **version_major** values. The value of **version_major** can range from 0 through 15. Timing and Management packets constructed according to this version of this standard shall have the value of **version_major** set to 0.

version_minor, in conjunction with **version_major**, shall indicate the version of the protocol used to construct the Timing and Management Data packet. Increments in the value of **version_minor** are intended to indicate changes in the structure that are fully backward compatible with lower-ordered **version_minor** values paired with the same **version_major** value. The value of **version_minor** can range from 0 through 15. Timing and Management packets constructed according to this version of this standard shall have the value of **version_minor** set to 0.

maj_log_rep_cnt_pre shall indicate the number of repetitions of Preamble data in the Preamble Stream at UDP port 30064 prior to emission of the Preamble. Permitted values shall be 1, 3, 5, 7, and 9. Note that the value of **L1B_lls_flag** may be correct only in the final copy of the Preamble data sent to Transmitters prior to emission. Consequently, majority logic error correction can be applied reliably to all portions of the Preamble Stream data except the flag value noted. See Section 10.2 for details of placement of the repeated data.

maj_log_rep_cnt_tim shall indicate the number of repetitions of Timing and Management data in the Timing and Management Stream at UDP port 30065 prior to emission of the next Bootstrap. Permitted values shall be 1, 3, 5, 7, and 9. Note that values for the **ea_wakeup** bits may be correct only in the final copy of the Timing and Management data sent to Transmitters prior to emission. Consequently, majority logic error correction can be applied reliably to all portions of the Timing and Management Stream data except the **ea_wakeup** values noted. See Section 10.1 for details of placement of the repeated data.

bootstrap_major shall indicate the value of the **bootstrap_major_version** of the Bootstrap symbols that introduce the Physical Layer frame identified by the **Bootstrap_Timing_Data ()**, which value shall be applied as the root of the Zadoff-Chu sequence of the Bootstrap symbols, as specified in [2].

bootstrap_minor shall indicate the value of the **bootstrap_minor_version** of the Bootstrap symbols that introduce the Physical Layer frame identified by the **Bootstrap_Timing_Data ()**, which value shall be applied as the seed for the pseudo-noise (PN) sequence of the Bootstrap symbols, as defined in [2].

min_time_to_next shall be the enumerated value indicating the minimum time until the next frame of the same type as defined in [2].

system_bandwidth shall be the enumerated value indicating the bandwidth of the RF Transmission channel as defined in [2].

bsr_coefficient shall be the binary value associated with the baseband sampling rate as defined in [2].

preamble_structure shall be the enumerated value indicating the Preamble configuration as defined in [3].

num_emission_tim shall indicate the number of sequential Bootstrap Reference Emission Times that are contained within the **Bootstrap_Timing_Data ()** 'for' loop. Up to 64 values may be indicated. Allowable values shall range from 0 thru 63 and shall be expressed as the number of values carried in the packet minus 1. At least the next Bootstrap Reference Emission Time shall be carried, and it shall be carried in index 0 of the 'for' loop.

ea_wakeup shall signal the states of the two EA Wakeup Bits to be included in the Bootstrap signal at the start of the next frame to be emitted.

num_xmtrs_in_group_minus_1 shall indicate the number of Transmitters minus one to which data is addressed in the **Per_Transmitter_Data ()** 'for' loop (i.e., 1 to 64 Transmitters are indexed 0 to 63). The value can be less than the total number of Transmitters in the Network, in which case data addressed to groups of Transmitters shall be sequenced in order across multiple Timing and Management Data packets.

xmtr_group_num shall indicate the ordinal number of a group of Transmitters to which information in the **Per_Transmitter_Data ()** loop is addressed. The value of the field can range from 0 through 127. Only a single value of **xmtr_group_num** shall apply to a given Timing and Management Stream data packet. Information for individual Transmitters shall be organized in groups identified by values of **xmtr_group_num** starting at 0 and incrementing by 1 from one Timing and Management Stream data packet to the next, until the highest-numbered group is reached, at which point the value shall start again at 0 in the following such packet.

Bootstrap_Timing_Data () shall contain a list of the Bootstrap Reference Emission Times of the next and, optionally, successive future frames, the list having a total number of entries equaling the value of **num_emission_tim**. The values of the Bootstrap Reference Emission Times shall strictly increase from the first entry in the list to the last.

maj_log_override shall indicate that all previous instances of Timing and Management Data and Preamble data for the next and following Physical Layer frames shall be ignored and that the information in the current Timing and Management packet and a subsequent Preamble data packet shall be used to configure the next Physical Layer frame. The non-override condition shall be indicated by a value of '000' in this field. An override condition shall be indicated by a value of '111' in this field.

num_miso_filt_codes shall be set to one less than the number of different MISO filter codes in use within an SFN, as represented by the variable '$\boldsymbol{N_{TX}}$' in Annex J of A/322. [3] For example, when $\boldsymbol{N_{TX}}$=2 is used, **num_miso_filt_codes** would be set equal to 1. The value '0' shall be reserved for future use.

tx_carrier_offset shall indicate the carrier offset of the Transmitter(s) in the frequency domain. The carrier offset shall be expressed in units of a positive or negative integer number of carriers, and it shall be a two's complement signed integer binary number having a range from –1 to +1 decimal, representing from –1 to +1 OFDM carriers. The carrier offset value shall be equal to the product of the value of **tx_carrier_offset** and the carrier frequency spacing in Hz of an 8K FFT for the value of **bsr_coefficient** and **system_bandwidth** indicated in the **Structure_Data ()** for the same frame. Carrier frequency spacing (in Hz) equals BSR (in Hz) divided by 8192. For example, in a system operating with a 6 MHz channel bandwidth and a BSR of 6.192 Mega-samples/second, **bsr_coefficient** = 2, the carrier frequency spacing of 8K carriers is 843.75 Hz, and the carrier frequency offset will be -843.75 Hz, 0 Hz, and +843.75 Hz for values of **tx_carrier_offset** of -1, 0, and +1, respectively. **tx_carrier_offset** = -2 shall be reserved for future use.

The **tx_carrier_offset** value also is used by the Scheduler to set Bootstrap Reference Emission Times as described in Section 10.3.3.2 below.

seconds shall carry a value equal to the 32 least significant bits (LSBs) of the seconds portion of the TAI time value [15] of the associated Bootstrap Reference Emission Time, as expressed using the Precision Time Protocol (PTP) defined in [11] and [12].

nanoseconds shall carry a value equal to the nanoseconds portion of the TAI time value [15] of the associated Bootstrap Reference Emission Time. It shall be expressed as a 32-bit binary value having a range from 0 through 999,999,999 decimal.

Per_Transmitter_Data () shall contain information addressed individually to one or a group of Transmitters, with the number of Transmitters for which data is included in the loop equaling the value in **num_xmtrs_in_group_minus_1** plus 1**.**

xmit_id shall indicate the address of the Transmitter to which the following values are being sent and shall correspond to the seed value used by the TxID code sequence generator of that Transmitter. The value of the address shall be an unsigned integer binary number having a range of possible values from 0 through 8191 decimal.

tx_time_offset shall indicate the emission time offset of the Transmitter to which it is addressed relative to the Bootstrap Reference Emission Times of all frames. The Transmitter time offset shall be expressed in units of positive or negative integer steps of 100 ns and shall be a two's complement signed integer binary number having a range from –32,768 through +32,767 decimal, representing time offsets from –3,276.8 through +3,276.7 microseconds.

txid_injection_lvl shall indicate the Injection Level of the TxID signal below the average power of the Preamble symbols emitted by the Transmitter to which its value is addressed. The Injection Level shall indicate the value in dB listed in A/322 [3] Table N.3.1 for the TxID Injection Level Code included in the **txid_injection_lvl** field (or Off for code value '0000').

miso_filt_code_index shall be set to one less than the specific MISO filter code assigned to the individual Transmitter, as represented by the variable 'h' in A/322 [3] Annex J. For example, when h=1 is used, **miso_filt_code_index** would be set equal to 0. The same value of MISO filter code index shall apply to a particular Transmitter regardless of whether 64-coefficient or 256-coefficient filters are in use.

pkt_rls_seconds shall be the seconds portion of the time of release from the Broadcast Gateway of the specific Timing and Management packet in which the value is found. Its value shall be expressed as 4 bits representing the 4 LSBs of the seconds value of the TAI time [15] when the first bit of the IP header of the T&M packets is released from the Broadcast Gateway.

pkt_rls_a-milliseconds shall be the milliseconds portion of the time of release from the Broadcast Gateway of the specific Timing and Management packet in which the value is found. Its value shall be expressed as 10 bits representing the 3rd through 12th MSBs of the nanoseconds value of the TAI time [15] when the first bit of the IP header of the T&M packets is released from the Broadcast Gateway. Its range will be from 0 to 953 (decimal) as a consequence of the Period of an **a-millisecond** being slightly longer than precisely a millisecond. (See the definition of an **a-millisecond** in Section 3.4.)

crc16 shall be the value resulting from application of the 16-bit cyclic redundancy check defined in [10], applied to all fields in the Timing and Management Packet payload from the **length** field through the field (and any reserved bits) immediately preceding the **crc16** field.

9.3.2 Bootstrap Emission Timing and Frame Identification

Bootstrap Reference Emission Time shall be used for Physical Layer frame identification. In the RTP headers of the STLTP, the timestamp fields are used to carry number patterns matching portions of a complete Bootstrap Reference Emission Time. The portions to be matched have been selected for unique identification of the Physical Layer frames with which the STLTP packets are associated. The portions of the Bootstrap Reference Emission Time used for identification are the 22 LSBs of the seconds value and the 3rd through 12th MSBs of the nanoseconds value, the latter of which represent approximately millisecond time increments. The selected portions of the Bootstrap Reference Emission Time are sufficient to assure no ambiguity in the association of STLTP packets with Physical Layer frames.

9.3.2.1 Baseband Packet Delivery

The Packetizer functionality accepts ALP packets from the transport layer as described in [5]. All ALP packets are converted to Baseband Packets as described in [3]. The result of this process is a complete description of the baseband data to be emitted within a specific PLP within an identified Physical Layer frame. The Baseband Packetizer then shall encapsulate and segment Baseband Packets into RTP/UDP/IP packets as per Scheduler instructions appropriate to the configuration of the associated destination PLP and Physical Layer frame.

9.3.3 PLP Data Stream

Each PLP carries parallel data, simultaneous in time and inherently packetized into Baseband Packets. The payload identity must be signaled for correct PLP processing. The port number used to deliver each ALP Stream, as described in Section 6, associates the ALP Stream with a particular PLP number.

9.3.4 Baseband Packet Data Stream Protocol

The Baseband Packet data shall be delivered in an RTP/UDP/IP multicast Stream conforming to RFC 3550 [6] with the constraints defined below. The Baseband Packet data structure size can exceed the typical 1500-byte MTU, so a mechanism is defined herein to segment the Baseband Packet data across multiple RTP/UDP/IP multicast packets. Note that this is required only to conform with typical MTU sizes of 1500 bytes. If the local Network allows larger multicast packets, this segmentation may not be required.

The payload data for each RTP/UDP/IP multicast packet shall be a fragment of the Baseband Packet data structure as defined in [3]. The collection of segmented packets representing a single Baseband Packet is referred to hereafter as the Baseband Packetizer Packet Set (BPPS). The resultant packet Stream shall have IP destination address 239.0.51.48 and IP destination ports 30000 through 30063, corresponding to the PLPs numbered from 0 through 63, respectively, before application of channel number offset of the port number in the case of multichannel carriage within a single STL Tunnel Packet Stream.

The Packetizer, in each path from the corresponding ALP Stream to the target PLP Stream, will form each Baseband Packet from ALP packets in the incoming ALP data Stream as described above. Once a Baseband Packet is ready for emission, it will be partitioned, if necessary, into multiple RTP/UDP/IP multicast packets each conforming, with the necessary headers, to the local Network MTU size. In summary, the resultant BPPS typically will consist of multiple packets of the same size followed by a smaller remainder packet. Constructing the packets in this way is not normative and any size or number of packets is permitted within a BPPS.

The RTP header fields of the BPPS shall be set as described below, with the **marker (M)** bit of the packet containing the beginning of the BPPS set to '1'. The **marker (M)** bit of the remaining packets shall be set to '0'. This allows the Transmission system on the Data Consumer side of the STL to reconstruct the Baseband Packet after any resequencing takes place. The timestamps of the packets of the BPPS shall be set to the same value indicating the Bootstrap Reference Emission Time of the Physical Layer frame. In addition to the **marker (M)** bit being set to '1' on the first BPPS packet, the **Synchronization Source (SSRC) Identifier** field of the first BPPS packet shall be set to the overall length of the Baseband Packet data structure in bytes across the entire BPPS for a given PLP.

On receipt at the Data Consumer, the BPPS shall be resequenced and extracted into the Baseband Packet data structure. The Data Consumer can accumulate RTP packets until it has received all of the bytes defined by the length field in the first BPPS packet. If a BPPS packet is missed as determined by a missing sequence number or if a packet with the **marker (M)** bit set to '1' is received prematurely, indicating the start of the next BPPS, then one or more packets have been lost and the entire Baseband Packet data structure has been lost. Any accumulated data shall be discarded.

Note that, depending on time interleaver modes, a Baseband Packet at the current timestamp may span into the subsequent Physical Layer frame due to the delaying nature of time interleaving. In such a case, the Baseband Packet at the current timestamp would belong to both the current and the subsequent Physical Layer frames.

The RTP header fields shall follow the syntax defined in RFC 3550 [6] with the following additional constraints:

The **Padding (P)** bit shall conform to the RFC 3550 [6] specification.

The **Extension (X)** bit shall be set to '0' to indicate that the header contains no extension and that the packet has not been signed. The **Extension (X)** bit shall be set to '1' to indicate that a signing (or other) extension is present. Refer to Section 6.4 for details regarding packet signing and the definition of the STLTP signing extension.

The **CSRC Count (CC)** shall be set to '0' as no CSRC fields are necessary.

The **marker (M)** bit shall be set to '1' to indicate that the first byte of the payload is the start of a Baseband Packet data structure. A value of '0' shall indicate that the payload is a continuation of the Baseband Packet data structure from the previous BPPS packet.

The **Payload Type (PT)** shall be set to 78 (0x4e) indicting the Baseband Packet payload type.

The **Sequence Number** shall conform to the RFC 3550 [6] specification.

The **Timestamp** shall be defined as in Table 9.2. The **Timestamp** shall be set to the same value for all packets of a given BPPS.

When the **marker (M)** bit is '0', the **Synchronization Source (SSRC) Identifier** shall be set to '0'. When the **marker (M)** bit is set to '1', indicating the first packet of the BPPS, the **SSRC** field shall contain the total length of the Baseband Packet data structure in bytes. This allows the Data Consumer to know how much data is to be delivered within the payloads of the BPPS.

If the Baseband Packet is signed, the Header Extension of the Tunneled Packet (see Section 6.4.1) shall be placed immediately following the **SSRC** field and the **Extension (X)** bit shall be set to '1'.

9.4 Studio to Transmitter Link (STL) Transport Protocol (STLTP)

The STL Transport Protocol (STLTP) shall be an RTP/UDP/IP Stream based on the Common Tunneling Protocol defined in Section 6 of this standard.

9.4.1 STL Transport Protocol Design

Figure 9.2 provides a detailed diagram of the portion of the broadcast PHY layer chain described by this section. Refer to Figure 4.2 for a complete diagram of the system architecture.

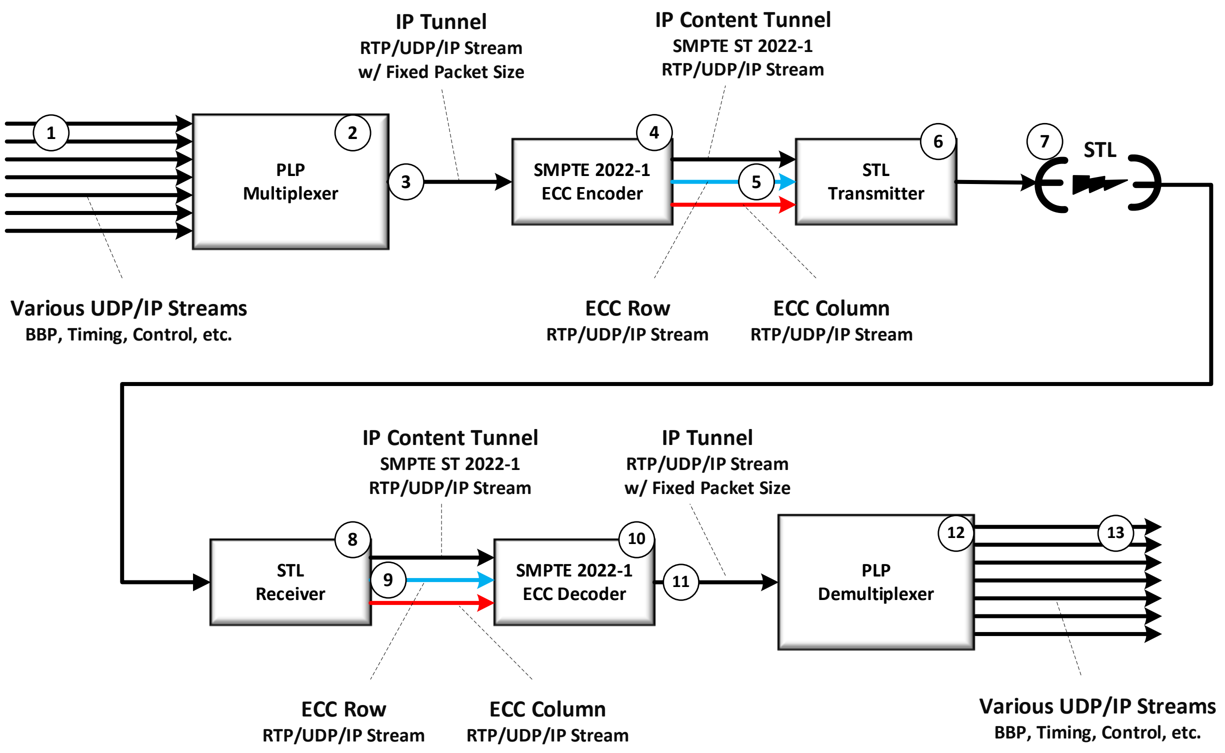


Figure 9.2 STL Transmission diagram

The following paragraphs describe each of the call-outs, (1) through (13), in Figure 9.2. These are an extension of the figure shown in the Common Tunneling Protocol specifically for the STL use case. Steps (1) through (5) and (9) through (13) correspond to the steps described in Figure 6.1 CTP Tunnel Encapsulation Process.

1) The multiple paths represent the RTP/UDP/IP Streams that are generated for each PLP as described in Sections 9.2, 9.3, and 9.3.4. These are referred to as the Tunneled Packet Streams.
2) The PLP Mux is configured to accept packets from multiple RTP/UDP/IP multicast Streams to be tunneled.
3) The Tunneled Packets are grouped into fixed-size payloads to accommodate the SMPTE ST 2022-1 ECC process [8]. See the CTP Section 6.3.2 for more details.
4) The process described in [8] buffers multiple fixed-sized payloads and performs XOR operations on groups of Tunnel Packets producing additional ECC packets.
5) The Tunnel Stream and up to two ECC packet Streams are sent to the STL Transmitter using up to three RTP/UDP/IP on three separate ports.
6) The three Streams are processed by an IP-capable STL Transmitter where they are
7) sent via the STL to
8) the STL Data Consumer where they are extracted into
9) up to three RTP/UDP/IP Streams; i.e., the Tunnel Packet Stream and up to two ECC Streams.

10) These Streams are processed by the SMPTE ST 2022-1 decoder. Note that the decoder detects missing packets from the RTP sequence numbers on the Tunnel Stream and reconstitutes them from the ECC Stream(s).
11) The resultant Tunnel Packet Stream is an in-order collection of fixed-size packets identical to the input Stream described in step (3) above.
12) The PLP Demux extracts each packet Stream from the payloads of the Tunnel Packets. The Tunnel Packet RTP headers contain information allowing the packet Streams to be reframed according to the CTP Section 6.3.1.
13) The reconstituted IP packets then are forwarded to the remaining stages of the Transmission system.

9.4.2 RTP Encapsulation Example

The following example shows how encapsulation is done using RTP to create a complete Tunneled Packet Stream for SMPTE ST 2022-1 ECC processing, specifically for the STLTP case. Please refer to the Common Tunneling Protocol (CTP) Section 6.3.1 for an example of the general encapsulation process. It examines the case of a Single-PLP emission, specifically, the details of the packing process, including the locations of the several headers of the different layered protocols within the Stream (in Figure 9.3).

Figure 9.3 shows the details of the encapsulated Tunnel Packet. In the top row of Figure 9.3 are packets from three separate Tunneled Packet Streams. In the center row of Figure 9.3 are the details of the construction of the Tunnel Packet that contains the Tunneled Packets or segments of Tunneled Packets from the top row. As shown by the bottom row, the middle row is a more detailed version of the Tunnel Packet structure shown within the dashed rectangle in CTP Figure 6.2.

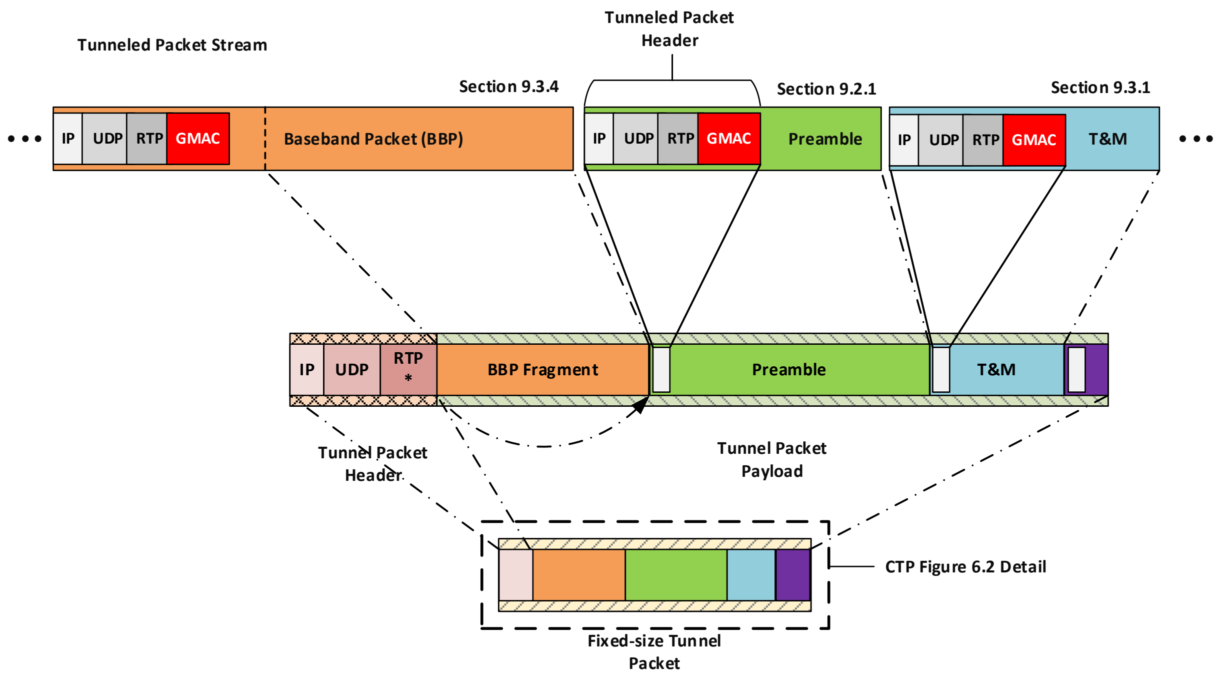


Figure 9.3 Tunneled Packet packing details

Starting from the top row of Figure 9.3, each of the three Tunneled Packet Streams has its own procession of packets, complete with IP/UDP/RTP header structures, including the Header

Extension containing the GMAC Tag resulting from the packet signing process (see Section 6.4). Either complete packets or segments of packets from the Tunneled Packet Streams are multiplexed together on the second row to form the payload of a Tunnel Packet, and another header is prepended to the payload to form a completely new packet. As shown in the bottom row, that packet is the one in a dashed rectangle shown in Figure 6.2 in CTP Section 6.3.1.

Examining the middle row of Figure 9.3 in more detail, it can be seen that there are four IP/UDP/RTP header sequences. The first, on the left, is the header sequence for the Tunnel Packet. The second header sequence is that of the complete Preamble packet, shown in green, that is fully encapsulated within the Tunnel Packet. The third header sequence is that of the complete T&M packet, shown in cyan, that is fully encapsulated within the Tunnel Packet. The fourth header sequence leads into an initial segment of a Baseband Packet, shown in purple. Also included in the packet described in the middle row is a segment of a Baseband Packet, shown in orange, that does not have its own header because it is a continuation from an earlier Tunnel Packet that contained the header sequence for that Tunneled Packet. Note that the packet headers from the Tunneled Packets may include a Header Extension that contains the GMAC Tag resulting from the Tunneled Packet signing process.

In the sequence of payload packets and segments described within the payload of the Tunnel Packet, there is a pointer from the Tunnel Packet RTP header to the start of the first byte of the header sequence (i.e., the IP header) that introduces the Preamble Packet. Since the header of the Preamble Packet is the first packet to appear within the Tunnel Packet, the Tunnel Packet RTP header pointer points there. Since the RTP header pointer did not point to the first byte following itself, the implication is that the first data segment within the Tunnel Packet payload is a segment of a Tunneled Packet that was not completed in an earlier Tunnel Packet. A Data Consumer of the STLTP Stream, upon examination of the Tunnel Packet RTP header, can determine both that the first data within the Tunnel Packet payload is a segment of a previously incomplete packet and where the next Tunneled Packet begins within the Tunnel Packet. By examining the length value contained within the header sequence of that next Tunneled Packet, the Data Consumer can locate the next header within the Tunnel Packet payload. This process can continue until the Data Consumer finds a Tunneled Packet header that indicates its length is longer than the space remaining available in the Tunnel Packet, implying that there will be at least a segment of the final Tunneled Packet contained within the next Tunnel Packet having the same port address (i.e., going to the same PLP) as the current Tunnel Packet.

9.5 Channel Bonding in Broadcast Gateways

When Channel Bonding is used, as specified in Section K of [3], a Broadcast Gateway shall include a Stream partitioning block that allocates Baseband Packets of Channel-Bonded PLP(s) to two RF channels, as depicted in Figure 9.4. For Channel Bonding, STLTP packet Stream generation from the Broadcast Gateway shall be enabled in one of two operation modes, which can be selected by setting **number_of_channels** as described in Table 6.1. When **number_of_channels** is set to '0', a Broadcast Gateway shall generate two outer Tunnel Data Streams, each of which shall carry one group of inner Tunneled Packet Streams corresponding to one RF channel, as depicted in Figure 9.4 (a). A group of inner Tunneled Packet Streams comprises the Preamble, Timing and Management, Baseband Packet, and Security Data Streams required for one RF channel. When **number_of_channels** is set to '1', a Broadcast Gateway shall generate one outer Tunnel Data Stream that carries two groups of inner Tunneled Packet Streams, one each for the two RF channels, as depicted in Figure 9.4 (b). When **number_of_channels** is set to '1', the two groups of inner Tunneled

Packet Streams shall be identified by different port assignments, such that one inner Tunneled Packet Stream group appears on ports 30000 – 30066, and the other group appears on ports 30100 – 30166. These two STLTP configurations are applicable to all the Channel Bonding modes defined in [3] (i.e., plain Channel Bonding and Channel Bonding with SNR averaging). Further details with examples of different Channel Bonding modes are described in Annex E.

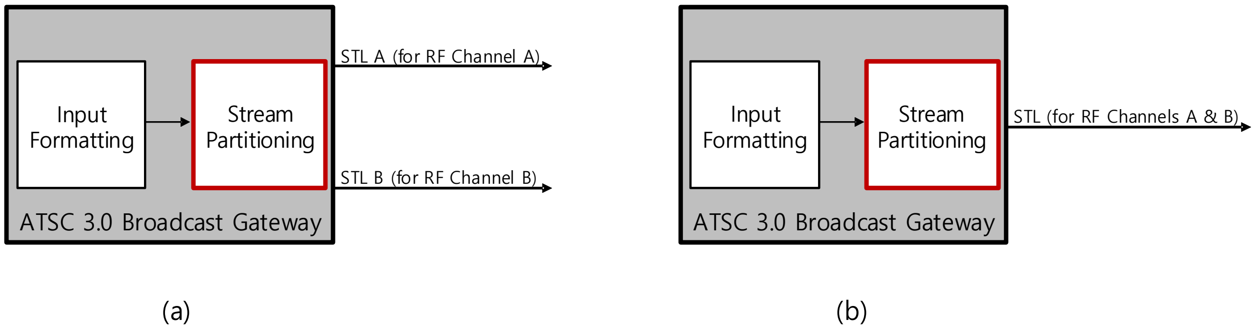


Figure 9.4 Channel Bonding operation modes in a Broadcast Gateway: (a) Sending two outer Tunnel Data Streams, and (b) Sending one outer Tunnel Data Stream that carries two groups of inner Tunneled Packet Streams

10 TRANSMITTER OPERATION NORMATIVE REQUIREMENTS

There are several requirements placed on Transmitter(s) that facilitate their use of the data arriving over the STL from a Broadcast Gateway and that enable operations of Transmitters in SFNs. The requirements relate to timing of emissions, buffer management, frequency control, offsetting of carrier frequencies for interference mitigation, and the like.

10.1 Timing Manager

A Timing Manager is required functionality in a Transmitter to permit use of the Timing and Management Data sent to Transmitters by a Broadcast Gateway. The Timing Manager shall receive, buffer, and process T&M Data arriving in the form described in Section 9.3 from the STL PLP Demux, as shown in Figure 4.2 (heavy, light-orange line). T&M Data must arrive at a Transmitter in advance of the time for its application. This data shall control the overall operation of the Transmitter and the times at which the numerous processes in the Transmitter are executed. The T&M Data that the Transmitter receives is not intended for emission but rather for control purposes only.

The Timing Manager also shall receive time information directly from a highly accurate source such as the Global Navigation Satellite System (GNSS) or communicated through a data Stream capable of precise time information transfer such as the Precision Time Protocol (PTP). The time information shall be used to accurately control the emission time of the leading edge of the first Bootstrap symbol of each Physical Layer frame. The same timing information used for control of emission timing also shall be capable of use for derivation of a precise frequency reference for control of the Transmitter emission frequency.

The T&M Data shall be stored in a buffer to permit Transmission of Bootstrap Reference Emission Times by the Scheduler to the Transmitter(s) well in advance of those emission times. The size of the buffer should be determined from the maximum advance of delivery time of the Bootstrap Reference Emission Time information permitted by the specification of the Timing and Management Data Stream protocol, as described in Section 9.3.1.

The primary data carried by the Timing and Management Stream is the Bootstrap Reference Emission Time at which each Physical Layer frame is to be emitted. Multiple instances of the Timing and Management Data for a given frame may be sent to Transmitter(s) by the Scheduler. Those instances will be identified as to the Physical Layer frame to which they apply by having a matching time value indicating the Bootstrap Reference Emission Time of the frame. The multiple instances can be held in a buffer and used by the Timing Manager to improve reliability of the T&M Data through application of majority logic or other processing of the redundant copies of the data.

The number of copies of the T&M Data that are sent by the Scheduler to the Transmitter(s) for each Physical Layer frame is indicated in the Timing and Management Data Stream itself, as specified in Table 9.3. Similarly, the number of copies of the Preamble data that are sent by the Scheduler to the Transmitter(s) for each Physical Layer frame also is indicated in the Timing and Management Data Stream. The number of Preamble copies sent by the Scheduler shall be communicated by the Timing Manager to the Preamble Parser so that it can utilize similar processing of the redundant data for purposes of improving reliability of the data.

Other functions of the Timing Manager are:

- Control of the configuration of Bootstrap symbols, waveforms, and the data carried
- Control of the emission times of Bootstrap symbols
- Compensation of the Transmitter-to-antenna emission delay (TAD)
- Control of buffer delays in the Transmitter data processing path
    - Including compensation for STL Network delivery delays in SFNs
- Control of Preamble data insertion following Bootstrap emission
- Parsing of Timing and Management Data for individual Transmitters in SFNs
- Control of Transmitter emission time offsets in SFN operation
- Control of TxID insertion
    - Including insertion level and off

10.2 Preamble Parser

A Preamble Parser is required functionality in a Transmitter to permit use of Preamble data from a Scheduler both to set up the modulation, coding, and other processes in the Exciter and to communicate to receivers the configuration of the signals emitted by the Transmitter. The Preamble Parser shall receive from the STL PLP Demux Preamble data in the form described in Section 9.2, as shown in Figure 4.2 (heavy, light-blue line) and shall buffer that data. Preamble data will arrive at a Transmitter in advance of the time for its emission. The Preamble data shall be used in two ways: (1) to set up the Transmitter data and signal processing, as controlled by the Scheduler through the delivered Preamble data, and (2) to deliver to the Preamble Inserter block the Preamble data Stream at the correct time for its emission in the transmitted signal.

To achieve the two required uses of the Preamble data, the Preamble Parser shall provide two primary functions. The first function shall be a buffer that holds the Preamble data Stream until it is needed for emission. The output of the buffer shall release the Preamble data Stream to the Preamble Inserter (thin, light-violet line in Figure 4.2) at the appropriate time, as determined by the Timing Manager, so that emission of the Preamble data occurs immediately following emission of the Bootstrap. Both the **length** data at the start and the **crc16** value at the end of the Preamble Payload data packet shall not be emitted by the Transmitter and can be deleted after any use by the Transmitter for error detection. The second function shall be parsing of the data in the Preamble into separate instructions for each of the data- and signal-processing stages of the Transmitter and delivery of those instructions to the processing stages at the appropriate times, as determined by the Timing Manager (thick, dark-blue line in Figure 4.2).

Advance arrival of the Preamble data at the Preamble Parser is for the purpose of providing time for the Exciter to set up its processing stages to the configuration specified by the Scheduler for the waveform, in advance of processing of the next Physical Layer frame. During that setup Period, the Preamble data Stream itself is held in the buffer until it is needed for emission in the appropriate frame.

Multiple instances of the Preamble data for a given frame may be sent to Transmitter(s) by the Scheduler. Those instances will be identified as to the Physical Layer frame to which they belong with a time value matching the Bootstrap Reference Emission Time of the frame. The multiple instances can be used by the Preamble Parser to improve reliability of the Preamble data through application of majority logic or other processing of the redundant copies of the data. The number of copies of the Preamble that are sent by the Scheduler to the Transmitter(s) is indicated in the T&M Data Stream that goes to the Timing Manager. The Timing Manager is expected to communicate that information to the Preamble Parser (thin, red line in Figure 4.2) when it is intended to process the redundant data for improved data reliability.

10.3 Other Transmitter Requirements

To enable use of Transmitters in Single Frequency Networks (SFNs) and to minimize interference between stations, certain additional requirements not covered elsewhere in this standard are placed on Transmitters. Those requirements relate to frequency accuracy, timing offsets for network management, and co-channel interference minimization through application of frequency and timing offsets.

10.3.1 Frequency Accuracy

The Center Frequency of the transmitted signal shall be either the frequency of the middle carrier in the signal's spectrum when an odd number of carriers is in use or shall be the frequency half-way between the two middle carriers when an even number of carriers is in use.[2] To enable both SFN operation and the co-channel interference mitigation methods discussed in Section 10.3.3, the Transmitter Center Frequency shall be maintained at the nominal Center Frequency ±0.5Hz, with a long-term-averaged error of zero. Use of GNSS as a time base and frequency reference can enable such precision.

10.3.2 Transmitter Timing Offsets

To cause Transmitters in a Network to emit their waveforms at times needed for Network shaping, offsets may be applied to emission time values individually at each Transmitter. A Transmitter timing reference with high precision, low jitter, and zero long-term drift can be derived from a single reference and may be delivered to each site or derived at each site. Each Transmitter must be locked to a source equal to the GNSS Time interval. It can be GPS time or TAI, but it must be independent of leap seconds.

A frequency reference with high precision, low jitter and zero long-term drift can be derived from the timing reference. GPS-like sources can be available simultaneously at multiple locations at the studio and Transmitter to enable such an accurate reference.

[2] In the current version of A/322 [3], all possible ATSC 3 Physical Layer frame configurations contain only an odd number of carriers.

10.3.3 Center-Frequency and Network-Timing Offsets for Co-Channel Interference Mitigation

The ATSC 3.0 Physical Layer protocol [3] supports low-SNR operation, which can extend coverage areas beyond those served by individual broadcast stations using earlier digital transmission systems. When there are two or more neighboring stations operating with low-SNR configurations on the same RF channel while transmitting different data, reception failure can occur due to channel estimation mismatches in strong co-channel interference environments. For example, when two co-channel stations use the same pilot pattern, FFT size, and guard interval length, after undergoing different propagation channel distortions, the frequency responses of the two arriving signals will be different. In areas where the pilot patterns of the two signals from the co-channel stations overlap, linear addition of the powers of the individual pilots will occur at receiving locations, resulting in creation of a channel model that does not represent the frequency response of either received signal. Receivers at those locations will attempt to correct the channel impairments of the resultant, combined "channel" represented by the summed pilots and consequently will not successfully receive either of the desired signals arriving at the receiver input.

Similarly, when Bootstrap signals from two neighboring co-channel stations are overlapped in the time dimension, receivers may not detect whether such overlap is due to multipath distortion of a single station or to interference from a co-channel station. To enable receivers to separate such interfering signals without requiring design changes, carrier and timing offsets, as described in the following sub-sections, shall be applied to Transmitters so that ATSC 3.0 receivers can successfully receive low-SNR signals in locations where strong, co-channel interference exists. The offsets must be assigned to triads of adjacent stations in such a way that each station in a triad has a different offset value. Given that most stations will belong to more than one triad, offsets must be organized on a large-area basis, considering all of the stations on each channel across an entire region.

10.3.3.1 Center Frequency Offset

Pilot pattern overlap always occurs when neighboring co-channel stations use the same FFT sizes, guard intervals, and pilot patterns. Even when neighboring co-channel stations use some different Physical Layer parameters (FFT sizes, guard intervals, or pilot patterns), there may be partial pilot overlap, depending on the choices of pilot patterns. Such pilot pattern overlap (both full and partial overlap) causes the sorts of channel estimation errors described above, which result in reception failure or performance degradation. To avoid overlapping pilot patterns, offset of Transmitter Center Frequencies shall be used. A Center Frequency offset parameter, having values of –1, 0, and +1 OFDM carriers, is carried in the **tx_carrier_offset** field of the Timing and Management Data. The actual offset frequency value to be applied to each Transmitter shall be the carrier frequency spacing in an 8K FFT for the channel bandwidth and baseband sampling rate in use. The offset frequency value in Hz is defined in the semantics for **tx_carrier_offset** found following Table 9.3. The Baseband Sampling Rate is indicated by the **bsr_coefficient** field of the Timing and Management Data. When signals from two or more neighboring, co-channel stations are receivable in an area, different carrier offset values shall be assigned to each of the co-channel stations. For a group of Transmitters that emits the same waveform (i.e., an SFN), the same carrier offset value shall be applied.

10.3.3.2 Network Timing Offset

Bootstrap signal overlap can occur when neighboring co-channel stations use the same frame size or integer multiples of a particular frame size, which is most likely to occur when frame sizes related to 1-second periods are used. To avoid such time-domain overlaps of Bootstraps from multiple stations, Bootstrap Reference Emission Time offsets of the station waveforms shall be applied by the respective Schedulers. Bootstrap Reference Emission Time offsets shall be applied when multiples or submultiples of Physical Layer frame lengths occur at integer multiples of 1-second periods. The directions of the Bootstrap Reference Emission Time offsets shall be the same as indicated by the **tx_carrier_offset** values sent by Schedulers to their Transmitters in the Timing and Management Data. Positive values of **tx_carrier_offset** shall indicate delays, and negative values of **tx_carrier_offset** shall indicate advances, in the Bootstrap Reference Emission Times sent by Schedulers to their Transmitters. When the assigned offset for a station is zero, its Bootstrap Reference Emission Time shall be 0 ± 1 msec of a Second Tick of TAI time [15]. When the assigned offset for a station is non-zero, its Bootstrap Reference Emission Time shall be ≥ ± (Bootstrap Length + 1 msec) up to ± (Bootstrap Length + 10 msec) of a Second Tick of TAI time [15], with the sign of '±' determined by the sign of **tx_carrier_offset.** Bootstrap symbols are 0.5 msec in length, and the minimum number of Bootstrap symbols is 4, making Bootstrap Length 2 msec or greater, depending upon whether the Bootstrap is extended per [2].

An example of using Center-Frequency and Network-Timing offsets among neighboring co-channel stations is shown in Figure 10.1.

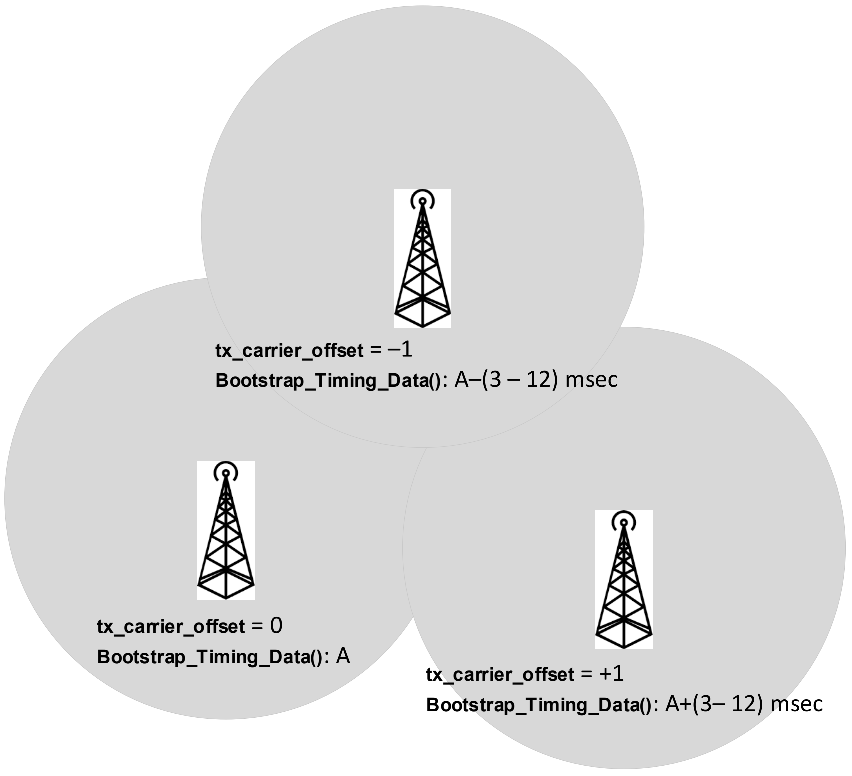

tx_carrier_offset = –1
Bootstrap_Timing_Data(): A–(3 – 12) msec
tx_carrier_offset = 0
Bootstrap_Timing_Data(): A
tx_carrier_offset = +1
Bootstrap_Timing_Data(): A+(3– 12) msec

Figure 10.1 Example use of carrier and timing offsets for neighboring co-channel stations

Annex A Physical Layer Control

A.1 PHYSICAL LAYER RESOURCES

System Manager pre-determined schedule consists of at least the control parameters listed in Section 9 of [3]. The control inputs to the Physical Layer are listed in the following sub-sections. Dependency of some parameters may be based on others from a formula. Those are identified when possible.

A.1.1 Bootstrap Signaling

bootstrap_major_version – This parameter is defined by [2] and constrained by [3]. Intent is to signal a structure change that is not backward compatible with existing version(s) of the Bootstrap.

bootstrap_minor_version – This parameter is defined by [2] and constrained by [3]. Intent is to signal a structure change that is backward compatible with existing version(s) of the Bootstrap.

ea_wake_up – This parameter is defined by [2] to signal the presence of emergency alert information.

min_time_to_next – This parameter is defined by [2] to signal when the next similar Bootstrap type (a Bootstrap that matches the same major and minor version number as the current Bootstrap) will be available in an emission.

system_bandwidth – This parameter is defined by [2] to signal the channel bandwidth of the post-Bootstrap portion of the current Physical Layer frame.

bsr_coefficient – This parameter is defined by [2] and constrained by Table 9.1 in [3] with broadcaster-intended Baseband Sample Rate.

preamble_structure – This parameter is defined by [2] and constrained by Table H.1.1 in [3] with settings for FFT size, guard interval and Preamble pilot spacing for L1 Preamble symbols chosen by the broadcaster. Preamble symbols should be at least as robust as the most robust payload settings.

A.1.2 L1-Basic Signaling

L1B_version – This parameter is defined and constrained by [3]. It is minor version signaling of the Preamble L1-Basic signaling structure.

L1B_mimo_scattered_pilot_encoding – This parameter is defined by [3] and signals MIMO pilot encoding scheme used by any MIMO subframes.

L1B_lls_flag – This parameter is defined by [3] to signal if Low Level Signaling (LLS) is available in the current frame. LLS is defined by [4] and is the starting point for finding which services are available on a given broadcast channel.

L1B_time_info_flag – This parameter is defined by [3] to signal the presence of Physical Layer timing information in the current frame.

L1B_return_channel_flag – This parameter is defined by [3] to signal the presence of a dedicated return channel.

L1B_papr_reduction – This parameter is defined by [3] to signal the technique used to reduce the peak to average power ratio within the current frame.

L1B_frame_length_mode – This parameter is defined by [3] to signal that the current frame is time-aligned with excess sample distribution to the guard intervals of data payload OFDM symbols or if the current frame is symbol-aligned with no excess sample distribution.

L1B_frame_length – This parameter is defined by [3] to signal the time Period measured from the beginning of the first sample of the Bootstrap associated with the current frame to the end of the final sample associated with the current frame when time-aligned frames are used. Frame length has many considerations like the longest segment length as defined in [4], or time interleaving depth that broadcasters must choose depending on desired content and robustness. Sizes are constrained by [3] to be between 50 msec and 5000 msec in 5-msec increments.

L1B_excess_samples_per_symbol – This parameter is defined by [3] to signal the additional number of excess samples included in the guard interval of each non-Preamble OFDM symbol of the post Bootstrap portion of the current frame when time-aligned frames are used.

L1B_time_offset – This parameter is defined by [3] to signal the number of sample periods between the nearest preceding or coincident millisecond boundary and the leading edge of the frame when symbol-aligned frames are used.

L1B_additional_samples – This parameter is defined by [3] to signal the number of additional samples added at the end of a frame to facilitate sampling clock alignment when symbol-aligned frames are used.

L1B_num_subframes – This parameter is defined by [3] to signal the number of subframes present within the current frame.

L1B_preamble_num_symbols – This parameter is defined by [3] to signal the total number of OFDM symbols contained within the Preamble, not including the first Preamble symbol.

L1B_preamble_reduced_carriers – This parameter is defined by [3] to signal the number of control units of carriers by which the maximum number of carriers for the FFT size used for the Preamble is reduced. It applies to all Preamble symbols except the first one of the current frame which uses the maximum possible carrier reduction.

L1B_L1_Detail_content_tag – This parameter is defined by [3] and signals new information is available in L1 Detail.

L1B_L1_Detail_size_bytes – This parameter is defined by [3] to signal the size (in bytes) of the L1-Detail information.

L1B_L1_Detail_fec_type – This parameter is defined by [3] to signal the FEC type for L1-Detail information protection. It is a combination of 16K LDPC code length with variety of non-uniform constellations and code rates.

L1B_L1_Detail_additional_parity_mode – This parameter is defined by [3] to signal the Additional Parity Mode which gives the ratio (*K*) of the number of additional parity bits for the next frame's L1-Detail that are carried in the current frame to half of the number of coded bits for the next frame's L1-Detail signaling.

L1B_L1_Detail_total_cells – This parameter is defined by [3] to signal the total size (specified in data cells) of the combined coded and modulated L1-Detail signaling for the current frame and any modulated additional parity bits for L1-Detail signaling of the next frame.

L1B_first_sub_mimo – This parameter is defined by [3] to signal whether MIMO is used for the first subframe of the current frame.

L1B_first_sub_miso – This parameter is defined by [3] to signal whether MISO is used for the first subframe of the current frame.

L1B_first_sub_fft_size – This parameter is defined by [3] to signal the FFT size associated with the first subframe of the current frame.

L1B_first_sub_reduced_carriers – This parameter is defined by [3] to signal the number of control units of carriers by which the maximum number of carriers for the FFT size used for the first subframe of the current frame is reduced.

L1B_first_sub_guard_interval – This parameter is defined by [3] to signal the guard interval length used for the OFDM symbols of the first subframe of the current subframe.

L1B_first_sub_num_ofdm_symbols – This parameter is defined by [3] to signal a value equal to one less than the total number of data payload OFDM symbols, including any subframe boundary symbol(s), present within the first subframe of the current frame.

L1B_first_sub_scattered_pilot_pattern – This parameter is defined by [3] to signal the scattered pilot pattern used for the first subframe of the current subframe whether it is SISO or MIMO.

L1B_first_sub_scattered_pilot_boost – This parameter is defined by [3] to signal the relative amplitude of the scattered pilots used for the first subframe of the current frame.

L1B_first_sub_sbs_first – This parameter is defined by [3] to signal whether or not the first symbol of the first subframe of the current frame is a subframe boundary symbol.

L1B_first_sub_sbs_last – This parameter is defined by [3] to signal whether or not the last symbol of the first subframe of the current frame is a subframe boundary symbol.

A.1.3 L1-Detail Signaling

L1D_version – This parameter is defined by [3]. It signals the minor version of the Preamble L1-Detail structure for the current frame.

L1D_num_rf – This parameter is defined by [3] to signal the number of BSIDs involved in channel bonding of the current ATSC 3.0 system, not including the BSID of the present channel.

L1D_rf_id – This parameter is defined by [3] to index the implicit IDs of the other RF channels involved in channel bonding.

L1D_bonded_bsid – This parameter is defined by [3] to signal the Broadcast Stream ID of a separate RF channel that is channel bonded with the current RF channel.

L1D_time_sec – This parameter is defined by [3] to signal the seconds component of the time information.

L1D_time_msec – This parameter is defined by [3] to signal the milliseconds component of the time information.

L1D_time_usec – This parameter is defined by [3] to signal the microseconds component of the time information.

L1D_time_nsec – This parameter is defined by [3] to signal the nanoseconds component of the time information.

L1D_mimo – This parameter is defined by [3] to signal whether MIMO is used for the given subframe.

L1D_miso – This parameter is defined by [3] to signal whether MISO is used for the given subframe.

L1D_fft_size – This parameter is defined by [3] to signal the FFT size associated with the given subframe.

L1D_reduced_carriers – This parameter is defined by [3] to signal the number of control units of carriers by which the maximum number of carriers for the FFT size used for the given subframe is reduced.

L1D_guard_interval – This parameter is defined by [3] to signal the guard interval length used for the OFDM symbols of the given subframe.

L1D_num_ofdm_symbols – This parameter is defined by [3] to signal the value equal to one less than the total number of data payload OFDM symbols, including any subframe boundary symbol(s), present within the given subframe.

L1D_scattered_pilot_pattern – This parameter is defined by [3] to signal the scattered pilot pattern used for the given subframe, whether it is SISO or MIMO.

L1D_scattered_pilot_boost – This parameter is defined by [3] to signal the relative amplitude of the scattered pilots used for the given subframe.

L1D_sbs_first – This parameter is defined by [3] to signal whether or not the first symbol of the given subframe is a subframe boundary symbol.

L1D_sbs_last – This parameter is defined by [3] to signal whether or not the last symbol of the given subframe is a subframe boundary symbol.

L1D_subframe_multiplex – This parameter is defined by [3] to signal whether the given subframe is time-division multiplexed / concatenated in time with adjacent subframes.

L1D_frequency_interleaver – This parameter is defined by [3] to signal whether the frequency interleaver is enabled or bypassed for the given subframe.

L1D_sbs_null_cells – This parameter is defined by [3] to signal the number of null cells in the subframe boundary symbols of the current subframe.

L1D_num_plp – This parameter is defined by [3] to signal a value equal to one less than the total number of PLPs used within the given subframe.

L1D_plp_id – This parameter is defined by [3] to signal a value equal to the ID of the given PLP, with a range from 0 to 63, inclusive.

L1D_plp_lls_flag – This parameter is defined by [3] to signal whether the given PLP carries the Low Level Signaling (LLS) defined by [4].

L1D_plp_layer – This parameter is defined by [3] to signal a value equal to the layer index of the given PLP.

L1D_plp_start – This parameter is defined by [3] to signal a value equal to the index of the data cell that holds the first data cell of the current PLP in the given subframe.

L1D_plp_size – This parameter is defined by [3] to signal a value equal to the number of data cells allocated to the given PLP.

L1D_plp_scrambler_type – This parameter is defined by [3] to signal the choice of scrambler type for the given PLP.

L1D_plp_fec_type – This parameter is defined by [3] to signal the Forward Error Correction (FEC) method used for encoding the given PLP.

L1D_plp_mod – This parameter is defined by [3] to signal the modulation used for the given PLP, whether SISO or MIMO.

L1D_plp_cod – This parameter is defined by [3] to signal the code rate used for the given PLP.

L1D_plp_TI_mode – This parameter is defined by [3] to signal the time interleaving mode for the given PLP. The Scheduler is responsible for selecting the appropriate time interleaving mode to use for each Core PLP configured for an RF channel. An Enhanced PLP follows the time interleaver mode of the Core PLP(s) with which it is layered-division multiplexed.

L1D_plp_fec_block_start – This parameter is defined by [3] to signal the start position of the first FEC Block that begins within the current PLP during the current subframe.

L1D_plp_CTI_fec_block_start – This parameter is defined by [3] to signal the position, after the CTI, of the first cell of the first complete FEC Block, before the CTI, for the current PLP in the current or subsequent subframe.

L1D_plp_num_channel_bonded – This parameter is defined by [3] to signal the number of RF channels, not including the present RF channel, involved in channel bonding of the given PLP.

L1D_plp_channel_bonding_format – This parameter is defined by [3] to signal the channel bonding format for the given PLP, whether plain channel bonding or SNR averaging channel bonding.

L1D_plp_bonded_rf_id – This parameter is defined by [3] to signal the RF IDs of the channels that are used for channel bonding with the given PLP.

L1D_plp_mimo_stream_combining – This parameter is defined by [3] to signal whether the Stream combining option of the MIMO precoder is used in the given PLP.

L1D_plp_mimo_IQ_interleaving – This parameter is defined by [3] to signal whether the IQ polarization interleaving option of the MIMO precoder is used in the given PLP.

L1D_plp_mimo_PH – This parameter is defined by [3] to signal whether the phase hopping option of the MIMO precoder is used in the given PLP.

L1D_plp_type – This parameter is defined by [3] to signal when the given PLP is non-dispersed (i.e., all data cells of the current PLP have contiguous logical addresses and subslicing is not used for the current PLP) or when the current PLP is dispersed (i.e., not all data cells of the current PLP have contiguous logical addresses and subslicing is used for the current PLP).

If **L1D_plp_type** has a value of '1', the number of subslices and subslice interval are set.

L1D_num_subslices – If the given PLP is dispersed, this parameter is defined by [3] to signal a value equal to one less than the actual number of subslices used for the given PLP within the given subframe.

L1D_subslice_interval – If the given PLP is dispersed, this parameter is defined by [3] to signal a value equal to the number of sequentially-indexed data cells measured from the beginning of a subslice for a given PLP to the beginning of the next subslice for the same PLP.

L1D_plp_TI_extended_interleaving – This parameter is defined by [3] to signal whether extended time interleaving is to be used for the given PLP.

L1D_plp_CTI_depth – This parameter is defined by [3] to signal the number of rows used in the Convolutional Time Interleaver.

L1D_plp_CTI_start_row – This parameter is defined by [3] to signal the position of the commutator at the start of the given PLP within the current subframe.

L1D_plp_HTI_inter_subframe – This parameter is defined by [3] to signal the hybrid time interleaving mode for the given PLP.

L1D_plp_HTI_num_ti_blocks – This parameter is defined by [3] to signal either the number of TI Blocks per interleaving frame, N_{TI}, when **L1D_plp_HTI_inter_subframe** = 0 or the number of subframes, P_I, over which cells from one TI Block are carried when **L1D_plp_HTI_inter_subframe** = 1 in the given PLP.

L1D_plp_HTI_num_fec_blocks_max – This parameter is defined by [3] to signal a value one less than the maximum number of FEC Blocks per interleaving frame for the given PLP.

L1D_plp_HTI_num_fec_blocks – This parameter is defined by [3] to signal a value one less than the number of FEC Blocks contained in the current interleaving frame for the given PLP.

L1D_plp_HTI_cell_interleaver – This parameter is defined by [3] to signal whether the Cell Interleaver is used in the given PLP.

L1D_plp_ldm_injection_level – This parameter is defined by [3] to signal the Enhanced PLP injection level relative to the Core PLP.

L1D_bsid – This parameter is defined by [3] to signal the Broadcast Stream ID of the current RF channel.

Annex B Network Configuration Examples

B.1 EXAMPLE STUDIO NETWORK TOPOLOGIES

The following discussion provides various Network topology examples to clarify how systems may be deployed to achieve various capabilities and robustness. It is informative only. Each deployment will be different and may change over time as new capabilities are added.

For environments with a single ALPTP Stream Producer and Broadcast Gateway, simply connecting the two units with a cross-over cable would suffice. The Network connections would have to be on the same subnet but would require no further setup. Typically, systems are connected using routers or switches as is depicted in Figure B.1. Note that Figure B.1 also shows using DSTP multicast to move various signaling, NRT data, audio and video components into the ALP Encapsulator. The ALP Encapsulator would process the DSTP as described in Section 7 prior to the ALP encapsulation stage, resulting in {4} in Figure B.1. Using DSTP allows the ALP Encapsulator to manage the order and timing of the contribution source Streams using information provided in the DSTP Tunneled Packet Information Headers (see Section 7.2 for details). The Data Sources may also supply timing and priority information in the DSTP Streams to enable reconstruction of packet spacing within the resultant ALP Streams.

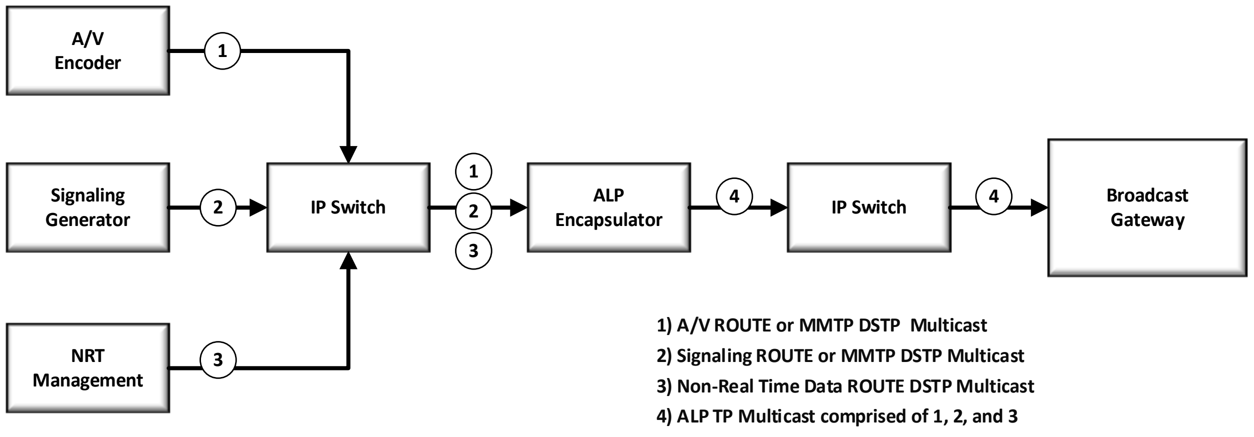

A/V Encoder
Signaling Generator
NRT Management
IP Switch
ALP Encapsulator
IP Switch
Broadcast Gateway
1
2
3
4
1) A/V ROUTE or MMTP DSTP Multicast
2) Signaling ROUTE or MMTP DSTP Multicast
3) Non-Real Time Data ROUTE DSTP Multicast
4) ALP TP Multicast comprised of 1, 2, and 3

Figure B.1 Simple ALP encapsulation.

Figure B.2 shows a case in which separate A/V encoding systems produce audio and video for two PLPs. In this example, a signaling system would produce signaling information that would be included in both PLP Streams. Signaling could be included in either Stream individually or in a dedicated PLP in the same manner. The non-real-time (NRT) data generation management system also would produce ROUTE Streams that could be included in either or both PLPs as desired. In Figure B.2, NRT data is only included in PLP 1. Note that in this example, all DSTP Streams are on separate multicast addresses on the input to the ALP Encapsulator. The ALPTP output of the ALP Encapsulator (5 and 6) would be on the same multicast address with different ports. Since there is only one source for each Stream in this example, there is no need for IGMPv3 Source

Specific Multicasting (SSM), and all network connections can be accomplished with relatively inexpensive routers or switches.

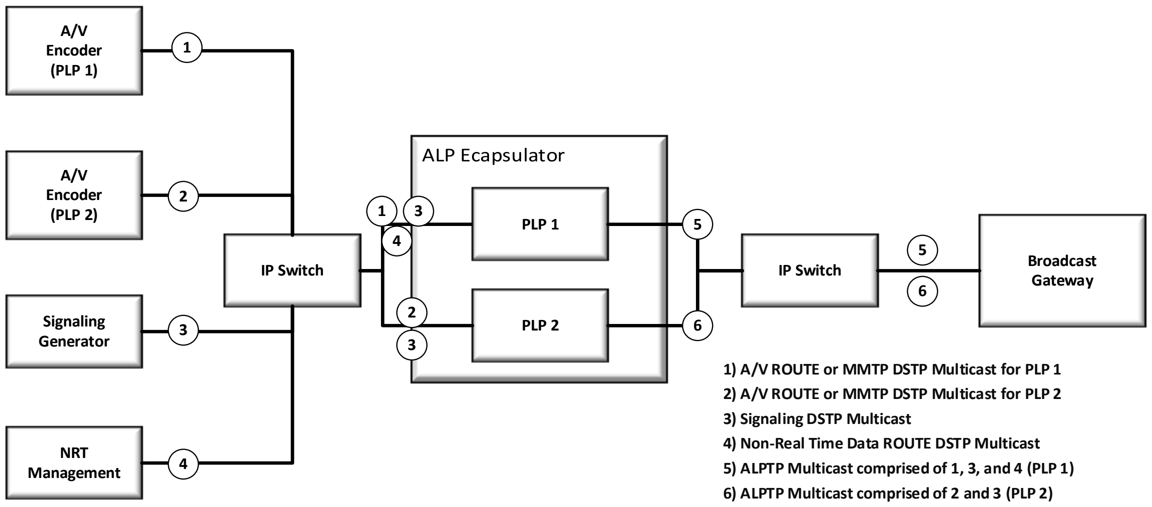

A/V Encoder (PLP 1)
A/V Encoder (PLP 2)
Signaling Generator
NRT Management
IP Switch
ALP Ecapsulator
PLP 1
PLP 2
IP Switch
Broadcast Gateway
1
2
3
4
5
6
1) A/V ROUTE or MMTP DSTP Multicast for PLP 1
2) A/V ROUTE or MMTP DSTP Multicast for PLP 2
3) Signaling DSTP Multicast
4) Non-Real Time Data ROUTE DSTP Multicast
5) ALPTP Multicast comprised of 1, 3, and 4 (PLP 1)
6) ALPTP Multicast comprised of 2 and 3 (PLP 2)

Figure B.2 Multiple PLP example.

A more typical production situation is diagrammed in Figure B.3. This example shows a redundant encoding and signaling environment, redundant ALP Encapsulators and redundant output systems feeding into separate STLs (not shown). The ALP Encapsulators combine various DSTP multicasts into individual ALPTP Streams, each destined for a unique PLP. A single physical ALP Encapsulator likely would create multiple ALPTP Streams for different PLPs, but, functionally, each PLP Stream can be considered separately.

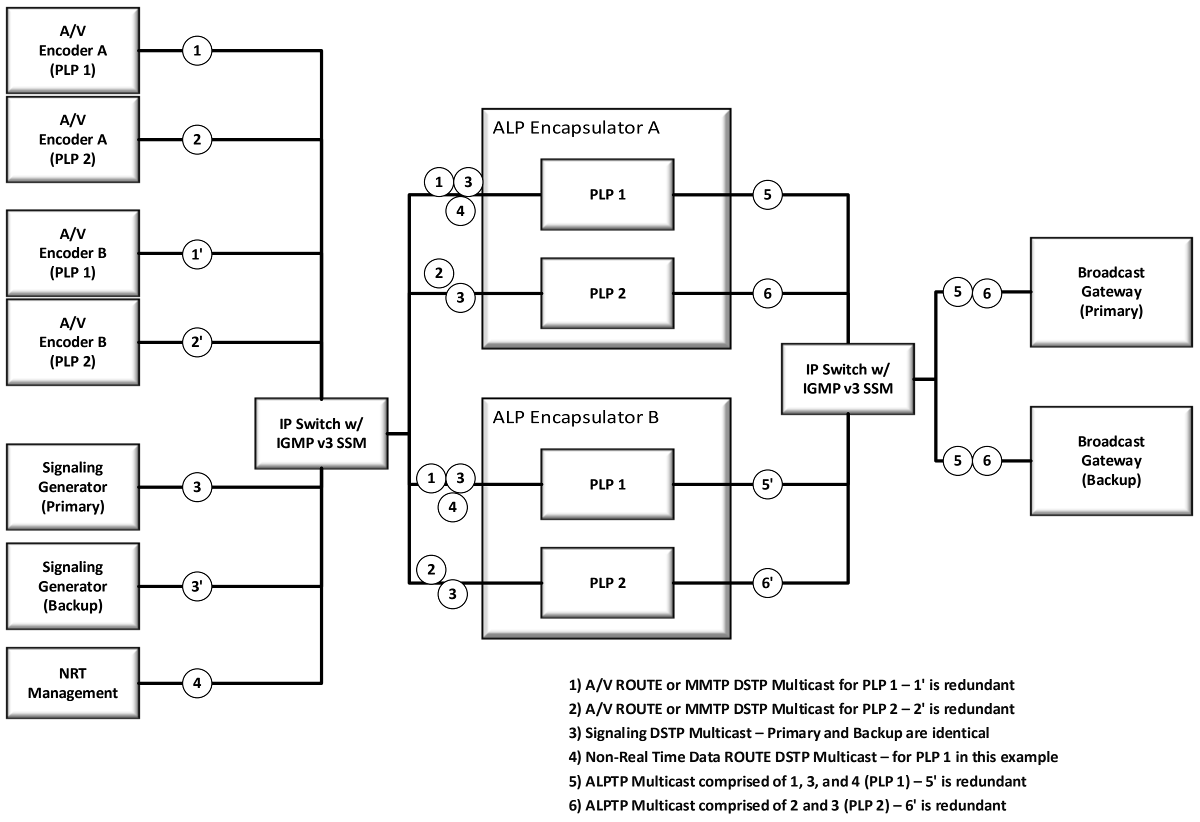

A/V Encoder A (PLP 1)
A/V Encoder A (PLP 2)
A/V Encoder B (PLP 1)
A/V Encoder B (PLP 2)
Signaling Generator (Primary)
Signaling Generator (Backup)
NRT Management
IP Switch w/ IGMP v3 SSM
ALP Encapsulator A
ALP Encapsulator B
PLP 1
PLP 2
IP Switch w/ IGMP v3 SSM
Broadcast Gateway (Primary)
Broadcast Gateway (Backup)
1) A/V ROUTE or MMTP DSTP Multicast for PLP 1 – 1' is redundant
2) A/V ROUTE or MMTP DSTP Multicast for PLP 2 – 2' is redundant
3) Signaling DSTP Multicast – Primary and Backup are identical
4) Non-Real Time Data ROUTE DSTP Multicast – for PLP 1 in this example
5) ALPTP Multicast comprised of 1, 3, and 4 (PLP 1) – 5' is redundant
6) ALPTP Multicast comprised of 2 and 3 (PLP 2) – 6' is redundant

Figure B.3 Fully redundant routing example.

Note that none of the redundant systems needs to be aware of its counterpart. For example, A/V Encoder A (PLP 1) can be configured identically to A/V Encoder B (PLP 1) except, of course, for its network address. The emitted DSTP multicasts (in Figure B.3, the circled numbers {1} and {1'}) are identical in all aspects except that the UDP headers would contain different source addresses. Both ALP Encapsulator A (PLP 1) and ALP Encapsulator B (PLP 1) would be configured with the addresses of the PLP 1 A/V Encoders. Initially, they would both begin ALP encapsulation of the A/V DSTP Stream {1} into ALPTP Streams {5} and {5'} by joining, via IGMP, the multicast address using the address of A/V Encoder A (PLP 1). If any problems were detected with {1} by either encapsulator, it would then switch to the second A/V Stream, {1'}. This would be accomplished by simply leaving the IGMP multicast group for A/V Encoder A (PLP 1) {1} and 'joining' the A/V Encoder B (PLP 1) multicast {1'}. No other communications are required between the redundant system elements. Similarly, the {5} or {5'} ALPTP multicasts would feed both Broadcast Gateway systems based on a join from each of them. Each Broadcast Gateway would be configured with the source addresses of both ALP Encapsulators A and B so that it could switch between the two sources if the quality of any of the Streams they were processing degraded.

Note that a single router or switch with IGMPv3 SSM capability could be used to route all the multicast traffic between all elements of a system. There are many options for duplicating routers and even having ring and tree configurations. Be aware that all multicast addresses would need to be unique within a broadcast plant network to enable a variety of routing options.

Annex C Scheduler Functional Description

C.1 OVERVIEW

Section 5 above described the high-level functions and requirements for the Scheduler and its place within a typical ATSC 3.0 headend. This annex provides more detail on the function of a reference Scheduler.

C.2 SCHEDULER OPERATION

The operation of the Scheduler is constrained by a combination of dynamic, quasi-static, and static parameters. The exact definition of these constraints is left to implementation. This document defines a required function, but not the parameters of that function.

There are two categories of metadata associated with content to be delivered. There is metadata associated with the content which must transit the link as they comprise information needed to successfully decode and render the media. There is another class of metadata that is exclusive to the task of scheduling media in an ATSC 3.0 system. This scheduling metadata is referred to as delivery metadata. This section discusses the functions and application of this delivery metadata. There is some discussion with respect to the order in which metadata and content should be delivered in order to optimize channel change speed.

The cascade of functions involved in the process of scheduling is shown below in Figure C.1. This figure contains no requirements as it is merely an example of an architecture. References to time are with respect to Server Current Time (SCT) as discussed in [4].

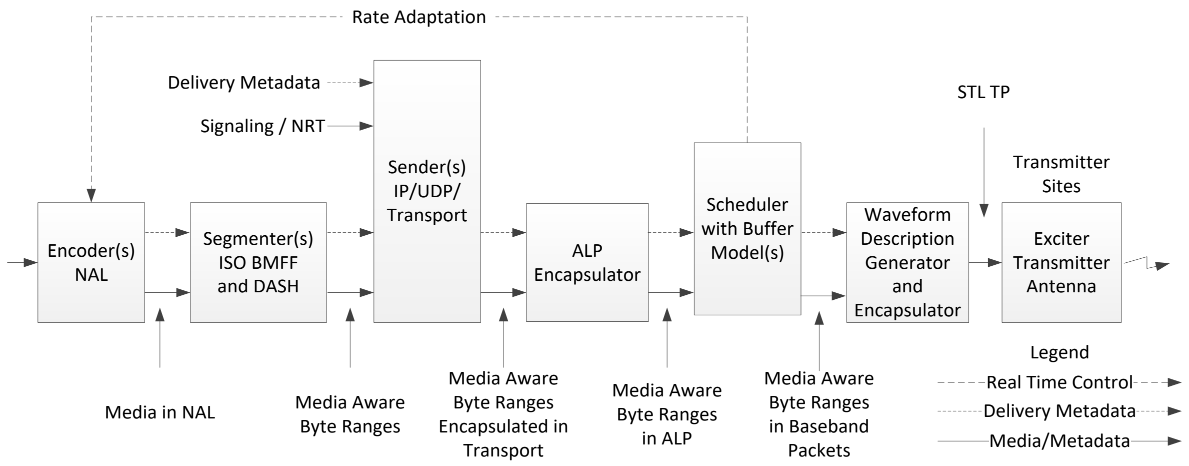

Rate Adaptation
Delivery Metadata
Signaling / NRT
Sender(s) IP/UDP/ Transport
STL TP
Transmitter Sites
Encoder(s) NAL
Segmenter(s) ISO BMFF and DASH
ALP Encapsulator
Scheduler with Buffer Model(s)
Waveform Description Generator and Encapsulator
Exciter Transmitter Antenna
Media in NAL
Media Aware Byte Ranges
Media Aware Byte Ranges Encapsulated in Transport
Media Aware Byte Ranges in ALP
Media Aware Byte Ranges in Baseband Packets
Legend
Real Time Control
Delivery Metadata
Media/Metadata

Figure C.1 Cascade of real-time functions involved with Scheduler.

Figure C.2 below shows a conceptual depiction of a Scheduler process flow. The central concept is that there are durations of time under construction with known target emission times. The process generates an endless sequence of Media Segments, which are mapped into Physical Layer frames and transmitted. There are no requirements contained in this figure; it is an example.

The point of the figure is there is a process in time which results in Media Segments being constructed and emitted. The construction of said Media Segments and Physical Layer frames is a series of intermediate processes that each run on a deadline. For the purposes of this figure, there is one set of Media Segments (e.g., audio, video, and captions) per Service per Physical Layer frame. This is not required and may not be optimum. This figure is depicted with a one-to-one correspondence among Media Segments and Physical Layer frames as a convenience to illustrating the process flow in time. The relationship between an Analyzed Media Duration and Physical Layer frames is discussed in some detail in Section C.4 below.

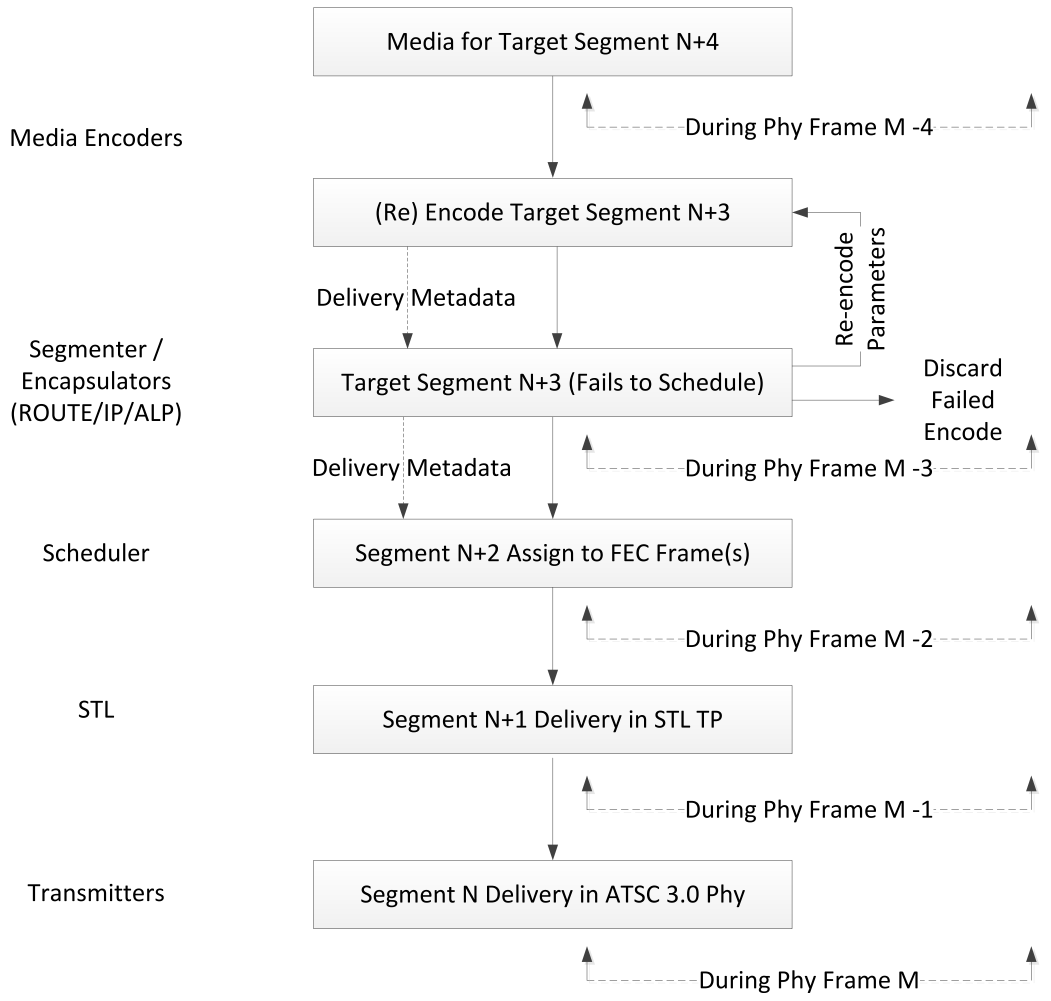

Media for Target Segment N+4
Media Encoders
During Phy Frame M -4
(Re) Encode Target Segment N+3
Re-encode Parameters
Delivery Metadata
Segmenter / Encapsulators (ROUTE/IP/ALP)
Target Segment N+3 (Fails to Schedule)
Discard Failed Encode
Delivery Metadata
During Phy Frame M -3
Scheduler
Segment N+2 Assign to FEC Frame(s)
During Phy Frame M -2
STL
Segment N+1 Delivery in STL TP
During Phy Frame M -1
Transmitters
Segment N Delivery in ATSC 3.0 Phy
During Phy Frame M

Figure C.2 Example depiction of a high-level Scheduler process flow.

C.3 KEY CONCEPTS OF SCHEDULER DELIVERY METADATA

The concept of earliest delivery at the Physical Layer is illustrated in Figure C.3 below. The data contained in this FEC Frame will be emitted from the ATSC 3.0 Transmitter after this Earliest Time. As shown, this description is inclusive of all processes comprised in the generation of the transmitted FEC Frame. If the FEC Frame were being discussed in the context of the Sync and Delivery [4] the Physical Layer FEC Frame contains data that is a Data Delivery Event at the receiver.

Latest Time at the Physical Layer is illustrated in Figure C.3 below. Conceptually, this is constructed and constrained in a manner similar to Earliest Time above. The data contained in this FEC Frame will be emitted from the ATSC 3.0 Transmitter before this Latest Time.

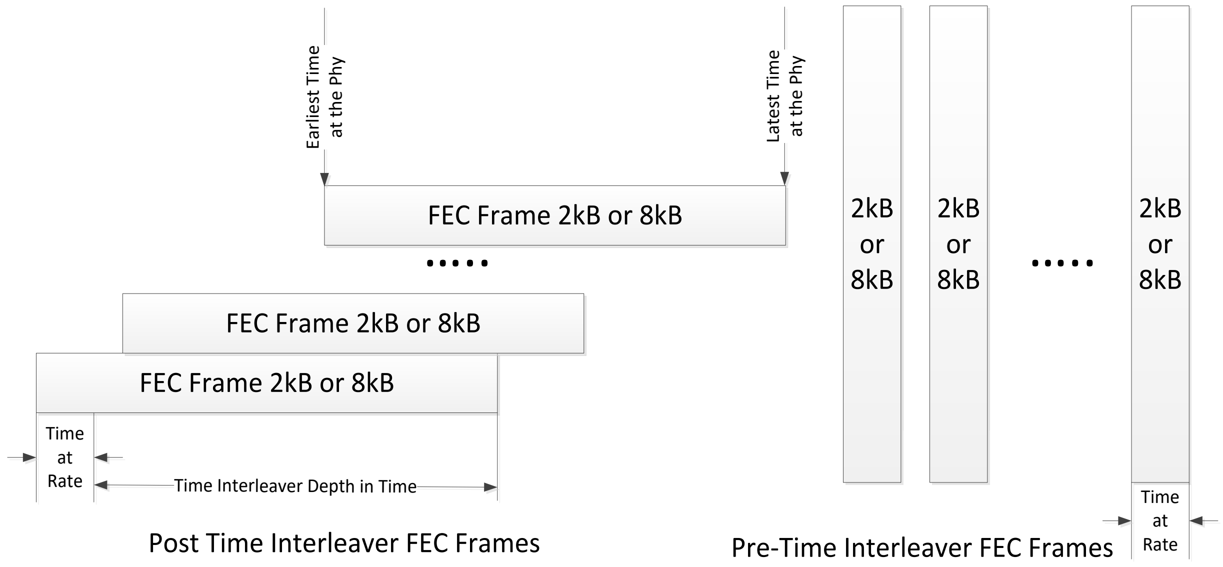


Figure C.3 Illustration of Earliest and Latest Times with a block interleaver.

Figure C.3 depicts the Earliest and Latest Times of a FEC Frame as realized in the Physical Layer. In practice the Scheduler function should receive a notably wider time range of delivery metadata, such that the Scheduler is not over-constrained.

Figure C.4 provides a conceptual view of Earliest and Latest Times in the context of a Media Segment playback duration. Figure C.4 depicts a broader notion of Earliest and Latest Times for multiple use cases.

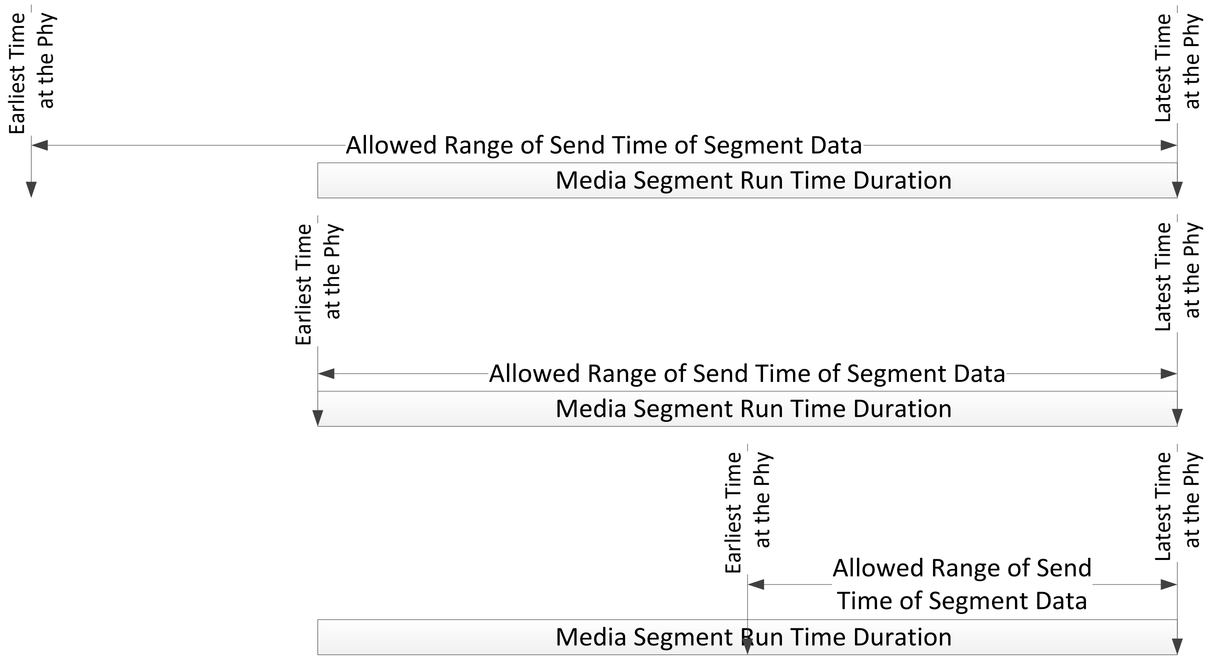

Earliest Time at the Phy
Latest Time at the Phy
Allowed Range of Send Time of Segment Data
Media Segment Run Time Duration
Earliest Time at the Phy
Latest Time at the Phy
Allowed Range of Send Time of Segment Data
Media Segment Run Time Duration
Earliest Time at the Phy
Latest Time at the Phy
Allowed Range of Send Time of Segment Data
Media Segment Run Time Duration

Figure C.4 Illustration of Earliest and Latest Time options relative to Media Segment play.

The use cases in Figure C.4 are intended to illustrate that the concept of Earliest and Latest Times is flexible. These use cases are not requirements. The three use cases depict that the mechanism can describe a variety of implementations. In the top use case, the specification of Earliest Time would allow the Scheduler to send Media Segment data significantly ahead of the actual demand time for the start of playback. The Latest Time in this example is bounded by the playback deadline for the Media Segment. The time shown is a duration – not the actual delivery or play time. The middle use case depicts a situation in which the media playback and media delivery are constrained to be within a Media Segment playback duration. This might be required, for instance, if there is any switching among PLPs, e.g., at a DASH Period boundary. The bottom use case illustrates a use case in which the delivery of a Media Segment is accomplished in less than a Media Segment run time duration. This sort of use case can result in faster channel change. These use cases are examples and are not intended to constrain the definition or usage of Earliest and Latest Times or actual implementation.

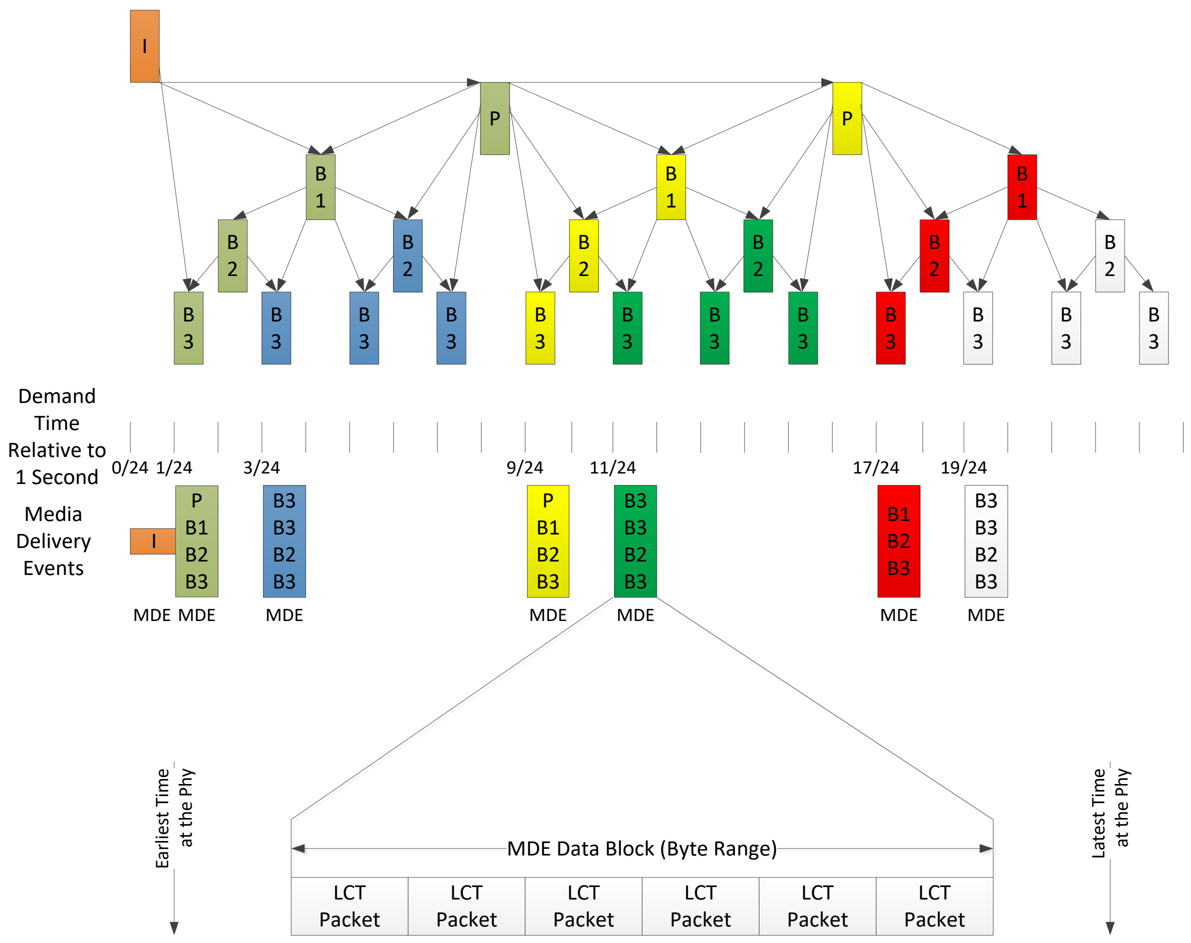


Figure C.5 Example of MDE and associated Earliest and Latest Times.

Figure C.5 illustrates the relationship of an individual MDE to its associated Earliest and Latest Times. The Latest Time in this case is related to 11/24ths of the media second as compared to a 0 Latest Time for the I frame MDE of the same "GoP". The 0 Latest Time repeats in this example on 1-second boundaries. If in this example, the Physical Layer frame and Media Segment boundaries conform to whole seconds, Earliest Time could be N.0000… and Latest Time N.99999... Given that in-order delivery is enforced for streaming media, the functional requirement is within this whole second for this one-second-duration Media Segment delivery.

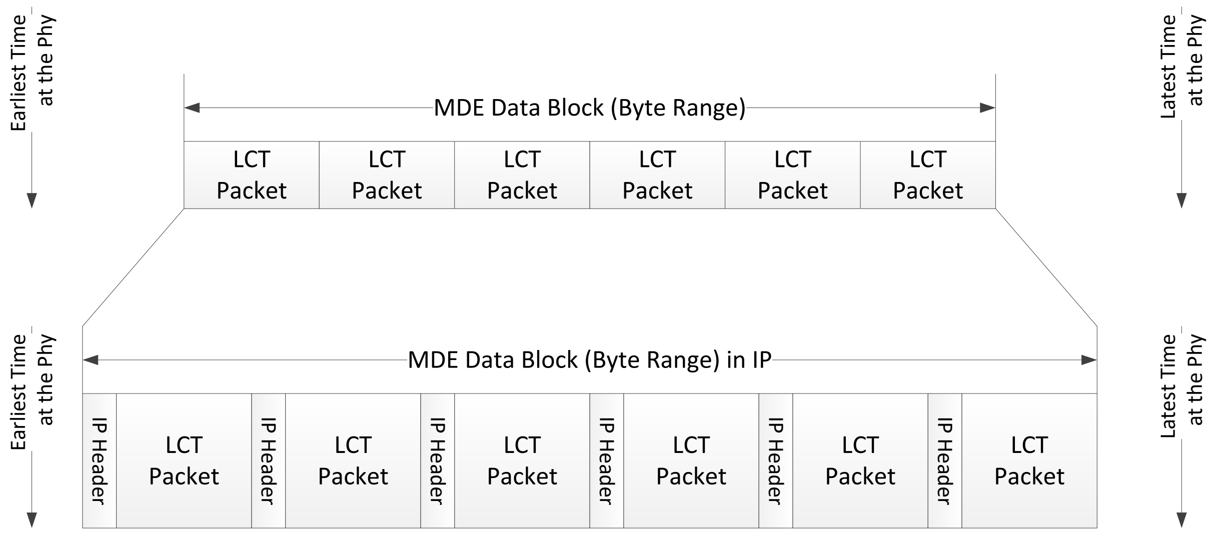


Figure C.6 Relationship of an MDE in LCT packets to IP encapsulation.

Figure C.6 is an example of an MDE with IP encapsulation shown. This figure contains no requirements beyond the Earliest and Latest Times being inherited from the MDE. There is no requirement that there is one LCT packet per IP header. There is no requirement that there is more or less than one LCT packet per IP Header. The only constraint is that IP packets containing the MDE are subject to the indicated Earliest and Latest Times of the source MDE data block.

C.4 HANDLING BOUNDARY CONDITIONS

To this point these concepts have been discussed in broad terms. Known metadata about the timeline of the encapsulated and to-be-encapsulated media is expressed down the stack to the Scheduler. This process may be summarized as the encapsulated media inherits the delivery requirements of contained media. There are assorted boundary conditions to consider. This section illustrates a few of these and describes how delivery metadata may be handled.

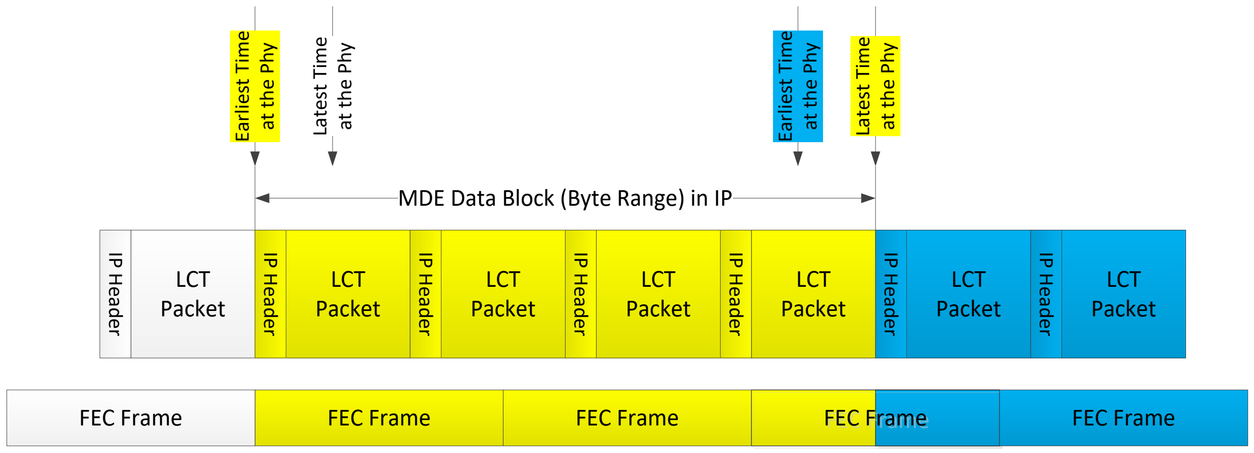


Figure C.7 Media sent at early boundary; e.g., Period start.

Figure C.7 depicts a use case for which the leading data of the first FEC Frame of the center MDE has a forward-justified IP packet (and ALP) in the first as-assigned FEC Frame. This is caused by the leading LCT packet in, for example, MDE with RAP having been identified as "Start in new FEC Frame". This could also occur because the Earliest Time of the center MDE in Figure C.7 is later than the Latest Time of the leading MDE, which is not shown and is more constraining for the Scheduler. The trailing FEC Frame after the center MDE contains a fragment of the trailing MDE. It should be noted that the sequence of FEC Frames may not be continuous in time and is only shown as such for the convenience of illustrating the mapping of IP packets to FEC Frames. The ALP encapsulation is not shown for convenience but is required. This sequence of encapsulated MDEs is assumed to be from a single content type with this media content having been declared as in-order delivery.

C.5 DELIVERY ORDER WITHIN AND ACROSS MULTIPLE SESSIONS

To this point in the discussion, there has been no statement with respect to whether a sequence of IP packets containing LCT packets represents one or more sessions. For the purpose of convenience, it has been assumed that examples have represented a single session, but possibly with multiple content types within the delivered Media Segment files. This section provides an example of a Presentation containing multiple sessions. Perhaps there is one content type per session.

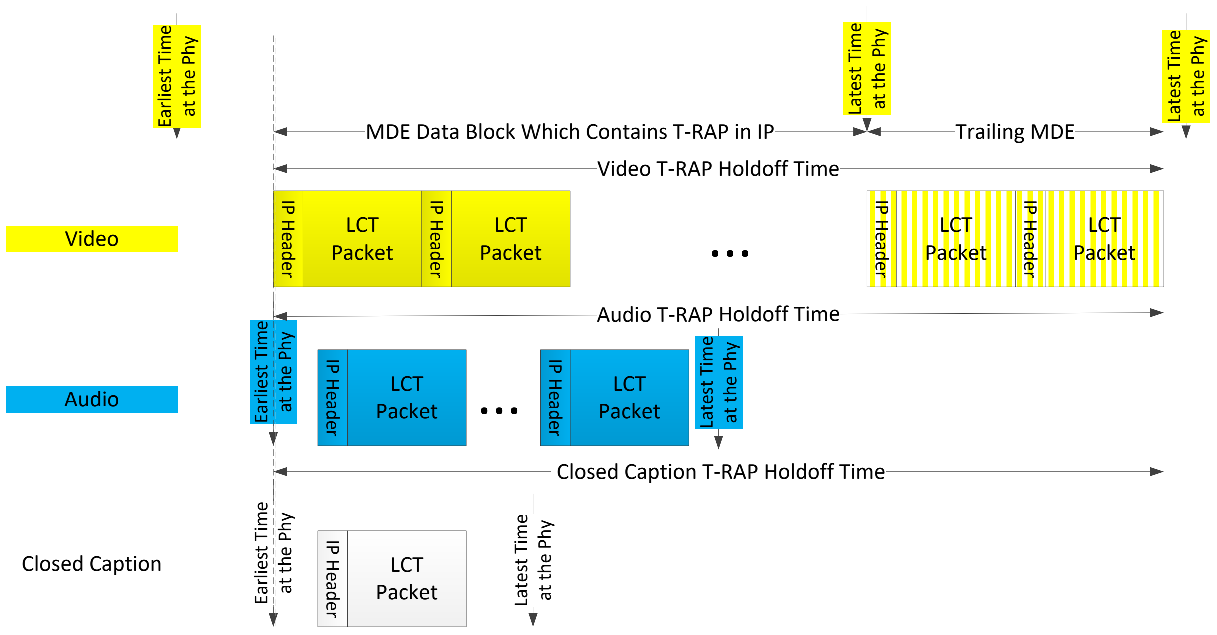


Figure C.8 Example of ordered delivery across multiple sessions.

In Figure C.8 an example is shown of delivery of multiple sessions across a single PLP or multiple PLPs. In this example the MPD is delivered in the T-RAP of video and the DASH client will see and request byte range delivery ahead of Media Segment availability start time. The video typically has the longest RAP cadence, so it is best to align MPD delivery to the video RAP. A Key requirement for MDE is the request of byte range delivery ahead of Media Segment availability start time. The MDE hold-off time due to `EXT_ROUTE_PRESENTATION_TIME - SCT` will expire ahead of Media Segment availability start time.

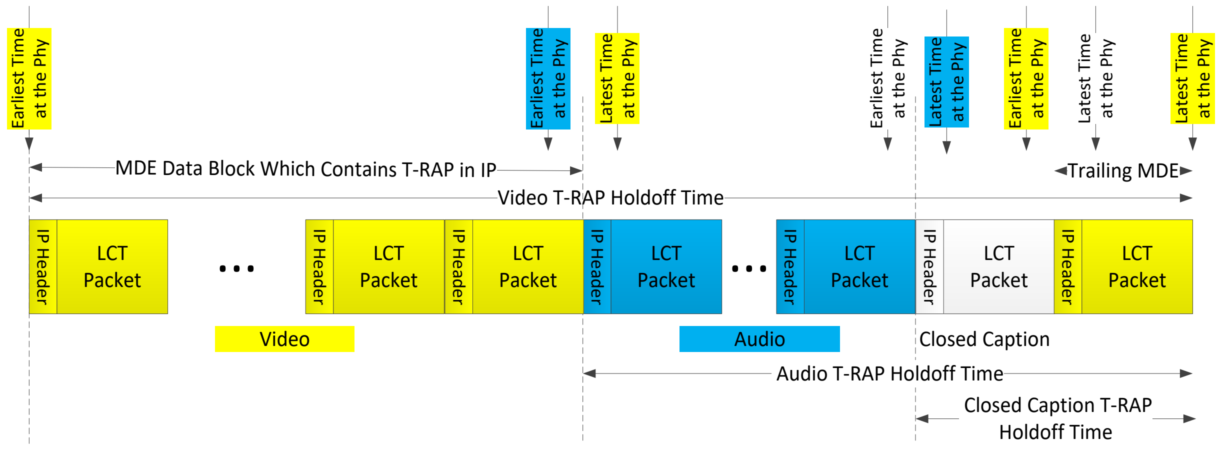


Figure C.9 Explicit delivery order on PLP with multiple sessions.

The order of the to-be-delivered IP packets may be preserved in a PLP. This is shown in Figure C.9. This is not relevant for a service spanning multiple PLPs. This requires that the sequence of LCT packets in IP or compressed IP is preserved in the emission. None of the timing description changes, but the PLP mapper is configured to observe in-order delivery of the IP Stream as delivered to a specific PLP at the Scheduler.

C.6 TIMELINES AND DEADLINES

The delivery of media as ISO BMFF files is an essentially endlessly repeating loop. Each Media Segment of media in Live Streaming is expressed in an ISO BMFF container that has an internal time line that starts at 0 and repeats on a, for example, N-second interval. A DASH MPD has a defined presentation timeline for the ISO BMFF files and the relationship of the send times at the Physical Layer has a nominally static relationship. The MDE method is adaptive to the stack delay of the receiver i.e., it starts as soon as possible without a stall. In each case, the correct timing must be defined by the Segmenter and encapsulator respectively.

If all the video "GoPs" run on the same time cycle, i.e., the IDR frames across Services align in time, most of the benefit of MDE mode will likely accrue to no time guard banding being required for stack delay variability. For Media Segment playback, the MPD must work for the slowest stack. MDE playback starts as soon after reception of a T-RAP as allowed. In order to achieve faster start-up, the use of staggered Media Segment start times likely is required as well as non-uniform bandwidth assignment priority vs. time. These are implementation details, which do not change the fundamental mechanisms, i.e., each MDE and Media Segment have a time deadline for delivery at the Physical Layer expressed by a Latest Time.

C.7 CONCEPT AND PRACTICE OF ANALYZED MEDIA DURATION

The discussion up to this point has assumed that media is encoded based on a one-second Media Segment duration; while this is conceptually convenient, it is a bit simplistic as compared to the capabilities of ATSC 3.0. The generalized construct for creating Physical Layer frames is an Analyzed Media Duration. As discussed above the delivery on the Physical Layer must meet a set of requirements driven by media timeline(s). The Physical Layer frames defined by the Scheduler do not have to correspond precisely to the Media Segment durations, as long as the time delivery requirements are met. The boundaries of Physical Layer frames may not or do not directly correspond to Media Segments, but they are related. The next paragraphs describe how the general method may be applied.

For each ALP Stream input, there is a Data Source. Video Data Sources can be the most demanding in terms of ISO BMFF Media Segment file size and therefore can drive requirements for ALP buffer size and Analyzed Media Duration. Large encoded Media Segment files may or may not be represented as MDE. An Analyzed Media Duration is a Period of time that is sufficient across all ALP Streams provided to the Scheduler such that an analysis Period is not dependent on other scheduling Periods. See Figure C.10 below as an example illustration of IDR positions and how the ALP buffer may go to zero (empty) at the end of each Media Segment. If it does not go empty, it is because a portion of the next Media Segment is already present. This description is valid for Media Segment delivery, irrespective of the ISO BMFF file delivery method for those Media Segments. If the Physical Layer has sufficient capacity and the Scheduler has access to later data, some data from the Period of a later Analyzed Media Duration may be sent during the Physical Layer frame(s) corresponding to previous Analyzed Media Duration(s).

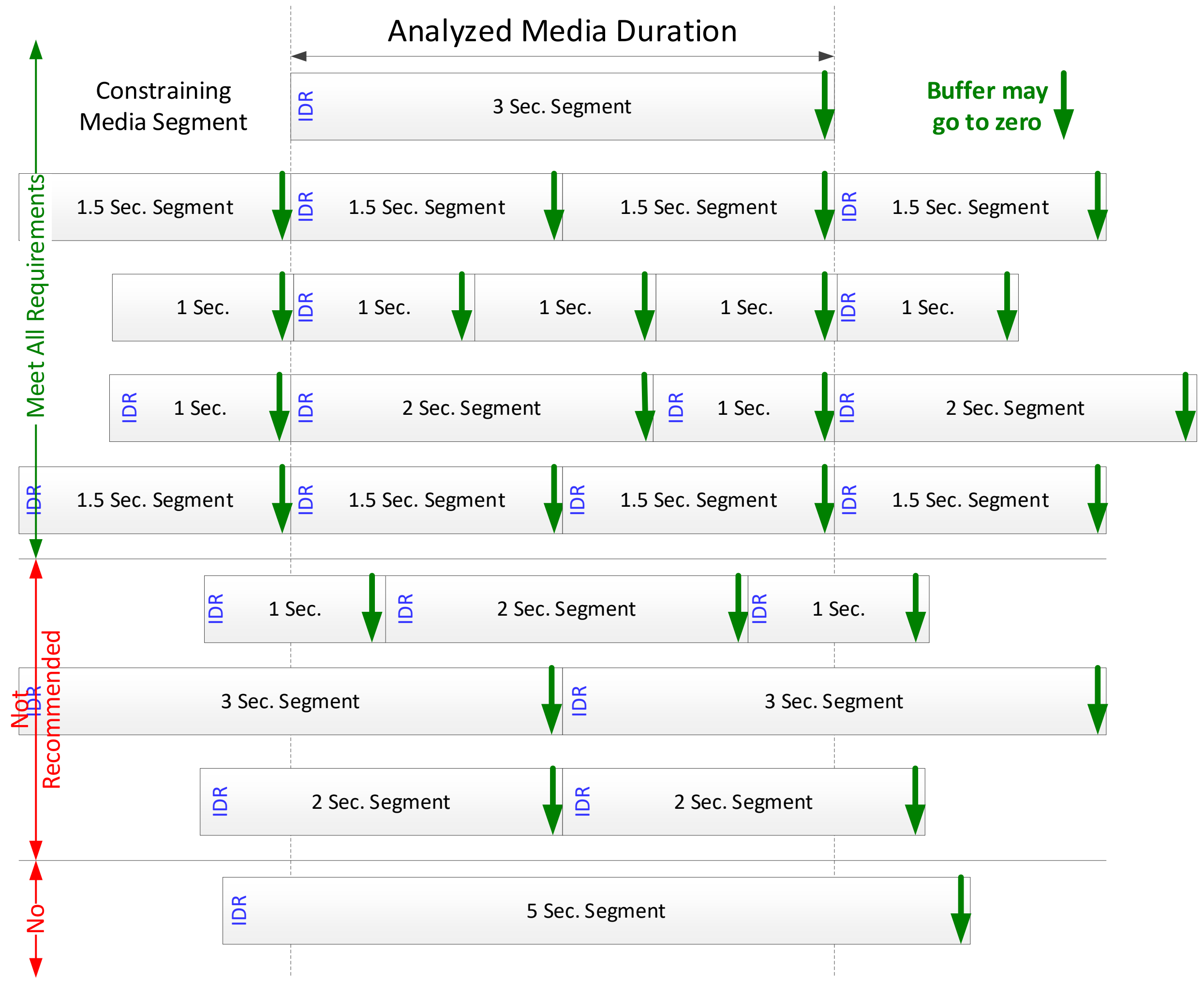


Figure C.10 Analyzed Media Duration.

Figure C.10 above has three main sections. The top section shows examples of Media Segment lengths providing an optimum ALP buffer length of one Media Segment (longest Media Segment among input ALP Streams). This Analyzed Media Duration starts with a Media Segment containing a SAP/RAP and ends with the ALP buffer going to zero. This is a sufficient set of conditions to allow single-Period scheduling. There is no need or requirement to have more than one Analyzed Media Duration in the ALP buffer. Such a configuration allows combinations of smaller Media Segments to be analyzed as well.

The middle 'Not Recommended' section of the figure shows examples of ALP buffer lengths that span more than one Media Segment. For the Scheduler to work properly, two Analyzed Media Durations need to be stored in this case.

The bottom section shows an example ALP buffer length that is less than the longest Media Segment present. This could require three Media Segment durations to schedule, which is not notionally supported. In this use case, the best approach would be to increase the Analyzed Media Duration and possibly align the Media Segment time boundary and the Analyzed Media Duration.

In general terms, the possibility of staggered Media Segment start times may be handled by a longer Analyzed Media Duration and use of two corresponding buffers.

There is an absolute time at which negotiations among Data Sources and the Scheduler must be completed and the data delivered. This time is a Data Delivery Deadline and the Scheduler needs to analyze entire Media Segments to meet this Data Delivery Deadline. Current Media Segments must be sent before next Media Segments are loaded to meet each Media Segment Data Delivery Deadline. The ALP buffer should send all Media Segment contents at the end of every Media Segment, independent of whether there is an IDR frame at the beginning. Every Media Segment may be used to ensure ALP buffer flushing. The time deadline for delivery of the complete Media Segment can result in the buffer going to zero. There is no requirement that the buffer go to zero, as it is possible for the Scheduler to "pull forward" media from a future Analyzed Media Duration. Consider that the earliest delivery time can allow the Scheduler to deliver media in a Physical Layer frame to be delivered ahead of the one that could meet the latest requirement.

On each input ALP Stream, Media Segment durations can vary (for example with layered coding), but they usually have an integer relationship in duration. For example, there can be 0.5-second Media Segments with 2-second enhancement Media Segments, or 1.5-second Media Segments with 3-second enhancement Media Segments, but preferably not 0.5-second Media Segments with 0.75-second enhancement Media Segments. Note that 0.5-second and 0.75-second segments could run in an Analyzed Media Duration of 1.5 seconds or 3 seconds. The constraining duration is the longest duration segment when all segment relationships are integer. The smallest Analyzed Media Duration allows all segment durations present to be present in integer numbers. Smaller Media Segments can combine to form the length of the largest Media Segment as shown in Figure C.10.

C.8 SEQUENCE OF REQUIRED DATA AND MEDIA EVENTS FOR ACQUISITION

Figure C.11 shows required data and associated events to achieve streaming media service with or without an application. There are two sequences of events. The first grouping is related to the Physical Layer. The Scheduler needs to understand that packets containing, for example, the SLT and SLS need to occur in tight time proximity to the Bootstrap and Preamble. The cyclic temporal location of the Bootstrap and Preamble likely is aligned to media T-RAP timeline, so as to minimize wait states. Multiple staggered media start times and T-RAPs may require that multiple Bootstraps and the associated signaling are required to minimize channel change time. If ROHC-U [16] header compression is being utilized, then there is a need to synchronize the context refresh to the T-RAP. This should be supported optionally as shown below in Figure C.11. The first Baseband Packets after the Preamble, as depicted below, may be delivered in a dedicated signaling PLP.

The delivery of an application may be incremental, which is to say that each of the utilized or utilizable Data Objects for the application may not be delivered entirely within the RAP. It is reasonable and good practice to define the T-RAP delivered portion of the app such that service acquisition and start may occur. Such details are out of scope for this document.

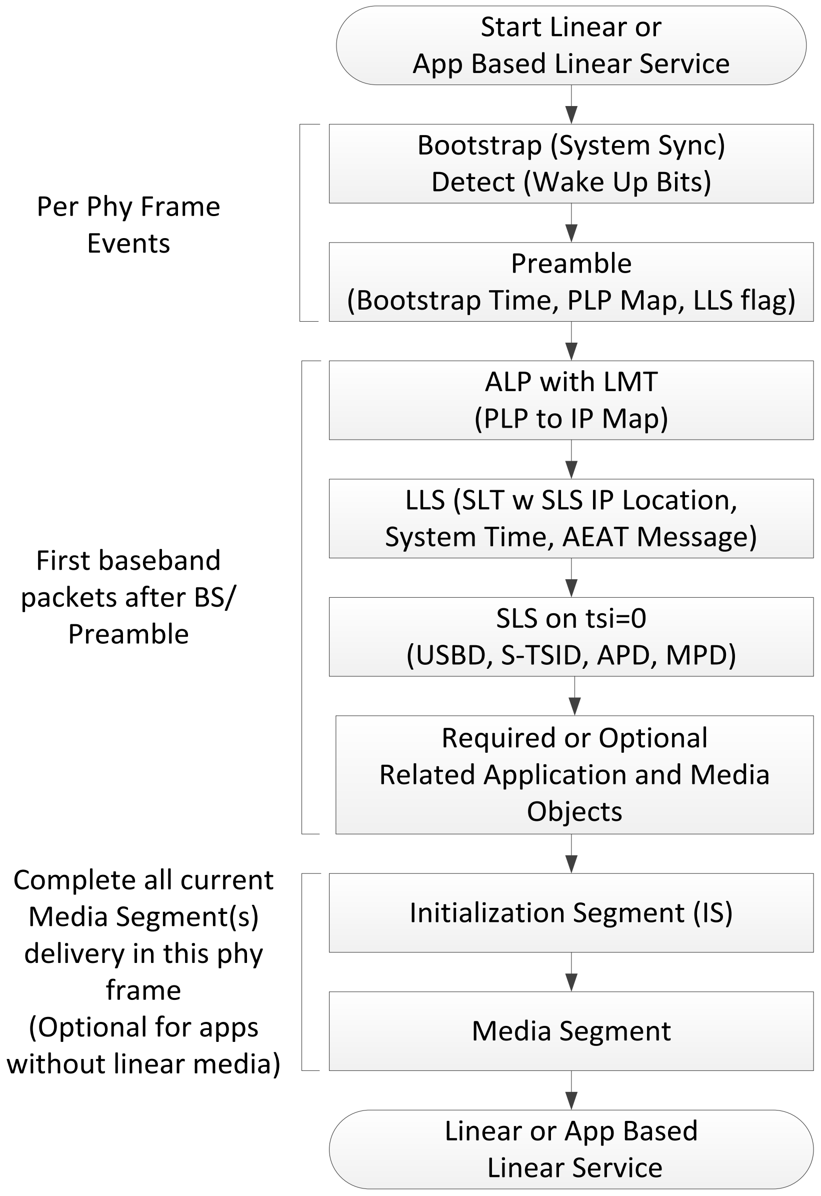

Start Linear or App Based Linear Service
Bootstrap (System Sync) Detect (Wake Up Bits)
Preamble (Bootstrap Time, PLP Map, LLS flag)
Per Phy Frame Events
ALP with LMT (PLP to IP Map)
LLS (SLT w SLS IP Location, System Time, AEAT Message)
SLS on tsi=0 (USBD, S-TSID, APD, MPD)
Required or Optional Related Application and Media Objects
First baseband packets after BS/ Preamble
Initialization Segment (IS)
Media Segment
Complete all current Media Segment(s) delivery in this phy frame (Optional for apps without linear media)
Linear or App Based Linear Service

Figure C.11 Order of events to optimize linear service acquisition w/wo an Application.

Annex D Unicast for Outer Tunneling Packets

As described in Section 6, the outer Tunnel Packets (CTP) are expected to be delivered through an RTP/UDP/IP Multicast protocol stack that uses a dedicated network link for reliable packet delivery. When such multicast capability is not available over the transmission link (e.g., when using a public network), the outer Tunnel Packets can be delivered through an RTP/UDP/IP Unicast protocol stack. Since a public network may not be reliable, which can cause packet drops over the transmission link, redundant Streams of the outer Tunnel Packets may be employed by enabling the **redundancy** field in Table 6.1, as shown in the example in Figure D.1.

In an ATSC 3.0 Transmitter, the redundant outer "Tunneling" Streams can be aggregated to further process in the ATSC 3.0 Physical Layer chain (BICM, Framing and Interleaving, and Waveform Generation). The ATSC 3.0 Transmitter may use the sequence numbers carried identically on each of the RTP/UDP/IP Unicast Streams to determine which packets have been received or lost along each of the multiple unicast paths. Note that since the intention of employing unicast is to use unidirectional delivery from a Broadcast Gateway to ATSC 3.0 Transmitter(s), the Broadcast Gateway will have to configure the unicast destination address(es) of the ATSC 3.0 Transmitter(s), which will be assigned by the Internet Service Provider provisioning each delivery service.

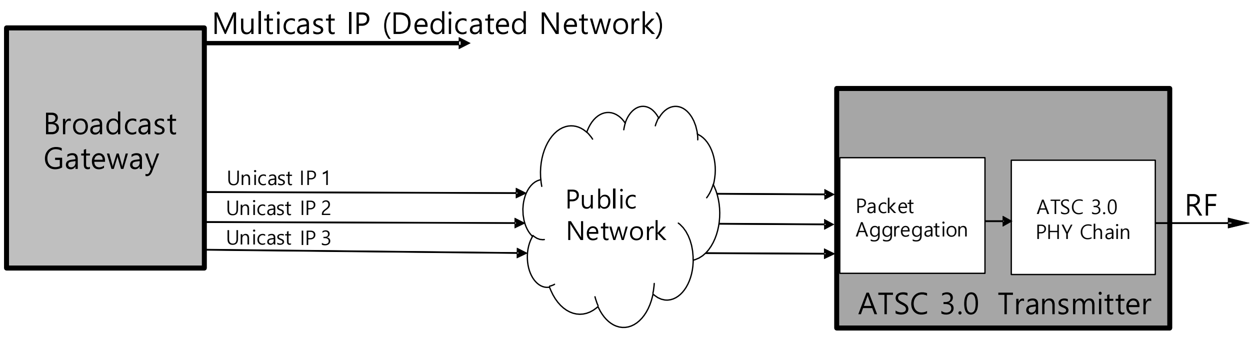


Figure D.1 RTP/UDP/IP Unicast for outer Tunnel Packets using CTP redundancy.

Annex E Bonded-Channel Distribution

Channel Bonding allows spreading of content over multiple channels. In the current version of this standard, the number of Bonded Channels is limited to two. See Figure E.1. Data is spread over the same PLP_IDs among these channels (per A/322 [3], Annex K).

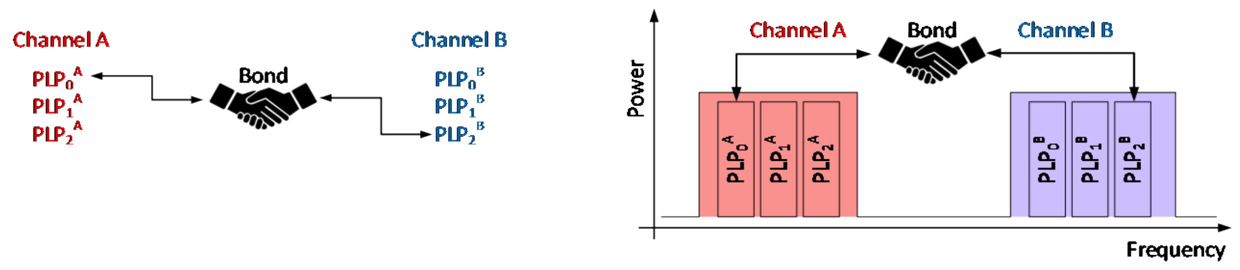


Figure E.1 Channel Bonding over PLP_ID 0 between Channels A and B.

As discussed in Section 9.5, the inner Tunneled Packet Streams for Channels A and B can be carried on two separate outer Tunnel Data Streams, one for each channel, as depicted below in Figure E.2.

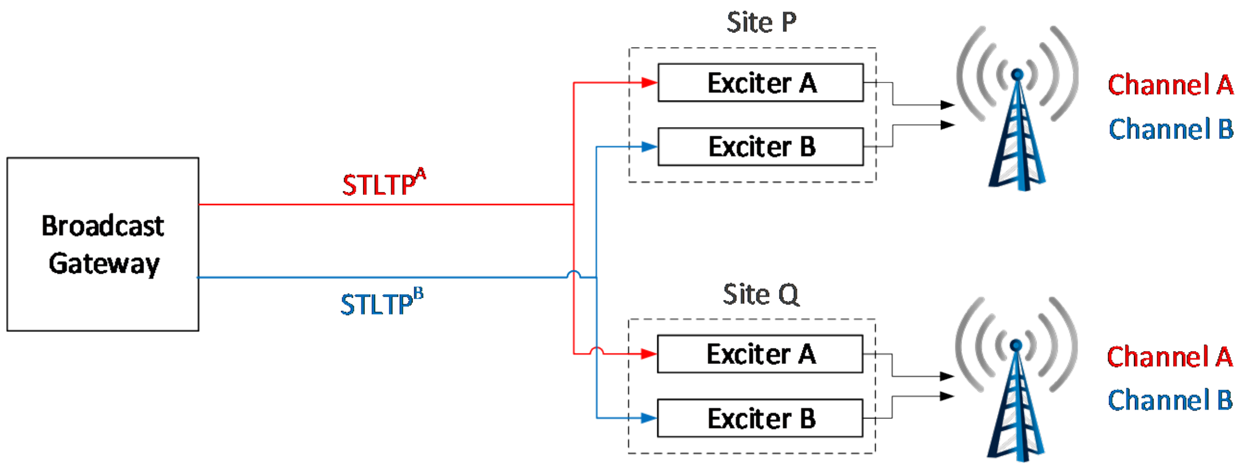


Figure E.2 Channel bonding with two separate STLTPs.

In the configuration of Figure E.2, the **number_of_channels** field is set to '0' in the RTP header of each Tunnel Data Stream. Each Tunnel Data Stream contains all the Tunneled Packet Streams required by Physical Layer processing.

Another strategy is to combine all the Tunneled Packet Streams inside one outer Tunnel Data Stream as depicted below in Figure E.3.

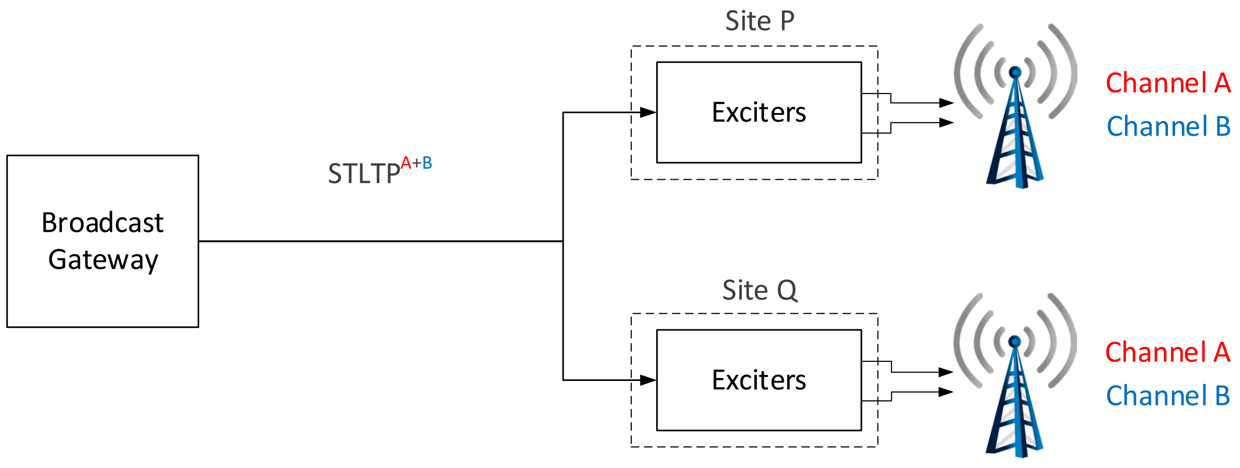


Figure E.3 Channel bonding with combined STLTP.

In this configuration, the **number_of_channels** field is set to '1' in the RTP header of the Tunnel Data Stream, indicating the presence of two complete sets of data Streams to populate two separate channels. The UDP port number ranges for the two groups of inner Tunneled Packet Streams are 30000 – 30066 and 30100 – 30166.

The Broadcast Gateway should support both the separate and combined STLTP configurations. The choice between them will depend on the Channel Bonding methods employed plus the network and Transmitter-site topologies.

For example, in the case of Plain Channel Bonding, the Broadcast Gateway may output two separate STLTP Tunnel Data Streams, one for each channel, as shown in Figure E.4. Since there is no cell exchange required in the modulator processing for Plain Channel Bonding, the Physical Layer chain generating the RF signal for a particular channel only needs the STLTP inner Tunneled Packet Streams for its channel.

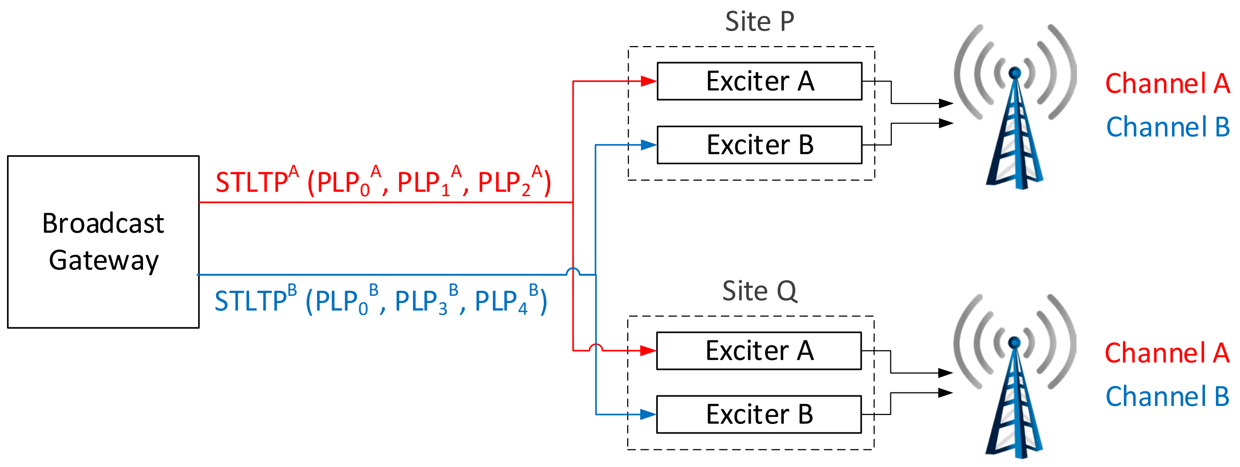


Figure E.4 Example of network topology for Plain Channel Bonding with co-located sites and separate STLTP Tunnel Data Streams.

In the case of SNR Averaging, cell-exchange processing between the two channels is required. Consequently, each exciter requires the inner Tunneled Packet Streams of the bonded PLPs of both channels. The Broadcast Gateway may output a single STLTP Tunnel Data Stream combining the inner Tunneled Packet Streams of both channels, as depicted below in Figure E.5.

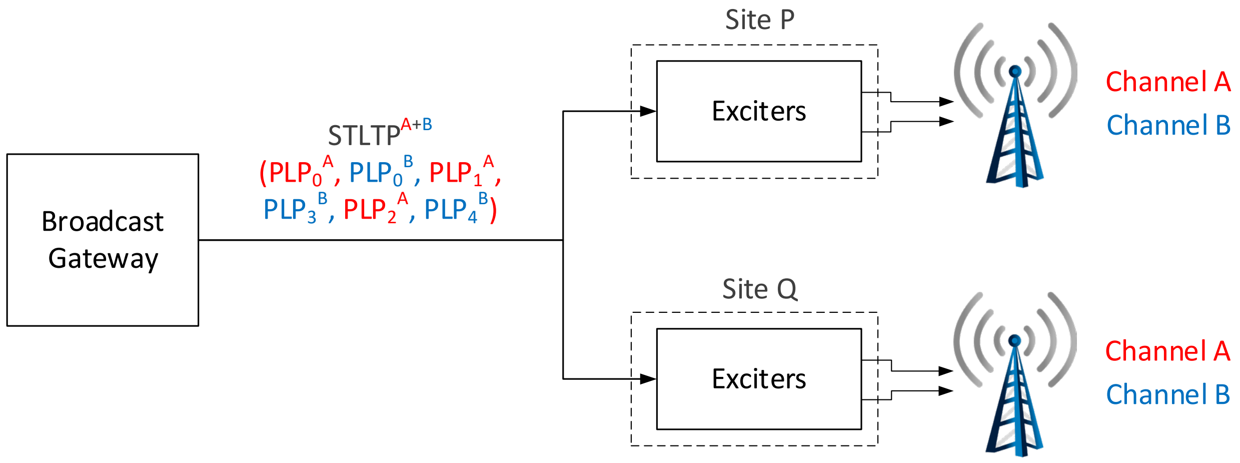


Figure E.5 Example of network topology for SNR Averaging with co-located sites and combined STLTP Tunnel Data Stream.

Annex F MIMO Configuration in Broadcast Gateway

When MIMO is used, as specified in Annex L of [3], a Broadcast Gateway generates a single STLTP Tunnel Data Stream that contains a single set of inner Tunneled Packet Streams intended for emission on a single channel (as opposed to the cases of Channel Bonding, in which two separate sets of Tunneled Packet Streams are sent to two functional exciters). In the current version of MIMO, two transmission chains are used after the generation of the bit-interleaved FEC Frames. Therefore, a Baseband Packet carried by a single STLTP Tunnel Data Stream is spread over the two transmission chains within a MIMO exciter, as depicted in Figure L.1.

For the parallel Framing & Interleaving stages that construct physical layer frames for transmitting antenna #1 (Tx1) and transmitting antenna #2 (Tx2), a Broadcast Gateway generates the same Preamble data and Timing & Management data for forming the signals to be emitted by Tx1 and Tx2. Since the data for a Baseband Packet is spread over two simultaneous physical layer frames emitted by Tx1 and Tx2, with equal amounts of data in each emission, care is required so that the Broadcast Gateway configures L1-Basic and L1-Detail parameters for the physical layer frames of only Tx1 or Tx2 but not both, rather than for the total of the input formatting data. (I.e., the values included in L1-Basic and L1-Detail will represent half of the total data transmitted within the combination of the two parallel physical layer frames.) For example, when MIMO is enabled (i.e., indicated via **L1B_first_sub_mimo** and **L1D_mimo**), **L1D_plp_size** configured by a Broadcast Gateway would indicate a PLP size contained in a physical layer frame of Tx1 (or Tx2) only, which represents half of the total input formatting data.

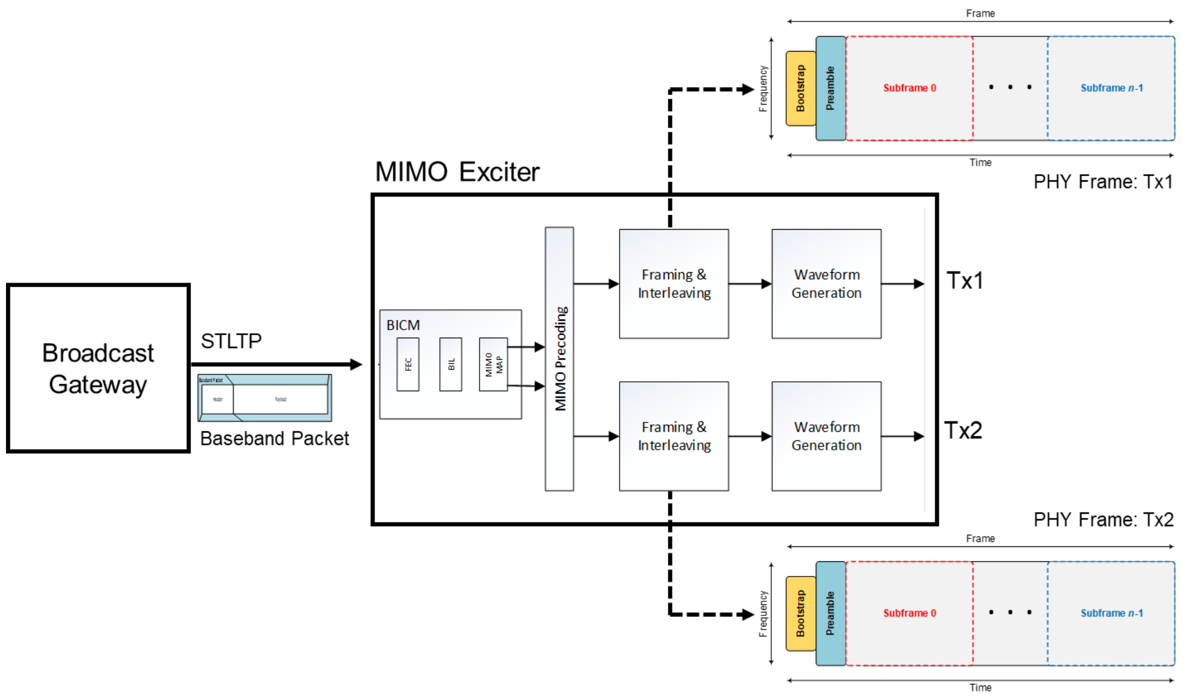


Figure F.1 MIMO Configuration in Broadcast Gateway and Exciter.

– End of Document –