Internet Engineering Task Force
Internet Draft
Document: draft-fairhurst-ipdvb-s2-gule-04.txt

Gorry Fairhurst
University of Aberdeen

Category: Draft

January 2006

An Encapsulation for Transmission of IP Datagrams over a DVB-S2 Generic Stream (GULE)

Status of this Draft

Abstract

This document describes a Generic stream Unidirectional Lightweight Encapsulation (GULE) mechanism for the transport of IPv4 and IPv6 Datagrams and other network protocol packets directly over the DVB S2 Generic Stream. This specifies a base encapsulation format and supports an extension format that allows it to carry additional header information to assist in network/Receiver processing.

Internet-Drafts can be and are used to describe technical issues and protocols that are of interest to the Internet community but that are not finally standardised within the IETF. This is such a document. It is intended for open discussion, but there is no current plan to publish this as an Internet RFC.

Table of Contents

1. Introduction

This document describes an encapsulation for transport of IP datagrams, or other network layer packets, over a link using the physical layer specified by DVB-S2 [ETSI-S2] in the Generic Mode (also known as the Generic Stream (GS)). Protocol requirements are defined in [S2-REQ], [S2-REQ-RCS], and [ID-S2-REQ]. Some recommended evaluation criteria are defined in [S2-REQ] and [S2-EVAL].

The method also allows TS-Packets [ISO-MPEG] to be sent within GULE SNDUs. This may be used to provide control information and/or Program Specific Information (PSI) for a Multiplex.

Note: Internet-Drafts can be and are used to describe technical issues and protocols that are of interest to the Internet community but that are not finally standardised within the IETF. This is such a document and is intended for open discussion.

1.1 SNDU and PDU Processing

Protocol Data Units, PDUs, (Ethernet Frames, IP datagrams or other network layer packets) for transmission are passed to an Encapsulator. This formats each PDU into a SubNetwork Data Unit (SNDU) by adding an encapsulation header and an integrity check trailer [RFC4259]. The SNDU may be fragmented into a series of fragments sent in one or more BaseBand (BB) frames that are sent over the DVB S2 link.

Although this document defines a different framing method to ULE [RFC4236], it maintains the same design philosophy as ULE for construction of the SNDUs. The important characteristics of this encapsulation are described in this subsection. This will allow subsequent additions to ULE to also be supported within GULE. The key features are that each SNDU has:

(i) D-bit
This permits both addressed and unaddressed SNDUs. The D=1 mode may be used as an enhancement to link efficiency (e.g. where IP-based address filtering is in use).

(ii) SNDU Length
This may be utilised in the SNDU framing, to skip unwanted SNDUs and to validate wanted SNDUs.

(iii) SNDU Type/Extension field
The Type field uses an extension processing in place of flag-parsing for options, providing a method for efficient option support. This field also permits the use of any IEEE format frame, including those

that control L2 infrastructure (QoS, VLAN, Network Management, ST, etc), bridging, and IETF-defined methods such as IPv4, IPv6, MPLS, arp, etc. The final use of this field is as a discriminator for IETF-defined encapsulation extensions. Currently envisaged

extensions are (a) Security extensions providing functions such as: encryption, identity hiding, and authentication methods; (b) Header compression methods for various formats of payloads.

An important consideration is that the Length and Type fields replicate information that may also be extracted from the network-layer protocol information. There are various reasons for this design:

(i) It is envisaged that SNDU parsing could be implemented in hardware/firmware or could use specialised logic to assist in parsing the SNDUs. In this case, access to the information may be required at a low-level to assure rapid filtering or unwanted packets and correct queuing of wanted packets. Omitting the information requires additional processing for all PDUs, irrespective of whether specific PDUs are forwarded to the specific Receiver, incurring significant additional computational cost.

(ii) Not all network-layer PDUs contain Type/Length information (e.g. DIX-format packets). When present, the fields are located at an offset from the start of each PDU (e.g. IPv4/IPv6). In some cases, this also requires computation (e.g. header skipping in IEEE/LLC).

(ii) Omitting the Type and Length field places a reliance of the SNDU framing on the PDU formats that are supported. This dependency will hinder evolution of the encapsulation for new applications.

If transmission efficiency is an important consideration, the duplicate protocol fields should be compressed at the PDU-level not the SNDU-level. This approach ensures protocol-independent framing of SNDUs; enable protocol-dependent compression (allowing the context of a particular protocol associations to provide sophisticated compression where needed); requires decompression only of those PDUs to be forwarded; and finally allows the decompression function to be off-loaded from the SNDU framing function (e.g. a design that does this as part of the Internet driver interface, rather than the device driver). The recommended approach is therefore to define well-founded and robust methods that may be used with both ULE and GULE.

1.2 Fragmentation Considerations

Three forms of fragmentation are provided in this encapsulation method:

(i) The simplest (default) provides efficient fragmentation and reassembly for consecutive fragmentation (adjacent BBFrames) [S2-REQ]. This follows a procedure that resembles that used by ULE [RFC4236], and common to other MPEG-2 TS formats [ISO-

MPEG]. This mode requires each Receiver to maintain one buffer for reassembly of the Current SNDU [ULE-RFC]. A constraint is that a BBFrame may contain only the fragment of a single new SNDU, which must be aligned to the end of the BBFrame.

(ii) The SNDU-Resume Extension Header defined in this document provides a method for non-consecutive fragmentation with multiplexing [S2-REQ] (i.e. where intervening BBFrames may be sent using an arbitrary ModCod value). When used in the latter way, an arbitrary large number of fragments may be resumed in the same BBFrame (i.e. multiple fragment support [S2-REQ]). A constraint is that a BBFrame may contain only the fragment of a single new SNDU, which must be aligned to the end of the BBFrame.

This method requires an additional protocol overhead within each resumed fragment. To utilise this method, Receivers need to maintain several buffers, up to one buffer for each MAC/NPA address that they expect to receive, plus one for the case of no specified MAC/NPA address (D=1). Receivers that allocate fewer buffers, may not be able to buffer all fragmented SNDUs, potentially resulting in some packet loss (depending on the fragmentation policy of the Gateway). This document defines a method to recover buffers that are occupied by incomplete fragments after a specified interval.

(iii) The SNDU-Suspend Extension Header defined in this document provides a method for non-consecutive fragmentation with multiplexing [S2-REQ] that also supports an arbitrary fragmentation of the first fragment of a new SNDU. This method must be used in combination with the SNDU-Resume method and has the same buffer management requirements as for the SNDU-Resume method. Using this method, a BBFrame may contain any number of fragments that start a new SNDU. New fragmented SNDUs do not need to be aligned to the end of the BBFrame.

To utilise this method, a Receiver needs to be able to accept an arbitrary large number of new fragmented new SNDUs within a BBFrame. This may add complexity to the Receiver design, but permits more flexibility in fragmentation by the Gateway.

These methods provide backwards compatibility: All Receivers MUST implement method (i). If a Receiver does not also implement methods (ii) and (iii), this Receiver will silently discard all PDUs sent using the SNDU-Resume or SNDU-Suspend method. A Receiver that implements the SNDU-Resume method may use method (i) and/or (ii), but will silently discard all PDUs sent using the SNDU-Suspend method (iii). A Receiver that implements the SNDU-Resume or SNDU-

Suspend method is able to receive and forward PDU encapsulated using any/all of the three methods.

2. Conventions used in this document

The capitalized key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

ACM: Adaptive Coding and Modulation (see ModCod).

b: bit. For example, one byte consists of 8b.

B: Byte. Groups of bytes are represented in Internet byte order.

BB: Baseband.

BBFrame payload [ETSI-S2]: The data field part of a Baseband frame that may be used for the communication of data. Typical BBFrames range in size from 3072 to 58192 bits according to the choice of modulation format and FEC in use.

BBH, Baseband Header [ETSI-S2]: A fixed format 10 byte header that starts each BBFrame.

Counter: An encapsulation protocol field located at a fixed position in all BBFrames utilising the GULE encapsulation. The value is incremented modulo 255 in each BBFrame sent with a specific ISI. It is used to monitor in-sequence reception of BBFrames within a Stream, and may be utilised for monitoring functions as a part of link-layer Operations and Management (OAM). In addition, the value is used to verify that BBFrames are consecutive within a Stream when the PP value is used to reassemble frames.

DF: Data Field [ETSI-S2]. The BBFrame payload. Note this abbreviation is also used for a different function at the IP layer.

DFL: Data Field Length [ETSI-S2]. The size of the BBFrame payload measured in bits. A set of DFL values have been specified by S.2, corresponding to the set of specified ModCod formats.

DVB: Digital Video Broadcast [ETSI-DVB]. A framework and set of associated standards published by the European Telecommunications Standards Institute (ETSI) for the transmission of video, audio, and data.

Encapsulator: A network device that receives PDUs and formats these into Payload Units (known here as SNDUs) for output in the Generic Mode of DVB-S2.

End Indicator [RFC4236]: A value that indicates to the Receiver that

there are no further SNDUs present within the current BBFrame.

Encapsulation Header: The logical group of fields that carry the protocol control information for the encapsulation protocol.

GULE: Generic Stream ULE. The protocol defined in this document.

GS: Generic Stream.

ISI: Input Stream Identifier, the second byte of the header of a BBFrame. This value uniquely identifies a specific logical channel within a DVB-S2 multiplex.

MAC: Medium Access Control [IEEE-802.3]. A link layer protocol defined by the IEEE 802.3 standard (or by Ethernet v2 [DIX]).

ModCod: Modulation/Coding. A combination of Modulation and FEC Coding that together determine the size of a BBFrame.

MPEG-2: A set of standards specified by the Motion Picture Experts Group (MPEG), and standardized by the International Standards Organisation (ISO/IEC 113818-1) [ISO-MPEG], and ITU-T (in H.220).

Next-Header: A Type value indicating an Extension Header [RFC4236].

NPA: Network Point of Attachment [RFC4236]. In this document, refers to a destination address (resembling an IEEE MAC address) within the DVB-S2 transmission network that is used to identify individual Receivers or groups of Receivers.

OAM: Operations and Management.

Packing Threshold: In GULE, this is a period of time an Encapsulator is willing to defer transmission of a partially filled BBFrame to accumulate more SNDUs, rather than use Padding. After the Packet Threshold period, the Encapsulator uses Padding to send the partially filled BBFrame.

Padding: A sequence of bytes with the value 0xFF that are used after an End Indicator to occupy the unused portion of a BBFrame payload. The DFL field may also be used in place of Padding.

PDU: Protocol Data Unit [RFC4259]. Examples of a PDU include Ethernet frames, IPv4 or IPv6 datagrams, and other network packets.

PP, Payload Pointer: In GULE, the PP is a 13-bit pointer located at a fixed position in the encapsulation header of all BBFrames utilising the GULE encapsulation. If the BBFrame contains the start of a GULE SNDU, the PP value SHALL be set to the position of the first byte of the first SNDU within the BBFrame. If the BBFrame contains neither the start of a SNDU (or an End Indicator), the value SHALL be assigned 0xFFFF.

PSI: Programme Specific Information [ISO-MPEG].

SI Table: Service Information Table [ISO-MPEG]. In this document, this term describes a table that is been defined by another

standards body to convey information about the services carried in a

SNDU: Subnetwork Data Unit [RFC4259]. In this document this is an encapsulated PDU sent using GULE.

Stream: A logical flow from an Encapsulator to a set of Receivers. The addresses at the MAC/NPA level and IP level need to be unique within a specific stream (i.e. denoted by a common S.2 ISI value).

SYNCD: Synchronisation Distance [ETSI-S2]. This is specified for a DVB-S2 packetized stream, where the distance in bits from the start of the DataField to the first start of packet contained in this DataField. This use resembles the MPEG-2 Payload Pointer, as defined in ULE. The use of SYNCD is not specified for continuous Generic Streams.

TS: Transport Stream [ISO-MPEG], a method of transmission at the MPEG-2 level using TS Packets; it represents layer 2 of the ISO/OSI reference model.

ULE: Unidirectional Lightweight Encapsulation (ULE) [RFC4236]. A scheme that encapsulates PDUs, into SNDUs that are sent in a series of TS Packets using a single TS Logical Channel. The encapsulation format defined by this document shares the same extension format, and basic processing rules and uses a common IANA Registry.

3. Description of the Method

PDUs (IP packets, Ethernet frames or packets from other network protocols) are encapsulated to form a Subnetwork Data Unit (SNDU) [RFC4236] associated with a specific Stream at the link layer. The SNDU is transmitted over an DVB-S2 link by placing it either in a single BBFrame, or if required, an SNDU may be fragmented into a series of BBFrames. A mode also permits an SNDU to be suspended and resumed in a later BBFrame, while preserving the original order of each SNDU sent to a specific MAC/NPA address over a Stream. Where there is sufficient space, the method permits a BBFrame to carry more than one SNDU (or part there of), sometimes known as Packing. All parts comprising an SNDU MUST be assigned to the same Stream.

If a SNDU finishes before the end of a BBFrame, but it is not intended to start another SNDU, a stuffing procedure fills the remainder of the BBFrame payload with bytes with a value 0xFF, known as Padding.

A Receiver that receives a value of 0xFF in place of the start of a SNDU, interprets this as Padding/Stuffing and silently discards the remainder of the BBFrame payload. The payload of the next BBFrame for the same stream will begin with a Payload Pointer of value 0x0000, indicating that the next SNDU immediately follows the encapsulation header. The ULE protocol [RFC4236] resembles this, but differs in the procedure that binds it to the MPEG-2 TS physical layer.

3.1 GULE Encapsulation Overhead

Each BBFrame includes a logical encapsulation header. This overhead comprises a set of fixed format fields defined by GULE, shown in the figure below.

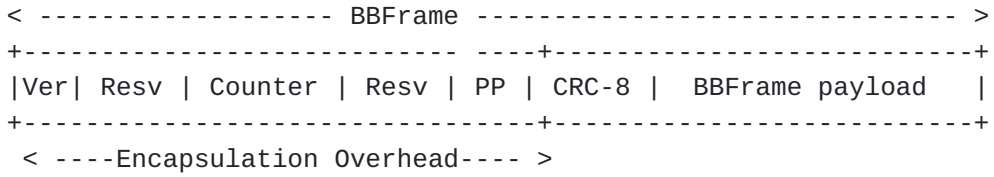


Figure 1: BBFrame Logical Encapsulation Overhead

The BBFrame encapsulation is identified by the 4-bit version field. There are also a BBFrame counter (size to be determined), a 13-bit Payload Pointer (aligned to an 8 byte boundary) and an 8-bit CRC-8. All unused fields are reserved for future use. The BBFrame payload is also known as the User Payload [ETSI-S2].

A CRC-8 provides an integrity check for the encapsulation overhead. It also guards against framing misalignment that may otherwise result following corruption of the PP value. The polynomial for computation of the CRC-8 will be defined in this document.

Field	Size of Field	Notes
-	-	Identifies the GULE format.
ver	(4 bits + 4 reserved bits)	*1
Counter	(8 bits)	For use in OAM *2
PP	(13 bits)	Measured in bytes.
	(3 bits)	Reserved for future use.
CRC	CRC-8	Overhead integrity check.

Notes:
*1 Usage to be confirmed.
*2 Also used to verify that BBFrames are consecutive within a Stream when the PP value is used to reassemble SNDUs.

Table 1: Encapsulation Overhead Protocol Fields

The specification described by this document does not define a transmission format for the logical encapsulation header. The approach recommended in this document is to include this encapsulation overhead within the BBFrame Header itself (Section 3.2). An alternative is to prefix as a fixed-sized protocol header within the BBFrame payload, placed following the BBFrame header defined by DVB-S2 (Section 3.3).

3.2 Placement of the Encapsulation Header within the BBHeader

This section describes the physical placement of the extension overhead fields within the header of a BBFrame [ID-S2-REQ], known as the BaseBand Header (BBH) [ETSI-S2]. These fields are placed into positions that are currently unused within the DVB S.2 BBH when sending in the Generic Mode. This placement eliminates the need for a separate Encapsulation header.

Field	Corresponding BB Header Field	Notes
	MATYPE (2B)	Identifies the GULE format.
	ISI (1B subfield)	Identifies the stream.
0000	User Payload Length, UPL (2B)*1	Identifies the GULE format.
	Data Field Length, DFL (2B)*2	As used in S.2 TS.
ver	Sync Pattern SYNCD (1B)*1	
Counter		For use in OAM.
PP	Sync Distance SYNCD (2B)*1	Equivalent function.
CRC	CRC-8 *2	Function already provided.

Notes:
*1 The usage of this BBH field is not specified in the DVB-S2 Specification for the Generic Mode [ID-S2-REQ].
*2 This BBH field is specified in the S.2 Specification and use in

the Generic Mode [ID-S2-REQ] is identical to use for Transport Streams [ISO-MPEG].

Table 2: BBFrame Header Structure

Usage of these fields (as in Table 2) within the BB header is subject to clarification of acceptable usage for the Generic Mode within the DVB-S2 Standard [ETSI-S2].

3.3 BB Encapsulation Header within the BBFrame Payload

This section describes an alternative method for communicating the encapsulation protocol overhead, in the case where the fields in the BBFrame header are not accessible to the GULE Gateway/Receiver.

The payload of each BBFrame starts with an encapsulation header defined by GULE. The GULE encapsulation header is sent as the first bytes of every BBFrame transmitetd by an encapsulation Gateway.

The BBFrame encapsulation header is of a fixed format. This comprises an 8-bit BBFrame counter, a 16-bit Payload Pointer field and an 8-bit CRC-8.

XXX Future revisions of this document may define additional fields within this header to accelerate processing of the GULE Receiver. XXX

The CRC-8 provides an integrity check for the entire GULE encapsulation header, and guards against framing mis-alignment that may otherwise result following corruption of the PP value. The polynomial for computation of the CRC-8 is identical to that specified for the BBHeader in [DVB-S2].

4. SNDU Format

This section is Informative, the formats described in this section are defined by ULE [RFC4236].

PDUs are encapsulated using ULE to form an SNDU. The encapsulation format to be used for PDUs is illustrated below:

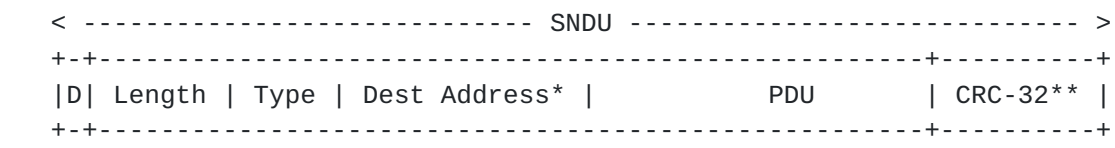


Figure 2: SNDU Encapsulation (* optional Destination Address) (** CRC-32 is optional for some formats of SNDU)

This format is identical that defined for ULE [RFC4236]. All multi-byte values (including the Length/End Indicator (4.2,4.3), Type (4.4), Destination Address (4.5), and Extension Headers (5)) are transmitted in network byte order (most significant byte first). The most significant bit of each byte is placed in the left-most position of the 8-bit field.

4.1 Destination Address Absent (D) Field

The most significant bit of the Length Field carries the value of the Destination Address Absent Field, the D-bit. A value of 0 indicates the presence of the Destination Address Field (see section 4.5). A value of 1 indicates that a Destination Address Field is not present.

An End Indicator (4.3) MUST be sent with a D-bit value of 1. Other SNDUs SHOULD be sent with a D-bit value of 0 (see 4.5).

4.2 Length Field

A 15-bit value that indicates the length, in bytes, of the SNDU counted from the byte following the Type field, up to and including the CRC (when present). Note the special case described in 4.3.

4.3 End Indicator

When the first two bytes of an SNDU have the value 0xFFFF, this denotes an End Indicator (i.e., all ones length combined with a D-bit value of 1). This indicates to the Receiver that there are no further SNDUs present within the current BBFrame (see section 6),

and that no Destination Address Field is present.

4.4 Type Field

The 16-bit Type field indicates the type of payload carried in an SNDU, or the presence of a Next-Header. The set of values that may be assigned to this field is divided into two parts, similar to the allocations for Ethernet.

The first set of ULE Type field values comprise the set of values less than 1536 in decimal. These Type field values are IANA assigned (see 4.4.1), and indicate the Next-Header.

The second set of ULE Type field values comprise the set of values greater than or equal to 1536 in decimal. In ULE, this value is identical to the corresponding type codes specified by the IEEE/DIX type assignments for Ethernet and recorded in the IANA EtherType registry.

4.4.1 Type 1: Next-Header Type Fields

The first part of the Type space corresponds to the values 0 to 1535 Decimal. These values may be used to identify link-specific protocols and/or to indicate the presence of Extension Headers that carry additional optional protocol fields. The format is defined by ULE and the ULE registry maintained by the IANA.

4.4.2 Type 2: EtherType Compatible Type Fields

The second part of the Type space corresponds to the values between 0x600 (1536 decimal) and 0xFFFF. This set of type assignments follow DIX/IEEE assignments (but exclude use of this field as a frame length indicator). All assignments in this space MUST use the values defined for IANA EtherType, the following two Type values are used as examples (taken from the IANA EtherTypes registry):

0x0800: IPv4 Payload (see 4.7.2)
0x86DD: IPv6 Payload (see 4.7.3)

4.5 SNDU Destination Address Field

The SNDU Destination Address Field is optional (see 4.1). This field MUST be carried (i.e. D=0) for IP unicast packets destined to routers that are sent using shared links (i.e., where the same link connects multiple Receivers). A sender MAY omit this field (D=1) for an IP unicast packet and/or multicast packets delivered to Receivers that are able to utilise a discriminator field (e.g. the IPv4/IPv6 destination address, or a bridged MAC destination address), which in

combination with the PID value, could be interpreted as a Link-Level address.

When the SNDU header indicates the presence of an SNDU Destination Address field (i.e. D=0), a Network Point of Attachment, NPA, field directly follows the fourth byte of the SNDU header. NPA destination addresses are 6 Byte numbers, normally expressed in hexadecimal, used to identify the Receiver(s) in a DVB-S2 transmission network that should process a received SNDU. The value 0x00:00:00:00:00:00, MUST NOT be used as a destination address in an SNDU. The least significant bit of the first byte of the address is set to 1 for multicast frames, and the remaining bytes specify the link layer multicast address. The specific value 0xFF:FF:FF:FF:FF:FF is the link broadcast address, indicating this SNDU is to be delivered to all Receivers.

IPv4 packets carrying an IPv4 subnetwork broadcast address need to be delivered to all systems with the same network prefix. When a SNDU Destination Address is present (D=0) the value MUST be set to the NPA link broadcast address (0xFF:FF:FF:FF:FF:FF).

When the PDU is an IP multicast packet and an SNDU Destination Address is present (D=0), the IP group destination address of the multicast packet MUST be mapped to the multicast SNDU Destination Address (following the method used to generate a destination MAC address in Ethernet). The method for mapping IPv4 multicast addresses is specified in [RFC1112]. The method for mapping IPv6 multicast addresses is specified in [RFC2464].

4.6 SNDU Trailer CRC

A SNDU may carry a 32-bit CRC field in the last four bytes of the SNDU. This position eases CRC computation by hardware. The CRC-32 polynomial is to be used. Examples where this polynomial is also employed include Ethernet, DSM-CC section syntax [ISO-DSMCC] and AAL5 [ITU-3563]. This is a 32 bit value calculated according to the generator polynomial represented 0x104C11DB7 in hexadecimal:

x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x^1+x^0.

The Encapsulator initialises the CRC-32 accumulator register to the value 0xFFFF FFFF. It then accumulates a transmit value for the CRC32 that includes all bytes from the start of the SNDU header to the end of the SNDU (excluding the 32-bit trailer holding the CRC-32), and places this in the CRC Field. In ULE, the bytes are processed in order of increasing position within the SNDU, the order of processing bits is NOT reversed. This use resembles, but is different to that in SCTP [RFC3309].

When present, the Receiver performs an integrity check by independently calculating the same CRC value and comparing this with

the transmitted value in the SNDU trailer. SNDUs that do not have a valid CRC, are discarded.

This description may be suited for hardware implementation, but this document does not imply any specific implementation. Software-based table-lookup or hardware-assisted software-based implementations are also possible. Annexe B provides an example of an Encapsulated PDU that includes the computed CRC-32 value.

The purpose of this CRC is to protect the SNDU (header, and payload) from undetected reassembly errors and errors introduced by unexpected software / hardware operation while the SNDU is in transit across the DVB-S2 subnetwork and during processing at the encapsulation gateway and/or the Receiver. It serves to verify the delineation (framing) of PDUs and may also detect the presence of uncorrected errors from the physical link (however, these may also be detected by other means, e.g. section 7.3). When a MAC?NPA address is present, it binds the SNDU to the specified address.

The presence of a CRC field is determined by the format of the SNDU (specified in the Type field of the Header). Formats by default SHOULD include a CRC, exceptions include amn End Indicator, the SNDU-Suspend and SNDU-Resume fragments.

4.7 Description of SNDU Formats

The format of an SNDU is determined by the combination of the Destination Address Absent bit (D) and the SNDU Type Field. The simplest encapsulation places a PDU directly into an SNDU payload. Some Type 1 encapsulations may require additional header fields. These are inserted in the SNDU following the NPA/MAC destination address and directly preceding the PDU.

Formats may be defined through relevant assignments in the IEEE and IANA registries.

4.7.1 End Indicator

The format of the End Indicator is defined by ULE [ULERFC]. It is shown in figure 2. This format MUST carry a D-bit value of 1.

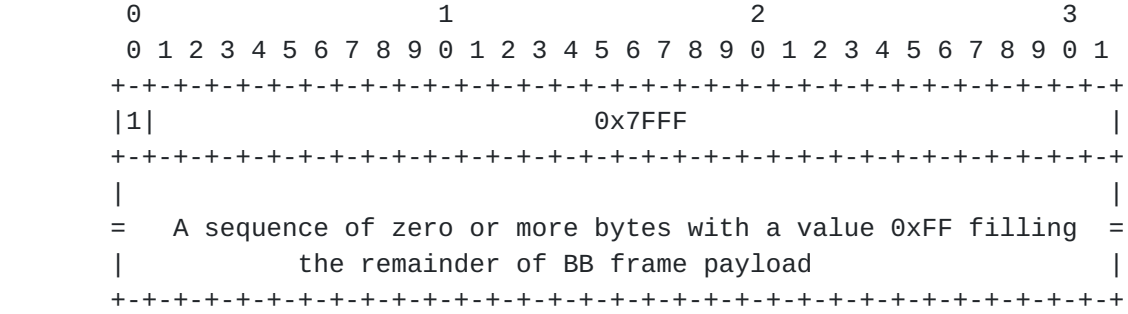


Figure 3: Format for a ULE End Indicator.

4.7.2 IPv4 SNDU

IPv4 datagrams are directly transported using one of the two standard SNDU structures, in which the PDU is placed directly in the SNDU payload. The formats are defined by ULE [ULERFC]. The two encapsulations are shown in figures 4 and 5. (Note that in this, and the following figures, the IP datagram payload is of variable size, and is directly followed by the CRC-32).

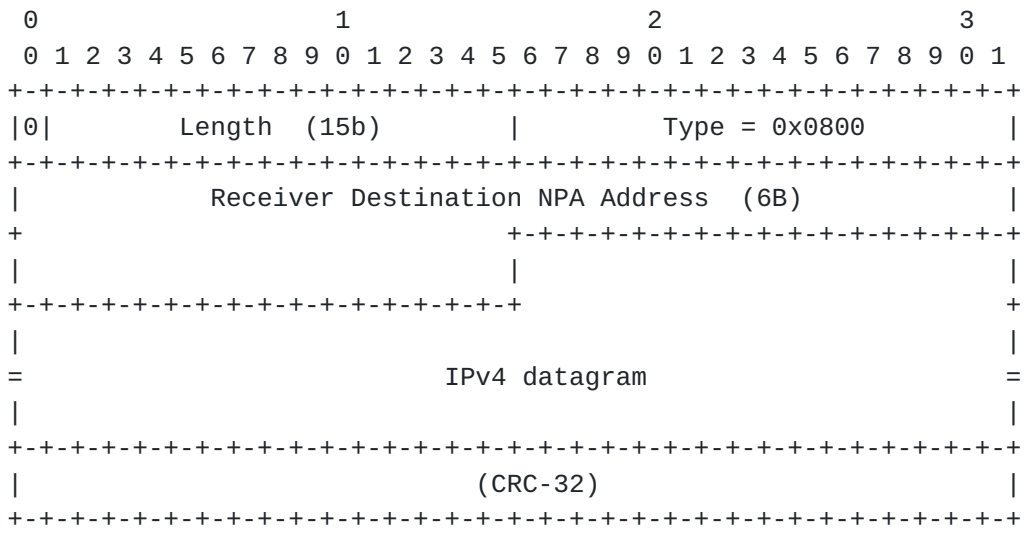


Figure 4: SNDU Format for an IPv4 Datagram using L2 filtering (D=0).

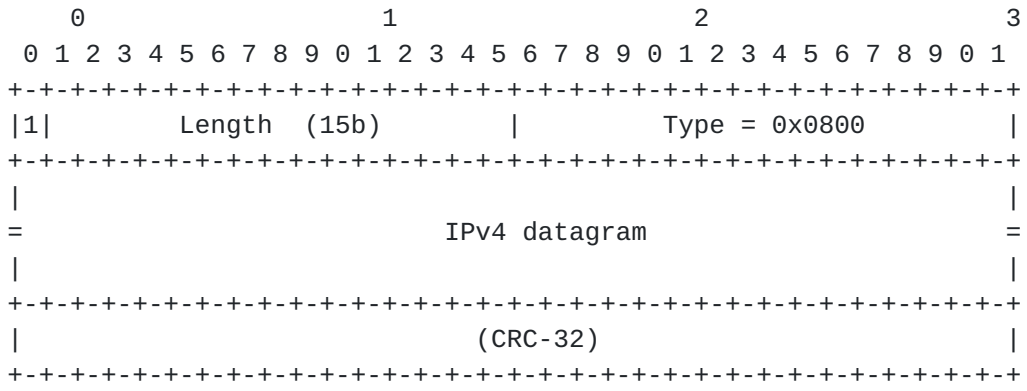


Figure 5: SNDU Format for an IPv4 Datagram using L3 filtering (D=1).

4.7.3 IPv6 SNDU Encapsulation

IPv6 datagrams are directly transported using one of the two standard SNDU structures, in which the PDU is placed directly in the SNDU payload, as defined by ULE [ULERFC]. The two encapsulations are shown in figures 6 and 7.

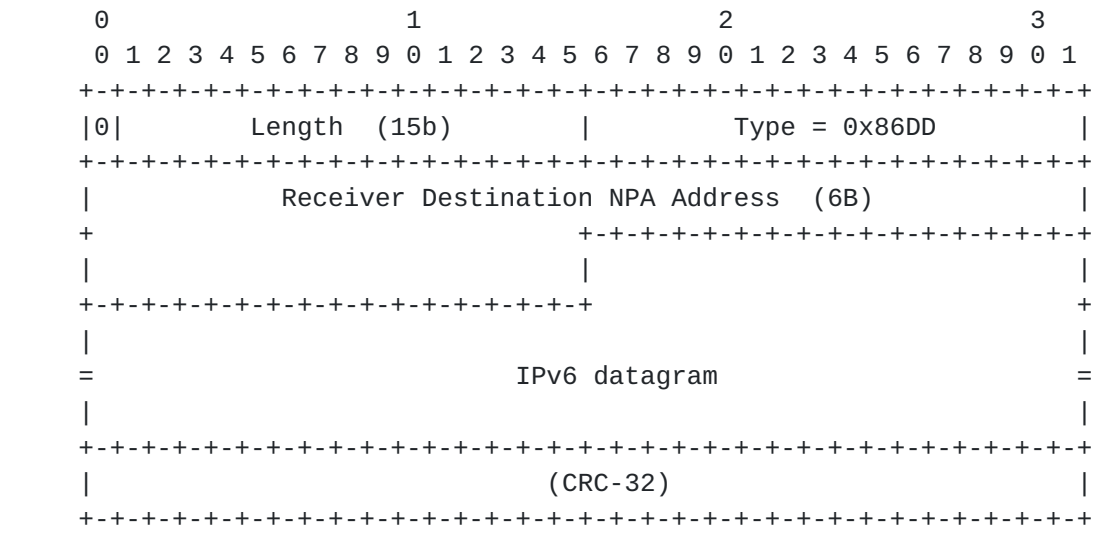


Figure 6: SNDU Format for an IPv6 Datagram using L2 filtering (D=0).

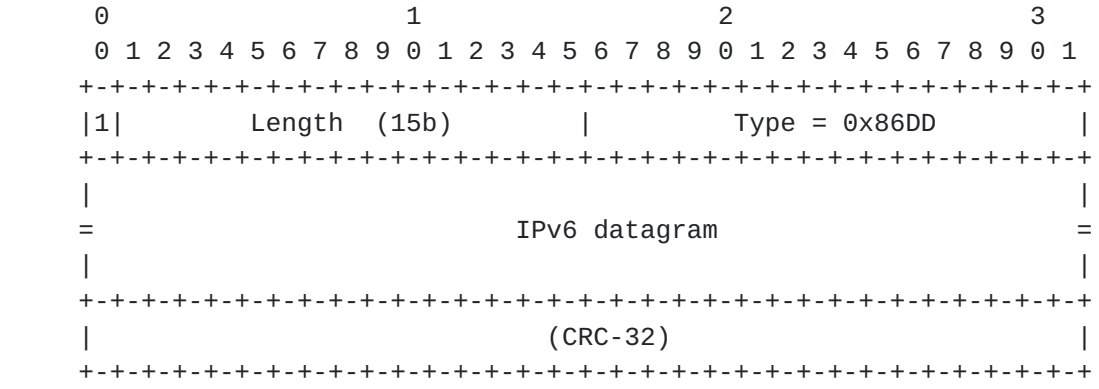


Figure 7: SNDU Format for an IPv6 Datagram using L3 filtering (D=1)

5. Extension Headers

This section describes an extension format for the ULE encapsulation. In ULE, a Type field value less than 1536 Decimal indicates an Extension Header. These values are assigned from a separate IANA registry defined for ULE [ELE-RFC].

The use of a single Type/Next-Header field simplifies processing and eliminates the need to maintain multiple IANA registries. The cost is that each Extension Header requires at least 2 bytes. This is justified, on the basis of simplified processing and maintaining a simple lightweight header for the common case when no extensions are present.

A ULE Extension Header is identified by a 16-bit value in the Type field. This field is organised as a 5-bit zero prefix, a 3-bit H-LEN field and an 8-bit H-Type field, as follows:

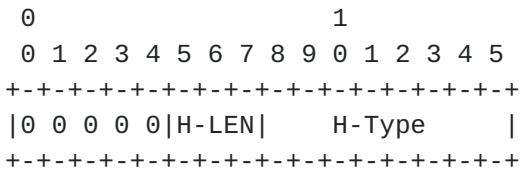


Figure 8: Structure of ULE Next-Header Field [RFC4236].

The H-LEN value indicates the total number of bytes in an Optional Extension Header (including the 2B Type field) [RFC4236].

An H-LEN value of zero indicates a Mandatory Extension Header. Each Mandatory Extension Header has a pre-defined length that is not communicated in the H-LEN field. No additional limit is placed on the maximum length of a Mandatory Extension Header. A Mandatory Extension Header MAY modify the format or encoding of the enclosed PDU (e.g. to perform encryption and/or compression).

The H-Type is a one byte field that is either one of 256 Mandatory Header Extensions or one of 256 Optional Header Extensions. This is defined by ULE [RFC4236].

The general format for an SNDU with Extension Headers is:

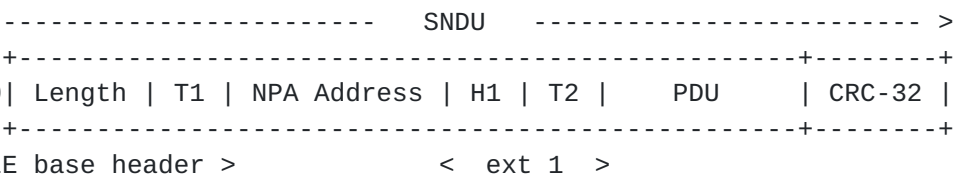


Figure 9: SNDU Encapsulation with one Extension Header (for D=0).

Where:
D is the ULE D_bit (in this example D=0, however NPA addresses may

also be omitted when using Extension Headers).
T1 is the base header Type field. In this case, specifying a Next-Header value.
H1 is a set of fields defined for header type T1. There may be 0 or more bytes of information for a specific ULE Extension Header.
T2 is the Type field of the next header, or an EtherType > 1535 B indicating the type of the PDU being carried.

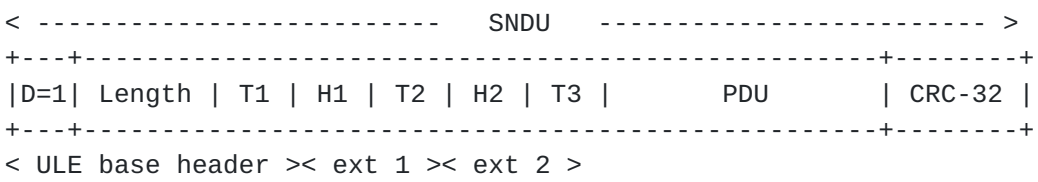


Figure 9: SNDU Encapsulation with two Extension Headers (D=1).

Using this method, several Extension Headers MAY be chained in series [RFC4236]. Although an SNDU may contain an arbitrary number of consecutive Extension Headers, it is not expected that SNDUs will generally carry a large number of extensions.

5.1 Test SNDU

A Test SNDU is a Mandatory Extension Header of Type 1, specified by ULE [RFC4236].

5.2 Bridge Frame SNDU Encapsulation

A bridged SNDU is a Mandatory Extension Header of Type 1 specified by ULE [RFC4236].

5.3 Extension-Padding Optional Extension Header

The Extension-Padding Optional Extension Header is s specified by ULE [RFC4236].

5.4 MPEG-2 TS Extension

The MPEG-2 TS Extension Header is specified by an IANA assigned H-Type value of <tbd>. This is a Mandatory Extension. The extension is used to communicate 1 or more MPEG-2 TS Packets over the DVB-S2 link when utilising the Generic Mode. The number of TS Packets carried in a specific SNDU is determined from the size specified by the remainder of the Payload following the MPEG-2 TS Extension. A Receiver MUST silently discard any data, if present, between the last complete encapsulated MPEG-2 TS Packet and the size denoted in the Length field of the SNDU fragment base header.

A value of D=1 indicates there is no NPA/MAC address in use. A value D=0 is also permitted, as defined in ULE.

The extension may be used to communicate one or more MPEG-2 TS Packets of arbitrary content, interpreted according to [ISO-MPEG]. One expected use is for the transmission of MPEG-2 SI/PSI signalling and clock references.

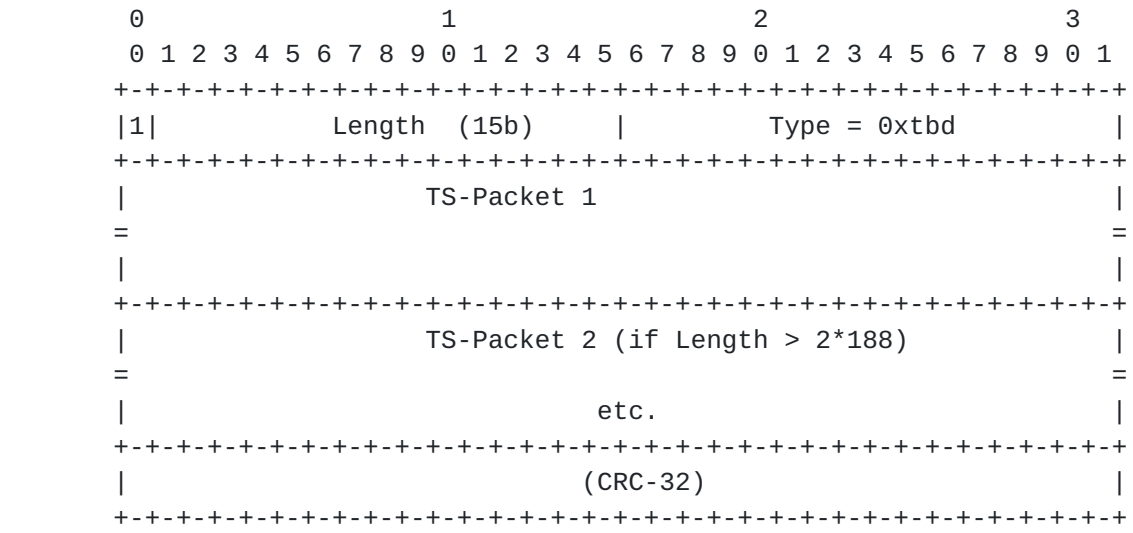


Figure 10: SNDU Format for a TS-Packet Payload (D=1)

5.5 SNDU-Resume Extension

The SNDU-Resume Extension is specified by an IANA assigned H-Type value of <tbd>. This is a Mandatory Extension. The extension is used to send the remainder of a suspended SNDU. It MAY be used at any time by Gateway which has already sent the first part of a SNDU.

The payload of an SNDU-Resume Extension contains a Fragment of a SNDU payload. The final fragment includes the SNDU CRC value

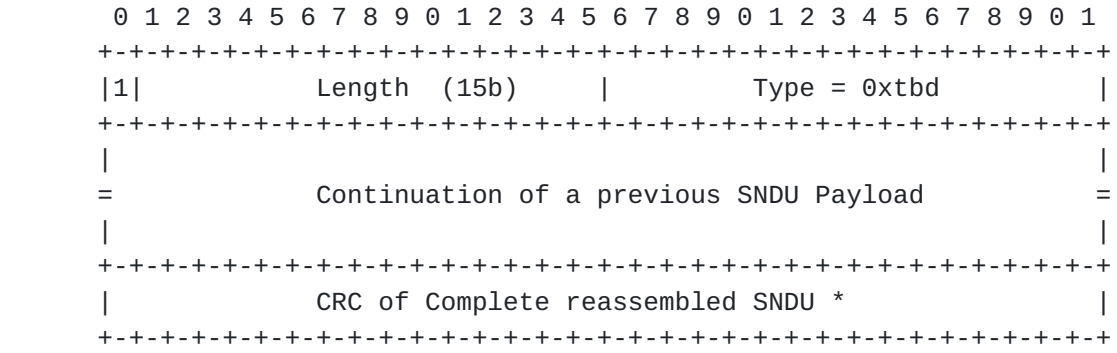
calculated over the entire (reassembled) SNDU. This CRC is used to verify the integrity and reassembly of the complete PDU. It also binds this to the MAC address for the case D=0.

The Length field in an SNDU-Resume Extension fragment specifies the size of the SNDU-Resume payload.

A Receiver that receives an SNDU-Resume fragment that it will not process/forward MUST silently discard all bytes from the fragment, it MUST NOT check the CRC value of the fragment, and proceeds with processing of the next in-sequence SNDU.

An SNDU that carries an SNDU-Resume Extension fragment that contains the final Fragment of a suspended SNDU carries the 4 byte CRC of the complete SNDU (that would have been sent at the end of the of the original SNDU, had the SNDU not been suspended). This CRC is follows the SNDU payload and is included in the calculated Length of the SNDU that carries this final Fragment (figure 11).

It is allowed to have a SNDU-Resume fragment with a Length less than the total remaining bytes (see Appendix for an example), however these fragments do NOT contain a CRC field.



* If the SNDU-Resume fragment contains the final byte of a SNDU, the CRC-32 of the original SNDU (prior to fragmentation) is appended to the SNDU-Resume fragment to form the CRC field (and included in the Length calculation). This is not the CRC of the fragment, but in this case the CRC of the complete reassembled SNDU. If further SNDU-Resume fragments would be required to complete the SNDU, the CRC field is omitted.

Figure 11: SNDU Format for a SNDU-Resume Payload (D=1).

If the number of bytes to be sent in an SNDU-Resume Extension fragment exceeds the unfilled space in the BBFrame payload, then the Encapsulator SHOULD by default continue transmission in the next BBFrame for the same stream. However, it MAY instead choose (e.g. utilising a local policy with ACM coding to optimise overall efficiency) to suspend the partially sent frame.

5.5.1 SNDU-Resume Extension fragment processing

A receiver may receive a BBFrame containing zero or more SNDUs that carry the SNDU-Resume Extension.

The Receiver SHOULD provisionally accept all SNDUs with a Length that is greater than the number of bytes remaining in the BBFrame, these are called Fragments. A Receiver MAY silently discard such SNDUs to recover the space in previously allocated reassembly buffers, or to limit processing, e.g. if it believes it is under a DoS attack, the resources consumed exceed some threshold, or other information indicates the frame has been corrupted by physical layer errors. A receiver MUST also configure a Fragment-timeout. Fragments that have been buffered for more than this predetermined period MUST be discarded (this procedure MUST be invoked when an encapsulation Gateway is restarted, and MAY be invoked via the control interface).

To reassemble the Fragments of a fragmented SNDU, a Receiver should buffer up to one full-sized SNDU for each NPA/MAC filter for which it expects to receive SNDUs. Reassembly is performed independently for each Stream that the Receiver is configured to receive. A Receiver may be configured to receive an arbitrary large number of multicast NPA/MAC addresses over a single Stream. This is in contrast to normal reassembly, where a Receiver needs to buffer at most one SNDU per Stream.

The Receiver detects the presence of a Fragment, when the SNDU Length is greater than the number of bytes remaining in a BBFrame, and the next BBFrame for the same Stream had a PP value of zero.

The processing of fragments uses the following rules:

1) Under-sized SNDU
The Receiver receives a subsequent SNDU-Resume Extension fragment with the same NPA/MAC address of a fragment that is pending reassembly for the Stream, and which contains less than the expected number of bytes required to complete the SNDU payload (SNDU Length of original base header). The Fragments SHOULD be added to the set of Fragments to be later reassembled.

2) Correct-sized SNDU
The Receiver receives a subsequent SNDU-Resume Extension fragment with a NPA/MAC address that matches a Fragment pending reassembly for the Stream, and which contains the expected number of bytes required to complete the SNDU payload. The Fragments are then reassembled and the CRC-32 is validated. An invalid CRC-32 MUST result in SNDU discard. The Receiver

continues processing the next SNDU in the BBFrame.

3) Over-Sized SNDU
The Receiver receives a subsequent SNDU-Resume Extension Fragment that matches the NPA/MAC address of a Fragment pending

reassembly for the Stream, and which contains more than the expected number of bytes required to complete the SNDU payload. This is a framing error. All Fragments for this combination of Stream and NPA/MAC address MUST be discarded. The Receiver continues processing the next SNDU in the BBFrame.

4) Unexpected Fragment
The Receiver receives a SNDU that is an SNDU-Resume Extension Fragment, but for which it has no Fragments pending reassembly with the same NPA/MAC address for the Stream. This is not a framing error. All Fragments for this combination of Stream and NPA/MAC address MUST be discarded and the Receiver will continue processing of the next received SNDU.

5) Unexpected SNDU
The Receiver receives a SNDU that is not an SNDU-Resume Extension fragment, but has the same NPA/MAC address of a Fragment that is pending reassembly for the Stream. This is a framing error. All Fragments for this combination of Stream and NPA/MAC address MUST be discarded. The Receiver then continues processing the next received SNDU.

6) Invalid Length
The Receiver receives a SNDU with an invalid Length of a Fragment. The SNDU MUST be discarded, and the Receiver enters the Idle state (see section 7).

7) Time-out
Within the period specified by the specified Fragment-timeout (see section 7), the Receiver does not receive an SNDU with a subsequent SNDU-Resume Extension for a NPA/MAC address for which it has Fragments outstanding. This is a framing error. All Fragments for this combination of Stream and NPA/MAC address MUST be discarded. The Receiver continues processing the next received SNDU.

8) Abort
The Receiver decides by other means that it will abort the reassembly processing. This is a not framing error. All Fragments for this combination of Stream and NPA/MAC address will be discarded. The Receiver continues processing the next received SNDU.

5.5.1 SNDU-Resume Extension reassembly

No specific algorithm for reassembly processing is mandated in this specification. Implementers may choose to reassemble SNDU Fragments as they are received, or only when all fragments are received.

When a Receiver has accumulated a set of Fragments with the Length specified in the SNDU in the base header of the first suspended Fragment, it must validate the reassembly. The total accumulated size of all SNDU fragments (including the SNDU CRC in the final fragment) MUST exactly match this Length. SNDUs with an inconsistent length MUST be discarded (a framing error may be recorded).

The Receiver MUST verify the SNDU CRC value carried in the final 4 bytes of the final fragment of a SNDU that it has reassembled. This value is used to verify the integrity of the reassembled SNDU.
A Receiver MUST ignore the SNDU CRC value of an SNDU-Resume fragment that is incomplete or has not been reassembled.

A successfully reassembled SNDU payload is processed according to the Type field specified in the base header of the first suspended Fragment (i.e. the start of the SNDU).

5.6 SNDU-Suspend Extension

The normal (continuous) method of transmission of a SNDU requires that a SNDU can only be fragmented if it is positioned at the end of a BBFrame. The sender can therefore only start one fragmented SNDU in each BBFrame, while this bounds processing costs per BBFrame, it can impose scheduler limitations.

Use of the method described in this section allows more scheduler flexibility. A Gateway using this method MUST use the SNDU-Resume method to transmit the remainder of the SNDU payload.

The SNDU-Suspend Extension is specified by an IANA assigned H-Type value of <tbd>. This is a Mandatory Extension. It enables a sender to start a SNDU at an arbitrary position within a BBFrame, and to then suspend processing to allow the SNDU to be completed in a subsequent BBFrame (or aborted). It MAY be used at any time by Gateway that wishes to start a new SNDU.

The extension header specifies the number of bytes sent in the current fragment (always less than the SNDU Length). The normal use of this extension is to allow a Gateway to start more than one different fragmented SNDU in the same BBFrame, in this case the Suspended-Length will be less than the number of bytes remaining within a BBFrame.

A value of D=1 indicates there is no NPA/MAC address in use. A value D=0 is also permitted, as defined in ULE, and in this case a MAC/NPA address is attached to the SNDU base header.

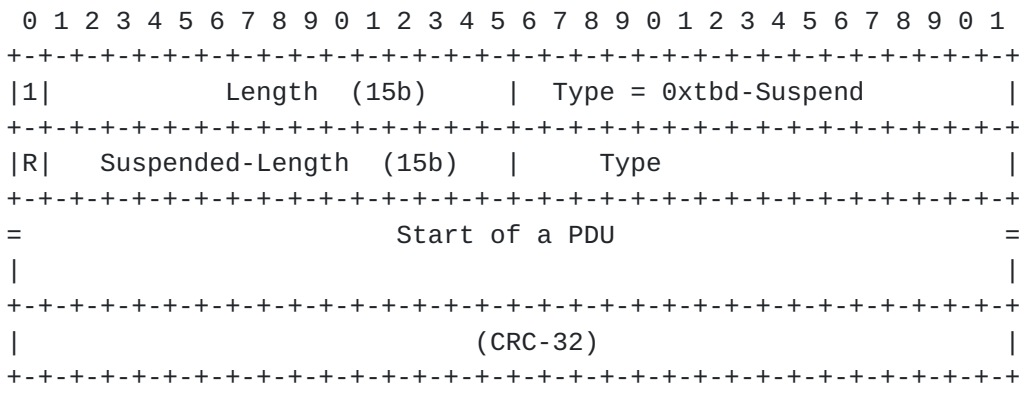


Figure 12: SNDU Format for a SNDU-Suspend Payload (D=1)

5.7 SNDU-Concat Extension

The SNDU-Concat Extension is specified by an IANA assigned H-Type value of <tbd>. This is a Mandatory Extension. It enables a sequence of (usually short) PDUs to be sent within a single SNDU payload. Each PDU is prefixed by its length in bytes. The Receiver processes this type of SNDU by extracting each SNDU in turn. The Receiver first verifies the Length and CRC of the SNDU. A Receiver MUST silently discard any data, if present, after the last complete encapsulated PDU.

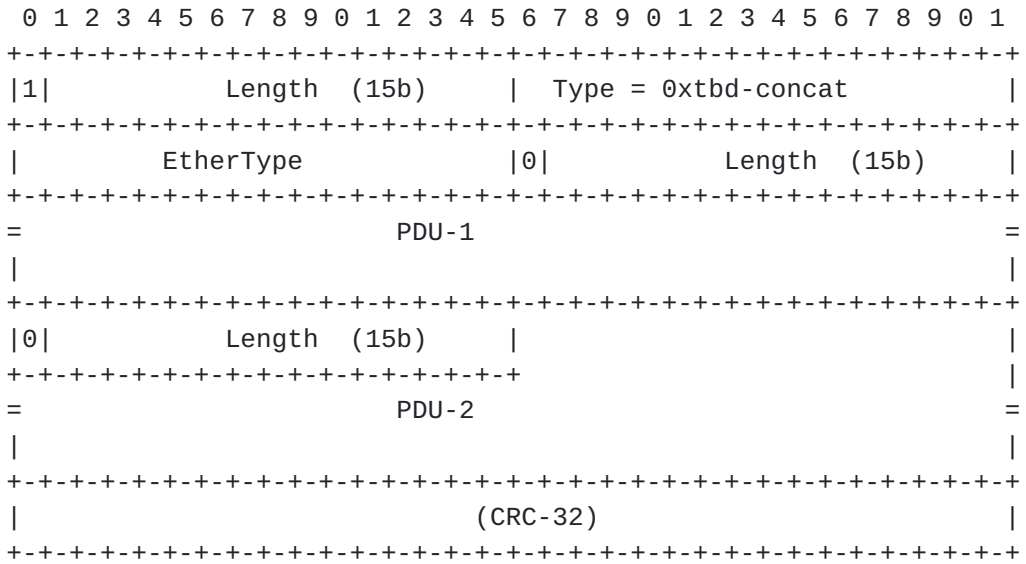


Figure 13: SNDU Format for a SNDU-Concat Payload (D=1)

A value of D=1 indicates there is no NPA/MAC address in use. A value D=0 is also permitted, as defined in ULE, and in this case a MAC/NPA address is attached to the SNDU base header. When D=1, the Receiver MUST associate the specified MAC/NPA address with all PDUs within

the SNDU Payload. This MAC/NPA address MUST also be forwarded with each PDU, if required by the forwarding interface.

6. Processing at the Encapsulator

The Encapsulator forms the PDUs queued for transmission into SNDUs by adding a header and trailer to each PDU (section 4). It then segments the SNDU into a series of BBFrame payloads (figure 9). These are transmitted using a single DVB-S2 Logical Channel (denoted by a specific ISI). Each BBFrame carries a Counter value, that is incremented modulo 255 after transmission of the BBFrame.

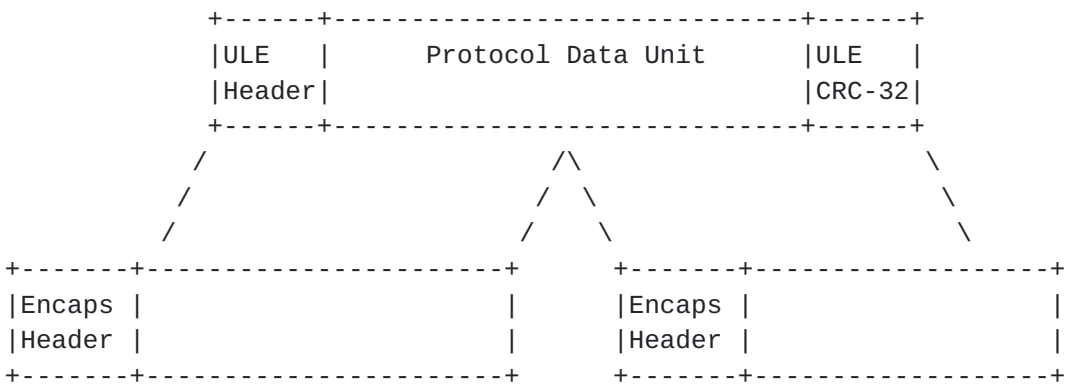


Figure 14: Encapsulation of an SNDU into a series of BBFrames

6.1 SNDU Encapsulation

When an Encapsulator has not previously sent a SNDU for a specific Stream, or after an Idle period, it starts to send an SNDU in the first available BBFrame. This BBFrame MUST carry a Payload Pointer value of zero indicating the SNDU starts in the first available byte of the BBFrame payload.

The Encapsulation MUST ensure that all BBFrames incremented by one (e.g. modulo 256 for an 8-bit value) the Counter carried in the Encapsulation header.

An Encapsulator MAY decide not to immediately send another SNDU, even if space is available in a partially filled BBFrame payload. This procedure is known as Padding (figure 11). The End Indicator informs the Receiver that there are no more SNDUs in this BBFrame payload. The End Indicator is followed by zero or more unused bytes until the end of the BBFrame payload. All unused bytes MUST be set to the value of 0xFF. The Padding procedure trades decreased efficiency against improved latency.

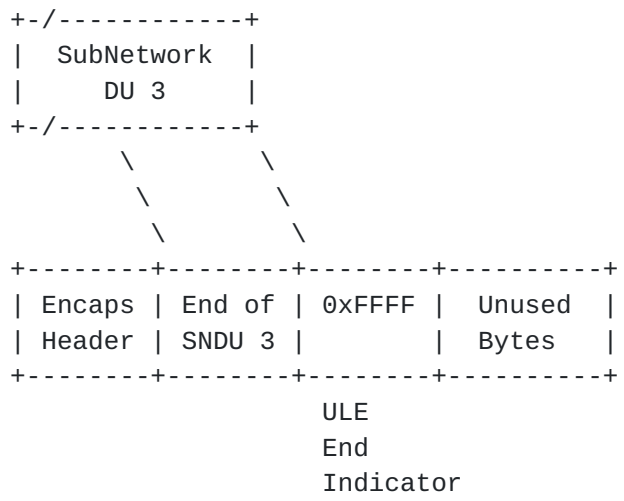


Figure 15: A BBFrame carrying the end of SNDU 3, followed by an End Indicator.

Alternatively, when more packets are waiting at an Encapsulator, and the BBFrame has sufficient space remaining in the payload, the Encapsulator can follow a previously encapsulated SNDU with another SNDU using the next available byte of the BBFrame payload (see 6.2). This is called Packing (figure 16).

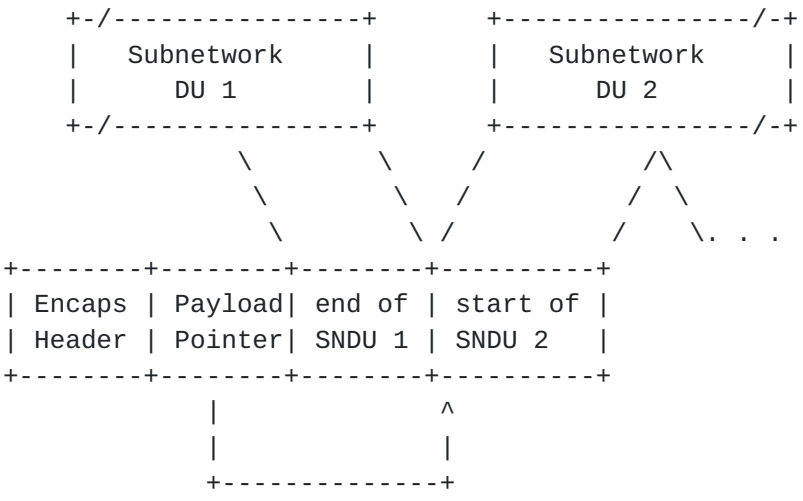


Figure 16: A BBFrame with the end of SNDU 1, followed by SNDU 2.

6.2 Procedure for Padding and Packing

Use of the Packing method by an Encapsulator is optional, and may be determined on a per-session, per-packet, or per-SNDU basis.

6.3 Suspend/Resume processing

An encapsulation gateway MAY also decide to suspend a SNDU at the end of BBFrame. In this case, it may continue transmission by

sending SNDUs to other MAC addresses within the same Stream. It MUST however preserve FIFO delivery of packets with the same MAC address by either sending a subsequent SNDU-Resume fragment (see section 5) or by aborting the suspended SNDU (by starting a new SNDU with the same MAC address).

7. Receiver Processing

A Receiver tunes to a specific S2 Multiplex and sets a receive filter to accept all Packets from a specific Stream. A single Receiver may be able to receive multiple Streams. In each case, reassembly MUST be performed independently for each Stream. To perform reassembly, the Receiver may use a buffer to hold the partially assembled SNDUs (when the SNDU-Resume extension is used, the buffers are managed separately for each active MAC/NPA address). The buffer is referred to here as the Current SNDU buffer. Other implementations may choose to use other data structures, but MUST provide equivalent operations.

Receipt of a BBFrame with a PP value not equal to 0xFFFF indicates that the BBFrame contains the start of a new SNDU. The value of the Payload Pointer indicates the number of bytes to the start of the first SNDU in the BBFrame. It is illegal to receive a Payload Pointer value greater than the size of the BBFrame payload, and this MUST cause the SNDU reassembly to be aborted and the Receiver to enter the Idle State. This event SHOULD be recorded as a payload pointer error.

A Receiver MUST support the use of both the Packing and Padding method for any received SNDU, and MUST support reception of SNDUs with or without a Destination Address Field (i.e. D=0 and D=1).

In GULE, the buffers that hold partially reassembled PDUs destined for each NPA/MAC address MAY be released at any time, and SHOULD be released after holding a buffer pending completion for a maximum of 256 BBFrames (with the same ISI). A simple way to implement this, is to mark each partially reassembled frame with the BBFrame Counter value of the BBFrame in which the SNDU started, and to abort (i.e. discard the memory buffer) when the BBFrame Counter of a later BBFrame increments to the stored value. This mechanism imposes a maximum fragment lifetime within the DVB-S.2 link, ensuring that badly formed fragments are purged from the Receiver. The value 255 was chosen as a balance between memory usage, and the need for flexible fragmentation of
large PDUs.

7.1 Idle State

After initialisation, errors, or on receipt of an End Indicator, the Receiver enters the Idle State. In this state, the Receiver discards all BBFrames until it discovers the start of a new SNDU (using the PP value), when it then enters the Reassembly State.

The Idle State does NOT normally result in the discard of any

buffered SNDU Fragments. The space in buffers occupied by Fragments that are no longer required is recovered by other means (section 7).

Figure 17 outlines these state transitions:

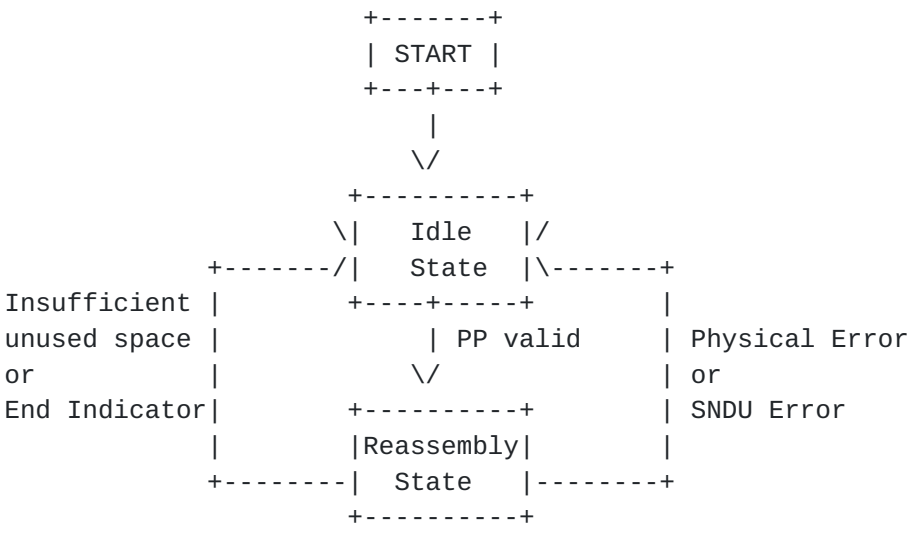


Figure 17: Receiver state transitions

7.1.1 Idle State Payload Pointer Checking

A Receiver in the Idle State MUST check the PP value in the BBFrame header of all received BBFrames. Following a loss of synchronisation, the Receiver MUST discard the number of bytes indicated by the Payload Pointer (counted from the first byte of the BBFrame payload field), before leaving the Idle State. It then enters the Reassembly State, and starts reassembly of a new SNDU at this point.

7.2 Processing of a Received SNDU

When in the Reassembly State, the Receiver reads a 2 byte SNDU Length Field from the BBFrame payload. If the value is less than or equal to 4, or equal to 0xFFFF, the Receiver discards the Current SNDU and the remaining SNDU payload (and any Fragments pending reassembly, see section 5) and returns to the Idle State. Receipt of an invalid Length Field is an error event and SHOULD be recorded as an SNDU length error.

If the Length of the Current SNDU is greater than 4, the Receiver accepts bytes from the BBFrame payload to the Current SNDU buffer until either Length bytes in total are received, or the end of the BBFrame payload is reached (see also 7.2.1). When Current SNDU length equals the value of the Length Field, the Receiver MUST calculate and verify the CRC value (see 4.6). SNDUs that contain an

invalid CRC value MUST be discarded. Mismatch of the CRC is an error event and SHOULD be recorded as a CRC error.

When the Destination Address is present (D=0), the Receiver accepts SNDUs that match one of a set of addresses specified by the Receiver (this includes the NPA address of the Receiver, the NPA broadcast address and any required multicast NPA addresses). The Receiver MUST silently discard an SNDU with an unmatched address.

After receiving a valid SNDU, the Receiver MUST check the Type Field (and process any Type 1 Extension Headers). The SNDU payload is then passed to the next protocol layer specified. An SNDU with an unknown Type value < 1536 MUST be discarded. This error event SHOULD be recorded as an SNDU type error.

The Receiver then starts reassembly of the next SNDU. This MAY directly follow the previously reassembled SNDU within the BBFrame payload.

(i) If the Current SNDU finishes at the end of a BBFrame, the Receiver MUST enter the Idle State.

(ii) If only one byte remains unprocessed in the BBFrame payload after completion of the Current SNDU, the Receiver MUST discard this final byte of BBFrame payload. It then enters the Idle State. It MUST NOT record an error when the value of the remaining byte is identical to 0xFF.

(iii) If two or more bytes of BBFrame payload data remain after completion of the Current SNDU, the Receiver accepts the next 2 bytes and examines if this is an End Indicator. When an End Indicator is received, a Receiver MUST silently discard the remainder of the BBFrame payload and transition to the Idle State. Otherwise this is the start of the next Packed SNDU and the Receiver continues by processing this SNDU (provided that the BBFrame has a PP value of less than 0xFFFF, see 7.2.1, otherwise the Receiver has detected a delimiting error and MUST discard all remaining bytes in the BBFrame payload and transitions to the Idle State).

7.2.1 Reassembly Payload Pointer Checking

A Receiver that has partially received an SNDU (in the Current SNDU buffer) MUST check the PP value in the header of the next consecutive BBFrames for the same Stream (consecutive frames are identified by verifying the Counter value associated with the BBFrame). If the Payload Pointer does NOT equal the number of bytes remaining to complete the Current SNDU, i.e., the difference between the SNDU Length field and the number of reassembled bytes, or the Counter value is not one larger, modulo 255, than in the previous BBFrame for the same Stream) the Receiver has detected a delimiting

error, or a suspended frame. The receiver follows the procedure described in section 5.

7.3 Other Error Conditions

The Receiver MUST check the BBFrame header and physical layer for information that indicates a BBFrame has been corrupted. If detected, a transmission error event SHOULD be recorded, and the entire BBFrame MUST be discarded.

The Receiver MUST check the BBFrame Counter of the Encapsulation. If the received value does NOT increment by one for successive BBFrames within a stream (modulo 255), the Receiver has detected a continuity error. A BBFrame counter error event SHOULD be recorded. A loss of continuity implies a Receiver has missed one or more SNDUs (e.g. because they was sent using a ModCod that the Receiver was unable to decode). The missed SNDUs do not necessarily have a NPA/MAC that the Receiver will forward. The Receiver MUST enter the Idle State, and use the PP value in the new BBFrame to recover framing alignment.

In Generic Mode, continuity errors may occur because an Encapsulation Gateway uses a ModCod [ETSI-S2, S2-REQ] that does not provide sufficient protection for the receive conditions experienced by a Receiver or a ModCod that is not implemented by a specific the Receiver.

8. Summary

This document defines an Generic Unidirectional Lightweight Encapsulation (GULE) to perform efficient and flexible support for IPv4 and IPv6 network services over networks built upon the DVB S2 physical layer specification operating in the Generic Mode. The encapsulation is also suited to transport of other protocol packets and bridged Ethernet frames.

GULE utilises Extension Header format defined by the Uni-directional Lihtweight Encapsulation (ULE) defined by RFC4236 and shares the associated IANA registry for support of both mandatory and optional SNDU headers.

9. IANA Considerations

This document will require IANA involvement for the assignment of two new ULE option headers. These options are defined for specific use cases envisaged by GULE, but are compatible with ULE.

10. Acknowledgments

This draft is based on the ULE protocol specification developed by the IETF ipdvb WG. The author gratefully acknowledges the inputs, comments and assistance offered by the members of the DVB-GBS ad hoc group on S2 encapsulation, and the inputs provided by the DVB-RCS WG in identifying appropriate protocol requirements. The author also wishes to thank Bernhard Collini-Nocker for his constructive email and to others who have patiently tried to explain DVB-S2. The author particularly wishes to thank Juan Cantillo & Jerome Lacan for their constant attention to detail and their contributions to the definition of the requirements for this protocol spec. The Suspend and Concat modes, and various other improvements followed discussions with Rita Ronaldo and Ulrik de Be.

11. Security Considerations

The security considerations for GULE resemble those of ULE, and security mechanisms developed as ULE extensions are expected to be appropriate also to the use of GULE.

12. References

12.1 Normative References

[RFC2119] Bradner, S., "Key Words for Use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, 1997.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.

[RFC4236] Fairhurst, G. and B. Collini-Nocker, "Unidirectional Lightweight Encapsulation (ULE) for transmission of IP datagrams over an MPEG-2 Transport Stream", Formerly <draft-ipdvb-arch-xx.txt, RFC 4236, December 2006.

[ETSI-S2] EN 302 307, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications", European Telecommunication Standards Institute (ETSI).

12.2 Informative References

[ETSI-DAT] EN 301 192, "Specifications for Data Broadcasting", European Telecommunications Standards Institute (ETSI).

[ID-S2-REQ] "Requirements for transmission of IP datagrams over DVB-S2", Internet Draft <draft-cantillo-ipdvb-s2encaps-01.rxr>, Work in Progress.

[ISO-MPEG] ISO/IEC DIS 13818-1:2000, "Information Technology; Generic Coding of Moving Pictures and Associated Audio Information Systems", International Standards Organisation (ISO).

[RFC4259 Montpetit, M.-J., Fairhurst, G., Clausen, H., Collini-Nocker, B., and H. Linder, "A Framework for Transmission of IP Datagrams over MPEG-2 Networks", RFC 4259, November 2006.

[S2-Eval] "Procedure for Comparative Evaluation of IP/DVB-S2 Encapsulation Protocol over Generic Streams", Technical Note, Work in Progress, DVB TM-GBS.

[S2-REQ] GBS0339, "Functional and performance requirements for IP/S2", DVB-TM GBS WG.

[S2-REQ-RCS] "Requirements for S2", Document 529r1, DVB TM RCS WG.

13. Authors' Addresses

Godred Fairhurst
Department of Engineering
University of Aberdeen
Aberdeen, AB24 3UE
UK
Email: gorry@erg.abdn.ac.uk
Web: http://www.erg.abdn.ac.uk/users/Gorry

14. IPR Notices

14.1 Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

14.2 Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

15. Copyright Statement

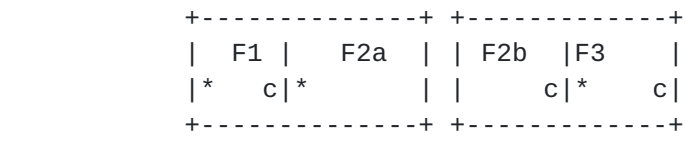
Appendix A: Scenarios for Fragmentation

This appendix provides a set of informative examples of the usage of this encapsulation.

A.1 CONSECUTIVE Fragmentation

This is the normal mode for fragmentation, where there is no additional header overhead resulting from SNDU fragmentation. SNDU fragments sent to the same NPA/MAC address follow in the next consecutive BBFrame sent to the same stream. These may be interleaved with other BBFrames associated with different streams.

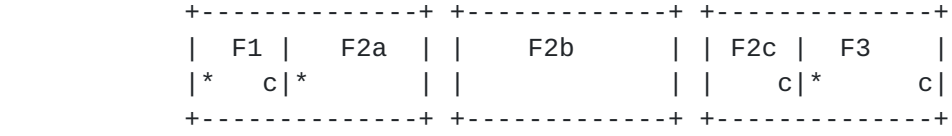
In the following figure, there are no additional overhead bytes consumed by Fragmentation of F2.



BBFrame 1 2
PP F2a F3
* = SNDU base header
c = SNDU payload CRC-32

Figure A.1.1 Fragments will follow in the next consecutive BBFrame

In the figure below, a large SNDU is fragmented across 3 consecutive frames, each sent using the same stream.



BBFrame 1 2 3
PP F2a 0xFFFF F3
* = SNDU base header
c = SNDU payload CRC-32

Figure A.1.2 Figure A.1.2 Fragments will follow in the next consecutive BBFrame

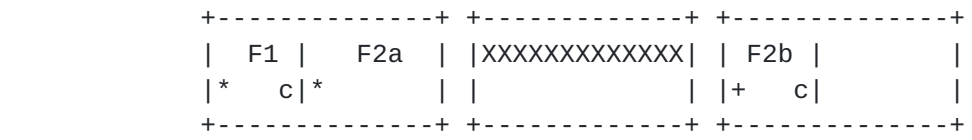
There is no additional overhead bytes consumed by Fragmentation of F2 in the case of fragmentation across an arbitrary number of

BBFrames.

A.2 GAP Fragmentation

In the case of a "gap" fragmentation, the Encapsulation sends a BBFrame after starting a fragment, that is not completed in the next adjacent BBFrame. Instead, one or more BBFrames are sent which are not directed at the same intended recipient (i.e. NPA/MAC address). It may be, for instance be that the Receiver can not decode the ModCod used for these intervening frames. In the figures in this section, it is assumed that all the BBFrames are directed to the same common stream.

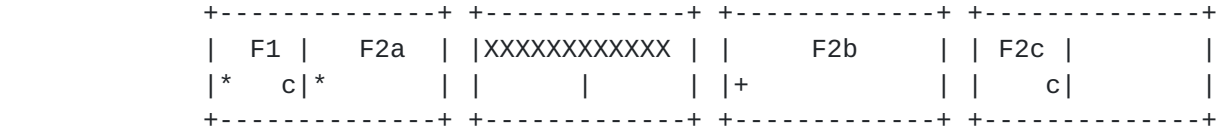
In this encapsulation, this requires a fragment header in the resumed segment (F2b shown in the figure below).



* = SNDU base header
\+ = SNDU-Resume Fragment
c = SNDU payload CRC-32

Figure A.2.1 BBFrames are not consecutive in the SNDU flow.

The additional overhead resulting from the fragmentation of F2 = 1 base header+CRC in F2b.



* = SNDU base header
\+ = SNDU-Resume Fragment
c = SNDU payload CRC-32

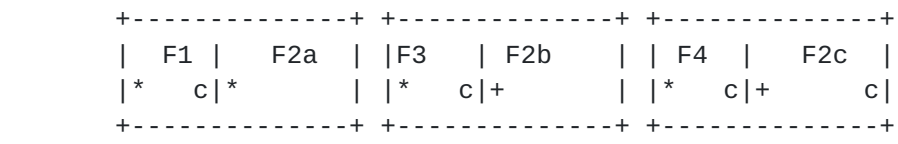
Figure A.2.2 BBFrames are not consecutive in the SNDU flow.

The additional overhead resulting from the fragmentation of the SNDU into three BBFrames is the same as that for the frame depicted in Figure A.2.2. Since the fragment F2c is consecutive with that of F2c (i.e. it is the next in-sequence BBFrame that was sent with the same stream identifier), there is no additional fragmentation header.

In some cases the Encapsulation Gateway may choose a transmission

ordering that eliminates the need for a gap transmission (in this case, scheduling the 2nd burst in figure A.2.2 before the start of the first). Where this is possible, it eliminates the need for the additional overhead associated with suspending and resuming a SNDU.

It is also permitted to suspend and later resume a SNDU on more than one occasion, should the Encapsulation Gateway wish to send a frame using space available in several BBFrames.



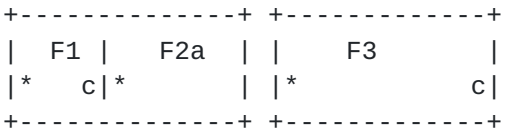
* = SNDU base header
+ = SNDU-Resume Fragment
c = SNDU payload CRC-32

Figure A.2.3 BBFrames that send F2 utilising the remaining space in 3 BBFrames.

In the above example, SNDUs F3 and F4 carry a unicast address that is not the same as that used for F2. In this encapsulation, this requires two fragment headers (and CRCs) in each of the resumed segments (F2b,F2c shown in the figure below). The extra flexibility may be justified, for example, in cases where the three BBFrames are sent using ModCod values that provide higher protection than required for transmission of F2, but where there is no higher priority traffic pending transmission in these frames. In this case, there will be a higher physical-layer cost, and the additional overhead only adds to this transmission cost.

A.3 Abort Fragmentation

These examples show the cases where the encapsulation process is terminated, resulting in discard of the partially received fragment. All the examples refer to a sequence of BBFrames associated with the same stream.



BBFrame 1 2
PP F2a 0 (First byte of F3)

* = SNDU base header
+ = SNDU-Resume Fragment
c = SNDU payload CRC-32

Figure A.3.1 Aborted Frame by start of a new complete SNDU (F3) to

same NPA/MAC address

In the above example, F2a is a partially complete fragment, followed by a BBFrame that contains the start of a different SNDU that is has

the same Receiver NPA/MAC address. Receipt of the SNDU F3 results in F2a being aborted.

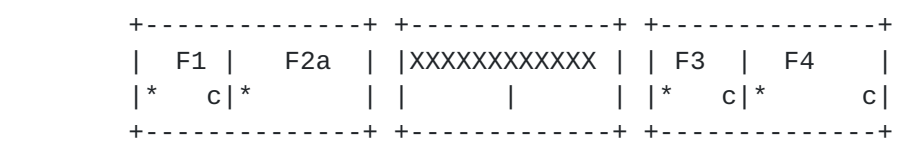


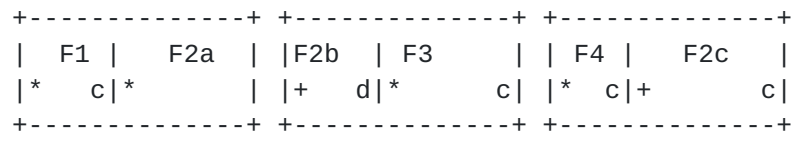
Figure A.3.2 Aborted Frame by start of a new complete fragment to same MAC address

In the above example, F2a is a partially complete fragment, followed by a BBFrame that is not decoded by the Receiver and F3 is a complete SNDU sent to a different Receiver NPA/MAC address. F4 is sent to the same MAC address as F2a, and results in F2a being aborted.

A.4 Informative examples of usage of SNDU-Resume

These examples illustrate use of the SNDU-Resume extension. All the examples refer to a sequence of BBFrames associated with the same Stream.

In the following example, SNDU 2 is fragmented into three parts (F2a,F2b,F2c). The first part of the SNDU, F2a contains the SNDU base header, followed by a sequence of bytes until the end of the 1st BBFrame. Fragment F2b follows in a later BBFrame as a SNDU-Resume fragment F2b, allowing transmission of F3 also within the second BBFrame. The final part of the SNDU is also sent in an SNDU-Resume fragment, F2c.



BBFrame 1 2 3
PP 0 0 0

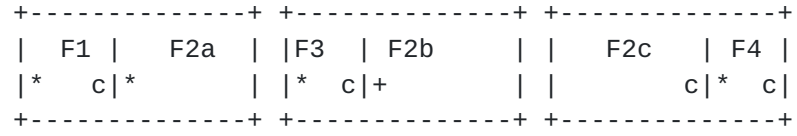
* = SNDU base header
\+ = SNDU-Resume header
c = CRC-32 used to validate integrity of SNDU.
d = CRC-32 used only to validate framing.

Figure A.4.1 Example use of SNDU-Resume

In the example above, a decision was made by the Gateway to send fragment F2c as a SNDU-Resume fragment. If the Fragment F2b had been

sent at the end of the second BBFrame, the Length field of the SNDU-Resume fragment could have been set to the total number of remaining bytes, allowing F2c to be sent as a continuation fragment, eliminating the need for a SNDU-header for

F2c (and hence improving efficiency).



BBFrame	1	2	3
PP	0	0	(1st byte of F4)

* = SNDU base header
\+ = SNDU-Resume header
c = CRC-32 used to validate integrity of SNDU.

Figure A.4.2 Example use of SNDU-Resume (optimised placement)

Appendix B: Alternative design options.

This section identifies a set of issued that will require further consideration.

B.1 Expanded D-Field

00	No MAC
01	Mac 3B * **
10	Mac 6B
11	Currently Unused

* Note: Subject to a satisfactory use-case being defined.
** Note: The short-form NPA/MAC address may also be represented in long-form by prefixing this with a well-known/reserved IEEE OUI field. The mapping of multicast IP addresses to the short-form OUI is to be specified. This mode is also outlined in [S2-REQ].

B.2 Reduction of CRC Field

The size of the CRC field may be reduced in future revisions of this document, subject to alternative methods being found that provide equivalent protection. A CRC-16 could be used in combination with appropriate and reliable feedback from the BBFrame processing layer.

A method could be defined that does not employ a CRC for complete (unfragmented) SNDUs.

Selection of the appropriate method requires inputs from experts with knowledge of the DVB-S.2 channel.

B.3 Encapsulation Header Field

Two methods of adding encapsulation overhead are defined. Selection of the appropriate method requires inputs from experts with knowledge of the DVB-S.2 BBHeader format.

[RFC EDITOR NOTE:
This section must be deleted prior to publication]

DOCUMENT HISTORY

Draft 00
This draft is intended as a study item for proposed future work by the DVB-GBS in this area. Comments relating to this document will be gratefully received by the author(s) and may also be sent to ip-dvb mailing list at: ip-dvb@erg.abdn.ac.uk

Draft 01
This draft was updated to include recommended placement of the encapsulation overhead within the BBFrame header, improved readability, correction of the counter field use (this does not trigger realignment, and is only for OAM), improved fragmentation text.

Draft 02
The author wishes to acknowdledge the detailed review by Juan Cantillo & Jerome Lacan, and their comments and contributions. Many references to MPEG-2 TS were corrected, and other minor mistakes were also corrected.

Draft 03
This draft was updated following discussion at the DVB S.2 GS Study Group.
The additions include:
SNDU-Resume - addition of payload format and informative examples
SNDU-Suspend - new extension
SNDU-Concat - new extension
The Idle state does not result in discard of the Currenmt SNDU (as in GULE).
Section 1 now includes implementation notes/constraints.
It also includes a number of editorial corrections.

Draft 04

This rev of the Spec does not use the CRC for validating frame alignment (i.e.delineating SNDU boundaries), and also therefore the SNDU Suspend option has no CRC-32. The SNDU-Resume option only carries a CRC when this is the final fragment of a SNDU. This modification came from comments from Axel J. and Rita R.

Text was also updated on the Counter   to clarify use in combination with the PP value (U.de.B).

[END of RFC EDITOR NOTE]