# Technical Descriptions for Cut-Through Forwarding in Bridges

**DCN 1-22-0042-12-ICne**

Author: Johannes Specht

November 14, 2022

# Contents

# List of Figures

# List of Algorithms

# Part I.

# Introduction

# 1. Purpose

Purpose of this document is to provide input for technical discussion in pre-PAR activities of IEEE 802, the *IEEE 802 Network Enhancements for the Next Decade Industry Connections Activity* (Nendica) in particular. The contents of this document are technical descriptions for the operations of Cut-Through Forwarding (CTF) in bridges. The intent is to provide more technical clarity, demonstrate technical feasibility, and thereby satisfy the request expressed by individuals during the IEEE 802.1 closing plenary meeting in July 2022.

# 2. Relationship to IEEE Standards

This document **IS NOT** an IEEE Standard or an IEEE Standards draft, it is an individual contribution by the author containing technical descriptions. This allows readers to focus on the technical contents in this document, rather than additional aspects that are important during standards development. For example:

1. The structure of this document does not comply with the structural requirements for such standards (e.g., this document does not contain mandatory clauses for IEEE Standards [1]).

2. Usage of normative keywords has no implied semantics beyond technical language. For example, usage of the words *shall*, *should* or *may* **DOES NOT** imply conformance requirements or recommendations of implementations.

3. This document contains references, but without distinguishing between normative and informative references.

4. This document does not contain suggestions for assigning particular contents to *vehicles* (e.g., IEEE 802 Working Groups, potential amendment projects for existing standards, or potential new standard projects). As a consequence, the clause structure of this document is intended for readability, rather than fitting into the clause structure of a particular Standard (which would especially matter for potential amendment projects).

# 3. Status of this Document

This document is work-in-progress. It contains technical and editorial errors, omissions, simplifications and certain descriptions can be enhanced. Readers discovering such issues are encouraged for making enhancement proposals, e.g. by proposing textual changes or additions to the author (johannes.specht.standards@gmail.com).

# Part II.

# Cut-Through Forwarding in Bridges

# 4. Overview and Architecture

This part of the document comprises technical descriptions for supporting CTF in bridges. While this document is not a standard, there are published IEEE 802.1 Standards describing the operation of bridges without the descriptions herein. For differentiation between bridges with support for CTF and bridges according to the published IEEE 802.1 Standards (e.g., IEEE Std 802.1Q[2]), term *CTF bridge* is used in this document to refer to the former, whereas term *S&F bridge* is used in this document to refer to the latter. Like in IEEE Std 802.1Q, CTF bridges may or may not support Virtual Local Area Networks (VLANs), and therefore terms *VLAN-aware* and *VLAN-unaware* are used to distinguish between bridges with and without support for VLANs.

The architecture of CTF bridges is widely aligned with the bridge architecture in IEEE Std 802.1Q [2, 8.2]. It is shown in Figure 4.1 in a compact form (see also the architectural figures in IEEE Std 802.1Q [2, Figure 8-2, 8-3, 8-4, ff.]).

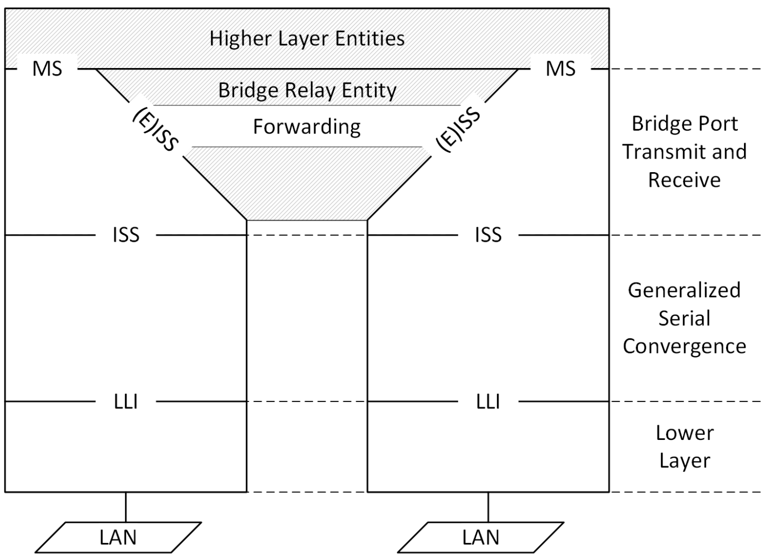


Figure 4.1.: Architecture of a Cut-Through Forwarding (CTF) Bridge.

This architecture comprises the following elements:

1. Higher layer entities using the MAC Service (MS) via the MAC Service interface defined in IEEE Std 802.1AC [3, clause 14].

2. A bridge relay entity (8) that relays frames between different bridge Ports.

3. Generalized serial convergence operations (6) per bridge Port that provide the Internal Sublayer Service (ISS) via the Internal Sublayer Service Interface defined in IEEE Std 802.1AC [3, clause 11].

4. Lower layers per bridge Port that are used by the generalized serial convergence operations for providing the ISS.

5. Bridge Port transmit and receive operations (7) per Bridge port that transform and transfer service primitive invocations between the bridge relay entity, higher layer entities and the generalized serial convergence operations.

Excluded from this document are several details on higher layer entities[1] above the MAC Service interface and elements of the bridge relay entity other than the forwarding process[2]:

– For frames to and from higher layer entities, the bridge port transmit and receive operations of a CTF bridge establish the behavior of S&F bridge at the MAC service interface (7.2), allowing higher layer entities to operate according to the behavior specified in IEEE 802.1 Standards unaltered.

– The forwarding process of a CTF bridges (re-)establishes the behavior of S&F bridges at interaction points with other elements of the bridge relay entity.

Furthermore excluded are hybrid CTF bridges where the ISS in different bridge Ports is provided by combinations of two or more of the following:

– Generic serialized convergence operations (6).

– Standardized and specific MAC procedures [3, clause 13][2, 6.7].

– Other technologies providing the ISS.

In general, this document limits on use of Cut-Through for a subset of operations standardized in IEEE Stds 802.1Q[2], 802.1AC[3] and 802.1CB[4] that is suitable for demonstrating technical feasibility and for which CTF is applicable[3].

[1] Examples for higher layer entities are Spanning Tree Protocols and Multiple Registration Protocols, supported by LLC entities above the MAC service interface [2, item c) in 8.2 and b) in 8.3].

[2] An example element of the bridge relay entity other than the forwarding process is the learning process [2, item c) in 8.2 and b) in 8.3].

[3] Defining CTF support for all protocols and procedures standardized by IEEE WG 802.1 and beyond is not intended. Some of these protocols and procedures are in contradiction with CTF, for example, if there is a strong dependency on the frame length. Fall-backs to S&F (5.4.3) are used for modeling interaction points with such protocols and procedures within CTF bridges.

# 5. Modeling Principles

## 5.1. Frame Types

If necessary, distinct terms for are used for frames for describing their current state, as follows:

**frame under reception** A frame that is being serially received from LAN for which reception began bit did not finish.

**received frame** A frame that was serially received from a LAN that finished reception.

**frame under transmission** A frame that is being serially transmitted to a LAN for which transmission began bit did not finish.

**transmitted frame** A frame that was serially transmitted to a LAN that finished transmission.

## 5.2. Modeling of Service Primitives

All invocations of service primitives in this document are atomic. That is, each invocation is non-decomposable (see also 7.2 of IEEE Std 802.1AC[3] and [5]). Semantics of the ISS (6.2.2) and EISS (7.4) in terms of service primitives, their parameters, etc. is refined in this document for the CTF operation, allowing for accurate description of operations within a CTF bridge. This refined model comprises the following:

1. The parameters of a service primitive are explicitly modeled as bit arrays.
2. The values of parameters during invocations of a service primitive are passed according to a call-by-reference scheme.
3. A service primitive provides two attributes[1], *'start* and *'end*. These attributes are used in subsequent descriptions to indicate the temporal start and the end of the indication, respectively.

In a series of sequential *processing stages* (e.g., the processes introduced in 6.1 or a sub-process of the forwarding process in 8), this model allows later processing stages to access contents in service primitive parameters that are incrementally added by an earlier processing stage. The 'start and end attributes can, but are not required to, be in temporal relationship with the duration of frames on the physical layer.

[1] The concept of *attributes* is inspired by the *Very High Speed Integrated Circuits Hardware Description Language*, VHDL[6], which provides predefined attributes (e.g., *'transaction*) that allow modeling over multiple VHDL simulation cycles at the same instant of simulated time.

## 5.3. Parameter-based Modeling

At higher processing stages, service primitives of frames and processing of these frames themselves is modeled at parameter level accuracy. The purpose of this model is to

1. provide means for compact description of temporal control (5.4) in and across processing stages,
2. enable re-use of existing transformation rules from IEEE 802.1 Stds, and
3. avoid low level details that would not provide any value to the clarity and unambiguous descriptions.

The parameter-based modeling uses the resolution of symbolic and/or numeric parameters instead of bit arrays (5.2). A parameter is said to be *complete* at the earliest instant of time at which the *minimal information* is available to *unambiguously* determine the parameter's value within the specified valid value range of such parameter. The minimal information may be

1. a coherent sequence of bits in a frame (e.g., eight subsequent bits forming an octet),
2. the result of composition and/or computation across bits located at various locations in a frame,
3. frame information not encoded in particular bits (e.g., frame length),
4. based on out-of-band information, or
5. combinations of the aforesaid.

As an example, the vlan_identifier parameter of EM_UNITDATA.indication (7.4) invocations can be derived from a subset of underlying bits of the associated SDU parameter of M_DATA.indication invocations (6.2.1) that are located in a VLAN Tag [2, 9.6] according to the specification of the Support for the EISS defined in IEEE Std 802.1Q [2, item e) in 6.9.1] or originate from out-of-band information like a configured per-Port PVID parameter [2, item d) in 6.9, item f) in 6.9.1 and 12.10.1.2]. If the VLAN tag is required to unambiguously determine the vlan_identifier parameter, the parameter is complete when all bits of the VID parameter[2] in the VLAN Tag where received. Most of the data transformations between bits in a frame, frame parameters and potential out-of-band information is already unambiguously specified in the relevant IEEE 802.1 Standards. This document omits repetition of already specified transformations and instead just refers to the relevant transformations in existing IEEE 802.1 Standards.

[2] The bits and potential out-of-band information form the minimal information, and exclude any redundant information, most prominently the (in-band) redundant encoding of the VID parameter in the frame's FCS parameter.

## 5.4. Temporal Control

### 5.4.1. Processing Stalls

Parameter-based modeling is used for onvenient formulation of temporal control statements in processing stages. A processing stage (5.2) may *stall* further processing of a frame under reception, including (but not limited to) passing this frame to a subsequent processing stage, until one or more parameters are complete (5.3), subject to the implicit discarding due to late errors (5.4.2). Most processing stalls are given due to the data dependencies already specified in IEEE 802.1 Standards (e.g., Ingress Filtering as part of the forwarding process in IEEE Std 802.1Q[2, 8.6.2] depends on the availability of a frame's VID, which therefore implicitly requires completion of the vlan_identifier parameter of EM_UNITDATA.indication invocations), however, explicit modeling of processing stalls may be expressed by formulations in natural language.

Example formulations:

1. *"Processing **stalls** pending the **vlan_identifier** parameter."*
2. *"Further execution in a CTF bridge is **stalled** pending the **destination address** of a frame under reception prior to the filtering database lookup of the destination ports."*

A processing stall does not become effective if all associated parameters of a frame are complete at the point where the processing stall is defined.

### 5.4.2. Late errors

In a sequence of processing stages, an earlier processing stage may discover an error in a frame under reception and then notify all subsequent (not antecedent) processing stages, which may then implement error handling upon this such notification. This is termed as a *late error*, which is raised by the earlier processing stage and associated with a particular frame under reception. If any of the subsequent stage stalls processing pending one or more parameters of the associated frame under reception when the error is raised, the frame is discarded in the subsequent stage and thereby neither further processed nor passed to any other following processing stage.

### 5.4.3. Fall-backs to S&F

The descriptions of the processing stages use *fall back to S&F* as a modeling shortcut to summarize the following sequence:

1. Processing of a frame under reception stalls pending the frame's end of reception, which is a shortcut by itself for stalling processing pending all parameters of a frame under reception, including the FCS.
2. Dependent on whether or not a late error was indicated by an earlier processing stage for that frame while processing stalls, processing continues or the frame is discarded:

a) Late error indicated:
The frame is discarded prior to any further processing by any stage.

b) No Late error indicated:
Processing of the frame continues through subsequent processing steps and stages according to the standardized behavior of an S&F bridge.

### 5.4.4. Instantaneous Operations

In absence of processing stalls, processing stages in this document perform their operations instantaneously. It is clear that idealistic instantaneous operations, in terms of 0-delay at an infinite high resolution[3], are not possible in real world implementations. Physics, design decisions and design constraints introduce additional delays in such implementations. The model is not intended to upper limit such delays. It is there for describing data dependencies, late error handling and the resulting externally visible behavior. Additional delays (e.g., real world implementations starting transmissions on a physical medium later than the model) are not described by the model, but could be determined by observation/measurement and are available as management parameters (9.3).

[3]The semantics of "instantaneous" depends on the resolution [7, p.11].

# 6. Generalized Serial Convergence Operations

## 6.1. Overview

The generalized serial convergence operations are described by a stack of processes that interact via global variables (see 6.4) and service primitive invocations (see 6.2). These processes provide an Internal Sublayer Service [3, clause 1] for the upper layers of a CTF bridge, and are intended to support a broad range of lower layers, including (but not limited to) physical layers. Figure 6.1 provides an overview of these processes

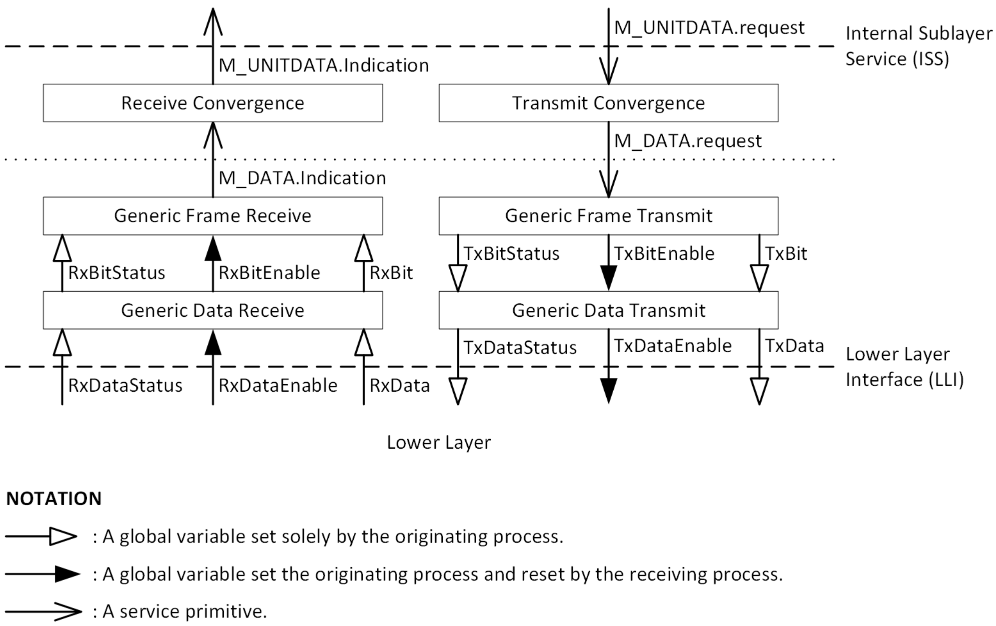


Figure 6.1.: Overview of the generalized serial convergence operations.

and their interaction[1]. The processes can be summarized as follows:

1. A Receive Convergence process (6.8) that translates each invocation of the M_DATA.-

[1] This interaction model is inspired by clause 6 and 8.6.9 of IEEE Std 802.1Q[2].

indication service primitive (6.2.1) into a corresponding invocation of the M_UNITDATA.indication service primitive (6.2.2).

2. A Generic Frame Receive process (6.7) that generates M_DATA.indication invocations for bit sequences originating from the Generic Data Receive process of at least LEN_MIN (6.3.5) bits.

3. A Generic Data Receive process (6.6) that translates a lower layer-dependent[2] serial data stream into delineated homogeneous bit sequences of variable length, each typically representing a frame.

4. A Transmit Convergence process (6.11) that translates each invocation of the M_UNITDATA.request service primitive into a corresponding invocation of the M_DATA.request service primitive.

5. A Generic Frame Transmit process (6.10) that translates M_DATA.request invocations into bit sequences for the Generic Data Transmit process.

6. A Generic Data Transmit process (6.9) that translates bit sequences from the Generic Frame Transmit process into a lower layer-dependent serial data stream.

The generalized serial convergence operations are heavily inspired by the concepts described in slides by Roger Marks [8, slide 15], but follow a different modeling approach with more formalized description of the processes and incorporate some of the following concepts, as suggested by the author of this document during the Nendica meetings on and after August 18, 2022. Some differences can be summarized as follows:

- Alignment with state machine diagram conventions of IEEE Std 802.1Q[2, Annex E].
- Support for serial data streams from lower layers with arbitrary data word length (6.3.7)[3].
- Explicit temporal modeling of atomic ISS service primitive invocations (5).
- Relaxed frame length constraints (6.3.5 and 6.3.6).

By keeping ISS service primitive invocations atomic, the approach in this section provides compatibility with the definition from IEEE Std 802.1 AC [3, 7.2].

[2]Such a lower layer may be an entity on the physical layer (PHY), but the generalized receive operations are not limited to this.

[3]This generalization is intended to allow a wide range of lower layers. This includes physical layer interfaces (see A.1), but the support for word sizes (e.g., 8 bits, 32 bits or 64 bits) may be close to internal interfaces of real world implementation. It is subject to discussion whether this generalization over [8] introduced by the author are needed or not.

**Algorithm 6.1** Signature of the M_DATA.indication service primitive.

**M_DATA.indication(DA, SA, MSDU, FCS)**

**Algorithm 6.2** Signature of the M_DATA.request service primitive.

**M_DATA.request(DA, SA, MSDU, FCS)**

## 6.2. Service Primitives

### 6.2.1. M_DATA.indication and M_DATA.request

The M_DATA.indication service primitive passes the contents of a frame from the Generic Frame Receive process to the Receive Convergence process. The M_DATA.-request service primitive passes the contents of a frame from the Transmit Convergence process to the Generic Frame Transmit process. The parameter signatures of the service primitives are as shown in Algorithm 6.1 and Algorithm 6.2[4].

The parameters are defined as follows:

#### 6.2.1.1. DA

An array of zero to LEN_ADDR (6.3.3) bits, containing the destination address of a frame.

#### 6.2.1.2. SA

An array of zero to LEN_ADDR (6.3.3) bits, containing the source address of a frame.

#### 6.2.1.3. MSDU

An array of zero or more bits, containing a service data unit of a frame. The number of bits after complete reception of a frame is an integer multiple LEN_OCT (6.3.2).

#### 6.2.1.4. FCS

An array of zero to LEN_FCS (6.3.4) bits, containing the frame check sequence of a frame.

### 6.2.2. M_UNITDATA.indication and M_UNITDATA.request

As specified in IEEE Std 802.1AC[3, 11.1], with the identical parameter signatures as shown in Algorithm 6.3 and Algorithm 6.4.

[4]The parameters in this version of this document limit to those introduced in Roger Marks' GSCF slides [8]. Future versions may introduce more flexibility (e.g., for IEEE Std 802.11 [9, 9.2]).

**Algorithm 6.3** Signature of the M_UNITDATA.indication service primitive.

**M_UNITDATA.indication(**
    **destination_address,**
    **source_address,**
    **mac_service_data_unit,**
    **priority, drop_eligible,**
    **frame_check_sequence,**
    **service_access_point_identifier,**
    **connection_identifier**
**)**

**Algorithm 6.4** Signature of the M_UNITDATA.request service primitive.

**M_UNITDATA.request(**
    **destination_address,**
    **source_address,**
    **mac_service_data_unit,**
    **priority, drop_eligible,**
    **frame_check_sequence,**
    **service_access_point_identifier,**
    **connection_identifier**
**)**

## 6.3. Global Constants

### 6.3.1. PREAMBLE

A lower layer-dependent array of zero[5] or more bits, containing the expected preamble of each frame.

### 6.3.2. LEN_OCT

The integer number eight (8), indicating the number of bits per octet.

### 6.3.3. LEN_ADDR

An integer denoting the length of the DA and SA parameters of M_DATA.indication parameters, in bits. For example,

$$\mathrm{LEN\_ADDR} = 48 \tag{6.1}$$

indicates an EUI-48 addresses.

[5]Including length zero permits to support lower layers that do not expose a preamble to the Generic Data Receive process.

### 6.3.4. LEN_FCS

An integer denoting the length of frame check sequence and the length FCS parameter of M_DATA.indication parameter, respectively, in bits. For example,

$$\text{LEN\_FCS} = 32 \tag{6.2}$$

indicates a four octet frame check sequence.

### 6.3.5. LEN_MIN

A lower layer-dependent integer, denoting the minimum length of a frame, in bits. Invocation of the M_DATA.indication service primitive starts once the Generic Frame Receive process received the first LEN_MIN bits of a frame. Values for LEN_MIN with

$$\text{LEN\_MIN} \geq \text{PREAMBLE.length} + \text{LEN\_FCS} \tag{6.3}$$

are valid.

### 6.3.6. LEN_MAX

A lower layer-dependent integer, denoting the maximum length of a frame, in bits. Invocation of the M_DATA.indication service primitive ends at latest once the Generic Frame Receive process received at most LEN_MAX bits of a frame. Values for LEN_MIN with

$$\text{LEN\_MAX} \geq \text{PREAMBLE.length} + 2\text{LEN\_ADDR} + \text{LEN\_FCS} \tag{6.4}$$

are valid.

### 6.3.7. LEN_DATA

A lower layer-dependent integer, denoting the data width of the RxData and TxData variables, in bits.

## 6.4. Global Variables

### 6.4.1. RxBitEnable

A Boolean variable, set by the Generic Data Receive process and reset by the Generic Frame Receive process, which indicates an update of the RxBit variable, RxBitStatus variable, or both.

### 6.4.2. RxBit

A bit variable used to pass a single bit value to the Generic Frame Receive process.

**Algorithm 6.5** Definition of data type `low_data_t`.

```
typedef struct {
  Boolean start;
  Boolean end;
  bit[]   value;
} low_data_t;
```

### 6.4.3. RxBitStatus

An enumeration variable used to pass the receive status from the Generic Data Receive process to the Generic Frame Receive process. The valid enumeration literals are as follows:

**IDLE** Indicates that the Generic Data Receive process does not pass bits of a frame to the Generic Frame Receive process.

**RECEIVING** Indicates that the Generic Data Receive process passes bits of a frame to the Generic Frame Receive process without knowledge of the frame length.

**TRAILER** Indicates that the Generic Data Receive process passes bits of a frame to the Generic Frame Receive process with the knowledge that LEN_FCS or less bits follow.

### 6.4.4. RxDataEnable

A Boolean variable, set by a lower layer and reset by the Generic Data Receive process, which indicates an update of the RxData variable, RxDataStatus variable, or both.

### 6.4.5. RxData

A variable of composite data type *low_data_t*, used for serially passing data words of frames from a lower layer to the Generic Data Receive process. Type low_data_t is defined in Listing 6.5. The semantics of the constituent parameters is as follows[6]:

**start** Indicates whether the data word is the first word of a frame (TRUE) or not (FALSE).

**end** Indicates whether the data word is the last word of a frame (TRUE) or not (FALSE).

**value** A lower layer-dependent non-empty array of up to LEN_DATA (6.3.7) bits, containing a data word of a frame. An array length less than LEN_DATA bits is only valid if end is TRUE.

[6] RxData and RxDataStatus contain redundant information, which may disappear in a future version of this document.

### 6.4.6. RxDataStatus

An enumeration variable used to pass the receive status from lower layers to the Generic Data Receive process. The valid enumeration literals are as follows:

**IDLE** Indicates that data stream reception from lower layers is not active.

**RECEIVING** Indicates that data stream reception from lower layers is active.

### 6.4.7. TxBitEnable

A Boolean variable, set by the Generic Frame Transmit process and reset by the Generic Data Transmit process, which indicates an update of the TxBit variable.

### 6.4.8. TxBit

A bit variable used to pass a single bit value of a frame's bit stream to the Generic Data Transmit process.

### 6.4.9. TxBitStatus

An enumeration variable that indicates the transmission state from the Generic Frame Transmit process to the Generic Data Transmit process. The valid enumeration literals are as follows:

**IDLE** Indicates that the Generic Frame Transmit process is not generating the bit stream of a frame.

**TRANSMITTING** Indicates that the Generic Frame Transmit process is generating the bit stream of a frame.

### 6.4.10. TxDataEnable

A Boolean variable, set by the Generic Data Transmit process a lower layer and reset by the lower layer, which indicates an update of the TxData variable.

### 6.4.11. TxData

A variable of composite datatype low_data_t (6.5), used for serially passing data words of frames from the Generic Data Transmit process to a lower layer.

### 6.4.12. TxDataStatus

An enumeration variable that indicates the transmission state from the Generic Data Transmit process to the lower layer. The valid enumeration literals are as follows:

**IDLE** Indicates that the Generic Data Transmit process is not generating the data stream of a frame.

**TRANSMITTING** Indicates that the Generic Data Transmit process is generating the data stream of a frame.

## 6.5. Global Functions

### 6.5.1. append(bitArray,bit)

Appends a given bit at the end of a bit array variable and increases the length of the variable by one.

### 6.5.2. insert(bitArray,index,bit)

Inserts the bit at the given index into the given bit array variable.

### 6.5.3. remove(bitArray,index)

Removes and returns the bit at the given index of the given bit array variable.

## 6.6. Generic Data Receive process

### 6.6.1. Description

The Generic Data Receive process translates a lower layer dependent serial data stream into a uniform bit stream and implements delay line of LEN_FCS bits to determine the value of the RxBitStatus variable.

### 6.6.2. State Machine Diagram

The operation of the Generic Data Receive process is specified by the state machine diagram in Figure 6.2 , using the variables defined in subsequent sub-clauses.

### 6.6.3. Variables

#### 6.6.3.1. cnt

An integer counter variable, used for indexing bits in the RxData variable.

#### 6.6.3.2. buf

A bit array variable for buffering bits from the RxData variable and forming a delay line.

#### 6.6.3.3. rxDataEnd

A Boolean variable, set when the data stream of a frame ends and used to determine the transition to the trailer of a frame in the RxBitStatus variable.

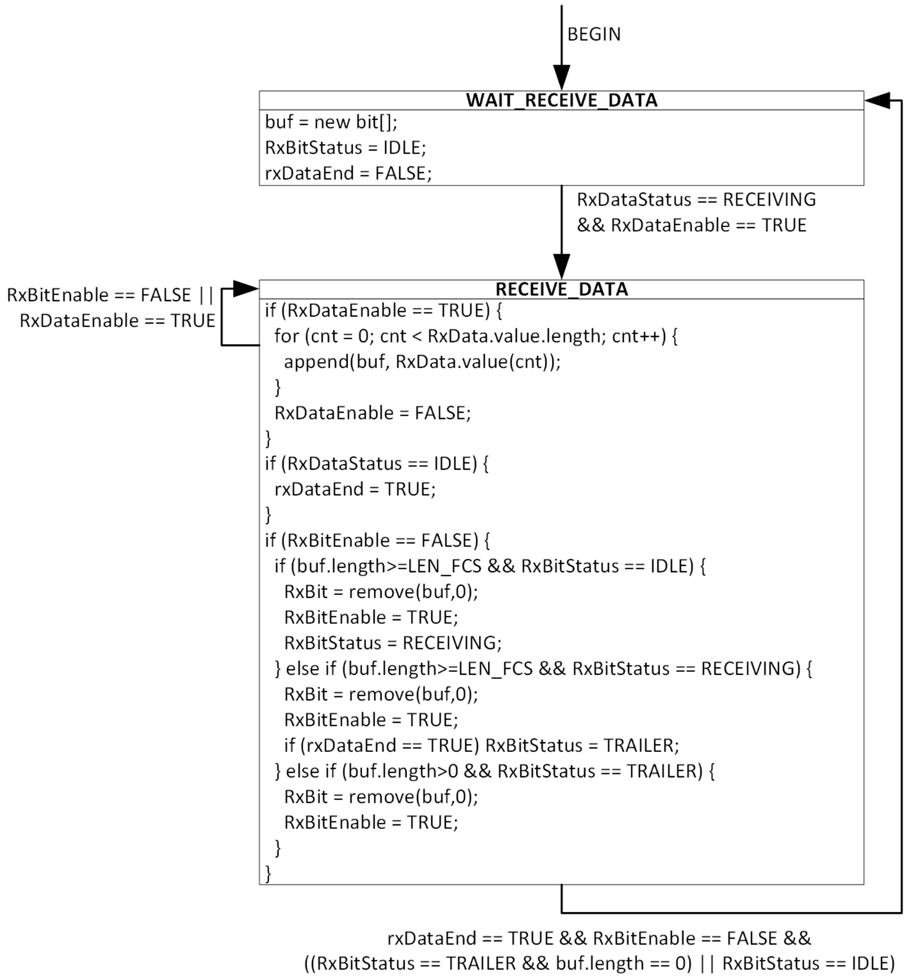


Figure 6.2.: State Machine Diagram of the Generic Data Receive process.

# 6.7. Generic Frame Receive process

## 6.7.1. Description

The Generic Frame Receive process transforms a serial bit streams of frames from the Generic Data Receive process into invocations of the M_DATA.indication primitive.

## 6.7.2. State Machine Diagram

The operation of the Generic Frame Receive process is specified by the state machine diagram in Figure 6.3 , using the variables and functions defined in subsequent subclauses.

## 6.7.3. Variables

### 6.7.3.1. cnt

An integer counter variable, used to count the number of bits in a parameter of a frame under reception.

### 6.7.3.2. len

An integer variable holding the actual length of a frame under reception, in bits.

### 6.7.3.3. buf

A bit array variable for buffering up to LEN_OCT bits of the MSDU parameter.

### 6.7.3.4. status

An enumeration variable holding the current status of the Generic Frame Receive process. The valid enumeration literals are as follows:

**Ok** Indicates that no error has been discovered prior or during frame reception.

**FrameTooLong** Indicates that a frame under reception exceeded LEN_MAX bits.

**FCSInvalid** Indicates inconsistency between the FCS parameter and the remaining parameters of a frame under reception.

## 6.7.4. Functions

### 6.7.4.1. FCSValid(FCS)

The FCSValid function determines if the FCS parameter consistent with the remaining parameters of the M_DATA.indication service primitive (TRUE) or not (FALSE). A late error associated with the frame under reception is raised (5.4.2) if the function returns FALSE.

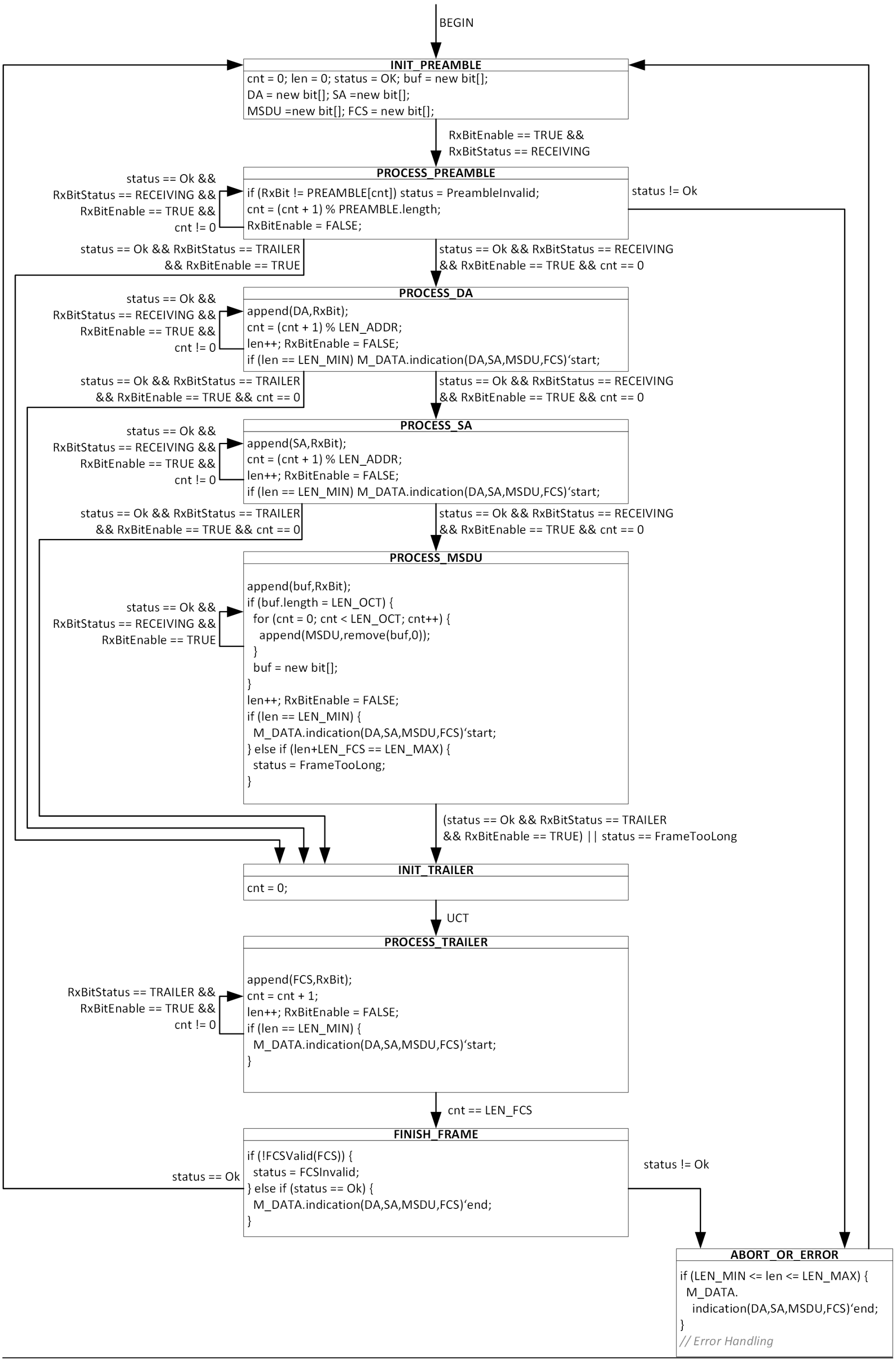




Figure 6.3.: State Machine Diagram of the Generic Frame Receive process.

## 6.8. Receive Convergence process

The Receive Convergence process implements the translation of M_DATA.indication invocations to M_UNITDATA.indication invocations. The supported translations are lower layer-dependent and include, but not limited to, those specified in clause 13 of IEEE Std 802.1AC[3].

Each M_DATA.indication invocation results in an associated M_UNITDATA.-indication invocation. During the translation, the M_UNITDATA.indication parameters are determined based on the the M_DATA.indication parameters according to the rules defined for the underlying lower layer[7].

## 6.9. Generic Data Transmit process

The Generic Data Transmit process translates a uniform bit stream into a lower layer-dependent serial data stream.

### 6.9.1. State Machine Diagram

The operation of the Generic Data Transmit process is specified by the state machine diagram in Figure 6.4.

### 6.9.2. Variables

#### 6.9.2.1. cData

A variable of type low_data_t (6.5), used for preparing the next data element passed to the lower layer via the TxData variable.

## 6.10. Generic Frame Transmit process

### 6.10.1. Description

The Generic Frame Transmit process transforms invocations of the M_DATA.request primitive from the Transmit Convergence Process into bit streams of frames.

### 6.10.2. State Machine Diagram

The operation of the Generic Frame Transmit process is specified by the state machine diagram in Figure 6.5 , using the variables subsequently defined.

[7] See also [8, p. 21].

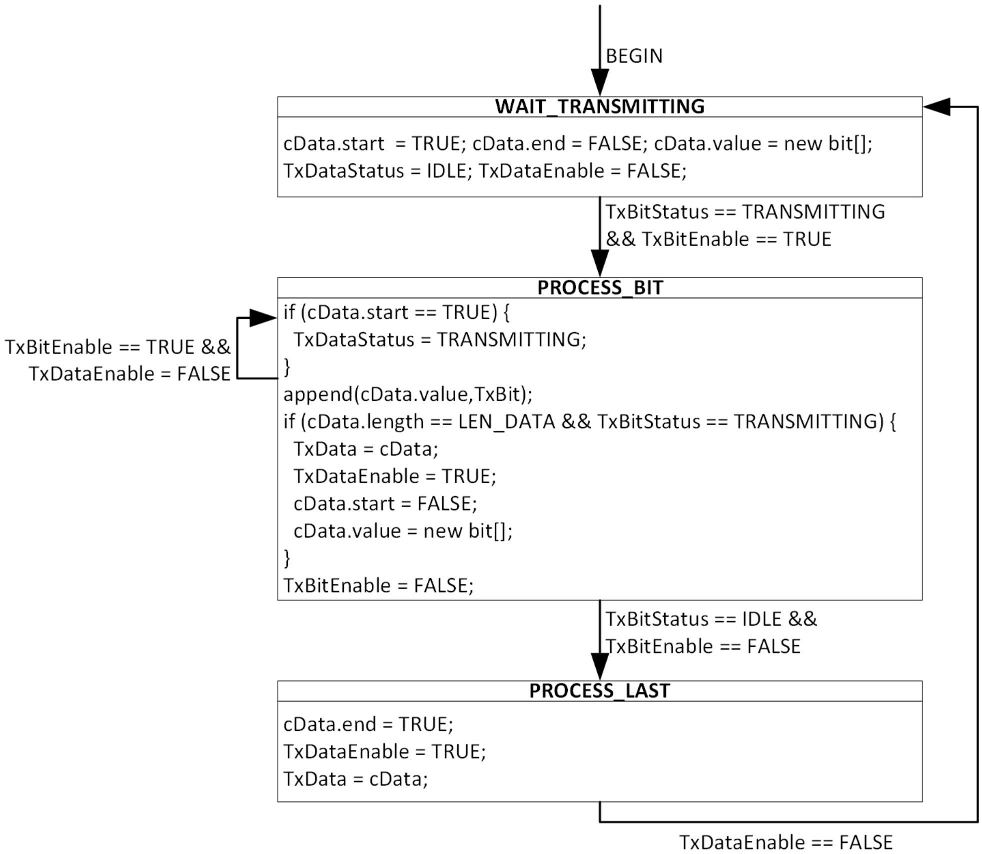


Figure 6.4.: State Machine Diagram of the Generic Data Transmit process.

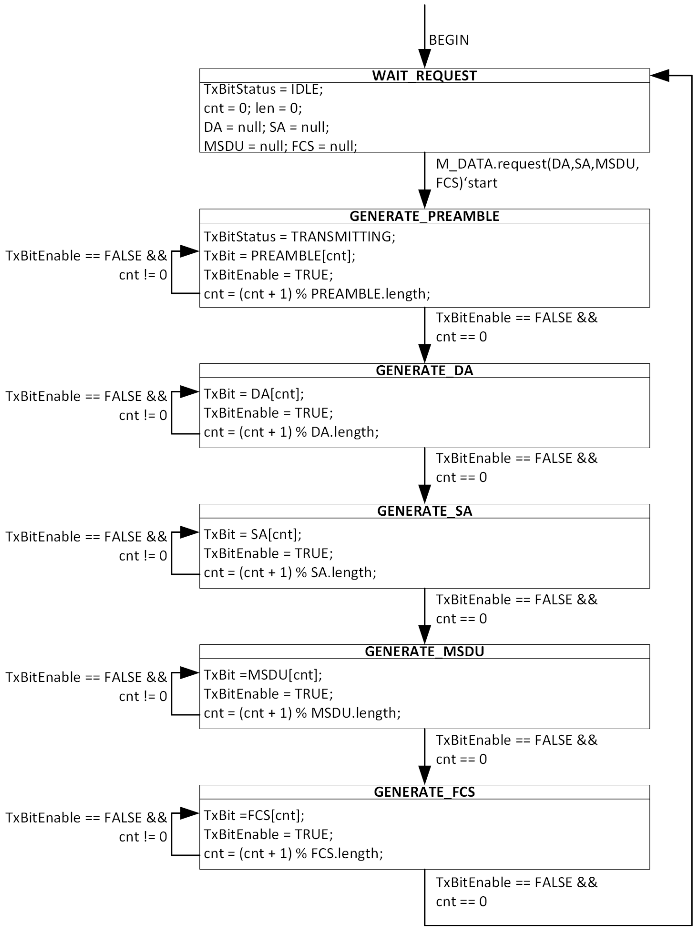


Figure 6.5.: State Machine Diagram of the Generic Frame Transmit process.

### 6.10.3. Variables

#### 6.10.3.1. cnt

An integer counter variable, used to count the number of bits in a parameter of a frame under transmission.

## 6.11. Transmit Convergence process

The Transmit Convergence process implements the translation of M_UNITDATA.-request invocations to M_DATA.request invocations. The supported translations are lower layer-dependent and include, but not limited to, those specified in clause 13 of IEEE Std 802.1AC[3].

M_UNITDATA.request invocations results in an associated M_DATA.request invocation. During the translation, the M_DATA.request parameters are determined based on the M_UNITDATA.request parameters according to the rules defined for the underlying lower layer[8].

[8] See also [8, p. 21].

# 7. Bridge Port Transmit and Receive Operations

## 7.1. Overview

The architecture of the bridge Port transmit and receive operations in CTF bridges is based on the architecture found in S&F bridges with additions for CTF. The architecture in CTF bridges is shown in Figure 7.1 and Figure 7.2 for VLAN-unaware and

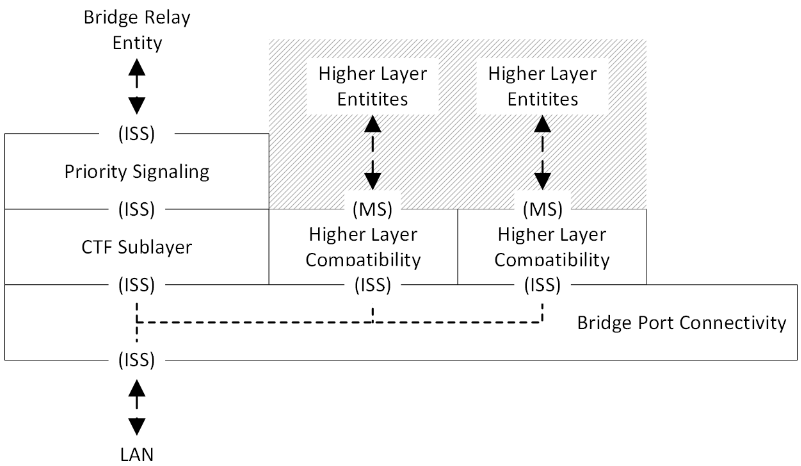


Figure 7.1.: Bridge Port Transmit and Receive (VLAN-unaware).

VLAN-aware CTF bridges, respectively. The elements contained are as follows:

1. Bridges Port Connectivity (7.2) between the access points of the ISS.
2. Priority Signaling in VLAN-unaware CTF bridges (7.4).
3. Translations between ISS and EISS in VLAN-aware CTF bridges (7.4).
4. Higher Layer Compatibility (7.5).
5. CTF Sublayer (7.6).

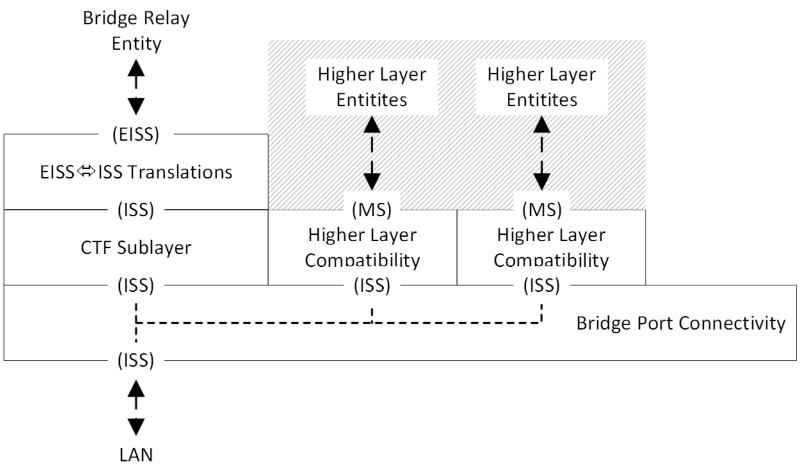


Figure 7.2.: Bridge Port Transmit and Receive (VLAN-aware).

## 7.2. Bridge Port Connectivity

Bridge Port connectivity in CTF bridges is as specified for S&F bridges specified in IEEE Std 802.1Q [2, 8.5.1] with the additional definitions as follows.

For frames under reception originating from the LAN, a copy of such frames for each upper access point is created prior to passing each copy towards the respective upper access point. Frames from the upper access points towards the LAN are passed instantaneously. The multiplexing rules towards the LAN are identical to those of S&F bridges with the addition that frames under reception originating from the bridge relay entity are treated as received frames.

## 7.3. Priority Signaling

### 7.3.1. Receive path operations

VLAN-unaware CTF bridges may or may not implement a shim for support of the ISS with signaled priority to determine values of the drop_eligible and priority parameters (6.2.2) from frames destined towards the bridge relay entity that contain a C-Tag (Customer VLAN Tag) or S-Tag (Service VLAN Tag or Backbone VLAN Tag).

If the shim is not implemented, frames under reception are passed towards the bridge relay entity instantaneously. If the shim is implemented, shim is as specified in IEEE Std 802.1Q [2, 6.20] with additional definitions for frames under reception as follows.

Frames under reception are stalled pending the initial four octets of the mac_-service_data_unit parameter. If the first two octets indicate a C-Tag [2, Table 9-1], the priority and drop_eligible parameters are decoded from the Tag's Control Information [2, 9.6] in the subsequent two octets prior to passing the frame towards the bridge relay entity instantaneously. For any other VLAN Tag [2, Table 9-1], processing falls back to S&F. In absence of any VLAN Tag, the frame is passed towards the bridge relay entity instantaneously.

For frames under reception, the invocation of M_UNITDATA.indication (M_UNITDATA.indication'start) towards the bridge relay entity starts when the frame is passed to the bridge relay entity according to the aforesaid definitions, and ends when the originating invocation of M_UNITDATA.indication ends (M_UNITDATA.indication'end)[1].

### 7.3.2. Transmit path operations

All frames originating from the bridge relay entity are passed towards bridge Port connectivity (7.2) instantaneously.

## 7.4. Translations between Internal Sublayer Service (ISS) and Enhanced Internal Sublayer Service (EISS)

### 7.4.1. Receive path operations

The translations from ISS to EISS can extract and decode C-Tags from the mac_service_data_unit parameter and discard tagged or untagged frames dependent on management parameters. The operations are as specified in IEEE Std 802.1Q[2, 9.6.1], with the following additional definitions for frames under reception.

Frames under reception are stalled pending the initial four octets of the mac_service_data_unit parameter. The frame is then discarded according to the rules specified in IEEE Std 802.1Q [2, 6.9.1], or further processed as follows:

- If the first two octets indicate a C-Tag [2, Table 9-1], the vlan_identifier, priority and drop_eligible parameters are decoded from the Tag's Control Information [2, 9.6] in the subsequent two octets, the first four octets are removed from mac_service_data_unit parameter [2], and the frame is passed towards the bridge relay entity instantaneously.
- If the first two octets indicate a VLAN Tag other than a C-Tag, processing falls back to S&F.
- In all other cases, the frame is passed towards the bridge relay entity instantaneously.

For frames under reception, the invocation of EM_UNITDATA.indication (EM_UNITDATA.indication'start) towards the bridge relay entity starts when the frame is passed

[1]This definition is intended to support the understanding of temporal relationships (e.g., distinction between "frame under reception" and "received frame").

[2]For illustration, removal can be translated to 32 invocations of the remove(mac_service_data_unit,0) function in 6.5.3.

to the bridge relay entity according to the aforesaid definitions, and ends when the originating invocation of M_UNITDATA.indication ends (EM_UNITDATA.indication'end).

### 7.4.2. Transmit path operations

The translations from EISS to ISS on the transmit path of S&F bridges can discard, encode and insert C-Tags into the mac_service_data_unit parameter[3]. The operations are as specified in IEEE Std 802.1Q [2, 9.6.2][4].

## 7.5. Higher Layer Compatibility

Higher layer compatibility ensures that only frames with consistent FCS are passed via the MAC Service Interface to higher layer entities. Therefore, a CTF bridge falls back to S&F prior to passing copies of frames under reception towards higher layer entities and performs the translation between the service primitives of the ISS and the MAC Service as defined in IEEE Std 802.1 AC [3, clause 14].

## 7.6. CTF Sublayer

### 7.6.1. Receive Path Operations

For frames under reception destined towards the bridge relay, the CTF sublayer can emit late errors and fall back to S&F based on the CTFReceptionEnable parameter (9.2.4).

If CTFReceptionEnable is FALSE, processing of a frame under reception is stalled pending all parameters of this frame, including the FCS. If the frame's FCS is consistent, the frame is passed towards the bridge relay instantaneously and discarded otherwise.

If CTFReceptionEnable is TRUE, a frame under reception is towards the relay (7.4 and 7.3) instantaneously.

The CTF sublayer maintains reference to frames under reception after passing these frames towards the bridge relay. If a frame's FCS is inconsistent, the following operations are performed:

- A late error associated with this frame is raised.
- A frame error counter is increased (7.6.3).

[3]Modifications of the mac_service_data_unit parameter in accordance with ISO/IEC 11802-5, IETF RFC 1042 (1988) and IETF RFC 1390 [2, 9.6.2] are incorporated into the queuing decision logic (8.12).

[4]For illustration, insertion can be translated to 32 invocations of the insert(mac_service_data_unit,0,bit) function in 6.5.2.

### 7.6.2. Transmit Path Operations

The transmit path of the CTF sublayer passes frames from the bridge relay entity towards the LAN instantaneously. For any frame that is a under transmission AND a frame under reception (i.e., Cut-Through), the transmit path operations of the CTF sublayer maintains reference to such frames and marks (7.6.3) each of these frames if a late error has been raised by an earlier stage. Such earlier stages include the CTF sublayer receive path (7.6.1) and other processing stages in the bridge relay entity (8).[5]

### 7.6.3. Inconsistent frame handling

Handling of inconsistent frames can increase diagnostic error counters on the receive path (7.6.1), CTFReceptionDiscoveredErrors (9.4.1) and CTFReceptionUndiscovered-Errors (9.4.2), as follows:

- If the frame has been marked by an upstream bridge and this mark was identified as such, CTFReceptionDiscoveredErrors is increased.
- In all other cases, CTFReceptionUndiscoveredErrors is increased.

Marking inconsistent frames on the transmit path (7.6.2) assigns a externally visible indicator to such frames, usually at the end of serial transmission. In existing implementations of CTF bridges, the marking mechanism varies. For example, an implementation may apply a modified FCS determined as follows:

1. Calculate a consistent FCS for the frame.
2. Modify the calculated consistent FCS in a deterministic manner. Examples:
   a) Exchange bits of the FCS at known positions.
   b) Invert bits of the FCS known positions.
   c) Perform an XOR operation between the FCS and a known constant value.
3. Replace the frame_check_sequence parameter of the associated M_UNITDATA.-request invocation with the modified FCS.

Proper interpretation of a marked frames by a receiving CTF bridge requires that the sending CTF bridge upstream is aware of the same marking mechanism. For example, if an sending bridge marks inconsistent frames by inverting all FCS bits, and the receiving bridge expects (FCS $\otimes$ C1-F4-80-21), the receiving bridge will increase CTFReceptionUndiscoveredErrors instead of CTFReceptionDiscoveredErrors even though the frame was marked by the sending bridge.

[5] Truncating frames under transmission is not part of this version of this document, but would be located in this section.

# 8. Bridge Relay Operations

## 8.1. Overview

The structure of the bridge relay entity of CTF bridges is aligned with that of an S&F bridge. Additional definitions for supporting frames under reception for Cut-Through exist primarily in the forwarding process (see also 4).

The structure of the forwarding process in CTF bridges, in terms of processing stages passed by frames, is likewise aligned with that of S&F bridges. It comprises processing stages symmetrical to those found in S&F bridges [2, 8.6 and Figure 8-12] with incorporated processing stages for FRER [4, 8.1 and Figure 8-2][1]. The forwarding process of a CTF bridge, additional elements in the bridge relay and indicated interactions between them are shown in Figure 8.1.

[1] The FRER stages used in this document limit to a subset of those described in IEEE Std 802.1CB when the FRER functions are integrated into the forwarding process, which limits the scope of this document. The given subset is intended to provide the minimum for having stream_handle and sequence_number parameters.

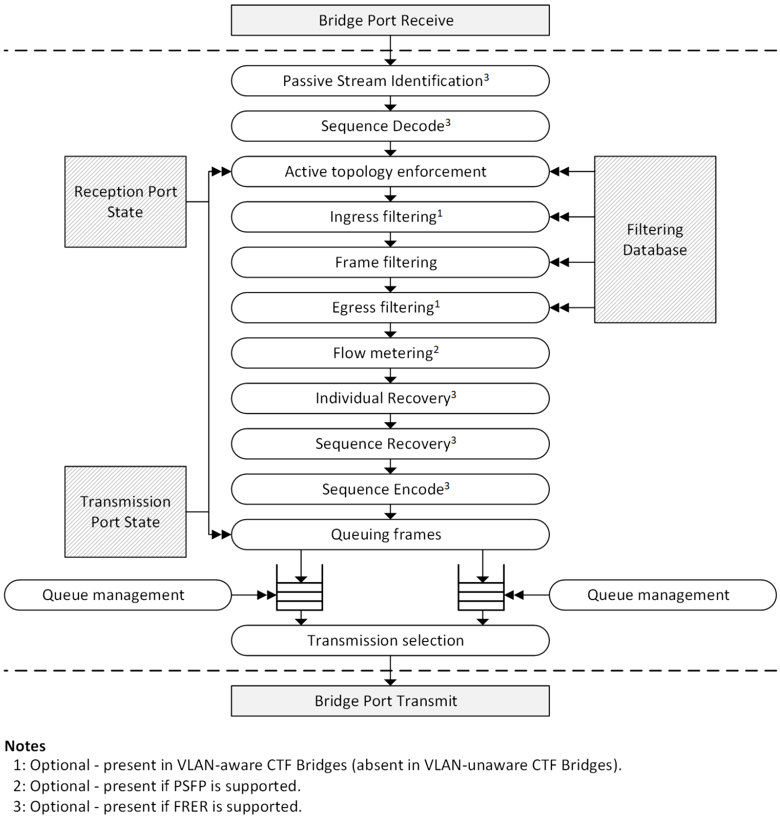


Figure 8.1.: Forwarding process of a CTF bridge.

The processing stages and their subsections are as follows:

1. Passive Stream Identification (8.2)
2. Sequence Decode (8.3)
3. Active topology enforcement (8.4)
4. Ingress filtering (8.5)
5. Frame filtering (8.6)
6. Egress filtering (8.7)
7. Flow classification and metering (8.8)
8. Individual recovery (8.9)
9. Sequence recovery (8.10)

10. Sequence encode (8.11)
11. Queuing frames (8.12), and associated additional definitions for queue management (8.13)
12. Transmission selection (8.14)

The sections of the processing stages are written in a manner that avoids replicating contents of the corresponding sections in the published IEEE 802.1 Standards. The sections provide reference to the corresponding section(s) in the published standards, followed by additional definitions for processing frames under reception. While the emphasis is on processing frames under reception, the stages are equally capable for processing received frames.

## 8.2. Passive Stream Identification

The passive stream identification stage can determine a stream_handle parameter and associate it with a frame. The operation of this stage is as specified in IEEE Std 802.1CB [4, 6.2, 6.4, 6.5, 8.1 and Figure 8-2] with the additional definitions for frames under reception described in the following.

Whether or not a frame under reception can be subject to passive stream identification is dependent on the associated management parameters [4, clause 9]. If it can be precluded that the frame is not subject to passive stream identification[2], the frame is forwarded to the next processing stage (8.3) instantaneously. If this cannot be precluded, processing of the frame stalls pending the necessary parameters of the frame (source_address, destination_address, vlan_identifier, msdu octets, etc.) that are required to determine the following:

1. Whether or not one or more stream stream identification function instance matches the frame, and
2. in case of multiple matching stream identification function instance, to the resolve ambiguity as defined in IEEE Std 802.1CB.

The exact set of parameters required to satisfy the aforesaid conditions is dependent on the stream identification function instances that are actually set in the stream identity table [4, 9.1] and the parameters of the underlying stream identification functions [4, clause 6]. If a stream identification function instance matches, a stream_handle parameter is associated to the frame before the frame is passed to the next processing stage instantaneously.

## 8.3. Sequence Decode

The sequence decode stage is not present in CTF bridges without support for FRER. The stage can extract redundancy tags[3] [4, 7.8] from frames, decode therein con-

[2]For example, if the stream identity table[4, 9.1] is empty.
[3]Consideration of tags other than R-Tag is excluded to limit the scope of this document.

tained sequence_number parameters [4, item b) in 6.1], and assign these parameters to frames. The operation of this stage is as specified in IEEE Std 802.1CB [4, 7.6] with the additional definitions for frames under reception described in the following.

If a frame under reception has no associated stream_handle parameter (8.2), the frame is passed to the next processing stage (8.4) instantaneously. If a frame under reception has an associated stream_handle parameter, processing stalls pending the initial six octets in the mac_service_data_unit parameter. If the first two octets indicate an R-Tag [4, Table 7-1], the sequence_number parameter is decoded from the 5th and 6th octet, the first six octets are removed from the mac_service_data_unit parameter, and the frame is passed to the next processing stage instantaneously.

## 8.4. Active Topology Enforcement

### 8.4.1. Overview

The active topology enforcement stage can determine if frames from reception Ports are submitted to learning, and determines the initial set of potential transmission Ports for each frame. Both operations are as specified in IEEE Std 802.1Q [2, 8.6.1] in CTF bridges, with the additions described in the following for learning (8.4.2) and the initial set of potential transmission Ports (8.4.3) separately.

### 8.4.2. Learning

Learning is based on the source_address (VLAN-unaware and VLAN-aware CTF bridges) and VID (VLAN-aware CTF bridges) parameters of frames for adding dynamic entries in the forwarding database (FDB) as specified in IEEE Std 802.1Q [2, 8.7]. The parameters are submitted to learning only if the following conditions are satisfied:

1. A frame under reception associated with the parameters reached the end of reception.
2. This frame's FCS is consistent.
3. All conditions of an S&F bridge for using the parameters for learning are satisfied [2, 8.4 and 8.6.1].

### 8.4.3. Initial set of potential transmission Ports

The initial set of potential transmission Ports is determined by CTF bridges as specified in IEEE Std 802.1Q [2, 8.6.1].

## 8.5. Ingress Filtering

The ingress filtering stage is not present in VLAN-unaware CTF bridges. The stage discards frames originating from reception Ports based on per frame VID parameters, if present. The conditions under which a frame is discarded by a CTF bridge are identical to those specified in IEEE Std 802.1Q [2, 8.6.2]. Non-discarded frames are passed to the next processing stage (8.6) instantaneously.

## 8.6. Frame Filtering

The frame filtering stage reduces the set of potential transmission Ports (8.4) associated with a frame based on the destination_address (VLAN-unaware and VLAN-aware CTF bridges) and VID (VLAN-aware CTF bridges) parameters, entries in the FDB and management parameters[4]. The operation of this stage is as specified in IEEE Std 802.1Q [2, 8.6.3] with the additional definitions for frames under reception as follows.

In VLAN-aware CTF bridges, an FDB query is performed for each frame under reception instantaneously. In VLAN-unaware CTF bridges, processing stalls pending a frame's destination_address parameter before performing an FDB query for this frame [2, 8.8.9]. Dependent on a query's result by the FDB, processing of the frame under reception falls back to S&F or passes the frame to the next stage instantaneously as follows:

- Whenever the query evaluation by the FDB results in flooding (i.e., query evaluation hits an "ELSE Forward" branch in 8.8.9 of IEEE Std 802.1Q), processing of the frame falls back to S&F[5].
- In all other cases, a frame under reception is passed to the next processing stage instantaneously.

## 8.7. Egress Filtering

The egress filtering stage is only present in VLAN-aware CTF bridges. The stage reduces the set of potential transmission Ports (8.4) associated with a frame based on this frame's VID parameter. The rules under which transmission Ports are removed from this set are identical to those specified in IEEE Std 802.1Q [2, 8.6.4].

## 8.8. Flow Classification and Metering

### 8.8.1. General

The flow classification and metering stage can can apply flow classification and metering to frames that are received on a Bridge Port and have one or more potential transmission ports. The stage is structured into multiple internal (sub)stages in CTF bridges, identical to the structure specified in IEEE Std 802.1Q [2, 8.6.5]. The internal stages and their relationships are shown in Figure 8.2 .

[4] flow_hash [2, item c) in 8.6.3] is excluded to limit the scope of this document.

[5] This fall back is intended to reduce the cases for circulation of inconsistent frames in topological loops, assuming that the performance benefits of CTF traffic that is subject to flooding are of little real-world use.

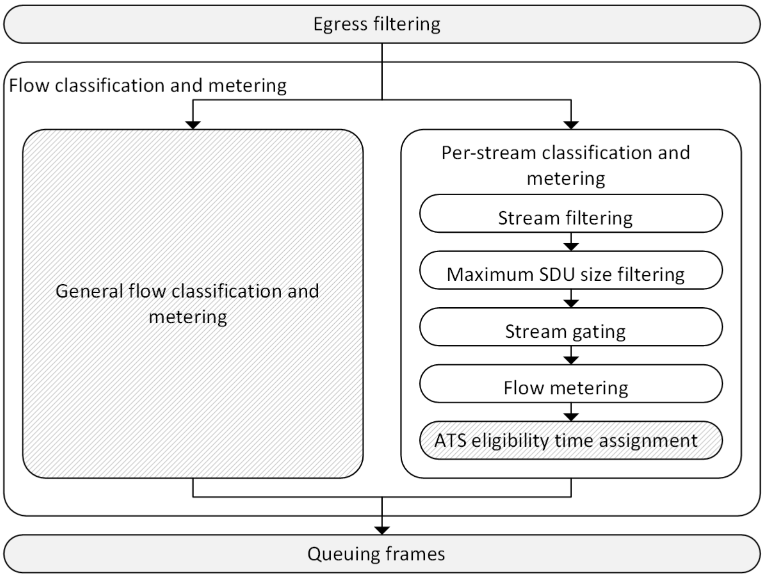


Figure 8.2.: Flow classification and metering.

Support for frames under reception is provided by CTF bridges for the following internal stages:

1. Stream filtering
2. Maximum SDU size filtering
3. Stream gating
4. Flow metering

Processing in CTF bridges falls back to S&F immediately if a frame under reception reaches any other internal stage prior to being processed by this stage.

The operation of stages with support for frames under reception is described in 8.8.2, 8.8.3, 8.8.4 and 8.8.5. All of these stages process frames under reception instantaneously (i.e., stall-free operation).

### 8.8.2. Stream Filtering

The operation of stream filtering for frames under reception is as specified in IEEE Std 802.1Q [2, 8.6.5.3].

### 8.8.3. Maximum SDU size filtering

The operation of maximum SDU size filtering for frames under reception is as specified in IEEE Std 802.1Q [2, 8.6.5.3.1] with the following additional definitions for frames under reception.

When a frame under reception reaches maximum SDU size filtering, an initial number of octets of this frame is already received. This number of octets is used by maximum SDU size filtering for the decision on whether or not this frame is passed to a subsequent processing stage or discarded. If a frame under reception already passed frame maximum SDU size filtering and the associated maximum SDU size limit is exceeded prior to the frame's end of reception, a late error for that frame is indicated for handling by subsequent processing stages in a CTF bridge.

### 8.8.4. Stream Gating

The operation of stream gates for frames under reception is as specified in IEEE Std 802.1Q [2, 8.6.5.4] with the following additional definitions for frames under reception.

When frame under reception reaches a stream gate, this frame is only passed to the next processing stage if the gate is in an open state. The frame is discard otherwise prior to being passed to the next processing stage. If a stream If a stream gate closes prior to the end of the frame under reception, a late error for this frame is indicated immediately for handling by subsequent processing stages in a CTF bridge.

### 8.8.5. Flow Metering

The operation of stream gates for frames under reception is as specified in IEEE Std 802.1Q [2, 8.6.5.5] with the following additional definitions for frames under reception.

When a frame under reception reaches flow metering, an initial number of octets of this frame is already received. This number of octets is used by the associated flow meter for the decision on whether or not this frame is passed to a subsequent processing stage or discarded. If a frame under reception already passed flow metering and the limit of the flow meter is subsequently exceeded prior to the frame's end of reception, a late error for this frame is indicated for handling by subsequent processing stages in a CTF bridge.

## 8.9. Individual Recovery

The individual recovery stage is not present in CTF bridges without support for FRER. If present, the stage can associate frames belonging to individual Member streams [4, 7.4.2] with therefore configured instances of the Base recovery function [4, 7.4.3], which then discard frames with repeating sequence_number parameters (8.3) on a per Member stream resolution. The operation of the individual recovery stage is as specified in IEEE Std 802.1CB [4, 7.5], with the following additions for CTF bridges.

If frames under reception are associated with a Base recovery function for individual recovery, processing falls back to S&F prior to performing individual recovery[6].

[6] Falling back to S&F ensures that individual recovery does not falsely discard a frame with correct sequence_number parameter (and consistent FCS) after accepting a frame with incorrect but identical sequence_number (and inconsistent FCS) earlier. The same rationale applies in 8.10.

## 8.10. Sequence Recovery

The sequence recovery stage is not present in CTF bridges without support for FRER. If present, the stage can associate frames belonging to sets of Member streams with therefore configured instances of the Base recovery function [4, 7.4.3], which then remove frames with repeating sequence_number parameters[4, item b) in 6.1] on a per Member stream set resolution. The operation of the sequence recovery stage is as specified in IEEE Std 802.1CB [4, 7.4.2], with the following additions for CTF bridges.

If frames under reception are associated with a Base recovery function for sequence recovery, processing falls back to S&F prior to performing sequence recovery.

## 8.11. Sequence Encode

The sequence recovery stage is not present in CTF bridges without support for FRER. If it is present, the stage can encode and insert R-Tags into the mac_service_data_unit parameter based on the sequence_number parameter associated with these frames. The operation of the sequence encode stage for frames under reception is as specified in IEEE Std 802.1CB [4, 7.6 and 7.8].

## 8.12. Queuing Frames

The queuing frames stage queues each received frame to a per-traffic class queue of each remaining potential transmission Port associated with the frame (8.4, 8.6 and 8.7). The stage operates as specified in IEEE Std 802.1Q [2, 8.6.6] with the following additional definitions for frames under reception.

Before a frame under reception is queued, a per-queue copy of a frame is created before queuing and considered separately according to Algorithm 8.1 . The algorithm determines whether or not subsequent atomic transmission (8.14 and 5.2) of frames under reception is possible and if not, discard such frames in case of configuration errors or fall back to S&F prior to queuing such frames.

## 8.13. Queue Management

The rules for removing frames from IEEE Std 802.1Q [2, 8.6.7] remain unaltered in CTF bridges.

In addition to this, CTF bridges may remove a frame from a queue if all of the following conditions are satisfied[7]:

1. The frame was queued while it was under reception.
2. A processing stage before queuing(8.12) raised a late error for that frame.

[7] Erroneous frames removed according to this additional rule will not become visible on the LAN of an associated transmission Port, because such frames can be removed before being selected by transmission selection .

**Algorithm 8.1** Queuing rules for frames under reception.

**IF**
(the associated CTFTransmissionEnable parameter [9.2.2] is FALSE) **OR**
(the associated transmission selection algorithm is not strict priority [2, 8.6.8.1])
**THEN**
Processing falls back to S&F before queuing the frame instantaneously.
**ELSE IF**
(the associated CTFTransmissionEnable parameter [9.2.2] is TRUE) **AND**
(**CTFInconsistencyCondition**)
**THEN**
The frame is discarded before queuing.
**ELSE**
The frame is queued instantaneously.
**END IF**

**CTFInconsistencyCondition** =
(transmission link speed of the frame $>$ reception link speed of the frame) **OR**
(mac_service_data_unit modification required in accordance with
ISO/IEC 11802-5, IETF RFC 1042 (1988) and IETF RFC 1390)

3. the end of reception of the frame was reached before the frame was selected for transmission (8.14).

## 8.14. Transmission Selection

Transmission selection determines whether frames in per traffic class queues are available for transmission, determines transmission ordering and transmission times of queued frames, de-queues frames for transmission and initiates transmission. Transmission selection in CTF bridges is as specified in IEEE Std 802.1Q [2, 8.6.8].

# 9. Management Parameters

## 9.1. Overview

The management parameters for CTF fall into three categories:

1. Control Parameters (9.2)
2. Timing Parameters (9.3)
3. Error Counters (9.4)

The control parameters allow to (i) determine whether CTF is supported on a per Port and per Port per Traffic Class resolution, and if CTF is supported, to (ii) enable and disable CTF on these resolutions. These parameters are available in reception Ports and transmission Ports. For a pair of bridge ports, frames can only be subject to the CTF operation if CTF is supported and enabled on both Ports.

The timing parameters expose the delays experienced by frames passing from a particular reception Port to another transmission Port. These parameters are primarily intended for automated network and traffic configuration, for example, by a Centralized Network Controller (CNC) using the associated mechanisms from IEEE Std 802.1Q [2, clause 46].

The error counters expose information on frames that were subject to the CTF operation in a bridge, even though such frames have consistency errors (i.e., a frame check sequence inconsistent with the remaining contents of that frame) during reception by this bridge. These counters are primarily intended for manual diagnostic purposes to support identifying erroneous links or stations, for example, by a human network administrator.

## 9.2. Control Parameters

### 9.2.1. CTFTransmissionSupported

A Boolean read-only parameter that indicates whether CTF on transmission is supported (TRUE) or not (FALSE). There is one CTFTransmissionSupported parameter for each traffic class of each transmission Port.

### 9.2.2. CTFTransmissionEnable

A Boolean parameter to enable (TRUE) and disable (FALSE) CTF on transmission. There is one CTFTransmissionEnable parameter for each traffic class of each transmission Port. The default value of the CTFTransmissionEnable parameter is FALSE for all traffic classes of all transmission Ports. It is an error if a CTFTransmissionEnable is set to TRUE if the associated CTF Transmission Supported parameter is FALSE.

### 9.2.3. CTFReceptionSupported

A Boolean read-only parameter that indicates whether CTF on reception is supported (TRUE) or not (FALSE). There is one CTFReceptionSupported parameter for each reception Port.

### 9.2.4. CTFReceptionEnable

A Boolean parameter to enable (TRUE) and disable (FALSE) CTF on reception. There is one CTFReceptionEnable parameter for each reception Port. The default value of the CTFReceptionEnable parameter is FALSE for all reception Ports.It is an error if a CTFReceptionEnable is set to TRUE if the associated CTFReceptionSupported parameter is FALSE.

## 9.3. Timing Parameters

### 9.3.1. CTFDelayMin and CTFDelayMax

A pair of unsigned integer read-only parameters, in units of nanoseconds, describing the delay range for frames that are subject to the CTF operation and encounter zero delay for transmission selection [2, 8.6.8]. This occurs when the queue for the frame's traffic class is empty, the frame's traffic class has permission to transmit, and the egress Port is idle (not transmitting). There is one pair of CTFDelayMin and CTFDelayMax parameters per reception Port per transmission Port traffic class pair.

## 9.4. Error Counters

### 9.4.1. CTFReceptionDiscoveredErrors

An integer counter, counting the number of received frames with discovered consistency errors. There is one CTFReceptionDiscoveredErrors parameter for each reception Port. A frame with discovered consistency errors has been identified as such by a bridge on the upstream path from which the frame originates and marked by that an implementation-dependent marking mechanism. The value of the counter always increases by one

1. if
    a) the upstream bridge that applied the marking,
    b) all bridges on the path of that bridge to the reception Port associated with the CTFReceptionDiscoveredErrors counter and

   c) the receiving bridge of which the reception Port is a part of are different instances of the same bridge implementation, and

2. the underlying marking mechanism is identical for all these instances if multiple marking mechanisms are supported by these instances.

If either of the conditions in items 1 through 2 is unsatisfied, CTFReceptionUndiscoveredErrors may be increased instead of CTFReceptionDiscoveredErrors[1].

### 9.4.2. CTFReceptionUndiscoveredErrors

An integer counter, counting the number of received frames with undiscovered consistency errors. There is one CTFReceptionUndiscoveredErrors parameter for each reception Port. This counter is increased by one if a frame with consistency errors is received at the associated reception Port and CTFReceptionDiscoveredErrors is not increased.

[1] It is assumed that there is a variety of options for implementing a frame marking mechanism. For example, by using physical layer symbols [10, l.121 - l.126] or special frame check sequences [11, p.54, 2.2.][12, p.17]. The current description in this document permits any marking mechanism, but the associated error counters are only consistent in networks with homogeneous implementation instances, and may be inconsistent in heterogeneous networks. However, term (CTFReceptionDiscoveredErrors + CTFReceptionUndiscoveredErrors) on a reception Port should be identical in several heterogeneous networks. A human network administrator may be able to localize erroneous links or stations solely by considering this term along multiple reception Ports across a network instead of its constituents.

# Part III.

# Cut-Through Forwarding in Bridged Networks

PLACEHOLDER, for contents on using CTF in networks [11, p.46 – p.49].

# Part IV.

# Appendices

# A. Interaction of the Lower Layer Interface (LLI) with existing Lower Layers

## A.1. PLS Service Interface

### A.1.1. Overview

This section summarizes how interfacing between the PLS service primitives on top of the Reconciliation sublayer [13, clause 22, clause 35, etc.] and LLI (6.1) is possible, similar to the interfacing of the original GSCF [8]. Interfacing between PLS service primitives and LLI can be established by three processes that translate between the LLI global variables (6.4) and the PLS service primitives. The processes and interactions are shown in Figure A.1.

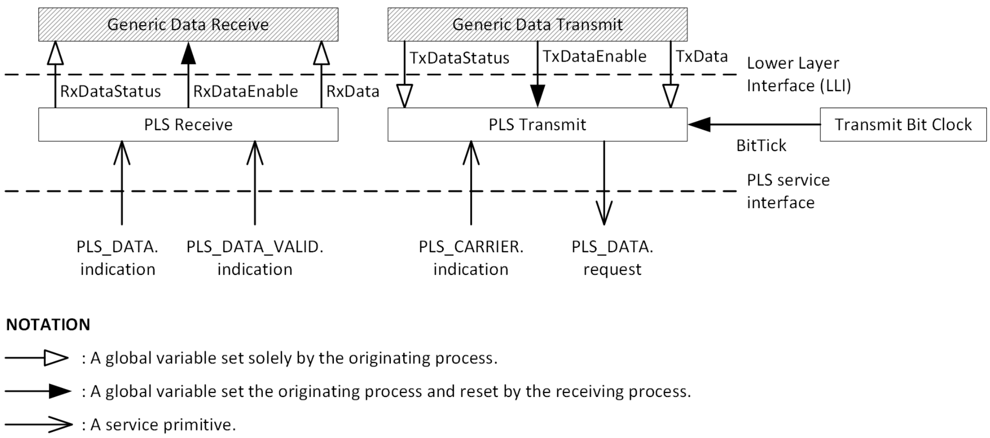


Figure A.1.: Processes and interactions for interfacing between LLI and PLS service primitives.

### A.1.2. Service Primitives

The PLS_DATA.indication, PLS_DATA_VALID.indication, PLS_CARRIER.indication and PLS_DATA.request service primitives are as specified in IEEE Std 802.3 [13, clause 6] limiting on full duplex mode[1].

### A.1.3. Global Variables and Constants

#### A.1.3.1. BitTick

A global Boolean variable, used to generate a bit clock for the PLS Transmit process.

#### A.1.3.2. LEN_FRAMEGAP

An integer constant defining the duration of the Inter-Frame Gap (IFG), in bits.

### A.1.4. Global Constraints

The following constraints are introduced for the Global Constants in sections 6.3 and A.1.3:

1. PREAMBLE = “10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011“[2]
2. LEN_MIN = 8*64 + PREAMBLE.length
3. LEN_MAX = 8*1500 + PREAMBLE.length
4. LEN_FCS = 32
5. LEN_DATA = 1
6. LEN_FRAMEGAP = 8*12

### A.1.5. Transmit Bit Clock process

The Transmit Bit Clock process periodically sets the BitTick variable to TRUE, where the period equals the duration of a Bit on the physical layer.

### A.1.6. PLS Transmit process

#### A.1.6.1. Description

The PLS Transmit process translates between global variables from the Generic Data Transmit process (6.9) and the PLS_CARRIER.indication and PLS_DATA.request service primitives (A.1.2).

#### A.1.6.2. State Machine Diagram

The operation of the PLS Transmit process is defined by the state machine diagram in Figure A.2.

[1] The PLS_SIGNAL.indication service primitive is effectively not required in this mode [13, 6.3.2.2.2]
[2] First bit in quotes is PREAMBLE[0], second bit in quotes is PREAMBLE[1], etc. whitespaces are ignored.

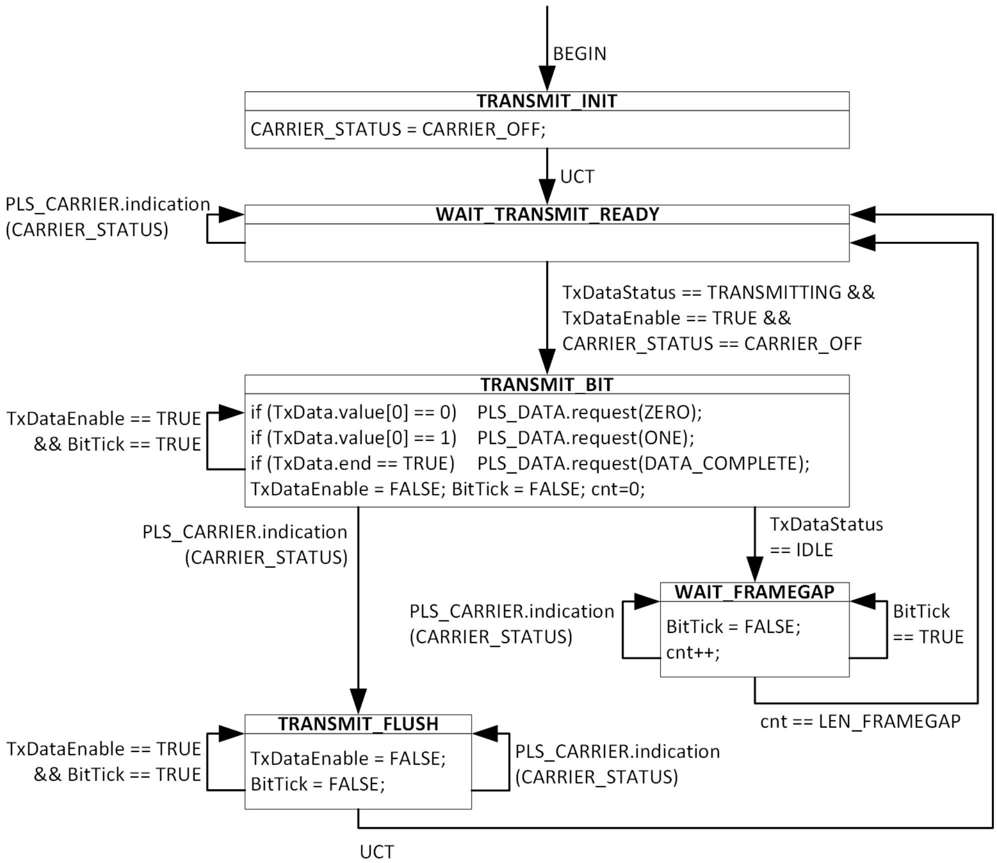


Figure A.2.: State machine diagram of the PLS Transmit process.

### A.1.6.3. Variables

**A.1.6.3.1. cnt** An integer variable for counting bits.

**A.1.6.3.2. CARRIER_STATUS** A variable holding to most recent value received by a PLS_CARRIER.indication invocation (A.1.2).

## A.1.7. PLS Receive process

### A.1.7.1. Description

The PLS Receive process translates between global variables from the Generic Data Receive process (6.6) and the PLS_CARRIER.indication and PLS_DATA.request service primitives (A.1.2).

### A.1.7.2. State Machine Diagram

The operation of the PLS Receive process is defined by the state machine diagram in Figure A.3.

### A.1.7.3. Variables

**A.1.7.3.1. cData** A variable of type low_data_t (6.5), used for implementing a delay line of a single bit.

**A.1.7.3.2. DATA_VALID_STATUS** A variable holding to most recent value received by a PLS_DATA_VALID.indication invocation (A.1.2).

**A.1.7.3.3. INPUT_UNIT** A variable holding to most recent value received by a PLS_DATA.indication invocation (A.1.2).

## A.1.8. Support for Preemption

Connecting to the MAC Merge sublayer [13, clause 99] instead of the Reconciliation sublayer for supporting preemption may be realized as shown in [8, p. 22] due to the identical service primitives and the re-composition of atomic per-frame bits streams in the pMAC.

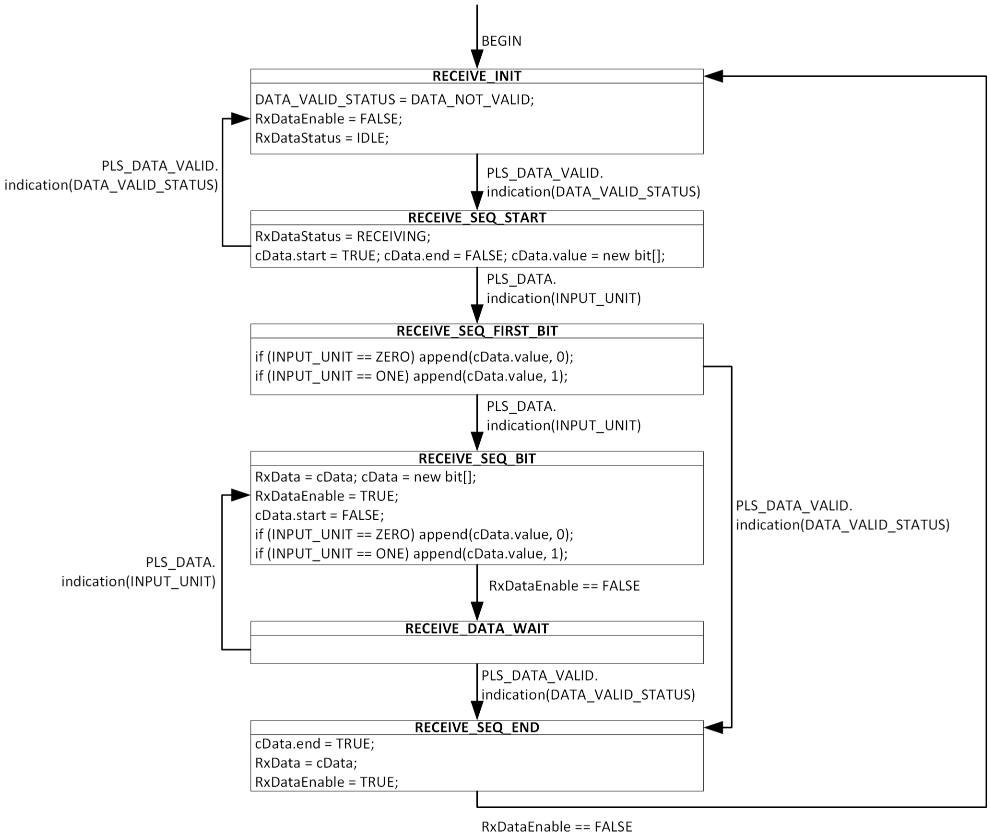


Figure A.3.: State machine diagram of the PLS Receive process.

# Bibliography

[1] IEEE Standards Association, *2021 IEEE SA Standards Style Manual.* [Online]. Available: https://mentor.ieee.org/myproject/Public/mytools/draft/styleman.pdf

[2] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014) and published amendments*, pp. 1–1993, 2018.

[3] "IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Service Definition," *IEEE Std 802.1AC-2016 (Revision of IEEE Std 802.1AC-2012)*, pp. 1–52, 2017.

[4] "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017 and published amendments*, pp. 1–102, 2017.

[5] E. Frank Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970. [Online]. Available: http://dl.acm.org/citation.cfm?id=362685

[6] "IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Service Definition," *IEEE Std 802.1AC-2016 (Revision of IEEE Std 802.1AC-2012)*, pp. 1–52, 2017.

[7] Johannes Specht (Self; Analog Devices, Inc.; Mitsubishi Electric Corporation; Phoenix Contact GmbH & Co. KG; PROFIBUS Nutzerorganisation e.V.; Siemens AG; Texas Instruments, Inc.), *An Idealistic Model for P802.1DU.* [Online]. Available: https://mentor.ieee.org/802.1/dcn/22/1-22-0015-01-ICne-idealistic-model-for-p802-1du.pdf

[8] Roger Marks (EthAirNet Associates), *Generic Serial Convergence Function (GSCF)*, 2022. [Online]. Available: https://mentor.ieee.org/802.1/dcn/22/1-22-0040-02-ICne-generic-serial-convergence-function-gscf.pdf

[9] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.

[10] Astrit Ademaj (TTTech) and Guenter Steindl (Siemens), *Cut-Through – IEC/IEEE 60802 – V1.1*, 2019. [Online]. Available: https://www.ieee802.org/1/files/public/docs2019/60802-Ademaj-et-al-CutThrough-0919-v11.pdf

[11] Johannes Specht, Jordon Woods, Paul Congdon, Lily Lv, Henning Kaltheuner, Genio Kronauer and Alon Regev, *IEEE 802 Tutorial: Cut-Through Forwarding (CTF) among Ethernet networks – DCN 1-21-0037-00-ICne*, 2021. [Online]. Available: https://mentor.ieee.org/802.1/dcn/21/1-21-0037-00-ICne-ieee-802-tutorial-cut-through-forwarding-ctf-among-ethernet-networks.pdf

[12] Peter Jones (Cisco), *802.3 NEA CTF: CTF concerns*, 2022. [Online]. Available: https://www.ieee802.org/3/ad_hoc/ngrates/public/calls/22_0427/jones_nea_01_220427.pdf

[13] "IEEE Standard for Ethernet," *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, pp. 1–5600, 2018.