Technical Descriptions for Cut-Through Forwarding in Bridges

DCN 1-22-0042-08-ICne

Author: Johannes Specht

October 27, 2022

Contents

List of Figures

Part I.

Introduction

1. Purpose

Purpose of this document is to provide input for technical discussion in pre-PAR activities of IEEE 802 (i.e., Nendica). 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, and thereby also address the desire expressed by some individuals during the IEEE 802 Plenary Meeting in July 2022 to a certain extent.

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 and simplifications. 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 Cut-Through Forwarding (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 (itself likewise aligned with the architectural figures in IEEE Std 802.1Q [2, Figure 8-2, 8-3, 8-4, ff.]) in a compact form.

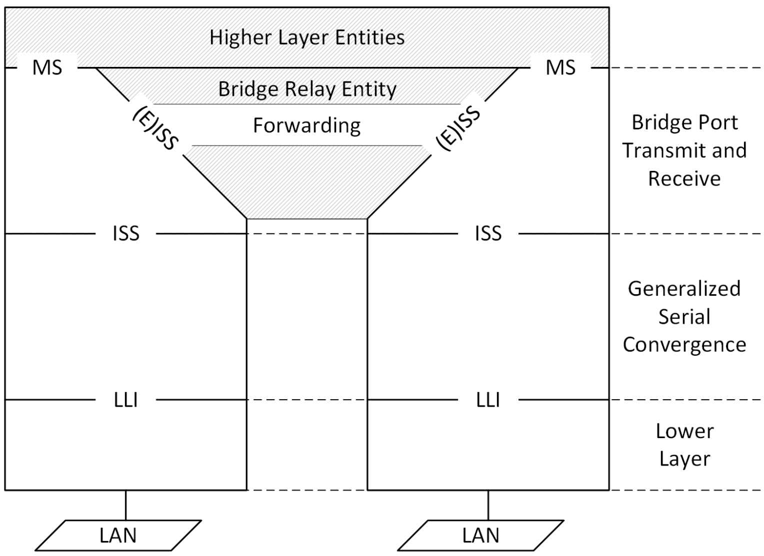


Figure 4.1.: Architecture of a Cut-Through Forwarding (CTF) Bridge.

This architecture comprises the following elements:

1. One or more higher layer entities above the MAC Service (MS) interface.
2. A bridge relay entity (8) that relays frames between different bridge Ports.
3. Generalized serial convergence operations (6) that translate between the Internal Sublayer Service (ISS) interface and Lower Layer Interface (LLI) per bridge Port.
4. 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.

The operation of CTF bridges is described in this document in the chapters referred to before, typically limiting on describing the additions and potential differences to the operations of S&F bridges.

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.

In general, this part of the document limits the use of Cut-Through to operations standardized in IEEE Stds 802.1Q[2], 802.1AC[3] and 802.1CB[4].

[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].

5. Modeling Conventions

5.1. Resolutions

5.1.1. Bit-Accurate Modeling

All invocations of service primitives in this document are atomic. That is, each invocation is non-dividable (see also 7.2 of IEEE Std 802.1AC[3]). Service primitive invocations are modeled more explicitly in this document, allowing for accurate description of operations within a Bridge, while retaining atomicity. This explicit model comprises the following:

1. A service primitive provides two attributes[1], *'start* and *'end*. These attributes are used in subsequent descriptions to indicate the start and the end of the indication, respectively.

2. The parameters of a service primitive are explicitly modeled as bit arrays.

3. The values of parameters during invocations of a service primitive are passed according to a call-by-reference scheme.

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.

5.1.2. Parameter-Accurate Modeling

At higher levels 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 (5.2) in and across processing stages,

2. enable re-use of existing transformation rules from IEEE 802.1 Stds by reference, and

3. avoid low level details that would not provide any value to the clarity and unambiguous descriptions.

[1]The concept of *attributes* is inspired by the *Very High Speed Integrated Circuits Hardware Description Language*, VHDL[5], which provides predefined attributes (e.g., *'transaction*) that allow modeling over multiple VHDL simulation cycles at the same instant of simulated time.

The parameter-accurate operates at the resolution of symbolic and/or numeric parameters instead of bit arrays (5.1.1). 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,
2. the result of composition and/or computation across bits located at various locations in a frame,
3. based on out-of-band information, or
4. any combination of the aforesaid.

As an example, the vlan_identifier parameter of EM_UNITDATA.indication (7.3) 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 data transformations in existing IEEE 802.1 Standards.

5.2. Temporal Control

5.2.1. Processing Stalls

Parameter-accurate modeling allows formulating temporal control in processing stages. A processing stage (5.1.1) may *stall* further processing of a frame, including (but not limited to) passing this frame to a subsequent processing stage, until one or more parameters are complete (5.1.2), subject to the implicit discarding due to late errors (5.2.2). 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

[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.

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 prior to the filtering database lookup of the destination ports."*

5.2.2. 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 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.

5.2.3. 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, itself a shortcut for stalling processing pending all parameters of a frame, including the FCS.
2. Dependent on whether or not a late error was indicated by an earlier processing stage for that frame:
   a) Late error indicated:
      The frame is discarded prior to any further processing by any stage.
   b) No Late error indicated:
      The frame continues subsequent processing through subsequent processing stages according to the standardized behavior of an S&F bridge.

5.2.4. Instantaneous Operations

In absence of processing stalls/data dependencies, processing stages as modeled in this document perform actions instantaneously. It is clear that instantaneous operations, in terms of 0-delay at an infinite high resolution[3], are not possible due to physical

[3] The semantics of "instantaneous" can vary dependent on the resolution [6, p.11].

constraints and in real world implementations. The latter introduce additional delays, and the model is not intended to limit such delays, other than 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 these delays could be determined by observation/measurement and are available as management parameters (9.3).

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 the translation between the Internal Sublayer Service (ISS) and a broad range of lower layers, including (but not limited to) physical layers. Figure 6.1 provides an overview of these processes and their interaction[1]. The processes can

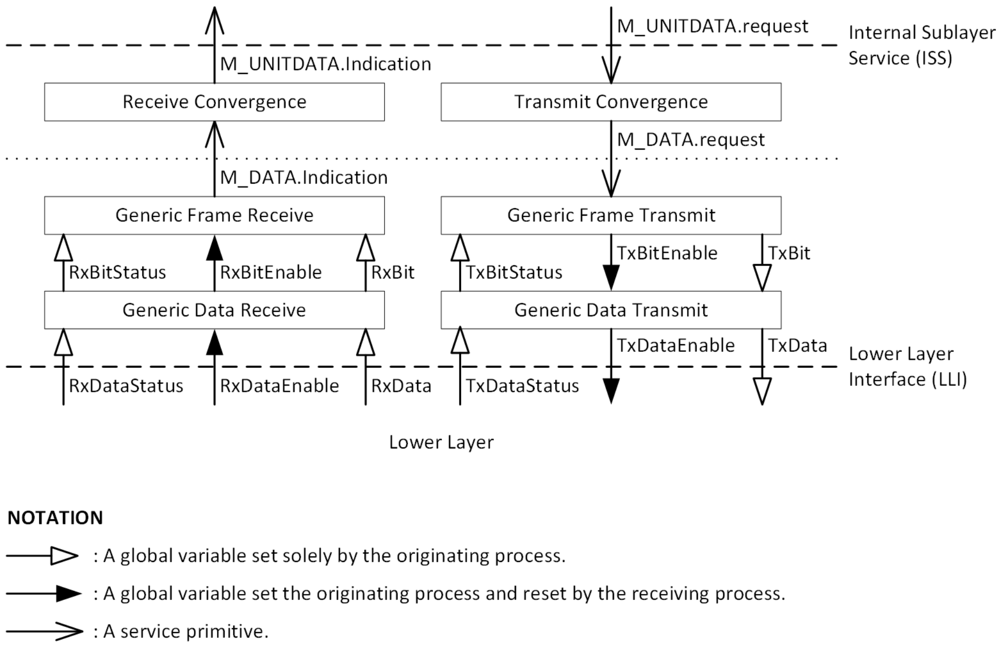


Figure 6.1.: Overview of the generalized serial convergence operations.

be summarized as follows:

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

1. A Receive Convergence process (6.7) that translates each invocation of the M_DATA.-indication service primitive (6.2.1) into a corresponding invocation of the M_UNIT-DATA.indication service primitive (6.2.2).

2. A Generic Frame Receive process (6.6) 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 (6.5) 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 (6.10) 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 (6.9) that translates M_DATA.request invocations into bit sequences for the Generic Data Transmit process.

6. A Generic Data Transmit process (6.8) that translates bit sequences from the Generic Frame Transmit process into a lower layer-dependent serial data stream.

The generalized serial convergence operations are inspired by the concepts described in slides by Roger Marks [7, slide 15], but follow a different modeling approach with more formalized description of these functions 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. The differences can be summarized as follows:

– Alignment with the state machine diagram conventions in Annex E of IEEE Std 802.1Q[2].

– Support for serial data streams from lower layers with arbitrary data word length[3].

– Explicit modeling of atomic ISS service primitive invocations.

By keeping ISS service primitive invocations atomic, the approach in this document is intended to provide a higher level of compatibility with existing IEEE 802.1 Stds, similar to the modeling approach via frame look-ahead of service primitive invocations/prescient functions[8, slides 7ff.].

[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. In addition, the support for word sizes (e.g., 8 bits, 32 bits or 64 bits) may be close to realities found in hardware implementation. It is subject to discussion whether this and other generalizations over [7] introduced by the author are considered to be helpful.

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. This parameter signatures of the service primitives are as follows[4]:

M_DATA.indication(DA, SA, SDU, FCS)

M_DATA.request(DA, SA, SDU, FCS)

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. SDU

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 parameter signatures summarized as follows:

[4]The parameters in this version of this document limit to those introduced in Roger Marks' GSCF slides [7]. Future versions may introduce more flexibility (e.g., for IEEE Std 802.11 [9, 9.2]).

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

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,

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

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 width of the RxData variable, 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 6.1 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:

RECEIVING Indicates that the Generic Data Receive process received data from lower layers in a serial stream without knowledge of the remaining length of the overall data stream.

TRAILER Indicates that the Generic Data Receive process received data from lower layers in a serial stream 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 6.1. The semantics of the constituent parameters is as follows:

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.

6.4.6. 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:

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

IDLE Indicates that data stream reception from lower layers is not 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 to the Generic Data Transmit process.

6.4.9. TxBitStatus

An enumeration variable that establishes a back pressure mechanism from the Generic Data Transmit process to the Generic Frame Transmit process. The valid enumeration literals are as follows:

READY Indicates that the Generic Data Transmit process can accept one or more bit(s) from the Generic Frame Transmit process.

BUSY Indicates that the Generic Data Transmit process cannot accept bits from the Generic Frame Transmit process.

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 (6.1), 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 establishes a back pressure mechanism from the lower layer to the Generic Data Transmit process. The valid enumeration literals are as follows:

READY Indicates that a lower layer can accept one or more bit(s) from the Generic Data Transmit process.

BUSY Indicates that a lower layer cannot accept bits from the Generic Data Transmit process.

6.5. Generic Data Receive

The Generic Data Receive process translates a lower layer-dependent[6] serial data stream into a uniform bit stream. In addition, it realizes the following functions:

– Determine the position in the serial data stream of a frame at which the frame check sequence begins (delay line modeling).

– Truncate excess bits to satisfy the frame length requirements implied by the parameter definition of the M_DATA.indication primitive (6.2.1).

6.6. Generic Frame Receive

6.6.1. 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.

6.6.2. State Machine Diagram

The operation of the Generic Frame Receive process is specified by the state machine diagram in Figure 6.2 , using the variables and functions defined in subsequent subclauses.

6.6.3. Variables

6.6.3.1. cnt

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

6.6.3.2. len

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

6.6.3.3. 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 an the remaining parameters of a frame under reception.

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

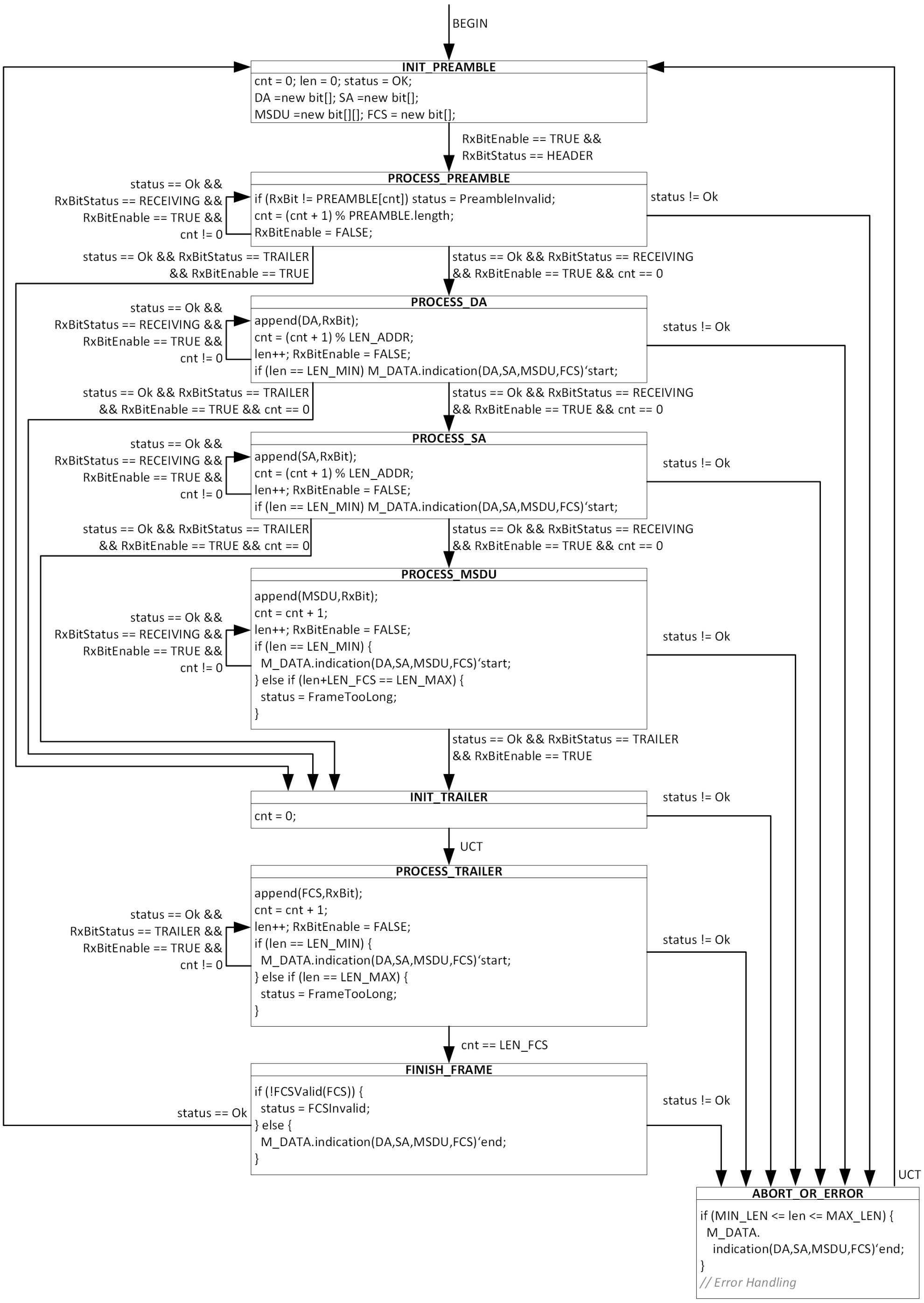


Figure 6.2.: State Machine Diagram of the Generic Frame Receive Process.

6.6.4. Functions

6.6.4.1. append(parameter,bit)

The append function appends a given bit at the end of a particular parameter of an M_DATA.indication service primitive.

6.6.4.2. 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).

6.7. Receive Convergence

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 are 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 extracted from the M_DATA.indication parameters according to the rules defined for the underlying lower layer.

6.8. Generic Data Transmit

PLACEHOLDER, for descriptions symmetrical to 6.5.

6.9. Generic Frame Transmit

PLACEHOLDER, for descriptions symmetrical to 6.6.

6.10. Transmit Convergence

PLACEHOLDER, for descriptions symmetrical to 6.7.

7. Bridge Port Transmit and Receive Operations

7.1. Overview

The architecture of the bridge port transmit and receive operations in CTF bridges is identical to the architecture of S&F bridges. The architecture is shown in Figure 7 and comprises the following elements:

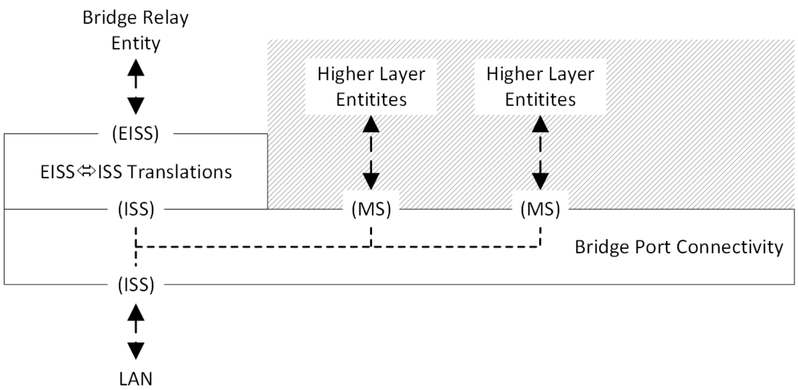


Figure 7.1.: Bridge Port Transmit and Receive.

1. Connectivity (7.2) between the access points of the generalized serial convergence operations (6), higher layer entities, and the bridge relay entity (8).
2. Translations between ISS and EISS (7.3).

7.2. Bridge Port Connectivity

Bridge Port connectivity in a CTF bridge is identical to S&F bridges specified in IEEE Std 802.1Q [2, 8.5.1] with the additions described in this section.

For frames under reception originating from the generalized serial convergence operations, a copy of such frames for each access point determined according to the rules in 8.5.1 IEEE 802.1Q. If a frame copy is destined to the bridge relay entity and the CTFReceptionEnable parameter (9.2.4) of the reception Port is set TRUE, this copy is passed instantaneously to the translation from ISS to EISS (7.3). In all other cases, CTF bridges fall-back to S&F for frames under reception originating from the generalized serial convergence operations prior to passing the respective copies to the associated access point.

Frames originating from the bridge relay entity or higher layer entities destined for the generalized serial convergence operations are passed instantaneously to the latter. The multiplexing rules in this case are identical to those of S&F bridges except that frames under reception originating from the bridge relay entity are deemed as complete frames for which no subsequent contents are expected.

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

7.3.1. Data translations

Data translation from service primitive invocations of the ISS and service primitive invocations of the EISS follows the associated rules specified in IEEE Std 802.1Q [2, 6.9].

7.3.2. Temporal relationship

7.3.2.1. Data indications

The temporal relationship (5.2) between M_UNITDATA.indication invocations of the ISS and the EM_UNITDATA.indication invocations of the EISS is as follows:

1. For EM_UNITDATA.indication invocations, EM_UNITDATA.indication'start and EM_UNITDATA.indication'end follow instantaneously after M_UNITDATA.-indication'start and M_UNITDATA.indication'end, respectively.

7.3.2.2. Data requests

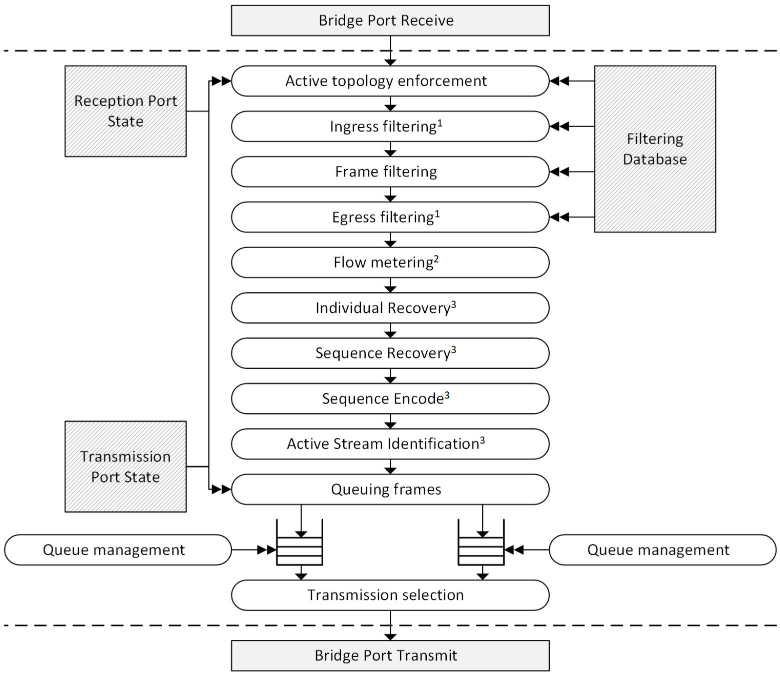
The temporal relationship between EM_UNITDATA.request invocations of the EISS and the EM_UNITDATA.request invocations of the ISS is as follows:

1. For EM_UNITDATA.request invocations, M_UNITDATA.request'start and M_UNIT-DATA.request'end follow instantaneously after EM_UNITDATA.indication'start and EM_UNITDATA.indication'end, respectively.

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. 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 Frame Replication and Elimination for Reliability [4, 8.1 and Figure 8-2]. The forwarding process of a CTF bridge, additional elements in the bridge relay and indicated interactions between them are shown in Figure 8.1.



Notes
1: Optional - present in VLAN-aware CTF Bridges (absent in VLAN-unaware CTF Bridges).
2: Optional - present if PSFP is supported.
3: Optional - present if FRER is supported.

Figure 8.1.: Forwarding process of a CTF bridge.

The processing stages and the respective sections in this document are as follows:

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

10. Queuing frames (8.11), and associated additional definitions for queue management (8.12)

11. Transmission selection (8.13)

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. Instead, section 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 to process frames for which the end of reception was reached. for the latter case, the behavior of the processing stages is identical to that of an S&F bridge.

8.2. Active Topology Enforcement

8.2.1. Overview

The active topology enforcement stage determines if frames from reception Ports are used for 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 (8.2.2) and the initial set of potential transmission Ports (8.2.3) separately.

8.2.2. Learning

Learning is based on the the source address and VID parameters of frames for adding entries in the forwarding database (FDB) as specified in IEEE Std 802.1Q [2, 8.7]. In CTF bridges, the source address and VID parameters are used for learning 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].

8.2.3. 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]. If this determination depends on the VID parameter of a frame under reception, processing stalls pending this parameter prior to passing the frame under reception to the next processing stage:

– Ingress filtering (8.3) for VLAN-aware CTF bridges

– Frame filtering (8.4) for VLAN-unaware CTF bridges

In absence of this dependency, the frame under reception is passed to the next processing stage instantaneously.

8.3. Ingress Filtering

The ingress filtering stage discards frames originating from reception Ports based on the VID parameters associated with these frames. 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]. Frames under reception are stalled by VLAN-aware CTF bridges pending the VID parameter and passed to the next processing stage (8.4) unless they are discarded and therefore not passed. The ingress filtering stage is only present in VLAN-aware CTF bridges.

8.4. Frame Filtering

The frame filtering stage reduces the set of potential transmission Ports associated with a frame based on parameters associated with this frame (destination address, VID, etc.) and querying the FDB of a bridge. The exact set of parameters of a frame is determined as specified in IEEE Std 802.1Q [2, 8.6.3]. If necessary, a CTF bridge stalls processing pending all necessary parameters of a frame under reception before performing an FDB query for this frame [2, 8.8.9].

Dependent on the query's evaluation by the FDB, processing of a frame under reception falls back to S&F or passes the frame to the next stage instantaneously:

– 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 prior to passing it to the next processing stage[1].

– In all other cases, a frame under reception is passed to the next processing stage.

8.5. Egress Filtering

The egress filtering stage reduces the set of potential transmission Ports 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]. Frames under reception are passed to the next processing stage once this reduction finished[2]. The egress filtering stage is only present in VLAN-aware CTF bridges.

[1] This fallback eliminates several cases for circulation of inconsistent frames in topological loops. A more conseverative approach could be to whitelist a FDB entry types and fall back to S&F in all other cases.

[2] It is not required to stall processing pending a frame's VID, because this already happened during ingress filtering (8.3).

8.6. Flow Classification and Metering

8.6.1. 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. This processing 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 .

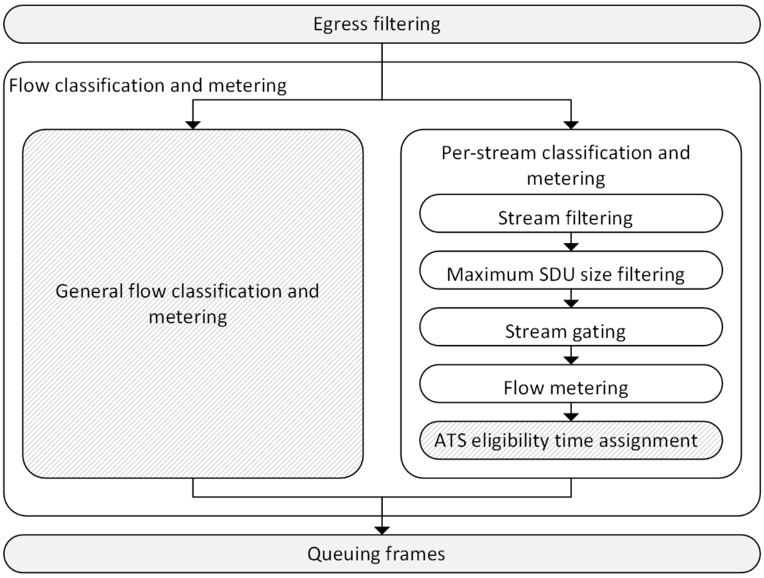


Figure 8.2.: Flow classification and metering.

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 8.6.2, 8.6.3, 8.6.4 and 8.6.5. With the exception of stream filtering, all subsequently described stages process frames under reception instantaneously (i.e., stall-free operation). When one of these stages passed a frame under reception to a subsequent processing stage, the associated frame counters of the stream filtering [2, items h) through m) in 8.6.5.3] are increased according to the rules specified in IEEE 802.1Q at the instant of time the frame is passed.

8.6.2. Stream Filtering

Frames under reception are associated with stream filters according to the same rules as specified in IEEE Std 802.1Q [2, 8.6.5.3]. If this association depends on a *stream_handle* parameter specified in IEEE Std 802.1CB [4], processing is stalled pending on this parameter prior to associating a stream filter. An associated stream filter then performs all necessary associations with subsequent internal stages passes these to the first associated internal stage instantaneously.

8.6.3. 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 additions in this section. 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 this frame is indicated for handling by subsequent processing stages in a CTF bridge.

8.6.4. 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 additions in this section. Once a 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.

8.6.5. 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 additions in this section. 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 immediately 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.

8.7. Individual Recovery

The individual recovery 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[4, item b) in 6.1] 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 executing this function[3].

8.8. Sequence Recovery

The sequence recovery 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 executing this function.

8.9. Sequence Encode

The sequence encode stage can insert externally visible tags into frames that represent the sequence_number parameter associated with these frames. The operations of the sequence encode stage and the tag formats for frames under reception are as specified in IEEE Std 802.1CB [4, 7.6 and 7.8].

8.10. Active Stream Identification

PLACEHOLDER, for describing differences and additions to 6.2 of IEEE Std 802.1CB. May be placed differently (in conjunction with incorporating stages for passive stream identification, sequence decoding and sequence generation [4, Figure 8-2]), subject to ongoing discussions in IEEE WG 802.1 at time of writing.

[3]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 8.8.

8.11. 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 (8.2, 8.4 and 8.5). The rules to determine the correct per-traffic queues for frames under reception are identical to the rules specified in IEEE Std 802.1Q [2, 8.6.6] with the following additions.

Before a frame under reception is queued, a per-queue copy of a frame before queuing. Each frame under reception resulting from this copy operation is then considered separately to allow for consistent transmission (8.13) as follows:

```
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 of the frame falls back to S&F before queuing it instantaneously.
ELSE IF
    (the associated CTFTransmissionEnable parameter [9.2.2] is TRUE) AND
    (the nominal transmit duration of the at the associated transmission Port
    would be less than the nominal duration of it's reception[4])
THEN
    The frame is discarded before queuing.
ELSE
    The frame is queued instantaneously.
```

8.12. 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[5]:

1. The frame was queued while it was under reception.
2. A processing stage before queuing(8.11) raised a late error for the frame.
3. the end of reception of the frame was reached before the frame was selected for transmission (8.13).

[4] This case avoids buffer under runs during transmission (e.g., due to untagging [2, clause 9] or different link speeds) in a conservative manner.

[5] Erroneous frames removed according to this additional rule will not become visible on the LAN of an associated transmission Port.

8.13. Transmission Selection

Transmission selection determines whether frames in per traffic class queues are available for transmission, determines transmission ordering and transmission times, initiates transmission of the frames, and removes transmitted frames from the per traffic class queues. 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

PLACEHOLDER, for describing the relationship between the LLI (6) and existing lower layers.

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] "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.

[6] 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

[7] 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

[8] Johannes Specht (Self; Analog Devices, Inc.; Mitsubishi Electric Corporation; Phoenix Contact GmbH & Co. KG; PROFIBUS Nutzerorganisation e.V.; Siemens AG; Texas Instruments, Inc.), *CTF - Considerations on Modelling, Compatibility and Locations.* [Online]. Available: https://mentor.ieee.org/802.1/dcn/22/1-22-0021-04-ICne-ctf-considerations-on-modelling-compatibility-and-locations.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