Technical Descriptions for Cut-Through Forwarding in Bridges

DCN 1-22-0042-06-ICne

Author: Johannes Specht

October 20, 2022

Contents

List of Figures

Part I.

Introduction

1. Purpose

This document is an individual contribution by the author, provided 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. 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. For example, it does not contain mandatory clauses for IEEE Standards [1].

2. Usage of normative keywords has no implied semantics beyond explicit description. For example, usage of the words *shall*, *should* or *may* **DOES NOT** imply requirements or recommendations for conformance of an implementation.

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 (i.e., which would matter for potential amendment projects).

3. Status of this Document

This document is work-in-progress in an early stage. It contains technical and editorial errors, omissions and simplifications. Readers discovering such issues are encouraged for making enhancement proposals, e.g. by sending such proposals to the author by email (johannes.specht.standards@gmail.com).

Part II.

Cut-Through Forwarding in Bridges

4. Overview

The architecture of a bridge with support for CTF is shown in Figure 4.1 , in symmetry with the architecture specified in IEEE Std 802.1Q [2, Figure 8-3].

This architecture comprises the following elements:

1. One or more higher layer entities above the MAC Service (MS) interface.
2. A bridge relay entity (7) that relays frames between different bridge Ports.
3. Generalized serial convergence operations (5) that translate between the Internal Sublayer Service (ISS) interface and Lower Layer Interface (LLI) per bridge Port.
4. Bridge Port transmit and receive operations (6) per Bridge port that transform and transfer service primitive invocations between the bridge relay entity, higher layer entities and the generalized serial convergence operations.

For differentiation between bridges with support for CTF and bridges without support for CTF, 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. CTF bridges may or may not support Virtual Local Area Networks (VLANs), similar to S&F bridges, and therefore terms *VLAN-aware* and *VLAN-unaware* are used to distinguish between bridges with and without support for VLANs.

The operation of S&F bridges is specified in the respective IEEE 802.1 Standards such as IEEE Std 802.1Q[2]. The operation of CTF bridges is described in this document in the chapters referred to before, typically limiting on describing the differences to the operations of S&F bridges.

Excluded from this document are technical details on higher layer entities above the MAC service interface. The Bridge port transmit and receive operations of CTF bridges limits on Cut-Through Forwarding to frames passing through the bridge relay entity. For frames passed to higher layer entities, the bridge port transmit and receive operations of a CTF bridge establish a S&F behavior visible via the MAC service interface, and the protocols of higher layer entities can operate according to the behavior specified in IEEE 802.1 Standards unaltered. In other words, the MAC service interface

On the path through the bridge relay of CTF bridges, this document limits the scope on the operations specified in 802.1Q[2], 802.1AC[3] and IEEE Std 802.1CB[4].

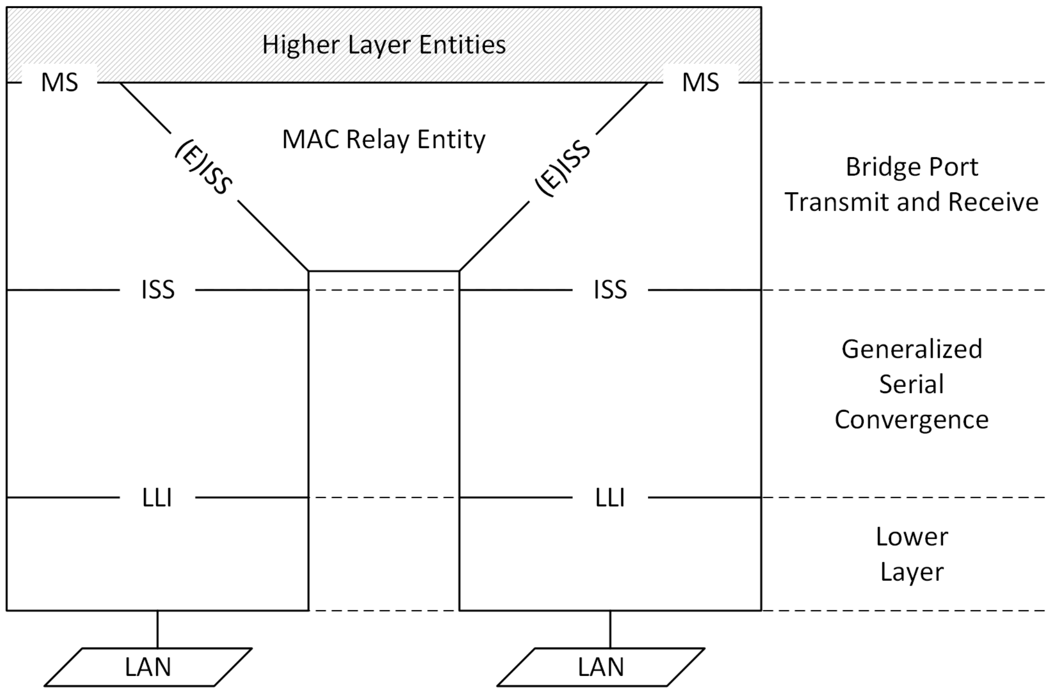

Figure 4.1.: Architecture of a Cut-Through Forwarding (CTF) Bridge.

5. Generalized Serial Convergence Operations

5.1. Overview

The generalized serial convergence operations are described by a stack of processes that interact via global variables (see 5.4) and service primitive invocations (see 5.2). 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 5.1 provides an overview of these processes and their interaction[1]. The processes can

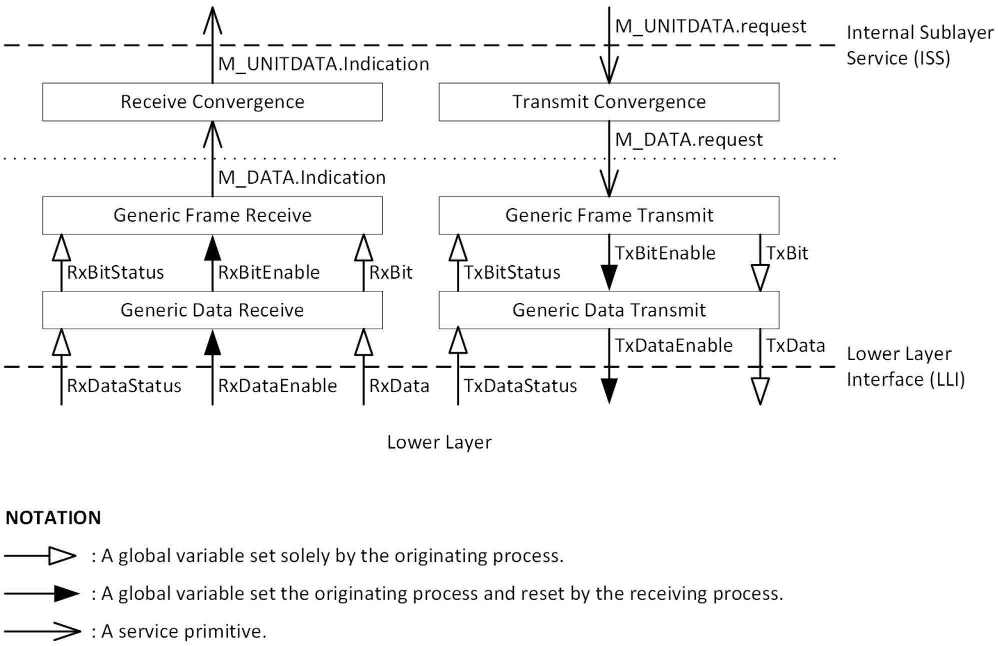

Figure 5.1.: Overview of the generalized serial convergence operations.

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 (5.7) that translates each invocation of the M_DATA.-indication service primitive (5.2.1) into a corresponding invocation of the M_UNIT-DATA.indication service primitive (5.2.2).

2. A Generic Frame Receive process (5.6) that generates M_DATA.indication invocations for bit sequences originating from the Generic Data Receive process of at least LEN_MIN (5.3.5) bits.

3. A Generic Data Receive process (5.5) 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 (5.10) 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 (5.9) that translates M_DATA.request invocations into bit sequences for the Generic Data Transmit process.

6. A Generic Data Transmit process (5.8) 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 [5, 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[6, 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 [5] introduced by the author are considered to be helpful.

5.2. Service Primitives

5.2.1. 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:

5.2.1.1. DA

An array of zero to LEN_ADDR (5.3.3) bits, containing the destination address of a frame.

5.2.1.2. SA

An array of zero to LEN_ADDR (5.3.3) bits, containing the source address of a frame.

5.2.1.3. 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 (5.3.2).

5.2.1.4. FCS

An array of zero to LEN_FCS (5.3.4) bits, containing the frame check sequence of a frame.

5.2.2. 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 [5]. Future versions may introduce more flexibility (e.g., for IEEE Std 802.11 [7, 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
)
```

5.2.3. Atomic Invocation Models

5.2.3.1. 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[5], *'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 5.1 or a sub-process of the forwarding process in 7), this model allows later processing stages to access contents in service primitive parameters that are incrementally added by an earlier processing stage.

[5]The concept of *attributes* is inspired by the *Very High Speed Integrated Circuits Hardware Description Language*, VHDL[8], which provides predefined attributes (e.g., *'transaction*) that allow modeling over multiple VHDL simulation cycles at the same instant of simulated time.

5.2.3.2. 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 (5.2.3.3) 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 unmabiguous descriptions.

The parameter-accurate operates at the resolution of symbolic and/or numeric parameters instead of bit arrays (5.2.3.1). 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 bit slice in the bit stream of a frame (i.e., a coherent sequence of bits in a frame's, in order),
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 (6.3) invocations can be derived from a subset of underlying bits of the associated SDU parameter of M_DATA.indication invocations (5.2.1) that are located in a VLAN Tag 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].

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.

5.2.3.3. Temporal Control

Parameter-accurate modeling allows formulating temporal control in processing stages. A processing stage (5.2.3.1) 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 (5.2.3.2). Most processing stalls are implicit 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 MAC bridge is **stalled** pending the **destination address** of a frame prior to the forwarding database lookup of the destination ports."*

5.3. Global Constants

5.3.1. PREAMBLE

A lower layer-dependent array of zero[6] or more bits, containing the expected preamble of each frame.

5.3.2. LEN_OCT

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

5.3.3. LEN_ADDR

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

$$\mathrm{LEN\_ADDR} = 48 \tag{5.1}$$

indicates an EUI-48 addresses.

5.3.4. 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,

$$\mathrm{LEN\_FCS} = 32 \tag{5.2}$$

indicates a four octet frame check sequence.

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

5.3.5. 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

$$\text{LEN\_MIN} \geq \text{PREAMBLE.length} + \text{LEN\_FCS} \tag{5.3}$$

are valid.

5.3.6. 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

$$\text{LEN\_MAX} \geq \text{PREAMBLE.length} + 2\text{LEN\_ADDR} + \text{LEN\_FCS} \tag{5.4}$$

are valid.

5.3.7. LEN_DATA

A lower layer-dependent integer, denoting the width of the RxData variable, in bits.

5.4. Global Variables

5.4.1. 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.

5.4.2. RxBit

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

5.4.3. 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.

Algorithm 5.1 Definition of data type `low_data_t`.

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

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.

5.4.4. 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.

5.4.5. 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 5.1. 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 (5.3.7) bits, containing a data word of a frame. An array length less than LEN_DATA bits is only valid if end is TRUE.

5.4.6. 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.

5.4.7. 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.

5.4.8. TxBit

A bit variable used to pass a single bit value to the Generic Data Transmit process.

5.4.9. 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.

5.4.10. 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.

5.4.11. TxData

A variable of composite datatype low_data_t (5.1), used for serially passing data words of frames from the Generic Data Transmit process to a lower layer.

5.4.12. 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.

5.5. Generic Data Receive

The Generic Data Receive process translates a lower layer-dependent[7] 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).

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

– Truncate excess bits to satisfy the frame length requirements implied by the parameter definition of the M_DATA.indication primitive (5.2.1).

5.6. Generic Frame Receive

5.6.1. 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.

5.6.2. State Machine Diagram

The operation of the Generic Frame Receive process is specified by the state machine diagram in Figure 5.2 , using the variables and functions defined in subsequent subclauses.

5.6.3. Variables

5.6.3.1. cnt

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

5.6.3.2. len

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

5.6.3.3. 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.

5.6.4. Functions

5.6.4.1. append(parameter,bit)

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

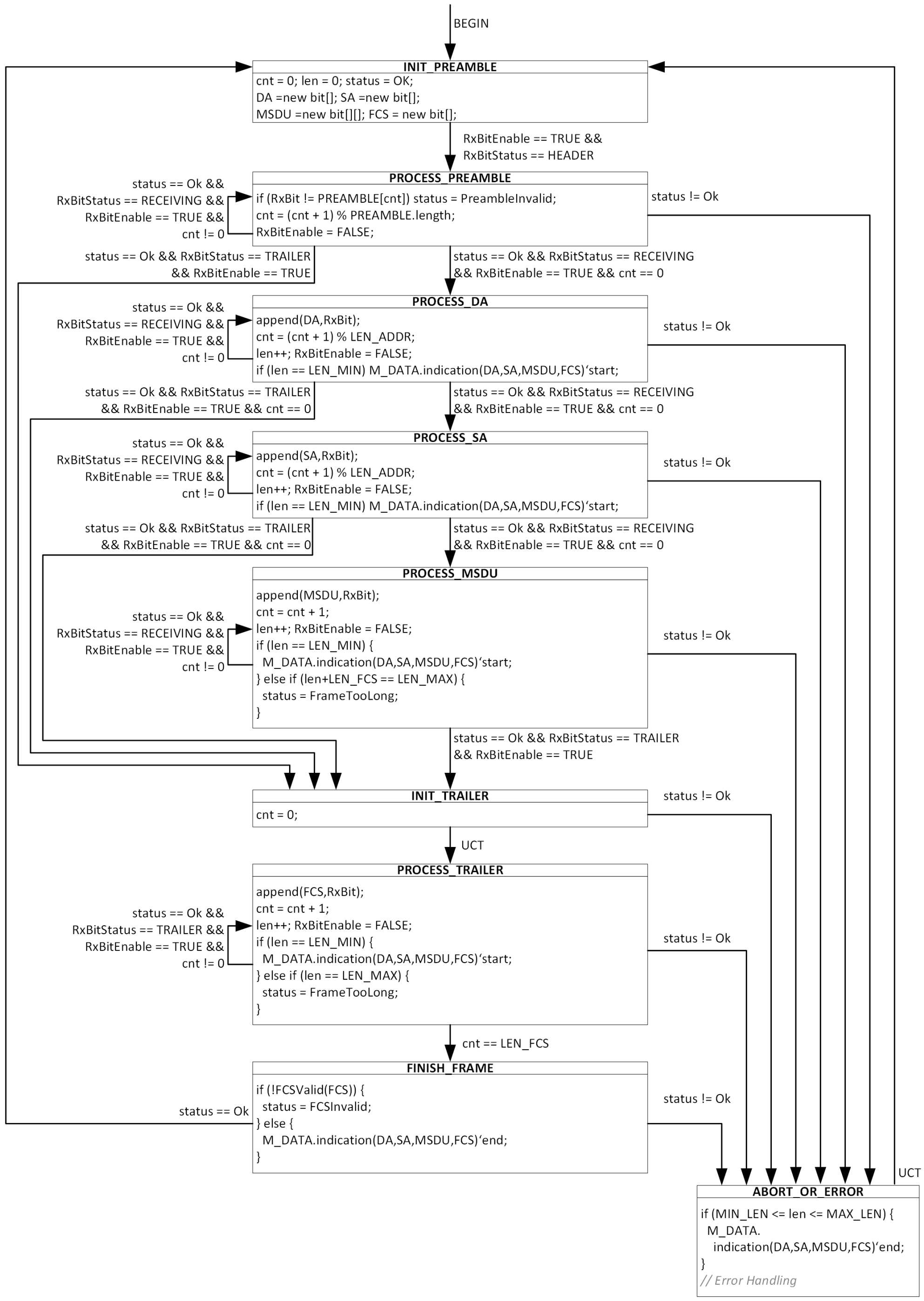

Figure 5.2.: State Machine Diagram of the Generic Frame Receive Process.

5.6.4.2. 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).

5.7. 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.

5.8. Generic Data Transmit

PLACEHOLDER, for descriptions symmetrical to 5.5.

5.9. Generic Frame Transmit

PLACEHOLDER, for descriptions symmetrical to 5.6.

5.10. Transmit Convergence

PLACEHOLDER, for descriptions symmetrical to 5.7.

6. Bridge Port Transmit and Receive Operations

6.1. Overview

The architeture 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 6 and comprises the following elements:

1. Connectivity (6.2) between the access points of the generalized serial convergence operations (5), higher layer entities, and the bridge relay entity (7).
2. Translations between ISS and EISS (6.3).

6.2. 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 destinied to the bridge relay entity and the CTFReceptionEnableParameter (8.2.4) of the reception Port is set TRUE, this copy is passed instantaneously to the translation from ISS to EISS (6.3). In all other cases, CTF bridges fall-back to S&F for frames under reception originating from teh generalized serial convergence operations prior to passing the respective copies to the associated acces point.

Frames originating from the bridge relay entity or higher layer entities destinied 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 subesequent contents are expected.

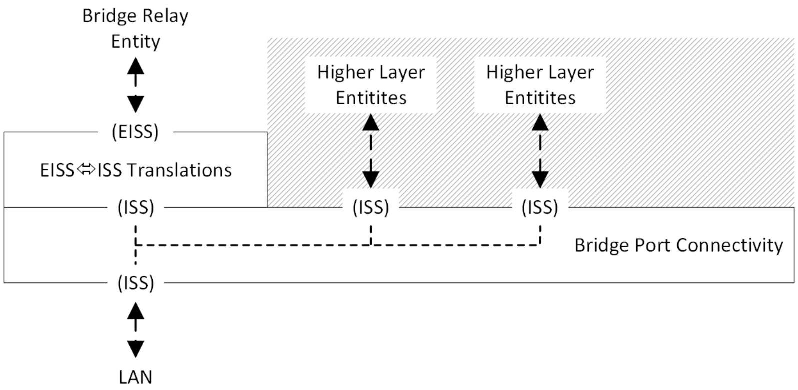

Figure 6.1.: Bridge Port Transmit and Receive.

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

6.3.1. 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].

6.3.2. Temporal relationship

6.3.2.1. Data indications

The temporal relationship (5.2.3.3) 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.

6.3.2.2. 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.

7. Bridge Relay Operations

7.1. Overview

The forwarding process of a CTF bridge is shown in Figure 7.1 , and comprises the processing stages of a CTF bridge [2, 8.6] with support for FRER [4, 8.1]:

1. Active topology enforcement (7.2)
2. Ingress filtering (7.3)
3. Frame filtering (7.4)
4. Egress filtering (7.5)
5. Flow classification and metering (7.6)
6. Individual recovery (7.7)
7. Sequence recovery (7.8)
8. Sequence encode (7.9)
9. Active stream identification (7.10)
10. Queuing frames (7.11)
11. Transmission selection (7.13)

7.2. Active Topology Enforcement

7.2.1. Overview

The active topology enforcement operations in a CTF bridge are as specified in IEEE Std 802.1Q [2, 8.6.1], with the differences described in the following for learning and forwarding separately.

7.2.2. Learning

Learning determines the source address and VID parameters for subsequent submission to the filtering database (FDB) according to the rules as specified in IEEE Std 802.1Q. The difference in CTF bridges are that

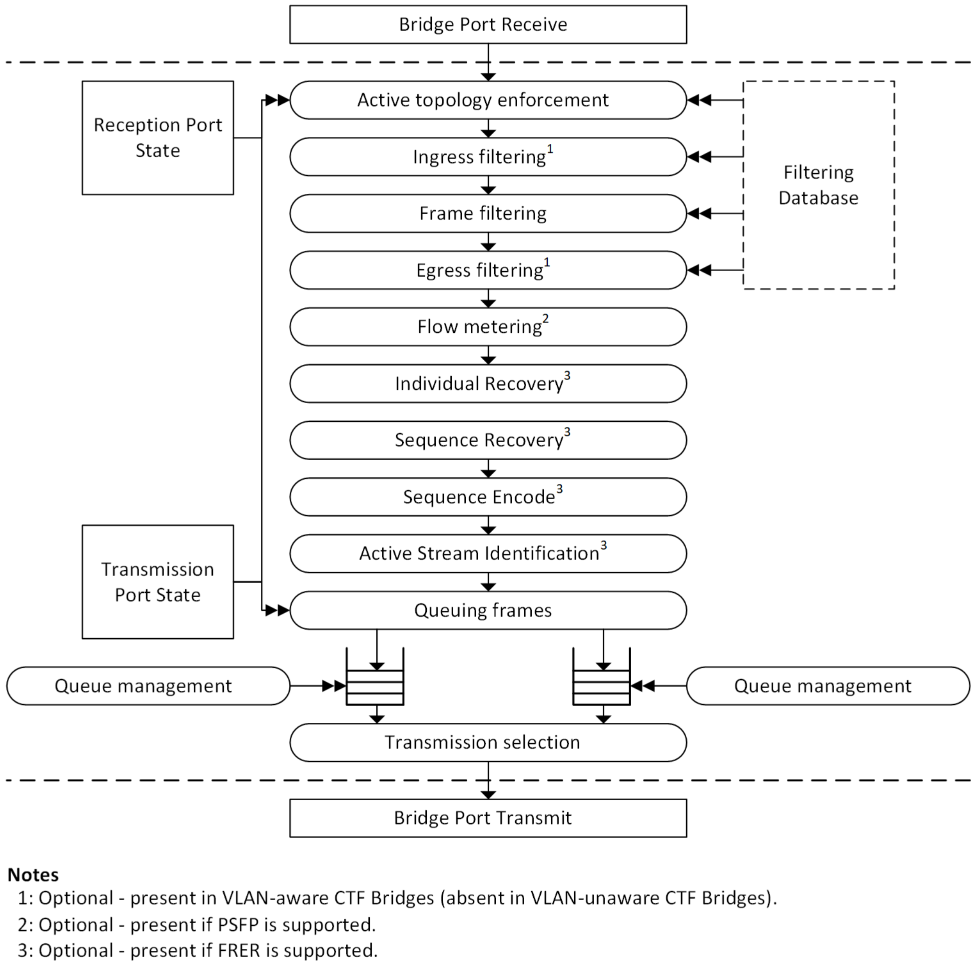

Figure 7.1.: Forwarding process of a CTF bridge.

– submission of source address and VID parameters does not take place prior to the entire associated associated frames, including FCS, are available, and

– only for frames with a consistent FCS.

7.2.3. Forwarding

Forwarding determines the set of potential transmission Ports according to the rules and controls specified in IEEE Std 802.1Q. If identification of this set depends on parameters of a frame such as VID or source address, processing is stalled pending all necessary parameters prior to passing the frame to the next processing stage, which is ingress filtering (7.3) for VLAN-aware CTF bridges and frame filtering (7.4) for VLAN-unaware CTF bridges. In absence of such dependencies, the frame is passed to the next processing stage instantaneously.

7.3. Ingress Filtering

Similar to IEEE Std 802.1Q [2, 8.6.2], ingress filtering operations are only available in VLAN-aware CTF bridges, and may discard frames received on Ports that are not in the member set [2, 8.8.10] associated with the VID parameter of these frames. The rules and controls for these operations are identical to those specified in IEEE Std 802.1Q. Frames are stalled by VLAN-aware CTF bridges pending the VID parameter and passed to the next processing stage (7.4) unless they are not in the member set and therefore not passed but instantaneously discarded instead.

7.4. Frame Filtering

Frame filtering operations in CTF bridges are as specified in IEEE Std 802.1Q [2, 8.6.1], with the differences and additions described in the following.

Similar to IEEE Std 802.1Q, frame filtering in CTF bridges can reduce the set of potential transmission Ports of each received frame on the basis of the rules and per-frame parameters specified in IEEE Std 802.1Q (destination address, VID, etc.). The exact set of parameters for each frames is determined identical to the rules specified in IEEE Std 802.1Q. If necessary, a CTF bridge stalls processing pending all necessary per-frame parameters before performing an FDB query for a frame [2, 8.8.9]. Dependent on the query's evaluation by the FDB, the frame is either passed to the next processing stage instantaneously, or processing of this frame falls back to S&F:

– Whenever the query evaluation 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, the frame is passed to the next processing stage instantaneously.

[1]This fallback eliminates several cases for circulation of inconstistent frames in topological loops.

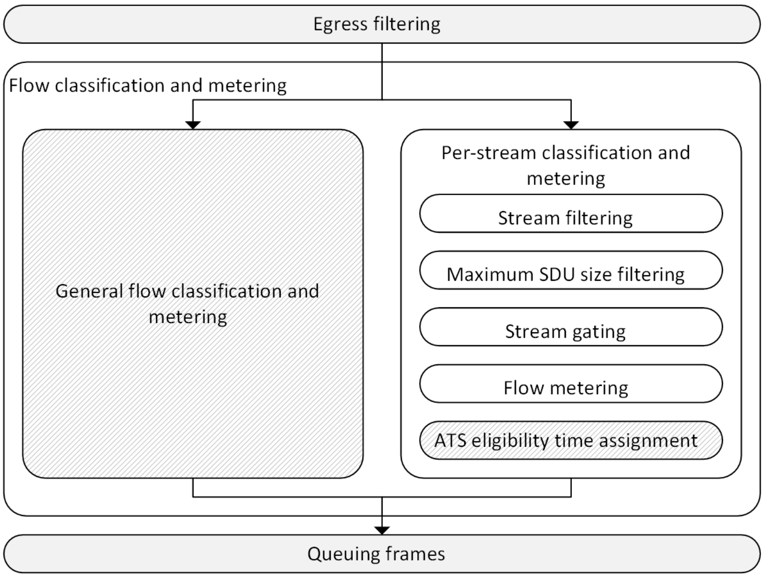

Figure 7.2.: Flow classification and metering.

7.5. Egress Filtering

Similar to IEEE Std 802.1Q [2, 8.6.2], egress filtering operations are only available in VLAN-aware CTF bridges, and reduces the set of potential transmission Ports of a frame by removing each Port that is not in the member set [2, 8.8.10] based on the frame's VID parameter. Frames are passed to the next processing stage instantaneously after completion of these operations[2].

7.6. Flow Classification and Metering

7.6.1. General

Flow classification and metering in CTF bridges comprises the same elements and processing stages as specified in IEEE Std 802.1Q [2, 8.6.5]. These elements and relationships are show in Figure 7.2 .

CTF bridges fall back to S&F for frames under reception that is subject to processing by elements and stages of flow classification and metering prior to being processed by such elements and stages, unless the elements and stages that process a frame under reception limits to the following ones:

1. Stream filtering

[2] It is not required to stall processing pending the a frame's VID, because this already happened during ingress filtering (7.3).

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

The operation of these stages for frames under reception is described in 7.6.2, 7.6.3, 7.6.4 and 7.6.5. With the exception of stream filtering, all aforesaid 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.

7.6.2. 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 streams gates, maximum SDU size filtering and flow metering, and passes frames to the associated stages instantaneously.

7.6.3. 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 immediately 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.

7.6.4. 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.

7.6.5. 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. The frame is discard otherwise prior to being passed to the next processing stage. If a frame under reception already passed flow metering and the limit of the flow meter would be exceeded prior to the frame's end of reception, a late error for this frame is indicated immediately for handling by subsequent processing stages in a CTF bridge.

7.7. Individual Recovery

PLACEHOLDER, for describing differences and additions to 7.5 of IEEE Std 802.1CB.

7.8. Sequence Recovery

PLACEHOLDER, for describing differences and additions to 7.4.2 of IEEE Std 802.1CB.

7.9. Sequence Encode

PLACEHOLDER, for describing differences and additions to 7.6 of IEEE Std 802.1CB.

7.10. Active Stream Identification

PLACEHOLDER, for describing differences and additions to 6.2 of IEEE Std 802.1CB.

7.11. Queuing Frames

The rules to determine transmission Ports, traffic classes of these Ports and the ordering requirements for frames under reception are identical to the ones specified in IEEE Std 802.1Q [2, 8.6.6] with the additions in this section.

Frames under reception and frames that were fully received are queued in the of arrival at the queuing frames processing stage. For each frames under reception, a per Port copy of these frames for each transmission port in the remaining set of potential transmission Ports is created before each individual copy is treated individually per transmission Port. For each copy, a CTF bridge determines whether or not it is queued by comparing the nominal transmission data rate of the transmission Port with the nominal transmission rate of all potential reception Ports from which the frame may originate and by evaluating the CTFTransmissionEnable management parameters (8.2.2) as follows:

If the CTFTransmissionEnable parameter of the traffic class for a frame under reception is FALSE, processing of this frames falls back to S&F before the frame is queued.

If the CTFTransmissionEnable parameter of the traffic class for a frame under reception is TRUE, the frame is queued if the nominal data rate of the transmission Port is less than or equal to the nominal data rate of all potential reception Ports.

If the CTFTransmissionEnable parameter of the traffic class for a frame under reception is TRUE, the frame is discarded if the nominal data rate of the transmission Port greater than the nominal data rate of at least one potential reception Ports of this frame.

7.12. Queue Management

The rules for removal of frames from queues are identical to the ones specified in IEEE Std 802.1Q [2, 8.6.7].

7.13. Transmission Selection

Queued frames are become available for transmission according the rules and operations specified in IEEE Std 802.1Q [2, 8.6.8], with the following additions. Frames under reception can only be selected for transmission by traffic classes that use the strict priority transmission selection algorithm [2, 8.6.8.1]. For traffic classes that other transmission selection, the frames become available for transmission selection no earlier than the end of reception is reached[3].

[3] This decision logic may be moved to 7.11 in the future.

8. Management Parameters

8.1. Overview

The management parameters for CTF fall into three categories:

1. Control Parameters (8.2)
2. Timing Parameters (8.3)
3. Error Counters (8.4)

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.

8.2. Control Parameters

8.2.1. 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.

8.2.2. 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.

8.2.3. 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.

8.2.4. 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.

8.3. Timing Parameters

8.3.1. 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.

8.4. Error Counters

8.4.1. 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].

8.4.2. 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 [9, l.121 - l.126] or special frame check sequences [10, p.54, 2.2.][11, 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 heterogenoues 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 [10, p.46 – p.49].

Part IV.

Appendices

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

PLACEHOLDER, for describing the relationship Generalized Serial Convergence (5) lower layer interface 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] 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

[6] 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

[7] "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.

[8] "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.

[9] 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

[10] 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

[11] 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