DCB Capability Exchange Protocol Specification

Rev 1.0

Intel Corporation
Cisco Systems
Nuova Systems

Legal Notice

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Terminology

Term	Description
BCN	Backward Congestion Management
CM	Congestion Management
DCB	Data Center Bridging
PROTOCOL	DCB Capability Exchange Protocol
LLDP	Link Layer Discovery Protocol, IEEE802.1AB
LLDPDU	An LLDP PDU
NIC	Network interface controller
OS	Operating System.
OUI	Organizationally Unique Identifier
PDU	Protocol Data Unit
PFC	Priority Flow Control (same as Per Priority Pause or Class Based Flow Control)
PG	Priority Groups
RX	Receive
SNMP	Simple Network Management Protocol
TLV	Type Length Value
TTL	Time to Live
TX	Transmit

Related Documents

DCB Feature Specifications	Company
Definition for new PAUSE function – v1.2 http://www.ieee802.org/1/files/public/dos2007/new-cm-barrass-pause-proposal.pdf	Cisco Systems
Packet Scheduling with Priority Grouping and Bandwidth Allocation for DCB Networks http://www.ieee802.org/1/files/public/docs2007/new-wadekar-priority-groups-1107-v1.pdf	Intel Corporation
Backward Congestion Notification Functional Specification	Cisco Systems
FCoE Specifications Http://www.t11.org/ftp/t11/pub/fc/bb-4/07/303v0.pdf	Cisco Systems / Nuova Systems
Network Interface Virtualization Functional Specification	Nuova Systems

DCB Capability Exchange Protocol ("Protocol") Related Contacts

Company	Contacts
Intel Corporation	Eric Multanen (eric.w.multanen@intel.com) Manoj Wadekar (manoj.k.wadekar@intel.com)
Cisco Systems	Dinesh Dutt (ddutt@cisco.com Sanjay Sane (sanjays@cisco.com)
Nuova Systems	J. R. Rivers (jrrivers@nuovasystems.com) Krishna Doddapaneni (krishna@nuovasystems.com)

1.

This document details the discovery and capability exchange protocol that is used by DCB devices to exchange configuration (referred to in this document as the "Protocol").. The document also lists the parameters of the various features in DCB. The DCB features covered in this document are:

- Priority Groups (PG)
- Priority Flow Control (PFC)
- Backward Congestion Notification (BCN)
- Applications
- Logical Link Down
- Network Interface Virtualization

This document does not provide background, motivation, and requirements for Data Center Bridging. The document also does not describe the various features. Please refer to the respective feature specifications for discussion of each feature.

1.1 Purpose

The following lists the purpose of PROTOCOL.

Allow discovery of DCB related peer capability: PROTOCOL is used to know about the capabilities of the peer device. It is a means to know if the peer device supports a particular feature like BCN or PFC. It can then take appropriate action based on this feature. For example, it can be used to determine if two link peer devices support Priority Flow Control.

Allow DCB configuration mismatch detection: PROTOCOL can be used to aid configuration synchronization of the link peers. It can be used to detect if both link peers have been configured consistently. For example, it can determine if the bandwidth groups and bandwidth allocations on both ends of the link are the same. This information can be used to notify the device manager in case of a conflict.

Allow configuration of DCB link peer: PROTOCOL can be used by a device to perform configuration of DCB parameters in its link peer. The goal is to provide basic peer to peer configuration through PROTOCOL in the initial version. Future versions of PROTOCOL or another higher layer application can build on top of this to provide more complex configuration distribution mechanisms.

Figure 1 shows a deployment scenario for a network that is using PROTOCOL. PROTOCOL capable links exchange DCB data and conflict alarms are sent to the appropriate management stations. As an example, a boundary is shown indicating which devices support BCN and which do not.

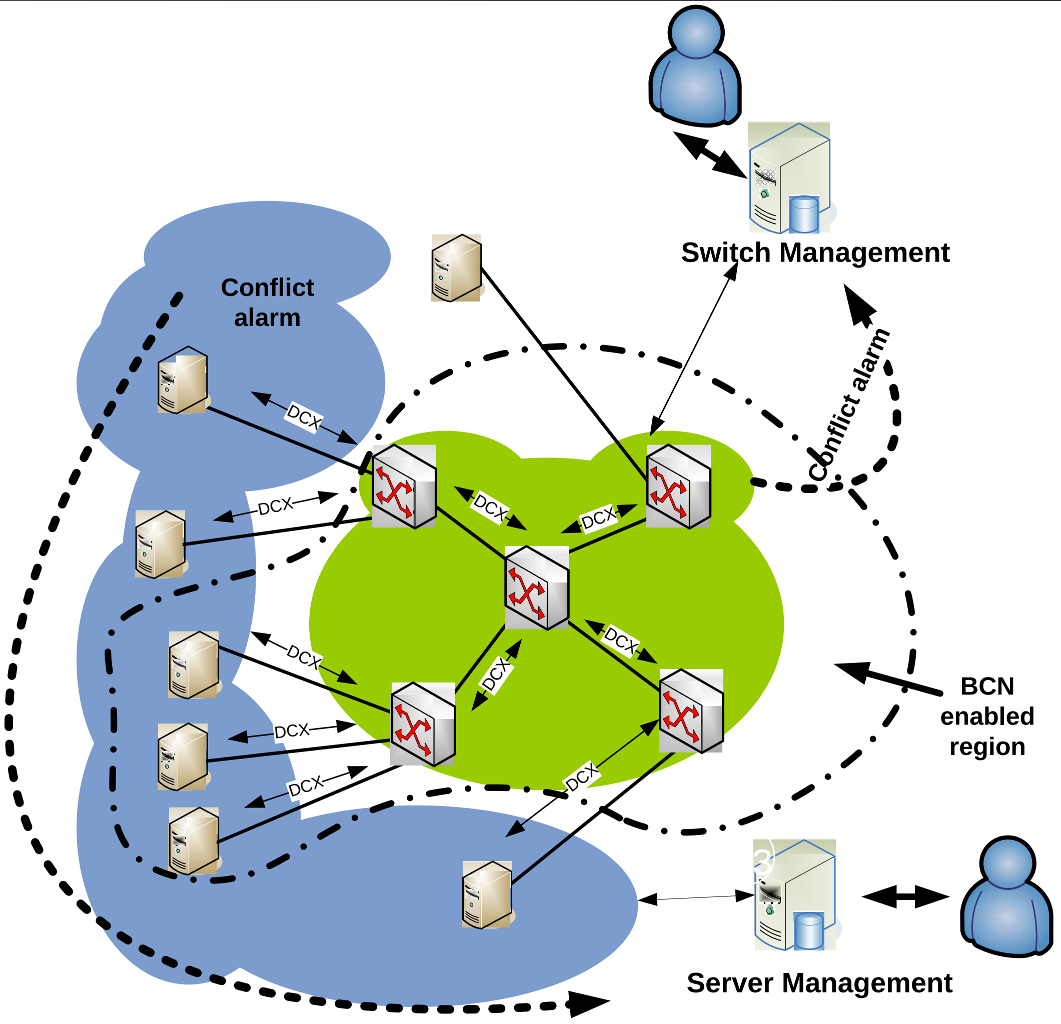

Figure 1 - PROTOCOL Deployment Scenario

1.2 Types of DCB Parameters

Each DCB feature has a set of parameters. DCB parameters are classified into two broad categories:

- **Exchanged parameters**: Exchanged parameters are sent to the peer. Within these parameters, there are two sub-groups:
  1. Administered parameters: These are the configured parameters.
  2. Operational parameters: This is the operational state of the related administered parameter. Operational state might be different than the administrative/configured state, primarily as a result of the PROTOCOL exchange with the peer. Operational parameters accompany only those administered parameters where there is a possibility that the operational state is different from what was set by their administrator. The operational parameters are included in the the LLDP message only for informational purposes. It might be used by a device to know what is the current operational state of the peer.
- **Local parameters**: Local parameters are not exchanged in LLDP messages.

Figure 2 shows the exchanged parameters that are sent to each peer via LLDP messages.

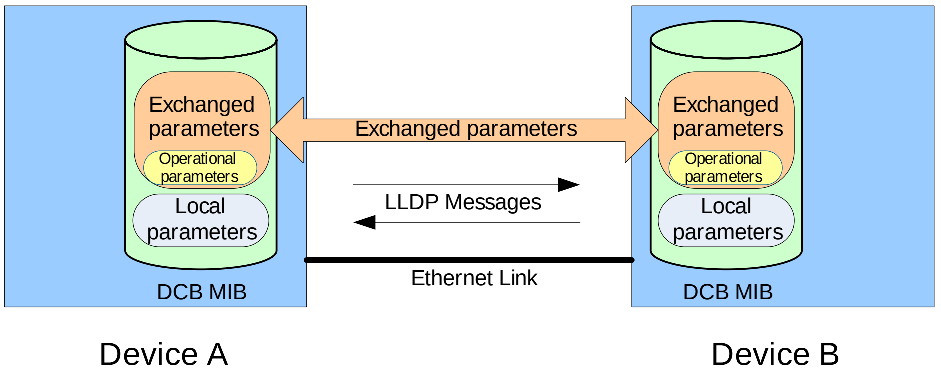

Figure 2 - Types of Parameters

1.3 PROTOCOL and LLDP

PROTOCOL uses Link Layer Discovery Protocol (LLDP) to exchange parameters between two link peers. LLDP is a unidirectional protocol. It advertises connectivity and management information about the local station to adjacent stations on the same IEEE 802 LAN.

LLDP PDUs carry Type Length Values (TLVs) classified as

1. Mandatory TLVs: Chassis ID, Port ID, TTL, End of LLDPDU.
2. Optional TLVs: Basic Management, 802.1 and 802.3 Organizationally Specific.

DCB exchanged parameters are packaged into Organizationally Specific TLVs. The OUI used for the PROTOCOL TLV is 0x00, 0x1B, 0x21 (an Intel OUI). Depending on the amount of data required for all features, one or more TLVs, with different sub-types, are defined for PROTOCOL. Within the PROTOCOL TLVs, sub-TLVs are defined for each feature carried by that TLV.

A device capable of any DCB feature must have PROTOCOL enabled by default with an option for PROTOCOL to be administratively disabled.

PROTOCOL is expected to operate over a point to point link. If multiple LLDP neighbors are detected, then PROTOCOL behaves as if the peer's PROTOCOL TLVs are not present until the multiple LLDP neighbor condition is no longer present. An LLDP neighbor is identified by its logical MAC Service Access Identifier (MSAP). The logical MSAP is a concatenation of the chassis ID and port ID values transmitted in the LLDPDU.

LLDP gives administrator control to enable/disable the protocol independently on the Rx side and Tx side. Since PROTOCOL is an acknowledged protocol which uses LLDP, for the protocol to operate correctly both LLDP Rx and Tx must be enabled on the interface on which PROTOCOL runs. The behavior of PROTOCOL is as follows with respect to LLDP Rx/Tx admin state controls:

* If either of Rx or Tx is in disable state, PROTOCOL is disabled on the interface. Neither the control nor feature state machines should run. The LLDP PDU's that are generated from this interface do not have any PROTOCOL TLVs. If the peer sends PROTOCOL TLVs they should be ignored as far as the PROTOCOL state machines are concerned.

* When PROTOCOL is currently running and LLDP TX is disabled, then according to the LLDP specification, a shutdown LLDPDU is sent. When the peer receives this PDU, PROTOCOL is determined to be disabled on the peer. This is equivalent to PROTOCOL TLV TTL expired in the Control State machine and Rx.Feature.present() = FALSE in the Feature state machine. If for some reason this frame is lost, then PROTOCOL depends on standard rxInfoTTL expiry of the peer's LLDP TLV's.

* When PROTOCOL is currently running and LLDP Rx is disabled, then all PROTOCOL TLV's including the control TLV should be withdrawn from the LLDP PDUs that the interface generates. The peer's behavior should be the same as discussed in the previous case.

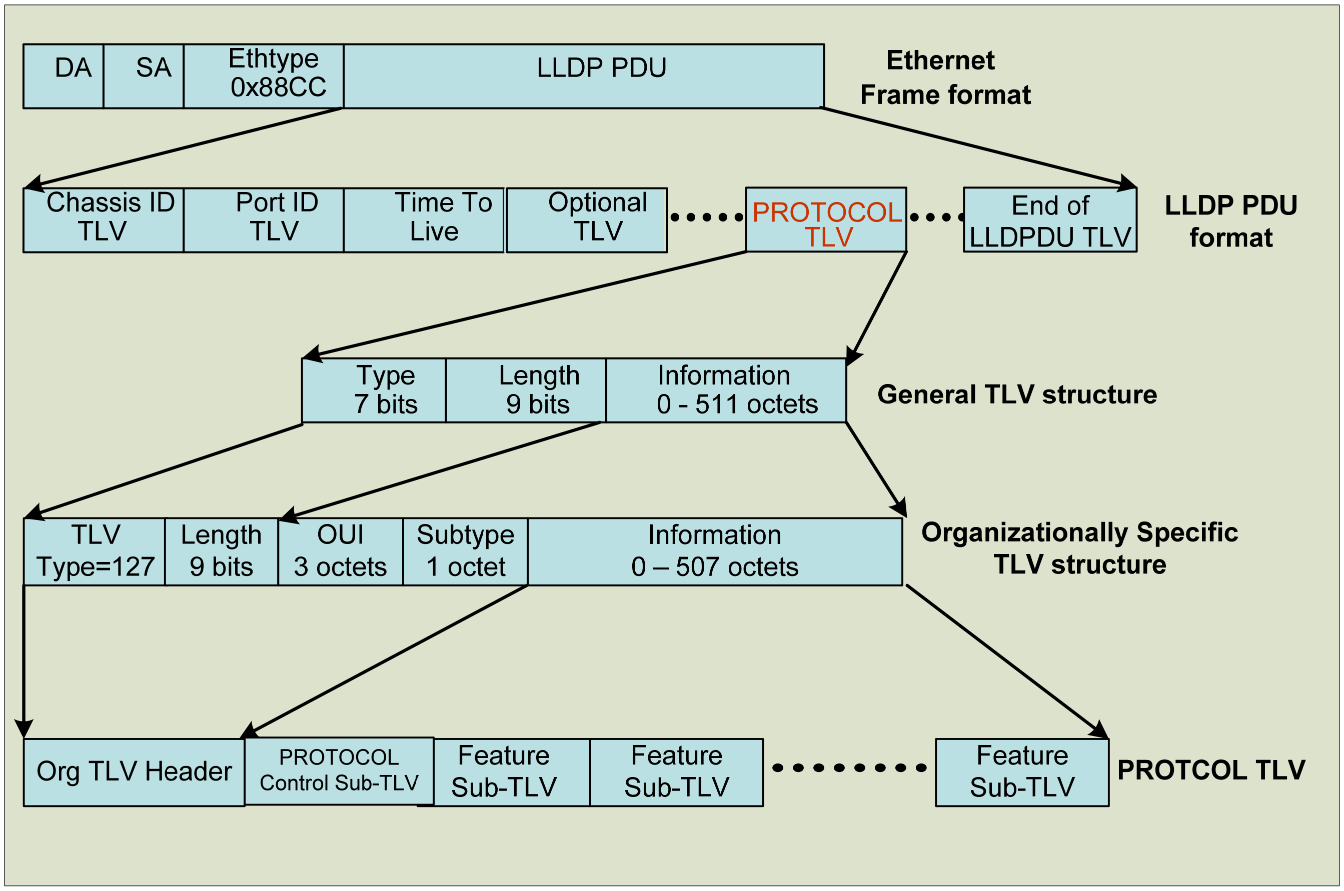

Figure 3 - LLDP Frame Format

1.3.1 LLDP Modifications

This section lists the proposed modifications to the LLDP protocol for use with PROTOCOL. IEEE 802.1AB REV project is currently working on these modifications. Once standard is published for IEEE 802.1AB-REV, PROTOCOL specification will be appropriately modified.

1.3.1.1 Fast initial LLDP Transmissions

The current LLDP protocol can result in a long delay before DCB parameters are exchanged and synchronized.

LLDP transmits a frame after:

- Transmit countdown timer expiration (recommended default value = 30 seconds) and txDelay expiration, OR
- A condition (status or value) change in one or more objects in LLDP local system MIB. (LLDP local system MIB contains only the objects that are sent in LLDP frames. This MIB could be merely a subset of a larger MIB).

After initialization, an LLDP frame is transmitted (considering the initialization as a status change).

An initial LLDP message might not be received by the peer due to different times at which their initialization completes (see Figure 4).

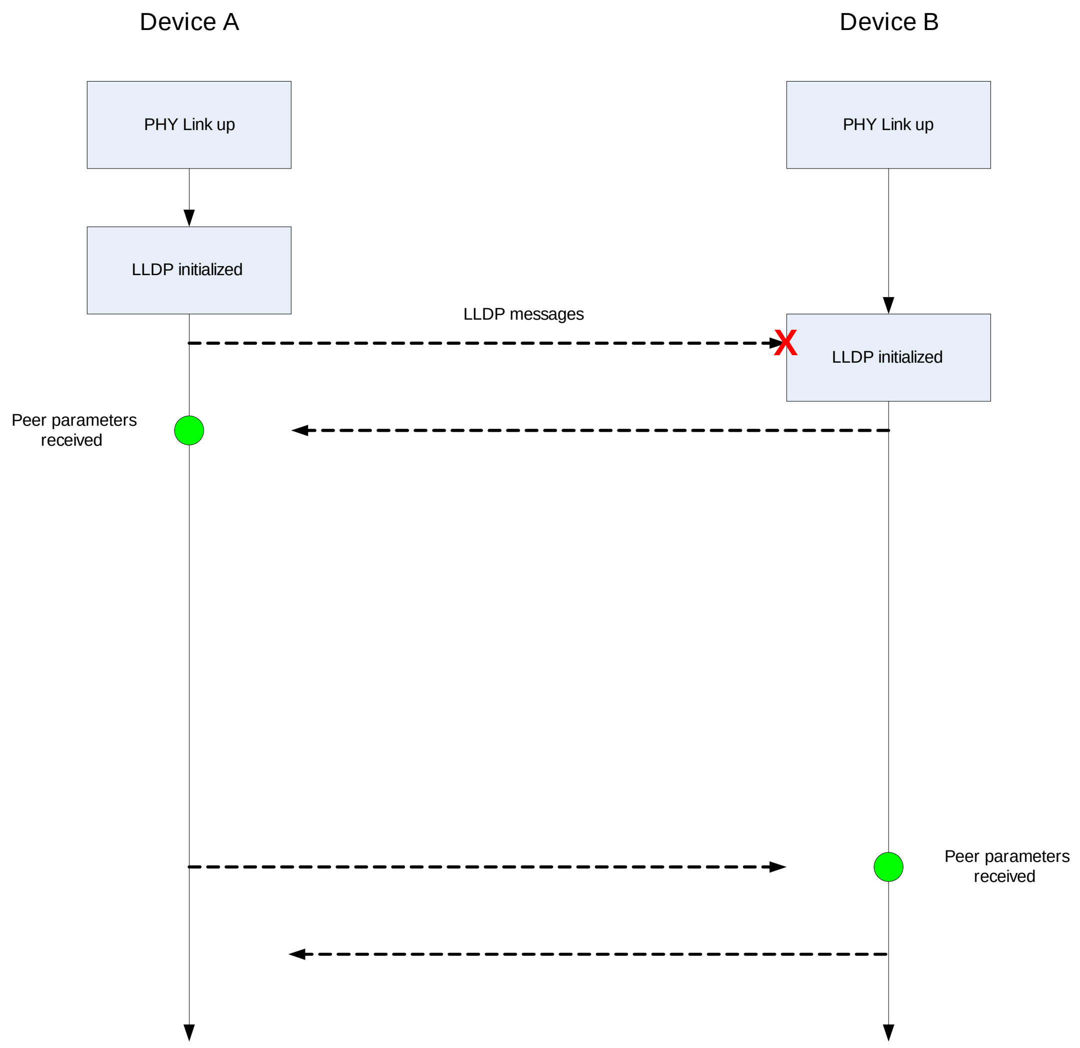

Figure 4 - Initial LLDP Exchange Delay Issue

In order to overcome this problem, the following modification is proposed. The interval for the LLDP transmission time to refresh the timer (msgTxInterval) is set to one second for the first five transmissions after LLDP initialization and then reset to the administratively configured value. The txDelay (minimum delay between successive transmitted LLDP frames) is also set to one second as a PROTOCOL default. This ensures that at start up both devices receive peer parameters within a short timeframe as shown in Figure 5.

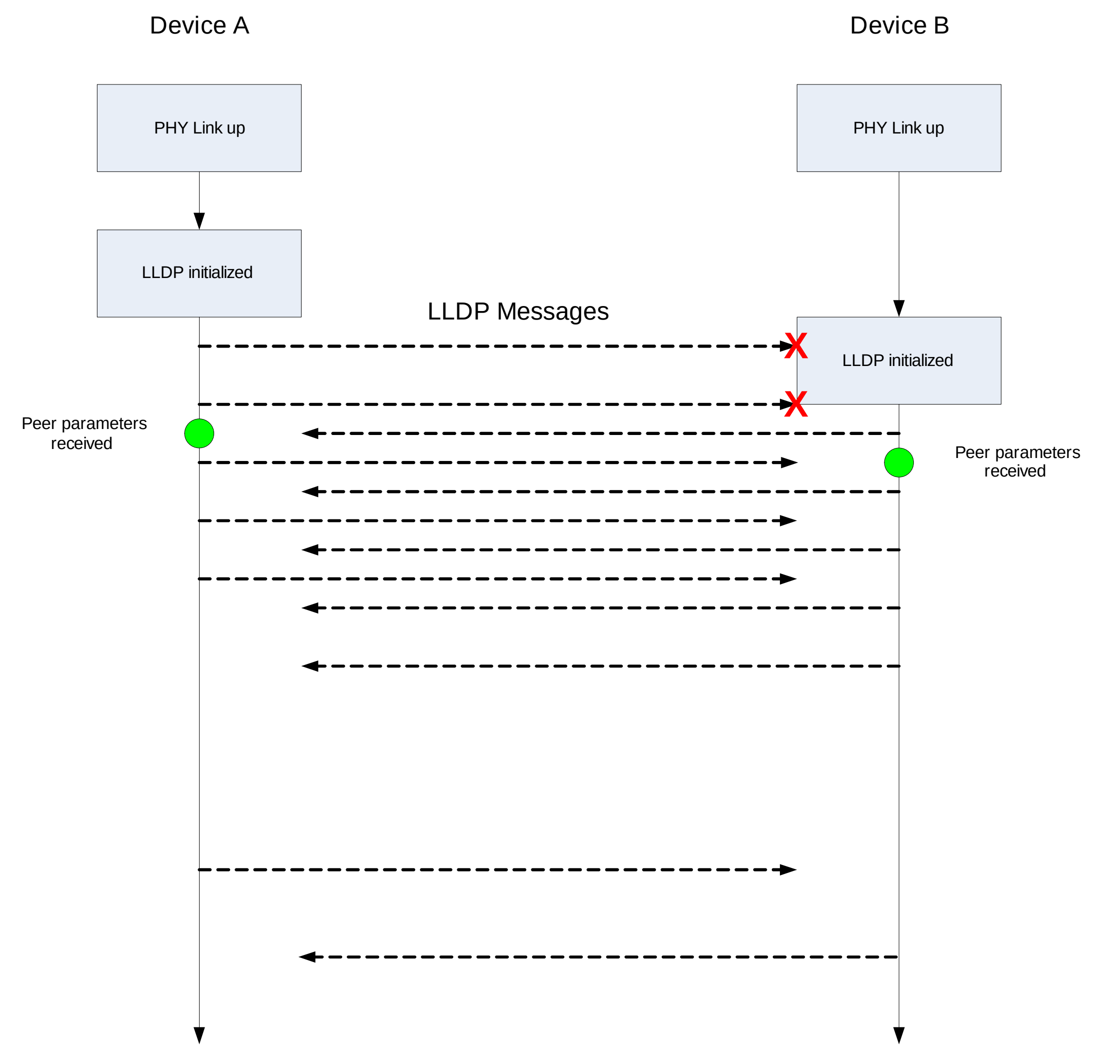

Figure 5 - Initial Fast Retransmission of LLDP Frames

Each time LLDP is initialized, such as link up, LLDP enters this fast transmission mode. LLDP operates in its normal transmission mode at all other times.

1.4 PROTOCOL Operation

PROTOCOL is defined as a PROTOCOL control state machine and a set of DCB feature state machines. The PROTOCOL control state machine handles ensuring that the two PROTOCOL peers get in sync by exchanging LLDP PDUs after link up or following a configuration change. The DCB feature state machines handle the local operational configuration for each feature by comparing and synchronizing with the peer's feature settings.

1.4.1 PROTOCOL TLV Format

Information about the PROTOCOL control state and DCB feature configuration are exchanged with the peer in PROTOCOL TLVs that are transmitted via LLDP PDUs. Figure 6 shows the general structure of the organizationally specific PROTOCOL TLV. The details of each sub-TLV are covered in the remainder of the document.

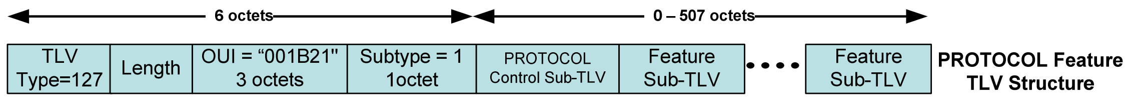

Figure 6 - High Level PROTOCOL TLV Structures

The PROTOCOL Control Sub-TLV and the set of Feature Sub-TLVs can be arranged in any order within the PROTOCOL TLV. Duplicate Sub-TLV's (such as more than one Sub-TLV for the same feature) are not allowed. Duplicates are handled as a configuration error for the feature. A duplicate PROTOCOL Control TLV causes an error for all features.
The PROTOCOL sub-TLVs follow the same format as an LLDP TLV – having type, length and information fields. The type field is meaningful within the context of a PROTOCOL TLV and the length specifies the number of octets in the information portion of the sub-TLV.

1.4.1.1 Bit and Octet Ordering Conventions

PROTOCOL uses the same bit and octet ordering conventions as LLDP.

```
[The PROTOCOL TLV] contain[s] an integral number of octets. The
octets in [a PROTOCOL TLV] are numbered starting from one and
increasing in the order they are put into the LLDP frame. The
bits are numbered from zero to seven, where zero is the low-order
bit.

When consecutive bits within an octet are used to represent a
binary number, the highest bit number has the most significant
value. When consecutive octets are used to represent a binary
number, the lower octet number has the most significant value.
All TLVs respect these bit and octet ordering conventions, thus
allowing communications to take place.
```

In the details that follow, the following data types are used to define structures which describe the elements of PROTOCOL sub-TLVs:

- u32 - unsigned 32 bit integer
- u16 - unsigned 16 bit integer
- u8 - unsigned 8 bit integer

Elements listed first in a structure have lower octet numbers then subsequent elements. Bit fields within an element occupy the highest to lowest order bits of the element in the order they are listed. The following structure shows an example.

```
struct example_tlv {
    u16 type         :7;     // high order bit field in u16 element
    u16 length       :9;     // low order bit field in u16 element
    u32 fieldA       :8;     // highest order bit field in u32 element
    u32 fieldB       :8;
    u32 fieldC       :3;
    u32 fieldD       :13;    // lowest order bit field in u32 element
};

SIZE = 6 octets
```

1.4.2 PROTOCOL Control State Machine

The PROTOCOL Control state machine uses the PROTOCOL Control sub-TLV to exchange information with the peer. In addition, it maintains some additional local state variables to manage the state machine operation. The TLV and state variables are defined in the sections that follow.

1.4.2.1 PROTOCOL Control TLV

Figure 7 shows the PROTOCOL Control TLV.

```
struct protocol_control_tlv {
    u16 type         :7;
    u16 length       :9;
    u16 oper_version :8;
    u16 max_version  :8;
    u32 seqno;       :32;
    u32 ackno;       :32;
};

SIZE = 12 octets
```

Figure 7 - PROTOCOL Control TLV Definition

The following table lists the fields in the PROTOCOL Control TLV.

Field	Type	Range	Default Value	Access (RO,RW, NA)	Description
Type	Integer	N/A	1	RO	Type code of the PROTOCOL Control sub-TLV.
Length	Integer	N/A	10	RO	Length of the PROTOCOL Control sub-TLV payload (not including the Type and Length fields). The length is less than the maximum possible value (511) as this TLV is packaged inside the PROTOCOL TLV along with other feature TLV's.
Oper Version	Integer	0..255	Highest supported version	RO	Operating version of the PROTOCOL protocol. The system adjusts as needed to operate at the highest version supported by both link partners.
Max Version	Integer	0..255	Highest supported version	RO	Highest PROTOCOL protocol version supported by the system. Version numbers start at zero. The PROTOCOL protocol must be backward compatible with all previous versions.
SeqNo	Integer	0 .. (2^{32} –1)	0	RO	A value that changes each time an exchanged parameter in one or more of the DCB feature TLV's changes.
AckNo	Integer	0 .. (2^{32} –1)	0	RO	The SeqNo value from the most recent peer PROTOCOL TLV that has been handled. This acknowledges to the peer that a specific SeqNo has been received.

Table 1 - PROTOCOL Control TLV Fields

1.4.2.2 PROTOCOL Control State Variables

The following table lists the local state variables used to maintain the PROTOCOL Control state machine.

State Variable	Type	Range	Default Value	Access (RO,RW, NA)	Description
MyAckNo	Integer	0 .. (2^{32} –1)	0	RO	The 'AckNo' from the most recent peer PROTOCOL TLV that has been handled. This is an acknowledgement from the peer that a specific SeqNo has been received.

Table 2 - PROTOCOL Control state variables

1.4.2.3 PROTOCOL Control State Machine

In addition to the TLV fields and state variables previously described, the PROTOCOL Control state machine uses the following mechanisms:

- **somethingLocalChanged** – this is an indication from the PROTOCOL state machine to the LLDP module that there is a new PROTOCOL TLV to transmit.
- **somethingRemoteChanged** – this is an indication from the LLDP module to the PROTOCOL Control state machine that there is new information from the peer (such as new PROTOCOL TLV or data has expired).
- **remoteFeatureChanged** – the PROTOCOL Control state machine provides this indication to the DCB Feature state machines after it has received the remoteFeatureChanged indication.
- The **SeqNo** and **MyAckNo** variables are visible to the DCB Feature state machines.

Figure 8 shows the operation of the PROTOCOL Control state machine. Note that the diagram is defined using an infinite loop model (such as no waiting). Implementations might use event waiting mechanisms as long as the function of the state machine is preserved.

A few notes concerning the notations used in Figure 8:

- PROTOCOL Control TLV fields and state variables are used directly such as SeqNo)
- Variables from the Feature state machines are identified by pre-pending Feature to the variable: For example, Feature.Syncd refers to a variable called Syncd from a Feature state machine.
- TLV fields received from the peer are identified as: Rx.<machine>.<variable> - i.e. 'Rx.Protocol.SeqNo.

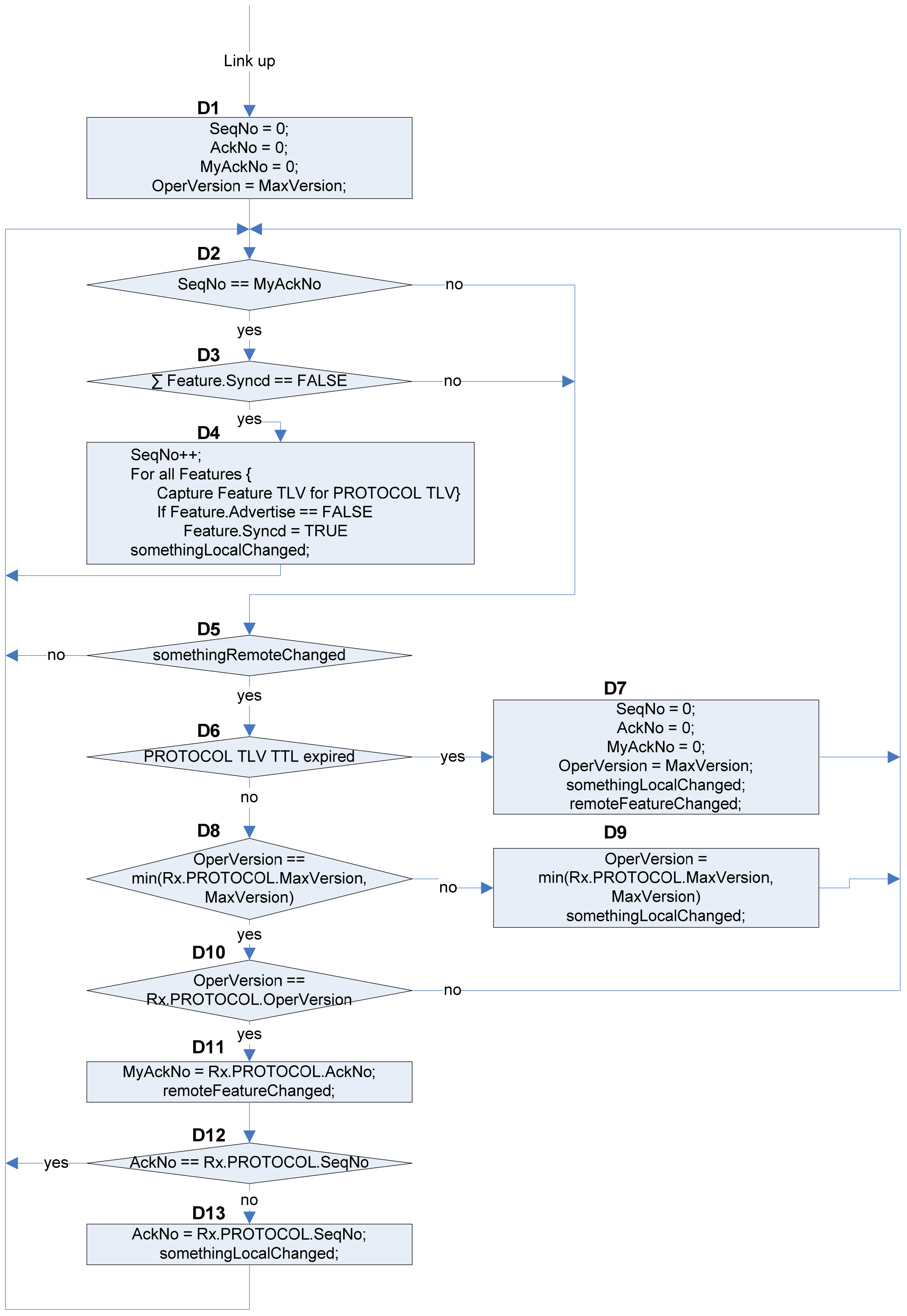

Figure 8 - PROTOCOL Control State Machine Diagram

Commentary on the PROTOCOL Control state machine diagram by reference label (D*x*):

- D6 and D7: If the PROTOCOL TLV from the peer has expired, then the local side resets similar to a link up. This is a different case than an actual link down, which would cause this state machine to exit.
- D13: The peer has sent a Control TLV with a new sequence number. Send a new Control TLV with an updated AckNo field.

1.4.3 DCB Feature State Machine

This section defines the operation of the DCB Feature state machine. The configuration of each DCB feature is managed by an independent DCB Feature state machine; however, the operation of the DCB Feature state machine is common for all features.

1.4.3.1 DCB Feature TLV

Figure 9 shows the common TLV header structure used for all DCB feature sub-TLV's. Feature specific TLV parameters are pre-pended with this header.

```
struct PROTOCOL_tlv_header {
    u16 type          :7;
    u16 length        :9;
    u32 oper_version  :8;
    u32 max_version   :8;
    u32 enable        :1;
    u32 willing       :1;
    u32 error         :1;
    u32 reserved      :5;
    u32 sub_type      :8;
};

SIZE = 6 octets
```

Figure 9 – DCB Feature TLV Header Definition

Figure 10 shows a generic DCB Feature TLV structure. The details of each feature specific parameter structure are defined in the upcoming DCB Feature sections.

```
struct PROTCOL_feature_tlv {
    struct PROTOCOL_tlv_header h;
    struct PROTOCOL_feature_cfg
desired_cfg;
};

[illegible]
```

Figure 10 – Generic DCB Feature TLV

The following table lists the fields in the DCB Feature TLVs that are used to define the operation of the DCB Feature state machine.

Field	Type	Range	Default Value	Access (RO,RW, NA)	Description
Type	Integer	2..127	N/A	RO	Type code of the DCB Feature. Following is a list of defined types: 1 – PROTOCOL Control (not a feature) 2 – Priority Groups 3 – Priority Flow Control 4 – BCN 5 – Application 6 – Logical Link Down
Length	Integer	N/A	N/A	RO	Length of the DCB Feature sub-TLV payload (not including the Type and Length fields). The length is less than the maximum possible value (511) as this TLV is packaged inside the PROTOCOL TLV along with other feature TLV's.
Oper Version	Integer	0 .. 255	Highest supported	RO	Operating version of the feature. The system adjusts to operate at the highest version supported by both link partners.
Max Version	Integer	0 .. 255	Highest supported	RO	Highest feature version supported by the system. Version numbers start at zero. The feature must be backward compatible for all previous versions.
Enable	Boolean	Truth value	True	RW	Locally administered parameter that indicates whether the DCB feature is enabled or not.
Willing	Boolean	Truth value	True	RW	Locally administered parameter that indicates whether this feature accepts its configuration from the peer or not. When set to TRUE, the system uses the DesiredCfg supplied by a !Willing peer as the OperCfg. A system set to Willing must be capable of accepting any valid DesiredCfg for the feature from the peer. If both local and remote systems have the same value for the Willing flag,

					then the local DesiredCfg is used and the operational outcome of the exchange is determined by the Compatible method of the feature.
Error	Boolean	Truth value	False	RO	Indicates that an error has occurred during the configuration exchange with the peer. Error is set TRUE when the OperCfg and OperMode of a feature cannot be set as the protocol requires. Error is also set to TRUE when the Compatible method for the feature fails. The Feature turns OperMode to FALSE if either the local or remote Error flag is set to TRUE. Duplicate TLV's for the same Type/SubType or the PROTOCOL Control TLV also causes Error to be set to TRUE.
SubType	Integer	0..255	NA	RO	The Application TLV (Type == 5) and Logical Link Down TLV (Type == 6) might have multiple SubType values defined that represent specific types of network traffic. All other Feature TLV Types set the SubType field to zero. In general, the Type and SubType, taken together, identify a unique feature that is managed by an instance of the DCB Feature State Machine.
DesiredCfg	Structure	NA	NA	RW	This represents the locally configured values of the feature specific configuration.

Table 3 - DCB Feature TLV header field definitions

1.4.3.2 DCB Feature State Variables

The following table lists the additional state variables used to maintain each DCB Feature state machine.

State Variable	Type	Range	Default Value	Access (RO,RW	Description

				, NA)	
Advertise	Boolean	Truth value	True	RW	Locally administered parameter that indicates whether this feature is exchanged in the PROTOCOL TLV. When Advertise is False, received TLVs for this feature are ignored.
OperMode	Boolean	Truth value	False	RO	Operational state of the feature.
FeatureSyncNo	Integer	0 .. (2^{32} –1)		RO	When Syncd is False, this indicates the value that Protocol.SeqNo must become equal to before Syncd can become True.
Syncd	Boolean	Truth value	False	RO	Indicates whether the current DesiredConfig has been received by the peer.
OperCfg	Structure	NA	DesiredCfg	RO	The operating configuration of the feature – set to either DesiredCfg or PeerCfg.
PeerCfg	Structure	NA	N/A	RO	The DesiredCfg of the peer – as received in a PROTOCOL TLV from the peer.
PeerWilling	Boolean	Truth value	N/A	RO	The Willing state of the peer – as received in a PROTOCOL TLV from the peer.

Table 4 - PROTOCOL state variable definitions for the DCB Feature state machine

1.4.3.3 DCB Feature State Machine

In addition to the TLV fields and state variables previously described, the DCB Feature state machine uses the following mechanisms:

- **remoteFeatureChanged** – this represents an indication from the PROTOCOL Control state machine that there is new information from the peer (such as a new PROTOCOL TLV or data has expired).
- The **Syncd** variable from each DCB Feature state machine is visible to the PROTOCOL Control state machine.
- Each feature has a method called **Compatible** which is used to compare the DesiredCfg and PeerCfg.
- Each feature retrieves **LocalParams** from non-volatile storage or sets them to default values at link up.
- **Local parameter change** indicates that a configurable DCB Feature TLV field or state variable has been modified on the local system.

Figure 11 shows the operation of the DCB Feature state machine. Note that the figure is defined using an infinite loop model (such as no waiting). Implementations might use event waiting mechanisms as long as the function of the state machine is preserved.

A few notes concerning the notations used in Figure 11:

- TLV fields and state variables are used directly (e.g. SeqNo)
- State machines are identified as either PROTOCOL or Feature (e.g. PROTOCOL.SeqNo refers to the SeqNo variable from the PROTOCOL Control state machine).
- TLV fields received from the peer are identified as: Rx.<machine>.<variable> - such as Rx.PROTOCOL.SeqNo.

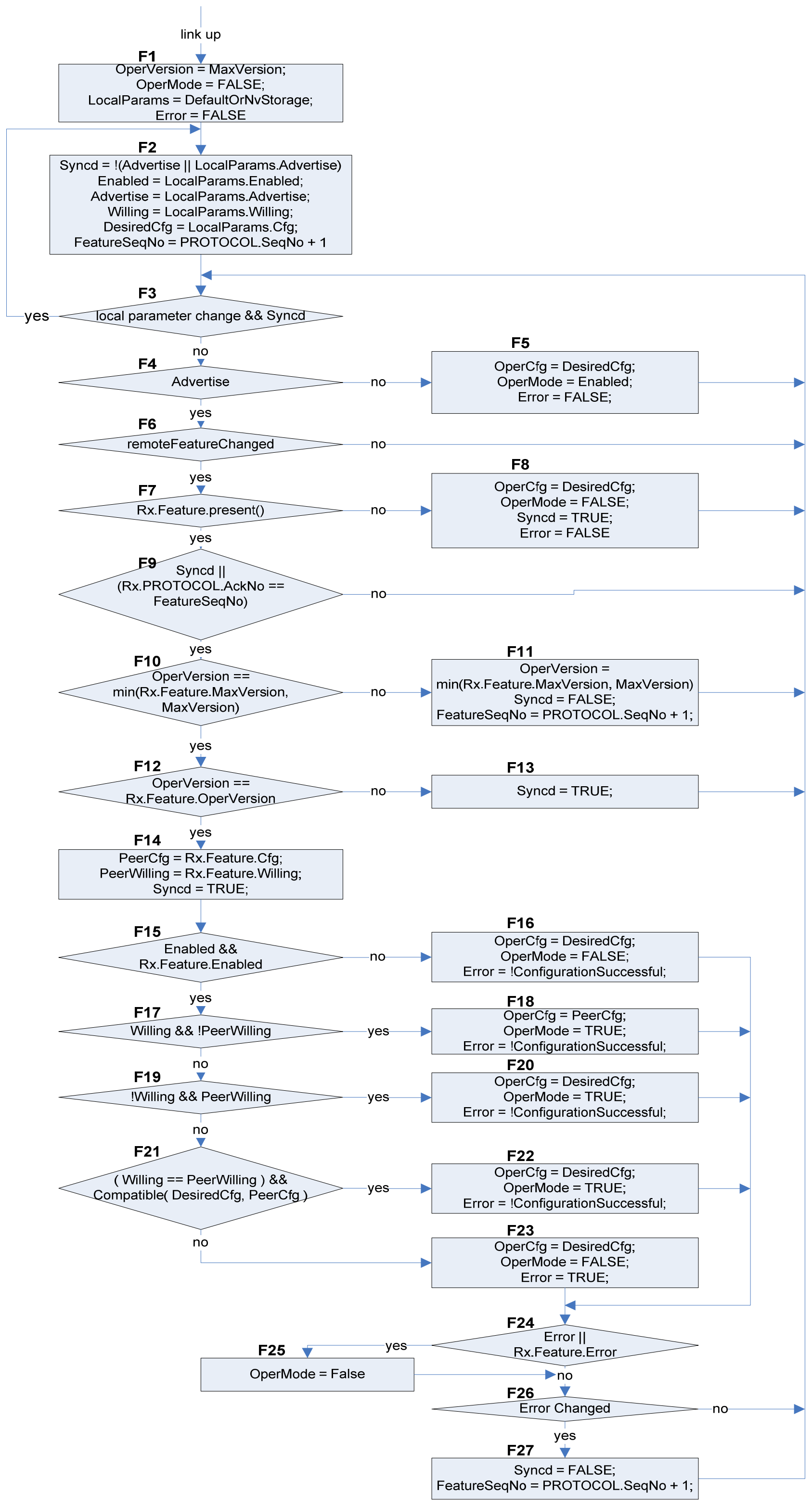

Figure 11 - DCB Feature State Machine

Commentary on the DCB Feature state machine by reference label (Fx):

- F2: If multiple features experience a change and set Syncd to FALSE, it is possible that the first change triggers the Control state machine to send an LLDP message. Additional pending changes do not get sent until the first change has been acknowledged (per D2 of Control state machine). In other words, the SeqNo's ratchet up and are acknowledged one value at a time. For example, the AckNo from the peer could be 9, the local SeqNo is 10, and multiple features could be pending with FeatureSeqNo at 11. A PDU with SeqNo 11 is not sent until an AckNo of 10 is received.
- F7: If 'remoteFeatureChanged' was the expiration of the PROTOCOL TLV TTL, then the feature is not present and OperMode is set to FALSE.
- F5 (any place OperCfg is set) – The hardware configuration for the feature takes place at the point OperCfg is set and the OperMode is set. The implementation might keep track of whether or not the OperCfg and OperMode have actually changed and require an update to the hardware configuration.
- F16, F18, F20, F22 – The Error flag is set based on the success or failure of updating the feature configuration, which is occurring implicitly here and is represented by the ConfigurationSuccessful flag. Error indicates that a problem outside the scope of the protocol has occurred, which prevented the feature configuration from being successful.
- F23 – Error is explicitly set to TRUE here to indicate that the two peers have a DCB configuration that is not compatible.

1.4.4 Manager Notifications

Implementations might choose to generate notifications when certain events occur. These types of events could include:

- Conditions indicating possible configuration error –for example, when the Compatible method fails.
- Conditions where the feature is not present on the peer. This can happen when a device does not support a feature (not really an error) or if the feature's Advertise flag is off (possible configuration error).
- The peer stops responding – as evidenced by an LLDP timeout event (delivered via the somethingRemoteChanged indication).
- Each time the Error flag is set to TRUE.

2. DCB Features

This section defines the DCB Feature parameters and statistics.

2.1 Priority Group Feature

This section describes the details of the Priority Group feature. The Priority Groups Specification provides configuration tables as well as a scheduling algorithm for managing bandwidth for various traffic classes on a converged link.

NOTE: Although it is expected that DCB devices will eventually provide scheduling functionality as specified in the Priority Group specification (or better), legacy implementations exist. To encourage wider adoption, this Priority Group Feature allows legacy implementations to match scheduler capabilities to the behavior implied by the Priority Group specification as close as possible. All PROTOCOL implementations must be capable of advertising the Priority Group TLV.

2.1.1 Priority Group Parameters

The following table lists the Priority Group parameters. Note that some parameters are defined with Local scope. This means that they are configured locally, are not exchanged via PROTOCOL, and do not interact with the PROTOCOL protocol state machines.

Parameter	**Syntax**	**Range**	**Default Value**	**Access (RO,RW, NA)**	**Scope**	**Description**
Bandwidth Group (BWG) Allocation	Table					
BWG ID (index)	Integer	0..7		RW	Exchanged	Queue bandwidth group
BWG Percentage	Integer	0..100		RW	Exchanged	Percentage of link bandwidth
User Priority Allocation	Table					
User Priority (index)	Integer	0..7		RW	Exchanged	
BWG ID	Integer	0..7		RW	Exchanged	BWG to which the priority belongs
User Priority Percentage	Integer	0..100		RW	Exchanged	Percentage of BWG bandwidth
Strict Priority	Integer	0..2		RW	Exchanged	Strict priority settings: 0 – no strict priority 1 – Group Strict Priority (GSP) 2 – Link Strict Priority (LSP)

Table 5 - Priority Group Parameters

2.1.2 Priority Group TLV

Figure 12 shows Priority Group parameters structure that is used in the Priority Group Feature TLV.

```
struct PROTOCOL_pg_cfg {
    u8  bwg_percentages[8];  /* percentage of link BW per BWG */
    struct {
        u8 bwg_id      :3;   /* BWG ID */
        u8 strict_prio :2;   /* 1: LSP, 2: GSP, 3: reserved */
        u8             :3;
        u8 bw_percentage;    /* percentage of BWG bandwidth */
    } pg_up_settings[8];     /* Index is user priority */
};

SIZE = 24 octets
```

Figure 12 - Priority Group Parameters Structure

2.1.3 Priority Group Parameter Comparison

Table 6 lists how the Priority Group parameters of the local and peer nodes are compared to determine if they match or not.

Table 6 - Priority Groups Parameter Comparison

Parameter	Comparison
Bandwidth Group (BWG) Allocation	
BWG ID (index)	Needs to match
BWG Percentage	Needs to match
User Priority Allocation	
User Priority (index)	Needs to match
BWG ID	Needs to match
User Priority Percentage	Needs to match
Strict Priority	Needs to match

2.2 Priority Flow Control (PFC) Feature

This section describes the details of the Priority Flow Control feature. This feature is important to provide “no-drop” packet delivery for certain traffic classes while maintaining existing LAN behavior for other traffic classes on converged link.

NOTE: Legacy implementations that do not support Priority Flow Control can signal this by setting "Enable" to FALSE. This effectively disables the Priority Flow Control feature at which time the peers fall back to configured 802.3x PAUSE behavior. All PROTOCOL implementations must be capable of advertising the Priority Flow Control TLV.

2.2.1 Priority Flow Control Parameters

Table 7 lists the Priority Flow Control parameters.

Parameter	Syntax	Range	Default Value	Access (RO,RW, NA)	Scope	Description
PFC Config	Table					
User Priority (index)	Integer	0..7		RW	Exchanged	
Admin mode	Integer	0..1	0	RW	Exchanged	Administrative PFC mode. 0: Disabled 1: Enabled PFC Enabled means that flow control in both directions (Rx and Tx) is enabled.

Table 7 - Priority Flow Control Parameters

2.2.2 Priority Flow Control TLV

Figure 13 shows the Priority Flow Control parameters structure that is used in the Priority Flow Control Feature TLV.

```
struct PROTOCOL_pfc_cfg {
    u8 admin_map;  /* bitmap of admin mode, bit position is user priority */
};

SIZE = 1 octets
```

Figure 13 - Priority Flow Control Parameters Structure

2.2.3 Priority Flow Control Parameter Comparison

Local and remote parameter comparison for Admin Mode is done as follows:

```
foreach (user_priority)
{
    if ((localAdminMode == Disabled == remoteAdminMode)
            ||
        (localAdminMode == Enabled == remoteAdminMode))
    {
        Comparison successful – configuration match …
    }
    else
    {
        Comparison fails – configuration mismatch …
        break
    }
}
```

2.3 BCN

This section describes the details of the BCN feature. Note that parameters that are marked as having local scope are not exchanged in the BCN sub-TLV.

2.3.1 BCN Parameters

Parameter	Syntax	Range	Default Value	Access (RO,RW, NA)	Scope	Description
BCNA field						
BCNA	Octets	8 octets	NA	RW	Exchanged	The BCNA value passed from switch to host.
BCN Admin Mode	Table					
userPriority (index)	Integer	0..7		RW	Exchanged	
CP Admin Mode	TruthValue			RW	Exchanged	This device is capable of generating BCN (This is used by the peer to determine if it needs to forward BCN tagged frames or strip BCN tag)
RP Admin Mode	TruthValue			RW	Exchanged	This device is capable of responding to BCNs (rate controllers can be installed on all/any sources of traffic).
BCN Oper Mode	Table			NA		
User Priority (index)	Integer	0..7		RW	Exchanged	
RP Oper Mode	TruthValue		True	RO	Exchanged	This is set for an edge switch if connected end station does not support Rate Controllers. If RP Admin == True Set to False If RP Admin == False Set to True

Remove Tag Oper Mode	TruthValue		False	RO	Exchanged	CM Tag needs to be removed before data is sent out on this port. Set based on CP Admin Mode of peer port. If CP Admin == False Set to True If CP Admin == True Set to False
BCN Params	Table			NA		BCN parameters apply for all User Priorities
Rp Gd	Double	0..1		RW	Exchanged	Reaction Point: Decrement Coefficient
Rp Gi	Double	0..1		RW	Exchanged	Reaction Point: Increment Coefficient
Rp W	Integer	1/8 - 8		RW	Exchanged	Reaction Point: Derivative Weight
Rp Ru	Integer	1-100		RW	Local	Reaction Point: Rate Unit
Rp Tmax	Integer	1 .. 1M		RW	Exchanged	Reaction Point: Maximum Time to backoff after BCN0
Rp Rmin	Integer	0..link_ rate		RW	Exchanged	Reaction Point: Default Rate to resume after first BCN0
Rp Wrtt	Integer	0..10		RW	Local	Reaction Point: RTT moving average weight
Rp Ri	Integer	0..link_ rate		RW	Local	Reaction Point: Initial rate. Fixed at 50% link rate.
Rp Alpha	Double	0..1		RW	Exchanged	Reaction Point: Maximum decrease factor
Rp Beta	Double	0..1		RW	Exchanged	Reaction Point: Maximum increase factor
Rp C	Integer	0..100K		RW	Local	Reaction Point: Link capacity Fixed
Rp Td	Integer	1..10K		RW	Exchanged	Reaction Point: Drift Interval
Rp Rd	Integer	1..100		RW	Exchanged	Reaction Point: Drift factor

Cp Bmc	Integer	0..Buf max		RW	Local	Congestion Point: Buffer Medium Congestion threshold (used to set M bit in BCN)
Cp Bsc	Integer	0..Buf max		RW	Local	Congestion Point: Buffer Severe congestion threshold (used to set S bit in BCN)
Cp Qeq	Integer	0..Qm ax		RW	Local	Congestion Point: Queue Equilibrium. Min is fixed, instantiated
Cp Qsc	Integer	0..Qm ax		RW	Local	Congestion Point: Queue Severe Congestion Threshold (Qsc, used to send BCN 0)
Cp Qmc	Integer	0..Qm ax		RW	Local	Congestion Point: Queue Mild Congestion Threshold (between Qmc and Qsc, Max feedback signal is sent)
Cp Qscale	Integer	1/8..8		RW	Local	Congestion Point: Queue Scale factor
Cp Sf	Integer	1..256K		RW	Exchanged	Congestion Point: Fixed portion of the sampling interval
Cp Sr	Integer	1..64K		RW	Local	Congestion Point: Random portion of the sampling interval Fixed (as a % of Sf)

Table 8 - BCN Parameters

2.3.2 BCN TLV

Figure 14 shows the BCN parameters structure that is used in the BCN Feature TLV. To keep the size of the BCN TLV down to a reasonable size, only one set of BCN parameters is exchanged. These parameters apply for all BCN enabled user priorities.

```
struct PROTOCOL_bcn_cfg {
    u8 bcna[8];              /* CM-Tag BCNA field */
    struct {
        u8 cp_admin     :1;  /* CP admin mode */
        u8 rp_admin     :1;  /* RP admin mode */
        u8 rp_oper      :1;  /* RP operational mode */
        u8 rem_tag_oper :1;  /* Remove CM tag operational mode */
        u8              :4;
    } bcn_up_settings[8];    /* Index is user priority */
    double rp_alpha;         /* RP max decrease factor */
    double rp_beta;          /* RP max increase factor */
    double rp_gd;            /* RP decrement coefficient */
    double rp_gi;            /* RP increment coefficient */
    u32 rp_tmax;             /* RP max time to backoff after BCN0 */
    u32 cp_sf;               /* CP sampling interval fixed */
    u16 rp_td;               /* RP drift interval */
    u16 rp_rmin;             /* RP default rate after 1st BCN0 */
    u8 rp_w;                 /* RP derivative rate */
    u8 rp_rd;                /* RP drift factor */
};

SIZE = 62 octets
```

Figure 14 - BCN Parameters Structure

2.3.3 BCN Parameter Comparison

If a host is connected to a switch whose BCN is disabled (BCN enable is off), then the host goes to Oper Mode off.

The BCN RP capability of the host is used by the switch to determine if it needs to support RP capability. Similarly, the BCN CP capability of the host is used by the switch to determine if it needs to forward BCN tagged packets to the host.
All of the other numerical parameters must match exactly.

2.4 Application TLV

Application TLVs are used to exchange information that is needed to enable specific Applications to operate optimally on the DCB network. For example, an Application TLV might specify the User Priorities that should be used for specific Applications traffic. Specific Applications are identified by the SubType field. The Application TLV parameters can vary based on the SubType.

The primary function of PROTOCOL is to exchange Application TLV's and coordinate with the peer based on the rules of the DCB Feature state machine. PROTOCOL does not need to understand the details of the Application parameters. Applications can set and query their Application TLV parameters via a local management interface with PROTOCOL.

Table 9 lists the Application SubTypes currently defined.

SubType Value	Description
0	Fiber Channel over Ethernet (FCoE)

Table 9 - Application SubTypes

The following sub-sections define the parameters and TLVs for the general Application TLV and for each of the defined Application SubTypes.

2.4.1 General Application Feature

2.4.1.1 Application Parameters

Table 10 lists the Application TLV parameters.

Parameter	Syntax	Range	Default Value	Access (RO,RW, NA)	Scope	Description
AppParams	Octet array	NA	NA	RW	Exchanged	An array of Application data

Table 10 – Application Parameters

2.4.1.2 Application TLV

Figure 15 shows the general Application parameters structure that is used in the Application Feature TLV.

```
struct PROTOCOL_app_cfg {
    u8 app_params[n];
};

SIZE = n octets
```

Figure 15 - Application Parameters Structure

2.4.1.3 Application Parameter Comparison

The local and remote Application parameters must match exactly in order for the local and peer Application TLVs to be considered compatible – per the DCB Feature state machine.

2.4.2 FCoE Application

2.4.2.1 FCoE Parameters

Table 11 lists the FCoE Application parameters.

Parameter	Syntax	Rang e	Default Value	Access (RO,RW, NA)	Scope	Description
User Priority Map	Bitmap	0..255	8	RW	Exchanged	Each bit represents a User Priority that is associated with FCoE traffic.

Table 11 – Traffic Type Parameters

2.4.2.2 FCoE TLV

Figure 16 shows the FCoE parameter structure that is used in the FCoE Application Feature TLV.

```
struct PROTOCOL_fcoe_cfg {
    u8 user_priority_map :8;
};

SIZE = 1 octets
```

Figure 16 - FCoE Parameters Structure

2.4.2.3 FCoE Parameter Comparison

As with the general Application TLV, the local and remote parameters must match exactly in order for the local and peer FCoE TLVs to be considered compatible.

2.5 Logical Link Down Feature

Logical Link Down is a mechanism that enables a networking component to relay the connectivity or reachability of a type of traffic to a peer. This enables the peer to make systemic changes based on the knowledge. Many of the network redundancy mechanisms deployed today rely on knowledge of physical failures.

Figure 17 shows an example use case. In the example, a dual up-link FCoE adapter is connected to two FCoE capable switches, which in turn connect to both a common Ethernet LAN and two Fibre Channel SANs.

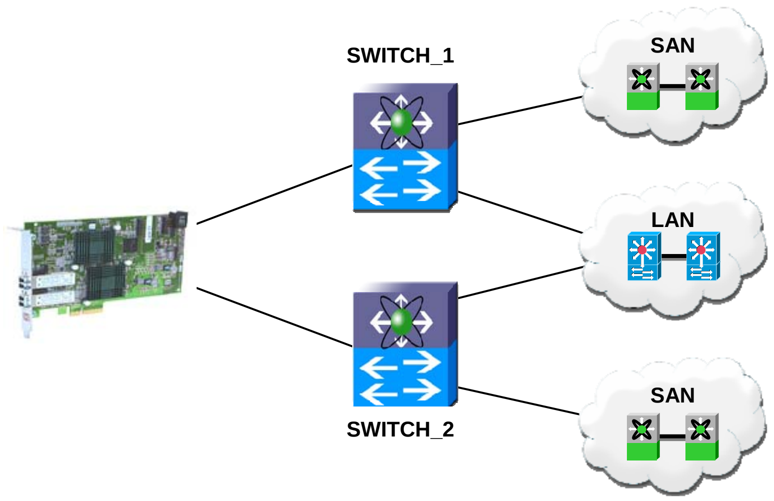

Figure 17 – LAN and FCoE SAN Use Case

The host includes the capability to access both the LAN and SAN via the DCB adapter. Per typical deployment mechanisms, a LUN aware multi-pathing mechanism exists above the SCSI layer for storage traffic, and an active-passive bonding driver is deployed for LAN redundancy. If the host's primary LAN connections through SWITCH_1 fail and the SAN connection stays intact, we'd like the LAN bonding driver to know about this situation and start using the alternate LAN interface while continuing to use the SAN interface on SWITCH_1. Logical Link Down is the mechanism used to signal this desire.

Table 12 lists the SubTypes currently defined for this TLV.

SubType Value	Description
0	FCoE Logical Link Status
1	LAN Logical Link Status

Table 12 - Logical Link Status TLV SubTypes

2.5.1 FCoE Logical Link Status

2.5.1.1 FCoE Logical Link Status Parameters

Table 13 lists the FCoE Logical Link Status parameters.

Parameter	Syntax	Range	Default Value	Access (RO,RW, NA)	Scope	Description
Logical Link Status	TruthValue		0	RW	Exchanged	This bit signifies whether the Logical Link Status of this type of network (FCoE) is up or not.

Table 13 - FCoE Logical Link Status Parameters

2.5.1.2 FCoE Logical Link Status TLV

Figure 18 shows the FCoE Logical Link Status parameter structure that is used in the FCoE Logical Link Status Feature TLV.

```
struct PROTOCOL_fcoe_logical_link_status_cfg {
    u8  logical_link_status:1;
    u8  resv:7;
};

SIZE = 1 octet
```

Figure 18 – FCoE Logical Link Status Parameters Structure

2.5.1.3 FCoE Logical Link Status Parameter Comparison

The DCB adapters with the ability to failover on the Logical Link level for the SAN network should advertise this TLV; otherwise the adapter should not advertise this feature TLV. Since this TLV is used for implementing network redundancy mechanisms by the adapter, the Willing bit should be set to TRUE on the adapter side. The adapter only acts on the logical_link_status bit defined above; it cannot generate this information.

If the adapter sets these bits, the bits are ignored by the switch. Since the switch pushes this information, the Willing bit is set to FALSE on the switch side. Except the combination above all other combinations are not valid. Since it is not expected that both Willing bits are the same, the comparison function always returns an error irrespective of the configurations.

2.5.2 LAN Logical Link Status

2.5.2.1 LAN Logical Link Status Parameters

Table 14 lists the LAN Logical Link Status parameters.

Parameter	Syntax	Range	Default Value	Access (RO,RW, NA)	Scope	Description
Logical Link Status	TruthValue		0	RW	Exchanged	This bit signifies whether the Logical Link Status of this type of network (LAN) is up or not.

Table 14 - LAN Logical Link Status Parameters

2.5.2.2 LAN Logical Link Status TLV

Figure 19 shows the LAN Logical Link Status parameter structure that is used in the LAN Logical Link Status Feature TLV.

```
struct PROTOCOL_lan_logical_link_status_cfg {
    u8  logical_link_status:1;
    u8  resv:7;
};
SIZE = 1 octets
```

Figure 19 – LAN Logical Link Status Parameters Structure

2.5.2.3 LAN Logical Link Status Parameter Comparison

The DCB adapters with the ability to failover on the Logical Link level for the LAN network should advertise this TLV; otherwise the adapter should not advertise this feature TLV. Since this TLV is used for implementing network redundancy mechanisms by the adapter, the Willing bit should be set to TRUE on the adapter side. The adapter only acts on the logical_link_status bit defined above; it cannot generate this information.

If the adapter sets these bits, the bits are ignored by the switch. Since the switch pushes this information, the Willing bit is set to FALSE on the switch side. Except the combination above, all other combinations are not valid. Since it is not expected that both Willing bits are the same, the comparison function always return an error irrespective of the configurations.

3. Additional Feature TLV's

This section defines additional optional DCB feature TLV's that might be supported.

Table 15 lists a summary of the additional feature TLV's listed in this appendix.

Type field	TLV description
7	Network Interface Virtualization

Table 15 - Summary of additional Feature TLV's

3.1 Network Interface Virtualization (NIV) Control Virtual Interface Feature

This section describes the details of the Network Interface Virtualization Feature. A DCB adapter and associated network device peer can choose to enable VN-Tag encapsulation using this feature. For more information on the benefits of using VN-Tag encapsulation and details of the VN-Tag are specified in the NIV Specification (contact Nuova Systems for more details about NIV). The advertisement of this feature TLV by a PROTOCOL entity implies the support for NIV. If the VN-Tag encapsulation is enabled on an interface all frames (including the control frames like PROTOCOL PDU's) should be using this encapsulation. In this case there is a need for negotiation of a control channel virtual interface to distinguish this traffic from the normal data traffic.

3.1.1 NIV Control Virtual Interface Parameters

Table 16 lists the NIV parameters.

Table 16 - Network Interface Virtualization Parameters

Parameter	Syntax	Range	Default Value	Access (RO,RW, NA)	Scope	Description
Control Channel Virtual Interface	Integer	0..4095	0	RW	Exchanged	The value of this parameter is the virtual interface that control frames are sent/received on this interface. The control channel frames should be priority tagged.

3.1.2 NIV Control Virtual Interface TLV

Figure 20 shows the NIV Parameters structure that is used in the NIV Feature TLV.

```
struct PROTOCOL_niv_control_virtual_interface_cfg {
    u16 control_virtual_interface:12;
    u16 resv:4
};

SIZE = 2 octets
```

Figure 20 - Network Interface Virtualization Parameters Structure

3.1.3 NIV Control Virtual Interface Parameter Comparison

In a normal mode of operation, a DCB adapter does a configuration push of the control channel VN-Tag to the peer (switch). This means that the Willing bit on the DCB adapter side is set to FALSE and is set to TRUE on the switch side. If both peers' Willing bit is the same then Table 17 should be used for the Comparison.

Parameter	Comparison
control_virtual_interface	Needs to match

Table 17 - Network Interface Virtualization Parameter Comparison