**IEEE P802.1Qdw**

**November, 2024**

(Amendment to IEEE Std 802.1Q-2022 as amended by

IEEE Std Qcz-2022, IEEE Std 802.1Qcw-2023, IEEE Std 802.1Qcj-2023, and IEEE Std 802.1Qdt-2023)

## Proposed text for

## Local and metropolitan area networks—

# Bridges and Bridged Networks

# Amendment: Source Flow Control

Unapproved contribution, prepared by

**Lihao Chen, Huawei Technologies Co., Ltd.**

Individual contribution to IEEE 802.1

# Contents

# 1. Overview

## 1.3 Introduction

Insert the following text at the end of 1.3 and renumber accordingly:

This standard specifies protocols, procedures and management objects that support flow control of congesting data flows within data center environments. This is achieved by enabling systems to signal congestion and its expected duration directly to the sender end-station of congesting data flow. Source Flow Control applies flow control to congesting flows of higher layer protocols at the sender end-station by providing sufficient information to pause individual congesting end-to-end flows. It reduces head-of-line blocking and PFC storm in the network compared to hop-by-hop flow control mechanism while it helps to reduce queue build-up and packet loss in the network. A method for PFC-based signaling to pause PFC priorities instead of end-to-end flows is provided for backwards compatibility with PFC-only enabled end stations.

To this purpose Source Flow Control does:

a) Defines a means for bridges to signal congestion directly to the sender of a specific traffic flow or priority class contributing to the congestion.

b) Specifies for senders of traffic flows to react to congestion signals.

c) Defines a means for bridges connected to senders of traffic flows to react to congestion signals and convert them to PFC signals if the sender is not capable of processing the congestion signals.

# 3. Definitions

*Insert the following definitions in the appropriate collating sequence, re-numbering as appropriate:*

**3.1 Source Flow Control Signaling System:** An End Station or Bridge Component conforming to the Source Flow Control signaling provisions of this standard.

**3.2 Source Flow Control Reception Station:** An End Station or Bridge Component conforming to the Source Flow Control signal processing provision of this standard.

**3.3 Source Flow Control Message (SFCM):** A message transmitted by a Source Flow Control Aware System, conveying congesting flow information used to flow control a congesting flow.

**3.4 Source Flow Control Point (SFCP):** A Virtual Local Area Network (VLAN) Bridge or end station Port function that monitors a set of queues for congesting flows, and can generate Source Flow Control Messages or identify Source Flow Control Messages to execute flow control.

# 4. Abbreviations

*Insert the following abbreviations in the appropriate sequence, re-ordering as appropriate:*

| | |
|---|---|
| **SFC** | Source Flow Control |
| **SFCM** | Source Flow Control Message |
| **SFCP** | Source Flow Control Point |

# 5. Conformance

### 5.4.1 VLAN Bridge component options

*Insert the following at the end of the lettered list in 5.4.1, renumbering accordingly:*

a) Support Source Flow Control (SFC) (5.4.9)

*Insert the following sub-clause at the end of 5.4, renumbering as appropriate:*

### 5.4.9 Source Flow Control (SFC) operation (optional)

A VLAN Bridge implementation that conforms to the provisions of this standard for Source Flow Control (Clause 52) shall:

a) <<TBD>>

*Insert the following at the end of Clause 5, renumbering as appropriate:*

## 5.34 End station requirements - SFC

<<TBD>>

# 6. Support of the MAC Service

### 6.10.1 Data indications

*Change the first paragraph of Clause 6.10.1 as follows:*

On receipt of an M_UNITDATA.indication primitive from the PIP-ISS, if the PIP is congestion aware (5.4.1.4) and the initial octets of the mac_service_data_unit contain a valid CNM encapsulation, the received frame is processed according to 32.16. If the PIP is congestion isolation aware (5.4.7) and the initial octets of the mac_service_data_unit contain a valid CIM encapsulation (49.4.3), the received frame is processed according to 49.4.2.6. If the PIP is Source Flow Control aware (5.4.9) and the initial octets of the mac service data unit contain a valid SFCM encapsulation (<<52.5.3>>), the received frame is processed according to <<52.5.2.5>>. Otherwise, the received frame shall be discarded if: …

# 12.Bridge management

## 12.xx Source Flow Control managed objects

Several variables control the operation of Source Flow Control in a source flow control aware Bridge. The managed objects are as follows:

a) SFC entity managed object (12.xx.1)

b) SFC Source Table (12.xx.2)

c) SFCP entity managed object (12.xx.3)

### 12.xx.1 SFC entity managed object

<<TBD>>

### 12.xx.2 SFC Source Table

<<TBD>>

### 12.xx.3 SFCP entity managed object

<<TBD>>

*Insert a new Clause "52. Source Flow Control" as follows:*

# 52.Source Flow Control

Source Flow Control (SFC) pauses congesting flows at the source by reacting to a Source Flow Control Message (SFCM) sent across a data center network from a port with a congesting traffic class that is enabled with the feature. The SFCM identifies a congesting flow at the source end station and provides a pause interval for which transmissions of frames for the flow are to be paused. The receiver of the SFCM asserts flow control on the traffic class used to transmit the flow or may provide more advanced implementation specific per-flow control if implemented. The externally visible behavior of SFC is that the bridge or end station processing a received SFCM will pause the traffic class of the identified flow for the specified interval.

The models of operation in this clause provide a basis for specifying the externally observable behavior of SFC and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified.

This clause introduces the concepts and protocols essential to source flow control as follows:

a) The objectives for source flow control (52.1).
b) Principles of source flow control (52.2).
c) Source flow control aware forwarding process (52.3, 52.4).
d) Source flow control protocol (52.5).

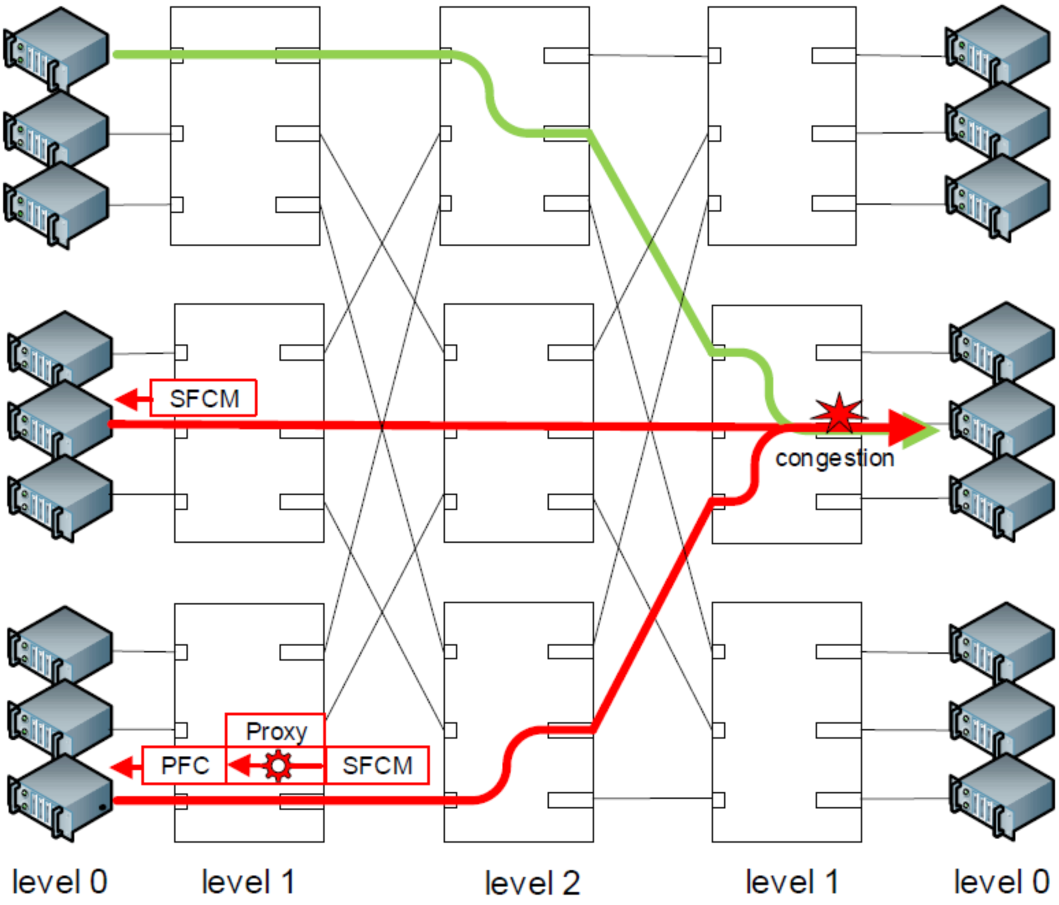


**Figure 52-1—Source flow control example operation**

Figure 52-1 shows an example operation of SFC. In the figure, all of the relay systems are layer-3 routers and may be SFC aware. The example shows a network of layer-3 routers, but SFC also applies to Bridged Networks. It is most important for the edge bridges (level 1 systems in Figure 52-1) are SFC aware, however when all systems are SFC aware different reasons for congestion may be detected. In the figure, server to server traffic is flowing from left to right across a data center network. When congestion is detected, SFC attempts to identify the congesting flows (52.2.1). For each identified congesting flow (e.g. the red flows in Figure 52- 1), the SFC aware system sends a Source Flow Control Message (SFCM) to the sending end station (level 0 system in Figure 52-1). If the end station is SFC aware, the SFCM is forwarded through the edge bridge directly to the end station (as seen in the middle of the diagram). If the end station is not SFC aware, the edge bridge should intercept the SFCM and convert the message to a traditional PFC message (as seen in the lower part of the diagram). The edge bridge converting the SFCM to a PFC message is known as SFC proxy mode and should be implemented by the edge bridges directly attached to end stations. The SFCM contains information necessary for the source end station or the proxy edge bridge to identify the traffic class of the congesting flow. The flow information may also be used to identify and invoke per-flow implementation specific traffic controls.

## 52.1 Source flow control objectives

The operation, procedures and protocols of source flow control are designed to meet the following objectives by category:

Functionality

a) Provide a means for low-latency flow control signaling for individual, congesting end-to-end flows across a network.

b) Reduce queue build up in congesting queues with minimal side effects on other flows in the network.

c) Move queueing and congestion from in the network to the edge, into the end stations.

d) Reduce the likelihood of frame loss.

e) Avoid the triggering of PFC, not replace PFC.

f) Invoke flow control on frames of congesting flows before the frames experience congestion in the queue management system.

g) Do not keep per-flow state in the congested device other than a probabilistic data structure to limit the messaging overhead per flow.

h) Specify the signaling of the pause duration of traffic classes.

i) Do only specify the minimum reaction to a source flow control message.

Compatibility

j) Be oblivious of protocols above the network layer. Do not be specific to higher layer protocols.

k) Work in existing lossless environments using PFC without requiring additional traffic classes.

l) Work in layer-2 and layer-3 networks.

Performance

m) Reduce queueing in the network and thereby the flow completion time across the network.

n) SFC messages should be as small as possible to ensure low-latency forwarding as well as low signaling overhead.

o) Reduce head-of-line blocking when using in an environment with PFC enabled.

Scale

p) Work in arbitrary data center network topologies with a mix of link speeds.

q) Limit messaging overhead by restricting the number of congestion messages per flow to a fixed number per RTT.

r) Limit the messaging overhead by signaling the pause duration without causing underutilization.

Implementation complexity

s) The main goal of SFC is implementational simplicity.

t) Require changes only on congested bridges as well as sender NICs, or the edge bridge directly attached to the sender end station. No modifications on intermediate bridges are required.

Manageability

u) Limit the ability to configure an inoperable environment.

v) Provide auto discovery of SFC message processing capability between last hop switches and NICs using existing LLDP messages and without creating additional hello and auto-configuration protocols.

## 52.2 Principles of source flow control

This clause introduces the principles of source flow control. Items a) through d) describe the life of a congesting flow from identification through pausing of the flow. Items e) through g) compare and contrast Source Flow Control with Priority-based Flow Control (Clause 36), Congestion Notification (Clause 30) and Congestion Isolation (Clause 49).

The following items describe the principals of source flow control:

a) Congesting flow identification (52.2.1).
b) Source flow control signaling (52.2.2).
c) End station SFCM reception (52.2.3).
d) Proxy SFCM reception (52.2.4).
e) Comparison to Priority-based Flow Control (52.2.5).
f) Comparison to Congestion Notification (52.2.6).
g) Comparison to Congestion Isolation (52.2.7).

### 52.2.1 Congesting flow identification

<<reference or leverage text from Clause 49. Congestion Isolation>>

There are many potential methods of identifying congesting flows and interoperable implementations can exist using different approaches. The SFCP Congestion Detection function (52.3.1) of the Source Flow Control Aware Forwarding Process (52.3) is responsible for the implementation.

Many modern data centers utilize encapsulated overlay networks, such as those described in IETF RFC 8014[B46]. An overlay network can carry multiple encapsulated flows within a single encapsulation flow. The congesting and non-congesting flows identified by SFC are the outer encapsulation flow as seen by the underlay network. The inner encapsulated flows might not be visible to the bridges and routers within the data center network, and are therefore not separated into congesting and non-congesting flows.

### 52.2.2 Source flow control signaling

Once a frame has been identified as being part of a congesting flow, an SFCM is created and sent to the source of the congesting flow. The SFCM carries information including the pause duration of the related traffic class that are used by the end station SFCM reception or by the proxy SFCM reception to trigger a PFC request. The SFCM PDU is encapsulated in a UDP packet and the destination address is the source address of the congesting frame.

The SFCM is addressed to the source end station. The system generating the SFCM must locate SFCM format information and the address parameters of the upstream peer by indexing into the SFC Source Table (12.xx.2) using the source address as the index.

The SFCM is identified at the sender end station or the edge bridge directly attached to the end station, by examining the contents of the SFCM.

*NOTE—A desirable approach is to use a specific UDP port number encapsulated in the SFCM PDU, see 52.5.1.1.5. And this number is used to identified the signaling message, i.e., the SFCM.*

The SFCM may carry optional information that may be used by the end station or the edge bridge directly attached to the end station to perform flow-based flow control or other actions.

The system sending SFC frames should be configurable to limit the SFCM sending rate to a specific receiver.

### 52.2.3 End station SFCM reception

SFCM aware end stations receive SFCM messages and parse them to execute flow control accordingly. The end station prohibits subsequent frames from transmission by pausing the associated traffic class for the specified amount of time.

### 52.2.4 Proxy SFCM reception

SFC may be extended to end stations that are not SFC aware, but are PFC enabled by an SFC proxy function in directly attached bridges. The SFC proxy function enabled on the bridge attached to the SFC unaware end station is responsible for converting the SFCM addressed to the end station to a PFC frame that pauses the appropriate traffic class. To perform this operation, the SFC proxy function decapsulate the SFCM and retrieve contents for triggering a PFC request.

### 52.2.5 Comparison to Priority-based Flow Control

PFC aims at eliminating frame loss. SFC does not have that goal. Instead, SFC aims at controlling the queue buildup during congestion events. Limiting queue build up reduces the likelihood of packet drops because of limited buffer space, it does not eliminate it though. For lossless network behavior, SFC should be used together with PFC enabled on the same traffic classes in the network.

In contrast to PFC, which uses hop-by-hop propagation of the flow control signal, SFC signals flow control directly to senders of congesting flows. Thereby, operational side effects of flow control that exist in hop-by-hop designs, such as PFC’s head-of-line blocking are eliminated. The queue build up is moved to the sender end station, minimizing the in-network side effects of flow control.

### 52.2.6 Comparison to Congestion Notification

Both Source Flow Control and Congestion Notification (Clause 30) generate congestion messages by the congestion monitoring point and send the messages upstream. While both direct the message across the Bridged Network to the source station, the addressing and the reaction on receiving the message differs. CN is designed to operate across a large layer 2 domain where reaction points in end-station network adapters regulate traffic. SFC is intended to operate in layer 3 networks and the reaction point invokes flow control, i.e. a pause of traffic injection. Moreover, SFC supports proxy mode so that SFC can be extended to end stations that are not SFC aware, but are PFC enabled.

### 52.2.7 Comparison to Congestion Isolation

Both Source Flow Control and Congestion Isolation (Clause 49) generate congestion messages by the congestion monitoring point and send the messages upstream. Both schemes supports L3 message formats. However, SFC signals to end-station (or via proxy bridges) while CI signals to the neighbor. CI does use an additional traffic class to isolate frames and does not directly rate control the sending host, while SFC pauses the sending host.

## 52.3 Bridge Component Source Flow Control Aware Forwarding Process

This clause specifies the architecture of the Source Flow Control Point (SFCP) in the Forwarding Process of a source flow control aware Bridge. In this architecture, a router is as a higher layer entity that relays frames using layer-3 information but uses the forwarding process of the underlying source flow control aware bridge to deliver frames to peers and end stations.

The models of operation in this clause provide a basis for specifying the externally observable behavior of SFC, and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified. Conformance of equipment to this standard is purely in respect of observable protocol.

Figure 8-12 illustrates the Bridge Forwarding Process at its highest conceptual level. Figure 52-2 focuses on the operation of a single Bridge Port and the relationship of new elements to the queuing and classification functions. Three new elements are specified for a SFC aware Bridge as follows:

a) SFCP Congestion Detection (52.3.1).
b) SFCM Multiplexer (52.3.2).
c) SFCM Demultiplexer (52.3.3).

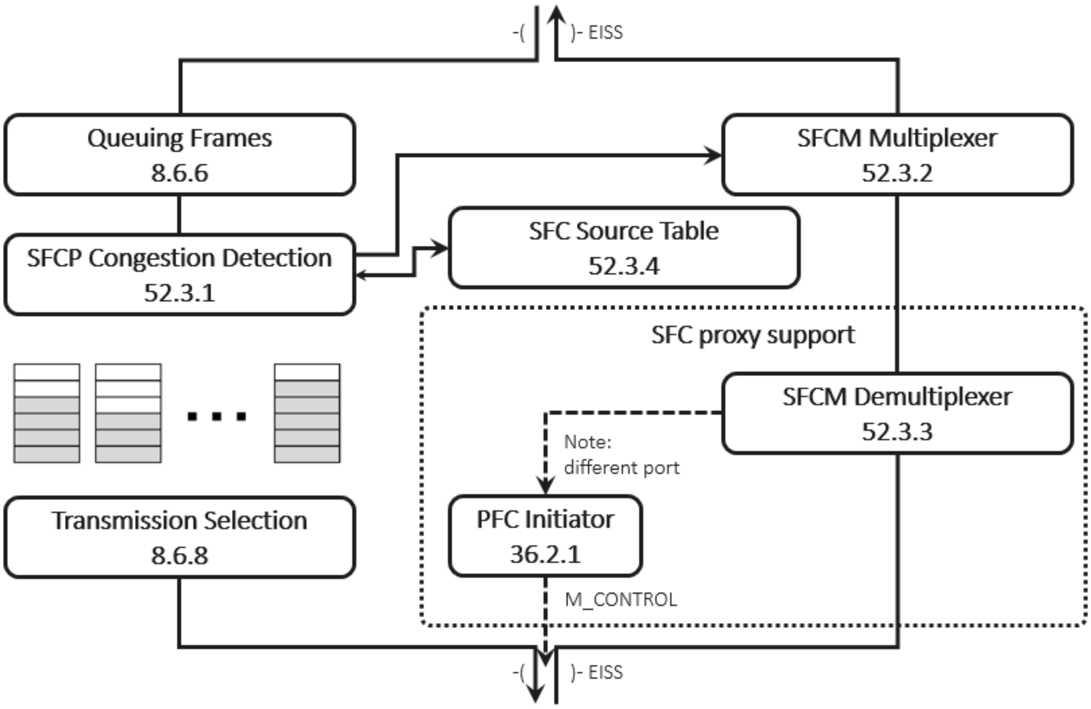


**Figure 52-2—Bridge component SFC reference diagram**

#### 52.3.1 SFCP Congestion Detection

SFCP Congestion Detection identifies frames of a congesting flow and generate an SFCM based on the contents of the frame as well as the information related to the congestion. SFCP Congestion Detection is also responsible for creating entries in the SFC Source Table (12.xx.2) for newly identified congesting flows. As described in 52.2.1, frames of a congesting flow are identified by a suitable method implemented by SFCP Congestion Detection.

Frames given to SFCP Congestion Detection by the Queuing Frames entity (8.6.6) in an EM_UNITDATA.request (52.5.2.2) may be identified as being part of a congesting flow. SFCP Congestion Detection creates a new entry in the SFC Source Table for congesting flows received on a monitored queue. An SFCM may be generated from the parameters obtained with the received frame. The SFCM is sent to the source of the congestion flow via the SFC multiplexer.

On each EM_UNITDATA.request, SFCP Congestion Detection indicates to the SFC Procedures (52.5.2) whether a monitored queue is congested or not congested.

### 52.3.2 SFCM Multiplexer

The SFCM multiplexer inserts SFCMs generated by SFCP Congestion Detection among frames received from the LAN. Layer-3 encapsulated SFCMs (IPv4 and IPv6) are routed to the source by a higher-layer routing function that is beyond the scope of this standard.

### 52.3.3 SFCM Demultiplexer

The SFCM demultiplexer is part of the SFCM proxy mode support. It identifies SFCMs received from the LAN and extracts the content of the SFCM PDU and determines whether the destination end station of the SFCM is directly attached to this bridge. If the destination end station of the SFCM is directly attached, the proxy function will assert PFC on the port connected to the destination end station. The port is located using layer-3 routing tables or implementation specific structures that identify how to reach a particular IP address.

An SFCM PDU may be encapsulated by two different SFCM encapsulations; IPv4 and IPv6. Implementations supporting IPv4 and IPv6 encapsulations must be able to identify and validate IPv4 and/or IPv6 packets in the SFCM demultiplexer. The rules for validating received SFCMs are specified in <<xx>>.

*NOTE—A desirable approach is to use a specific UDP port number encapsulated in the SFCM PDU, see 52.5.1.1.5. And this number is used to identify the SFCM.*

### 52.3.4 SFC Source Table

The SFC Source Table (12.xx.2) contains entries for each congesting flow. The entries are indexed by the source address of the congesting flow provided as a subparameter in EM_UNITDATA.request (52.5.2.2) invocations at the EISS. Each entry contains controlling variables (52.5.1.5) that allow an implementation to manage a congesting flow. This includes a means of identifying the congesting queue used to invoke the source flow control.

## 52.4 End Station Source Flow Control Aware Forwarding Process

This clause specifies the architecture of the Source Flow Control Point (SFCP) in the Forwarding Process of a source flow control aware End station.

Figure 52-3 focuses on the operation of a single End station port and the relationship of new elements to the queuing and classification functions. One new element is specified for a SFC aware Bridge as follows:

a) End Station SFCM Demultiplexer (52.4.1).

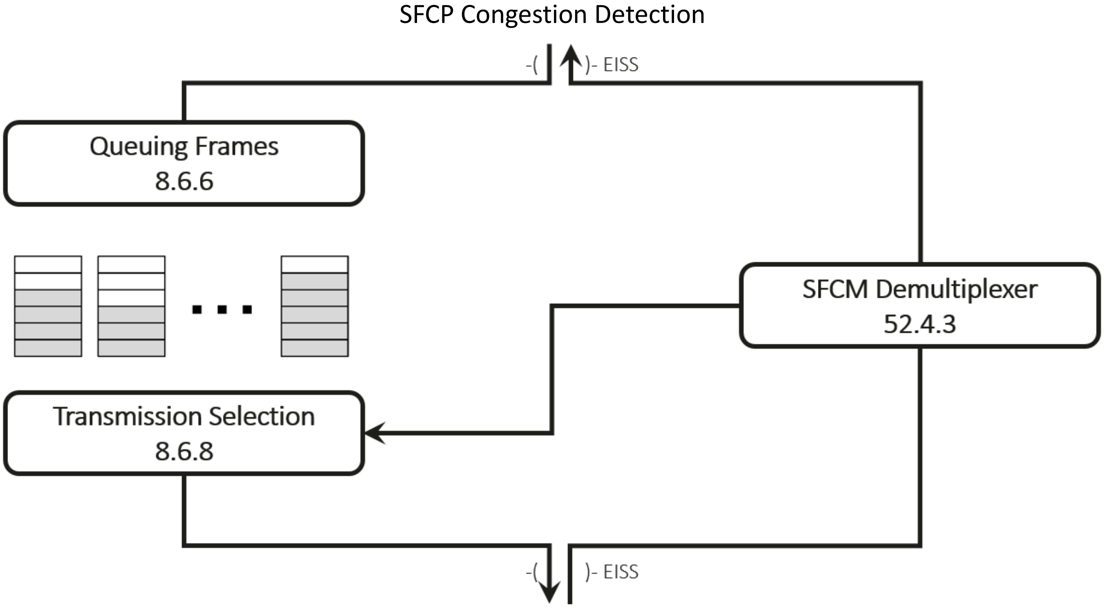


**Figure 52-3—End station SFC reference diagram**

### 52.4.1 End Station SFCM Demultiplexer

The SFCM demultiplexer identifies SFCMs received from the LAN and extracts the content of the SFCM PDU.

An SFCM PDU may be encapsulated by two different SFCM encapsulations; IPv4 and IPv6. Implementations supporting IPv4 and IPv6 encapsulations must be able to identify and validate IPv4 and/or IPv6 packets in the SFCM demultiplexer. The rules for validating received SFCMs are specified in <<xx>>.

*NOTE—A desirable approach is to use a specific UDP port number encapsulated in the SFCM PDU, see 52.5.1.1.5. And this number is used to identify the SFCM.*

After identifying, decapsulating, and validating the received SFCMs, the SFCM demultiplexer invokes a pause of transmission for certain traffic class for a certain period of time, as informed by the SFCM. The externally observable behavior on receiving a SFCM and a PFC should be the same. Any internal model of operating this behavior can be adopted.

## 52.5 Source Flow Control Protocol

Source flow control aware systems control forwarding elements and participate in SFC protocols and act upon the LLDP Source Flow Control TLV as specified in this clause. This includes:

a) Variables controlling operation (52.5.1).
b) SFCP procedures (52.5.2).
c) Encoding of the SFCM PDU and SFCM encapsulations (52.5.3).
d) LLDP Source Flow Control TLV (52.5.4).

### 52.5.1 Variables controlling operation

The source flow control variables control the operation of the SFC entity and the SFCP entity.

#### 52.5.1.1 SFC entity variables

Every source flow control aware system has a set of SFC entity variables to control the overall operation of SFC. These variables are included in the SFC entity managed object (12.xx). These include the following:

a) sfcMasterEnable (52.5.1.1.1).
b) sfcSFCMTransmitPriority (52.5.1.1.2).
c) sfcAddressIPv4 (52.5.1.1.3).
d) sfcAddressIPv6 (52.5.1.1.4).
e) sfcUDPPort (52.5.1.1.5).
f) sfcProxyEnable (52.5.1.1.6)

##### 52.5.1.1.1 sfcMasterEnable

A boolean value specifying whether SFC is enabled in this system. If sfcMasterEnable is FALSE all source flow control functionality is disabled; SFCMs and LLDP Source Flow Control TLVs are not generated and are ignored on receipt. If sfcMasterEnable is TRUE, the other managed objects and variables specified in the clause control the operation of SFC.

##### 52.5.1.1.2 sfcSFCMTransmitPriority

An integer specifying the priority value to be used when transmitting SFCMs from the system. The default is 6.

##### 52.5.1.1.3 sfcAddressIPv4

The IPv4 address, belonging to the system transmitting the IPv4 SFCM, used as the IPv4 source address in the IPv4 header (IETF RFC 791) of IPv4 SFCMs sent from the SFCP.

##### 52.5.1.1.4 sfcAddressIPv6

The IPv6 address, belonging to the system transmitting the IPv6 SFCM, used as the IPv6 source address in the IPv6 header (IETF RFC 8200) of IPv6 SFCMs sent from the SFCP.

##### 52.5.1.1.5 sfcUDPPort

<<It is desirable to have a well-known UDP number allocated by IANA rather than using a locally administrated value as described in the text below. The same value MUST be used by all systems in the data center network and will require configuration if a well known number is not allocated>>

The destination UDP port number in the UDP header (IETF RFC 768) of IPv4 and IPv6 SFCMs sent by the SFC aware system. The UDP port number must be selected from the range of dynamic port numbers, between 49152 and 65535, as specified in IETF RFC 6335. The port number must be currently available for use by the implementation. For example, an implementation may use UDP port 58623, if it is not currently being used by any other application in the system.

#### 52.5.1.1.6 sfcProxyEnable

A boolean value specifying whether SFC proxy is enabled in this system. If sfcProxyEnable is FALSE, SFCM Demultiplexer functionality is disabled for the bridge component. If sfcProxyEnable is TRUE, SFCM Demultiplexer functionality is enabled for the bridge component, and the other managed objects and variables specified in the clause control the operation of SFCM Demultiplexer.

<<NOTE: do we need the proxy mode to be enabled on per-port basis?>>

### 52.5.1.2 SFCP entity variables

<<These are per bridge component configuration values - none identified yet. NOTE: do we want the same set of traffic classes to have SFC enabled on every port? If so, we can define the TC enabled bit-mask here.>>

#### 52.5.1.2.1 sfcmMinInterval

A 32-bit value in nanoseconds specifying the minimum sending time interval between two consecutive SFCM constructed from the SFCP to the same target queue of the same source.

### 52.5.1.3 SFCP entity per-port variables

#### 52.5.1.3.1 sfcMonitorQueues

An 8-bit value specifying which traffic classes on the port will be monitored for congestion and potentially cause the generation of SFCM messages back to the source.

### 52.5.1.4 SFCP entity per-port per-traffic class variables

For each port and monitored queue in a source flow control aware system there is the following set of variables:

f) sfcCongesting (52.5.1.4.1)

#### 52.5.1.4.1 sfcCongesting

A boolean value that is set during the processing of a frame by the EM_UNITDATA.request (52.5.2.2) procedure. The value is set true when the monitored queue is indicated to be congested at the time the frame is processed. The variable is initialized to false.

### 52.5.1.5 SFCP entity per-stream variables

## 52.5.2 SFCP Procedures

Source flow control is implemented through the procedures of a SFCP. These include the following:

a) sfcInitialize() (52.5.2.1)
b) EM_UNITDATA.request (parameters) (52.5.2.2)
c) condTransmitSfcmPdu() (52.5.2.3)
d) pauseTimeCalc() (52.5.2.4)
e) processSfcmPdu() (52.5.2.5)

### 52.5.2.1 sfcInitialize()

<<TBD>>

### 52.5.2.2 EM_UNITDATA.request (parameters)

A SFCP offers an instance of the EISS (6.8) to the Queuing frames function (8.6.6). When called upon to enqueue a frame, the priority parameter specifies the target queue which represents the received priority of the frame. The SFCP determines if the target queue is a monitored queue by checking if the priority parameter has a corresponding bit set for the traffic classes in the sfcMonitorQueues variable.

If the corresponding bit in the sfcMonitorQueues variable has the value is 0, then the target queue is not participating in source flow control and the frame can be enqueued on the target queue with no further processing.

If the corresponding bit in the sfcMonitorQueues variable has the value is 1, then the target queue is a monitored queue and the SFCP determine if the frame is part of a flow that is creating congestion in the monitored queue. Frames that have been determined by the Congesting flow identification (52.2.1) to be creating congestion may cause a SFCM to be transmitted. The conditions by which a SFCM is transmitted are described in condTransmitSfcmPdu(52.5.2.3).

For each frame that is presented for queuing, the SFCP conditionally performs condTransmitSfcmPdu() (52.5.2.3) and buildAndSendSfcm() (52.5.2.4).

### 52.5.2.3 condTransmitSfcmPdu()

Called by the SFCP to conditionally generate and transmit a SFCM after the SFCP has determined that a frame is part of a flow that is creating congestion in the monitored queue. A SFCM will be generated and transmitted if the sfcmMinInterval has been passed since the last SFCM for the same target queue of the same source was sent. <<TBD - This function can also build the SFCM message with available parameters>>.

NOTE—Implementations may have additional conditions to restrict or allow the creation of SFCMs for adding congesting flows without risking interoperability.

The procedure performs the following:

a) If the sfcmMinInterval has been passed since the last SFCM for the same target queue of the same source was sent:

1) Call 52.5.2.4 buildAndSendSfcm()

### 52.5.2.4 buildAndSendSfcm()

This procedure is called by condTransmitSfcmPdu() to construct and send the SFCM. To properly generate the SFCM, the SFC Peer Table must be searched to retrieve the type of SFCM and the address information of the peer that will receive the SFCM. There are three formats for a SFCM; Layer-2, IPv4, and IPv6. All of the formats encapsulate a common SFCM PDU. The SFCM is constructed in the space provided by the parameters of the EM_UNITDATA.indication. The local variables holding the parameters from an EM_UNITDATA.request are expected to be loaded before the procedure is invoked, whether those parameters came from an actual EM_UNITDATA.request or were loaded from the SFC xx Table. The xx variable determines the contents of the xx field of the SFCM PDU.

The procedure performs the following:

a) Search the SFC Source Table (12.xx.2) … to obtain the ….

b) Fill the xxx field of the SFCM PDU with yyy.

c) The format of the SFCM constructed in the mac_service_data_unit of the EM_UNITDATA.indication depends upon the xxType variable as follows:

1) If the xxType is l2:

2) If the xxType is ipv4:

3) If the xxType is ipv6:

<<TBD>>

<<NOTE: The calculation of the pause time is left to be implementation specific.>>.

#### 52.5.2.5 processSfcmPdu()

The SFCM Demultiplexer (52.3.3) receives SFCMs from SFC aware systems and invokes processSfcmPdu() to process the SFCM. The procedure performs the following actions upon receipt of a SFCM:

a) The SFCM is validated according to 52.5.3.5 and is discarded if invalid.

b) Check if the SFCM reaches the destination. For end stations, compare the destination address on SFCM with the local address. For proxy mode bridges (sfcProxyEnable is TRUE), check if the destination address on SFCM matches one of the address on the forwarding table.

1) The SFCM reaches the destination, proceed to c).

2) The SFCM does not reach the destination, discard (end stations) or forward (bridges) the SFCM.

c) If the Type of the SFCM PDU is 0, <<TBD, for proxy mode bridge, calls PFC Initiator>>

#### 52.5.2.6 addSfcSource()

<<TBD: Register an entry indexed by the source address of the congesting flow for the SFC Source Table if the index does not exist.>>

#### 52.5.2.6 periodicTableCleanup()

<<TBD>>

### 52.5.3 Encoding of the SFCM PDU

This clause specifies the method of encoding source flow control message (SFCM) PDUs. There are two ways of encapsulating SFCM PDUs; a IPv4 and IPv6 SFCM PDU <<TBD - how do we know which type of SFCM to send? Must be another configuration option.>>. All SFCMs contain an integral number of octets.

The octets in a source flow control message PDU are numbered starting from 1 and increasing in the order they are put into the MSDU that accompanies a request to or indication from the instance of the MAC Internal Sublayer Service (ISS or EISS) used by a source flow control entity. The bits in an octet are numbered from 1 to 8 in order of increasing bit significance, where 1 is the LSB in the octet.

Where octets and bits within a source flow control message PDU are represented using a diagram, octets shown higher on the page than subsequent octets and octets shown to the left of subsequent octets at the same height on the page are lower numbered; bits shown to the left of other bits within the same octet are higher numbered.

Where two or more consecutive octets are represented as hexadecimal values, lower numbered octet(s) are shown to the left and each octet following the first is preceded by a hyphen, e.g., 01-80-C2-00-00-00. When consecutive octets are used to encode a binary number, the lower octet number has the more significant value. When consecutive bits within an octet are used to encode a binary number, the higher bit number has the most significant value. When bits within consecutive octets are used to encode a binary number, the lower octet number composes the more significant bits of the number. A flag is encoded as a single bit, and is set (TRUE) if the bit takes the value 1, and clear (FALSE) otherwise. The remaining bits within the octet can be used to encode other protocol fields.

### 52.5.3.1 Layer-2 SFCM PDU encapsulation

The means for identifying layer-2 encapsulated SFCM PDUs consists of two octets containing the EtherType value 89-A2 (in hexadecimal notation) shown in Table 52-1.

**Table 52-1—Source Flow Control Message EtherType**

| Name | Value |
|---|---|
| IEEE 802.1Q Source Flow Control Message (SFCM) | 89-A2 |

<<NOTE: The layer-2 encapsulated SFCM uses the same EtherType value as the layer-2 encapsulated Congestion Isolation Message shown in Table 49-1. The CIM uses Subtype value 0 while the SFCM uses Subtype value 1.>>

The layer-2 SFCM PDU encapsulation is shown in Figure

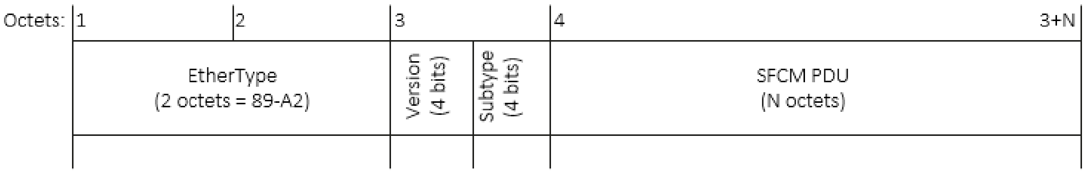


**Figure 52-4—Layer-2 SFCM encapsulation**

#### 52.5.3.1.1 Version

This field, 4 bits in length, shall be transmitted with the value 0 in this standard. If two Version fields are interpreted as unsigned binary numbers, the greater identifies the more recently defined

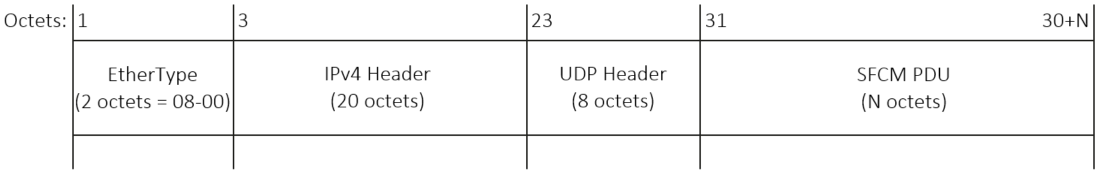
Version. The Version field occupies the most significant bits of the first octet of the layer-2 SFCM encapsulation.

**52.5.3.1.2 Subtype**

This field, 4 bits in length, shall be transmitted with the value 1 to indicate an encapsulated SFCM PDU. The Subtype field occupies the least significant 4 bits of the first octet of the layer-2 SFCM encapsulation.

52.5.3.2 IPv4 SFCM PDU encapsulation

The means of identifying IPv4 encapsulated SFCM PDUs consist of 2 octets containing the EtherType value for IPv4 packets (08-00) as well as the associated IPv4 header decoding for a UDP datagram carrying the SFCM PDU. The encoding of an IPv4 header is defined in IETF RFC 791. The destination address fields of the IPv4 header consists of 4 octets and indicates the destination of the SFCM. The value of the source address fields are taken from the sfcAddressIPv4 variable. IP options are not included in the IPv4 encapsulated SFCM PDU. The IP protocol field in the IPv4 header consists of 1 octet and identifies the UDP datagram with the value 17. The encoding of a UDP header is defined in IETF RFC 768. The source and destination port fields of the UDP header each consists of 2 octets and identifies the encapsulated SFCM PDU with the value from the sfcUDPPort variable. The IPv4 encapsulation is shown in Figure 52-5.

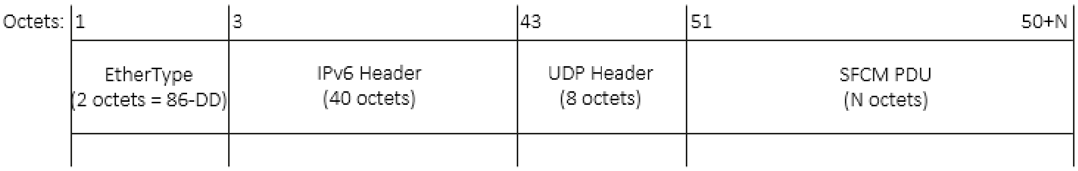


| Octets: | 1 | 3 | 23 | 31 30+N |
|---|---|---|---|---|
| | EtherType<br>(2 octets = 08-00) | IPv4 Header<br>(20 octets) | UDP Header<br>(8 octets) | SFCM PDU<br>(N octets) |

**Figure 52-5—IPv4 SFCM Encapsulation**

52.5.3.3 IPv6 SFCM PDU encapsulation

The means of identifying IPv6 layer-3 encapsulated SFCM PDUs consist of 2 octets containing the EtherType value for IPv6 packets (86-DD) as well as the associated IPv6 header decoding for a UDP datagram carrying the SFCM PDU. The encoding of an IPv6 header is defined in IETF RFC 8200. The destination address fields of the IPv6 header consists of 16 octets and indicates the destination of the SFCM. The value of the source address fields are taken from the sfcAddressIPv6 variable. IPv6 Extension Headers are not used in the IPv6 layer-3 encapsulated SFCM PDU. The next header field in the IPv6 headers indicates the upper layer protocol field and consists of 1 octet identifying the UDP datagram with the value 17. The encoding of a UDP header is defined in RFC 768. The source and destination port fields of the UDP header each consists of 2 octets and identifies the encapsulated SFCM PDU with the value from the sfcUDPPort variable. The IPv6 encapsulation is shown in Figure 52-6.



| Octets: | 1 | 3 | 43 | 51 50+N |
|---|---|---|---|---|
| | EtherType<br>(2 octets = 86-DD) | IPv6 Header<br>(40 octets) | UDP Header<br>(8 octets) | SFCM PDU<br>(N octets) |

**Figure 52-6—IPv6 SFCM Encapsulation**

52.5.3.4 Source flow control message PDU format

The format of a source flow control message (SFCM) PDU is illustrated in Figure 52-7.

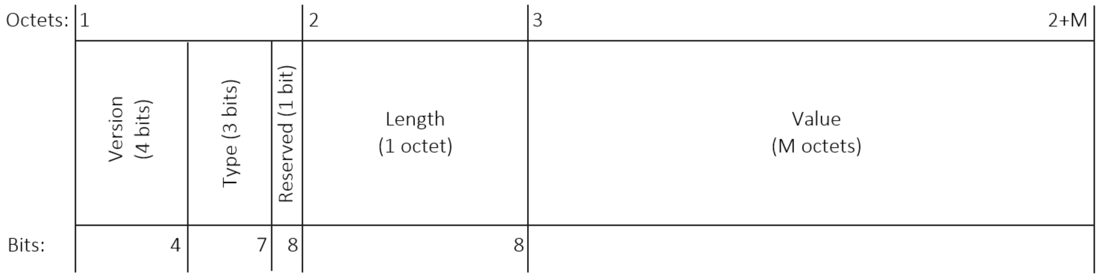


**Figure 52-7—SFCM PDU**

### 52.5.3.4.1 Version

This field, 4 bits in length, shall be transmitted with the value 0 in this standard. If two Version fields are interpreted as unsigned binary numbers, the greater identifies the more recently defined Version. The Version field occupies the most significant bits of the first octet of the SFCM PDU.

### 52.5.3.4.2 Type

The type is 3 bits in length and specifies the type used. Table 52-2 specifies the details of each TLV type and the relationship to other fields of the TLV.

### 52.5.3.4.3 Length

This field, 8 bits in length, is an unsigned integer providing the length of the value in octets.

### 52.5.3.4.4 Value

The value field contains between 0 and 255 binary octets representing the value. Each type has a specific value field definition.

### 52.5.3.4.5 TLV definitions

Table 52-2 provides the definition of the TLVs and the relationship between fields of the TLV. The layout of the value field and the bounds on the TLV length depend upon the TLV type.

**Table 52-2—TLV definitions**

| Type | Description | Length (octets) | TLV value layout |
|---|---|---|---|
| 0 | TBD | / | / |
| 1 | TBD | / | / |
| 2-6 | Reserved for future standardization | | |
| 7 | Vender-specific | | |

52.5.3.4.5.1 XXX

<<TBD>>

52.5.3.4.5.2 XXX

<<TBD>>

52.5.3.4.5.3 Reserved for future standardization

<<TBD>>

52.5.3.4.5.4 Vender-specific specific

<<TBD>>

52.5.3.5 SFCM Validation

The following condition shall not cause a received SFCM PDU to be considered invalid:

a) There are nonzero bits in the Version (52.5.3.1.1) field.
b) There are nonzero bits in the reserved fields of the SFCM PDU.

### 52.5.4 LLDP Source Flow Control TLV

<<TBD>>

52.5.4.1 LLDP Source Flow Control TLV Procedures

<<TBD>>

# Annex Z

(Temporary not for publication)

## Commentary

This is a temporary Annex, a place to record outstanding or recent technical issues and their disposition. It will be removed prior to SA Ballot. Because this is not a part of the proposed standard the editor will not accept comments on the text of this Annex itself, only on the issues raised. Discussion and resolution of the issues will result in modification of the contents.

The order of discussion of issues is intended to help the reader understand first what is the draft, secondly what may be added, and thirdly what has been considered but will not be included. In pursuit of this goal, issues where the proposed disposition is “no change” will be moved to the end. The description of issues is updated to reflect our current understanding of the problem and its solution: where it has been considered useful to retain an original comment, in whole or part, either to ensure that its author does not feel that it has not been sufficiently argued or the editor suspects there may be further aspects to the issue, that has been done as a footnote.

## Z.1 SFCM security and authentication considerations

A misbehaving congestion point, or perhaps an attacher that is masquerading as the congestion point could send an SFCM that could lead to denial of service. How do we prevent unauthorized senders to inject SFCM into the network?

Resolution: The data center is considered a single managed domain and congestion points within that domain will need to have some level of trust. Assume the switches in the network can be trusted. If NICs are controlled by the network operator, they are to be trusted as well. Stations masquerading as congestion points would need to be block or mitigated using any security mechanism that prevents an impostor from sending control frames in the network. Use ACLs on ports on the network connecting to non-trusted devices. Use mechanisms that work today.

## Acknowledgement

The author would like to express gratitude to Paul Congdon, the contributor of https://www.ieee802.org/1/files/public/docs2022/dw-congdon-individual-text-1122-v01.pdf, as this contribution references or leverages text from it.