**IEEE Std 802.6j-1995**
[Supplement to ISO/IEC 8802-6:1994
(ANSI/IEEE Std 802.6, 1994 Edition)]

# IEEE Standards for Local and Metropolitan Area Networks:

## Supplement to Distributed Queue Dual Bus (DQDB) Access Method and Physical Layer Specifications

# Connection-Oriented Service on a Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)

## IEEE Computer Society

Sponsored by the
Technical Committee on Computer Communications

*October 15,1995* *SH94230*

ISBN 1-55937-545-0

**IEEE Std 802.6j-1995**
[Supplement to ISO/IEC 8802-6:1994
(ANSI/IEEE Std 802.6, 1994 Edition)]

# IEEE Standards for Local and Metropolitan Area Networks:

## Supplement to Distributed Queue Dual Bus (DQDB) Access Method and Physical Layer Specifications

# Connection-Oriented Service on a Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)

Sponsor

**Technical Committee on Computer Communications
of the
IEEE Computer Society**

Approved March 16, 1995

**IEEE Standards Board**

**Abstract:** Enhanced Queued Arbitrated (QA) Functions, which can support applications requiring bandwidth guarantees and delay limits on a DQDB subnetwork, are specified. Connection-Oriented Convergence Functions (COCFs) using the enhanced QA Functions, which are necessary to support connection-oriented service, are also specified.
**Keywords:** Connection-Oriented Convergence Function (COCF), connection-oriented service, Distributed Queue Dual Bus (DQDB), Guaranteed Bandwidth (GBW), Queued Arbitrated (QA) Functions, Variable Bit Rate (VBR)

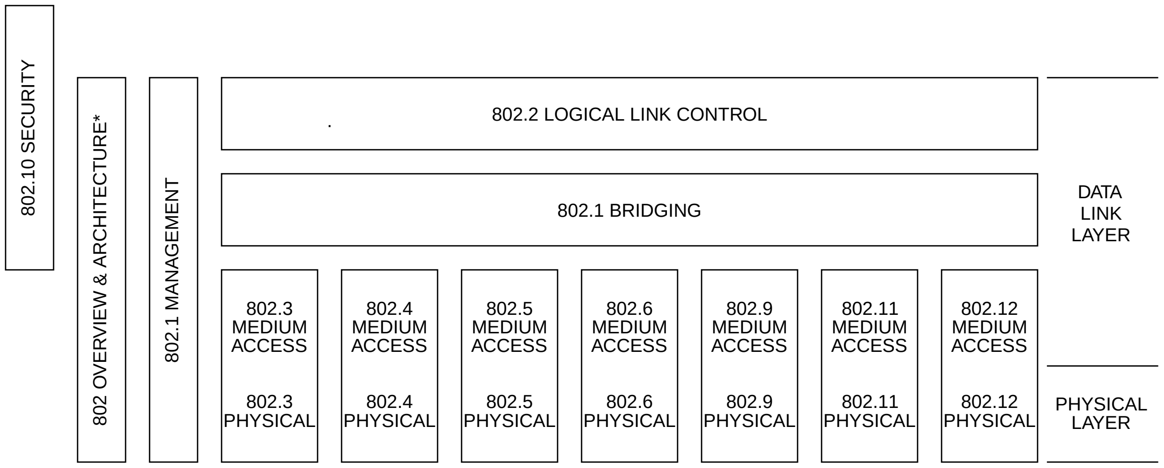
The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 1-55937-545-0

# Introduction

(This introduction is not part of IEEE Std 802.6j-1995.)

This standard is part of a family of standards for local and metropolitan area networks. The relationship between the standard and other members of the family is shown below. (The numbers in the figure refer to IEEE standard numbers.)



* Formerly IEEE Std 802.1A.

This family of standards deals with the Physical and Data Link layers as defined by the International Organization for Standardization (ISO) Open Systems Interconnection Basic Reference Model (ISO 7498 : 1984). The access standards define several types of medium access technologies and associated physical media, each appropriate for particular applications or system objectives. Other types are under investigation.

The standards defining the technologies noted above are as follows:

| | |
|---|---|
| • IEEE Std 802[1]: | Overview and Architecture. This standard provides an overview to the family of IEEE 802 Standards. This document forms part of the 802.1 scope of work. |
| • ANSI/IEEE Std 802.1B [ISO/IEC 15802-2]: | LAN/MAN Management. Defines an Open Systems Interconnection (OSI) management-compatible architecture, and services and protocol elements for use in a LAN/MAN environment for performing remote management. |
| • ANSI/IEEE Std 802.1D [ISO/IEC 10038]: | MAC Bridging. Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary. |
| • ANSI/IEEE Std 802.1E [ISO/IEC 15802-4]: | System Load Protocol. Specifies a set of services and protocol for those aspects of management concerned with the loading of systems on IEEE 802 LANs. |

[1]The 802 Architecture and Overview standard, originally known as IEEE Std 802.1A, has been renumbered as IEEE Std 802. This has been done to accommodate recognition of the base standard in a family of standards. References to IEEE Std 802.1A should be considered as references to IEEE Std 802.

|  |  |
|---|---|
| • ANSI/IEEE Std 802.2 [ISO/IEC 8802-2]: | Logical Link Control |
| • ANSI/IEEE Std 802.3 [ISO/IEC 8802-3]: | CSMA/CD Access Method and Physical Layer Specifications |
| • ANSI/IEEE Std 802.4 [ISO/IEC 8802-4]: | Token Bus Access Method and Physical Layer Specifications |
| • ANSI/IEEE Std 802.5 [ISO/IEC 8802-5]: | Token Ring Access Method and Physical Layer Specifications |
| • ANSI/IEEE Std 802.6 [ISO/IEC 8802-6]: | Distributed Queue Dual Bus Access Method and Physical Layer Specifications |
| • IEEE Std 802.9: | Integrated Services (IS) LAN Interface at the Medium Access Control (MAC) and Physical (PHY) Layers |
| • IEEE Std 802.10: | Interoperable LAN/MAN Security, *Currently approved:* Secure Data Exchange (SDE) |
| • IEEE Std 802.12: | Demand Priority Access Method, Physical Layer and Repeater Specification for 100 Mb/s Operation |

In addition to the family of standards, the following is a recommended practice for a common Physical Layer technology:

|  |  |
|---|---|
| • IEEE Std 802.7: | IEEE Recommended Practice for Broadband Local Area Networks |

The following additional working groups have authorized standards projects under development:

|  |  |
|---|---|
| • IEEE 802.11: | Wireless LAN Medium Access Control (MAC) Sublayer and Physical Layer Specifications |
| • IEEE 802.14: | Standard Protocol for Cable-TV Based Broadband Communication Network |

The reader of this standard is urged to become familiar with the complete family of standards.

## Conformance test methodology

An additional standards series, identified by the number 1802, has been established to identify the conformance test methodology documents for the 802 family of standards. Thus the conformance test documents for 802.3 are numbered 1802.3, the conformance test documents for 802.5 will be 1802.5, and so on. Similarly, ISO will use 18802 to number conformance test standards for 8802 standards.

## IEEE Std 802.6j-1995

This standard is a supplement to ISO/IEC 8802-6: 1994 [ANSI/IEEE Std 802.6, 1994 Edition]. When approved by ISO and IEC, it will be incorporated into the base text of the standard and will no longer be available separately.

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material. Information on the current revision state of this and other IEEE 802 standards may be obtained from

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE 802 committee working documents are available from

IEEE Document Distribution Service
AlphaGraphics #35 Attn: P. Thrush
10201 N. 35th Avenue
Phoenix, AZ 85051
USA

The following is a list of participants in the IEEE 802.6 Working Group at the time this standard was approved.

## Contents

CLAUSE PAGE

1. Overview .......... 1

1.1 Scope .......... 1
1.2 Organization .......... 2
1.3 Normative references .......... 2
1.4 Abbreviations and acronyms .......... 2

2. Overview of the connection-oriented service .......... 3

2.1 Compressed video applications .......... 3
2.2 Multiservice support .......... 3
2.3 Multimedia application .......... 3
2.4 Design objectives .......... 4
2.5 Properties of the connection-oriented service types .......... 4
2.6 Characteristics of priority levels .......... 5

3. Connection-oriented service definition .......... 5

3.1 CO-DATArequest .......... 5
3.2 CO-DATAindication .......... 6

4. Physical Layer service definitions .......... 6

5. Functional architecture of a DQDB node supporting connection-oriented service .......... 6

5.1 COCF blocks .......... 7
5.2 Queued Arbitrated (QA) Functions block .......... 16
5.3 Common functions .......... 22

6. DQDB Layer Protocol Data Units (PDUs) in support of connection-oriented service .......... 22

6.1 DQDB Layer PDUs in support of connection-oriented service using AAL3/4 .......... 22
6.2 DQDB Layer PDUs in support of connection-oriented service using AAL5 .......... 25
6.3 DQDB Layer PDUs in support of COCFx .......... 25

7. DQDB Layer facilities in support of connection-oriented service .......... 27

7.1 Counters .......... 27
7.2 Local reservation queue (LOCRES_I_x) .......... 28
7.3 System parameters .......... 29
7.4 Flags .......... 30

8. DQDB Layer operations .......... 31

8.1 Distributed Queue operation .......... 31
8.2 Reassembly operations .......... 39

9. DQDB Layer management .......... 39

ANNEX PAGE

Annex A (normative) Protocol Implementation Conformance Statement (PICS) proforma ........................ 40
Annex B (informative) Recommended parameters ........................................................................ 51

# IEEE Standards for Local and Metropolitan Area Networks:

## Supplement to Distributed Queue Dual Bus (DQDB) Access Method and Physical Layer Specifications

# Connection-Oriented Service on a Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)

### Revisions to ISO/IEC 8802-6: 1994 [ANSI/IEEE Std 802.6, 1994 Edition]

The contents of this document will be incorporated into ISO/IEC 8802-6 in a future edition. The clauses of this document are ordered to parallel the order of clauses in the base standard. However, to maintain internal consistency within this document, subclauses will not correspond identically with subclauses of the base standard, and the base standard will have to be renumbered upon incorporation of this document by the Working Group. This supplement is intended to be used in conjunction with ISO/IEC 8802-6: 1994.

Editing instructions in this document are provided in ***bold italics.*** It is noted whether a clause or subclause is an addition to ISO/IEC 8802-6: 1994 or replaces a clause or subclause of it in order to specify the provision of connection-oriented service.

## 1. Overview

### 1.1 Scope

This standard is a supplement to ISO/IEC 8802-6: 1994. Applications requiring bandwidth guarantees and delay limits are common and the need has been identified to support such applications on a DQDB subnetwork. Those requirements can be achieved by using either the Pre-Arbitrated (PA) functions (see IEEE Std 802.6h-1993[1]) or the Queued Arbitrated (QA) functions. This standard focuses on the use of enhanced QA functions.

The Guaranteed Bandwidth (GBW) protocol is specified as an enhancement of the QA Functions block. Moreover, Connection-Oriented Convergence Functions (COCFs) using the enhanced QA functions are necessary to support connection-oriented service. Different COCFs optimized for different types of

[1]Information on references can be found in 1.3.

applications are envisaged on top of the QA Functions block, using for example, AAL3/4 or AAL5. However, in this standard, only the COCF block using AAL3/4 is fully specified.

This standard does not specify procedures for establishing, maintaining, and releasing connections.

## 1.2 Organization

This standard in essence contains two parts, which are not explicitly separated:

a) *Enhanced QA functions.* The enhanced QA functions are necessary for providing services with bandwidth guarantees and delay limits. The enhanced QA functions are compatible with the QA functions as defined in ISO/IEC 8802-6: 1994. Due to the layered approach, the enhanced QA functions can be used by different convergence functions.

b) *COCFs.* The protocol functions and frame formats used to support different connection-oriented service types are defined. In this edition of the standard, only the Convergence Function block providing an AAL3/4 like service [ITU-T Recommendation I.363 (1993)] is fully specified.

## 1.3 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of ISO and IEC maintain registers of currently valid International Standards.

IEEE Std 802.6h-1993, IEEE Standards for Local and Metropolitan Area Networks—Supplement to Distributed Queue Dual Bus (DQDB) Access Method And Physical Layer Specifications—Isochronous Service on a Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN).[2]

ISO/IEC 11172: 1993, Information technology—Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s.[3]

ITU-T Recommendation I.363 (1993), B-ISDN ATM Adaption Layer (AAL) Specification, Rev. 1.[4]

## 1.4 Abbreviations and acronyms

For this standard, the abbreviations and acronyms in ISO/IEC 8802-6: 1994 apply. In addition, the following abbreviations and acronyms are used within this standard.

| | |
|---|---|
| CF | Convergence Function |
| COCF | Connection-Oriented Convergence Function |
| COCF3/4 | Connection-Oriented Convergence Function Block using AAL3/4 |
| COCF5 | Connection-Oriented Convergence Function Block using AAL5 |

[2]This publication is packaged with IEEE Std 802.6c-1993 and this volume is available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

[3]ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[4]ITU-T publications are available from the International Telecommunications Union, Sales Section, Place des Nations, CH-1211, Genève 20, Switzerland/Suisse. They are also available in the United States from the U.S. Department of Commerce, Technology Administration, National Technical Information Service (NTIS), Springfield, VA 22161, USA.

| | |
|---|---|
| COCFx | Connection-Oriented Convergence Functions (Stream-based) |
| CODU | Connection-Oriented Service Data Unit |
| COSU | Connection-Oriented Service User |
| DCOPDU | Derived Connection-Oriented Service Protocol Data Unit |
| FIFO | First In First Out |
| GBW | Guaranteed Bandwidth |
| ICOPDU | Initial Connection-Oriented Service Protocol Data Unit |
| MV | Maximum Value |
| PICS | Protocol Implementation Conformance Statement |
| PLCP | Physical Layer Convergence Procedure |
| TSSM | Traffic Shaping State Machine |
| VBR | Variable Bit Rate |

# 2. Overview of the connection-oriented service

***Add the following subclauses to ISO/IEC 8802-6: 1994.***

This clause provides an overview of the connection-oriented service. Although only video applications are described in more detail, other applications such as time-critical data communication or frame relay interworking are not precluded. Several applications are described that can use the Variable Bit Rate (VBR) connection-oriented service defined in this standard. The design objectives used in the development of this standard are also described. Finally, the properties of the different connection-oriented service types envisaged in this standard are described.

This standard complements the base standard, ISO/IEC 8802-6: 1994 [IEEE Std 802.6, 1994 Edition], which provides for connectionless data transfer without realtime delivery constraints.

## 2.1 Compressed video applications

Applications that are expected to be supported by this service fall into the category of low to medium bandwidth compressed digital video, i.e., 348 kbit/s to 44.2097 Mbit/s. There are two types of compressed video information, namely constant bit rate/variable image quality and variable bit rate/constant image quality. These two types have different service requirements and hence may be supported by different services on a DQDB subnetwork. The requirements of the constant bit rate/variable quality users are met by the isochronous service described in IEEE Std 802.6h-1993. The connection-oriented service types defined in this standard, in particular the stream-oriented service, are intended to meet the requirements for variable bit rate/constant image quality users.

## 2.2 Multiservice support

Multiservice support refers to the ability to simultaneously support multiple services on the same subnetwork from any given station, e.g., video and bursty data. By meeting the requirements of all other services, this type of service can be automatically supported.

## 2.3 Multimedia application

Multimedia application requires the simultaneous, integrated use of two or more of the following: voice, video, text, and graphics. This results in generating signals that are a sequence of isochronous and bursty information, which may be carried in separate channels or may be multiplexed in one channel. They are in specific timing relationships. These timing relationships are such that the information remains meaningful and intact, e.g., a voice describing a specific graphic image, or narrating a text, must be played while the image or text is on the screen. Another example is maintaining synchronization between a motion picture and a voice that accompanies it. However, when video and voice are multiplexed by the terminal into a single data stream—as is the case in ISO/IEC 11172: 1993—the network performance does not affect synchronization. When voice and video are carried in separate channels, the network performance has to be such that synchronization at the receiving end is possible whether it is carried out by the terminals or by the network. This implies that the differences in delay must be bounded.

## 2.4 Design objectives

The following design objectives are seen as the most important towards the definition of connection-oriented service capability on a DQDB subnetwork. Other design objectives may also apply.

### 2.4.1 Support of variable data rates

Connection-oriented service capability is desirable for several applications. In the near future, many likely applications are expected to use variable data rates. The connection-oriented service as defined in this standard supports any peak data rate by specific setting of system parameters.

### 2.4.2 Bounded delay, guaranteed throughput

The connection-oriented service shall be able to provide a bounded delay and/or a guaranteed throughput for transmission of Connection-Oriented Service Data Units (CODUs). This feature is ensured by the GBW protocol. A formula for calculating the maximum delay of a segment is provided in B.2.

The GBW protocol makes use of higher priorities and requires peak rate shaping. The peak rate shaping is located in the QA Functions block. Additionally, it supports mean rate shaping as a supplement to the peak rate shaping of the GBW protocol.

### 2.4.3 Conservation of bandwidth

The connection-oriented service shall not waste bandwidth that was reserved at connection setup but is not needed due to variable bit rates generated by the Connection-Oriented Service User (COSU). This will support statistical multiplexing and efficient network utilization. The GBW protocol ensures this feature since each slot used for transmission must be reserved individually.

### 2.4.4 Support of point-to-multipoint connections

In addition to point-to-point connections, the connection-oriented service shall be able to provide point-to-multipoint connections that may be used in a distribution service.

### 2.4.5 Compatibility with base standard

These enhancements of the QA functions are compatible with the QA functions of ISO/IEC 8802-6: 1994 (i.e., a node sending at priority level 0, implementing either the base standard or enhanced QA functions, can communicate with any other node).

## 2.5 Properties of the connection-oriented service types

Different service types are optimized for different applications. For example, data communication will prefer to use a packet-oriented, connection-oriented service type based on AAL3/4 or AAL5. The AAL3/4 like service (COCF3/4) conveys information regarding packet start and end points with error checking on each slot transmitted as well as an optional 32-bit Cyclic Redundancy Check (CRC) on the whole packet. The AAL5 compatible service (COCF5) uses 48 bytes payload in each slot as well as packet delimitation and error checking with a 32-bit CRC. Other applications (e.g., video applications) will prefer to use the Stream-Oriented Service (COCFx), which is also a connection-oriented service but does not expect the information stream to be demarked into packets, and which does not provide explicit error checking. Of course, the application can provide its own error checking and correcting facilities if desired.

### 2.6 Characteristics of priority levels

*Priority level 2.* This priority level guarantees its user a certain throughput and a maximum delay.

*Priority level 1.* This priority level guarantees its user a certain throughput.

*Priority level 0.* There are no throughput guarantees for this priority level.

## 3. Connection-oriented service definition

***This clause replaces 3.3 of ISO/IEC 8802-6: 1994.***

The connection-oriented service supports the transfer of CODUs between a pair of COSUs over an already established connection.[5] The size of a CODU depends on the chosen connection-oriented service type.

Note that it is intended that all connection-oriented service types described in this standard use the CO-DATArequest/indication primitives. However, since the COCF5 and COCFx are under study, different primitives for those service types may be defined if needed.

The abstract description does not constrain an implementation in any way. For example, a primitive may be implemented such that several primitive calls are necessary to forward one CODU to a Convergence Function block. This would allow pipelining.

The primitives used to describe the service are

CO-DATArequest
CO-DATAindication

### 3.1 CO-DATArequest

*Function:*

This primitive requests the transfer of a CODU over an established connection.

*Semantics of the service primitive:*

```
CO-DATArequest      (
                    CODU
                    )
```

The CODU parameter conveys a single CODU.

*When generated:*

This primitive is generated by a COSU whenever a CODU must be transferred.

[5]The procedures for establishing, maintaining, and releasing a connection are outside the scope of this standard, but will be specified by other standards. (See annex B of ISO/IEC 8802-6: 1994.)

*Effect on receipt:*

The receipt of this primitive by the DQDB Layer results in the DQDB Layer attempting to transfer the CODU over the established connection. CODUs are transferred in the same order in which they are submitted by the COSU.

*Additional comments:*

None.

### 3.2 CO-DATAindication

*Function:*

This primitive indicates the arrival of a CODU over an established connection.

*Semantics of the service primitive:*

```
CO-DATAindication   (
                    CODU
                    )
```

The CODU parameter conveys a single CODU.

*When generated:*

This primitive is generated by the DQDB Layer to deliver a CODU that has arrived over an established connection.

*Effect on receipt:*

The effect of receipt of this primitive is dependent upon the COSU. CODUs shall be delivered to the receiving COSU in the same order that they were sent by the sending COSU.

*Additional comments:*

None.

## 4. Physical Layer service definitions

The Physical Layer service definition used in support of the connection-oriented service is as described in clause 4 of ISO/IEC 8802-6: 1994.

## 5. Functional architecture of a DQDB node supporting connection-oriented service

Figure 1 describes the general architecture of a DQDB node as found in ISO/IEC 8802-6: 1994. This clause defines the Convergence Function blocks used in support of connection-oriented service. Moreover, enhancements to the QA Functions block as defined in ISO/IEC 8802-6: 1994 are included. The new and enhanced blocks covered in this standard are marked in figure 1. The COCF blocks are defined in 5.1, the enhanced QA Functions block is described in 5.2, and the Common Function blocks are defined in 5.3.

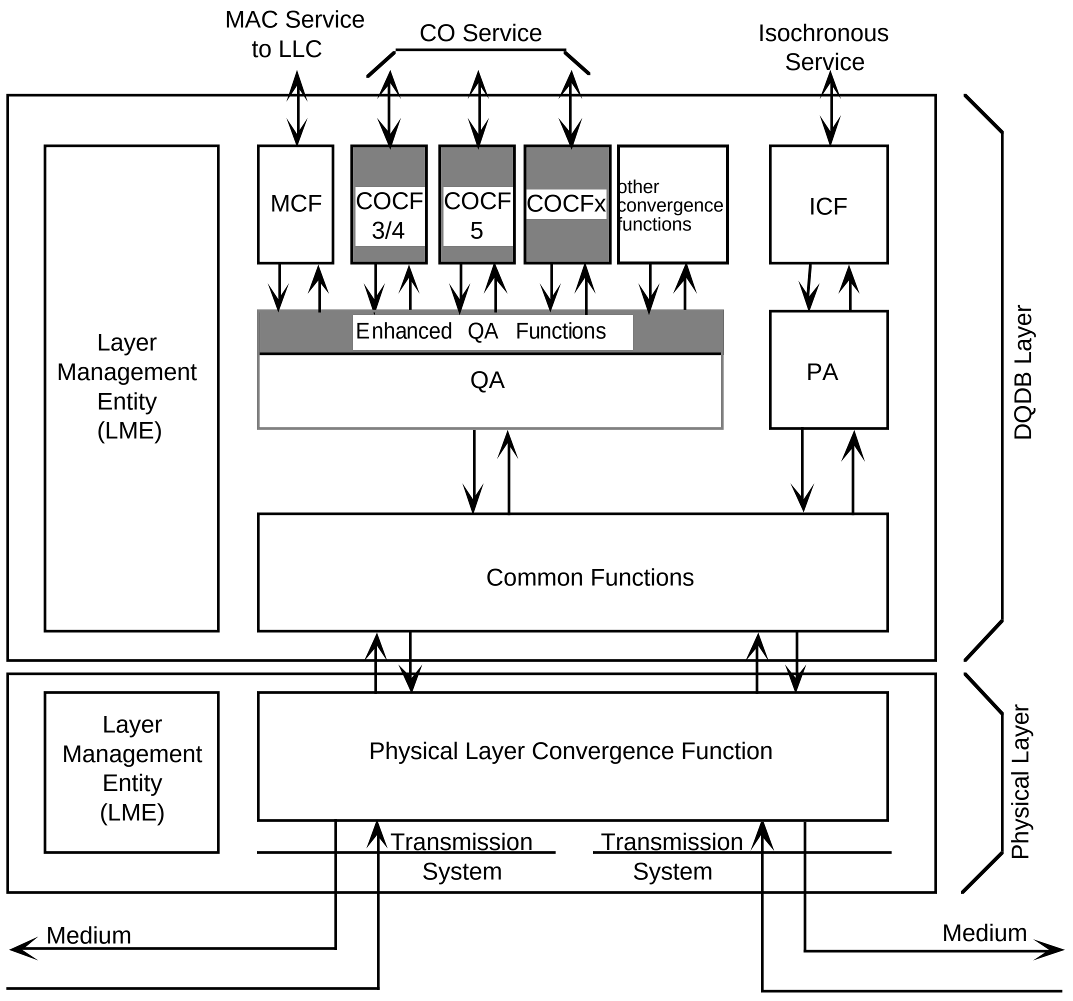


**Figure 1— DQDB node architecture**

## 5.1 COCF blocks

***This subclause replaces 5.3.1 through 5.3.1.2.1 of ISO/IEC 8802-6: 1994.***

Different COCFs described in this standard make use of the enhanced QA functions described in 5.2. Due to the independence of the Convergence Function blocks and the QA Functions block, other convergence functions may also use the service offered by the enhanced QA functions. The interface data unit exchanged between the convergence functions and the QA functions has a length of 48 octets.

The PDU of a connection-oriented service is received as the CODU in a CO-DATAindication, as defined in clause 3. Figure 2 summarizes the Transmit and Receive functions that are required of the COCF blocks and the QA Functions block to transfer a CODU between nodes. The CODU is delivered by each destination node in a CO-DATAindication, as defined in clause 3. (The bracketed numbers in the diagram refer to subclauses in this standard.)

### 5.1.1 COCF block using AAL3/4 (COCF3/4)

This subclause describes the functions performed by the DQDB Layer to support the transfer of a CODU from one DQDB node to one or more peer DQDB nodes. The protocol functions and frame formats are based on the AAL3/4 [ITU-T Recommendation I.363 (1993)].

The PDU of the connection-oriented service is received, either completely or in segmentation units of 44 octets, as the CODU in a CO-DATAindication, as defined in 3.1. Figure 2 summarizes the Transmit and Receive functions that are required of the COCF block using AAL3/4 and the QA Functions block to transfer a CODU between nodes. The CODU is delivered by each destination node in a CO-DATAindication, as defined in 3.2.

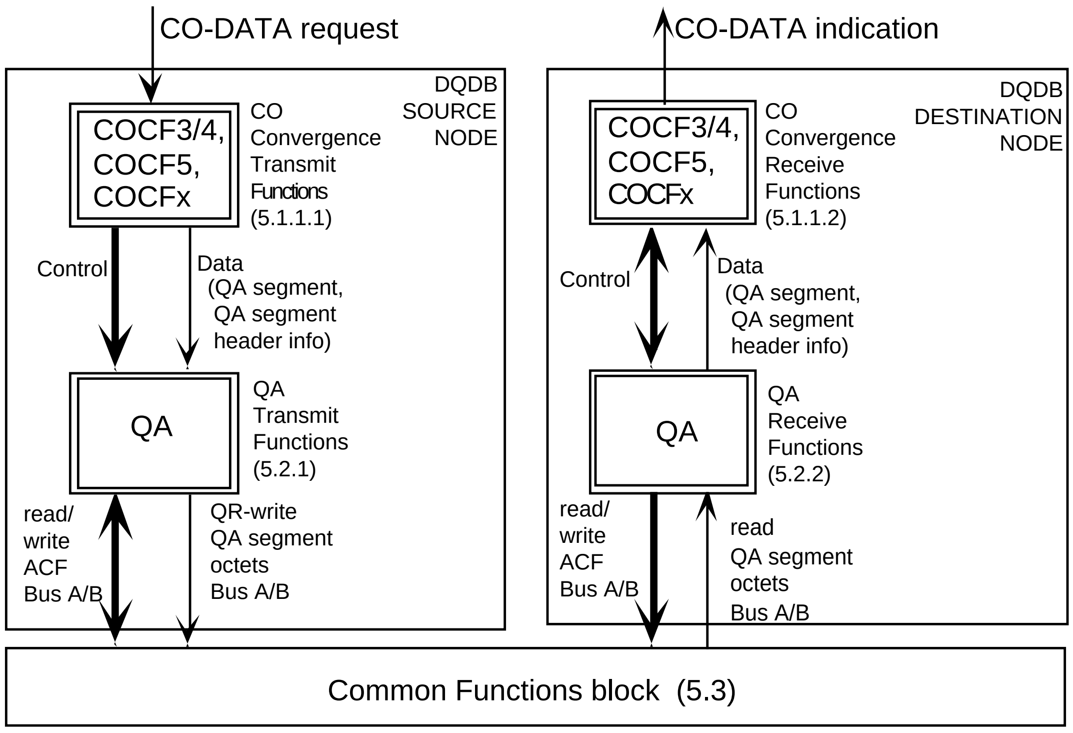

CO-DATA request
CO-DATA indication
DQDB SOURCE NODE
COCF3/4, COCF5, COCFx
CO Convergence Transmit Functions (5.1.1.1)
Control
Data (QA segment, QA segment header info)
QA
QA Transmit Functions (5.2.1)
read/ write ACF Bus A/B
QR-write QA segment octets Bus A/B
DQDB DESTINATION NODE
COCF3/4, COCF5, COCFx
CO Convergence Receive Functions (5.1.1.2)
Control
Data (QA segment, QA segment header info)
QA
QA Receive Functions (5.2.2)
read/ write ACF Bus A/B
read QA segment octets Bus A/B
Common Functions block (5.3)

**Figure 2—DQDB Layer functions to support connection-oriented service**

#### 5.1.1.1 COCF3/4 Transmit functions

This subclause specifies the functions performed by the COCF3/4 block upon the receipt of a single CODU in a CO-DATArequest, as depicted in figure 3.

The creation of an Initial Connection-Oriented Service Protocol Data Unit (ICOPDU) by addition of protocol control information to the CODU is specified in 5.1.1.1.1. The segmentation of the ICOPDU into fixed length segmentation units is specified in 5.1.1.1.2. The creation of a Derived Connection-Oriented Service Protocol Data Unit (DCOPDU) by addition of protocol information to a segmentation unit is specified in 5.1.1.1.3. How the DCOPDU is passed to the QA Functions block as a QA segment payload, along with data needed by the QA Functions block to generate the QA segment header, and other control information needed to transfer the DCOPDU, is specified in 5.1.1.1.4.

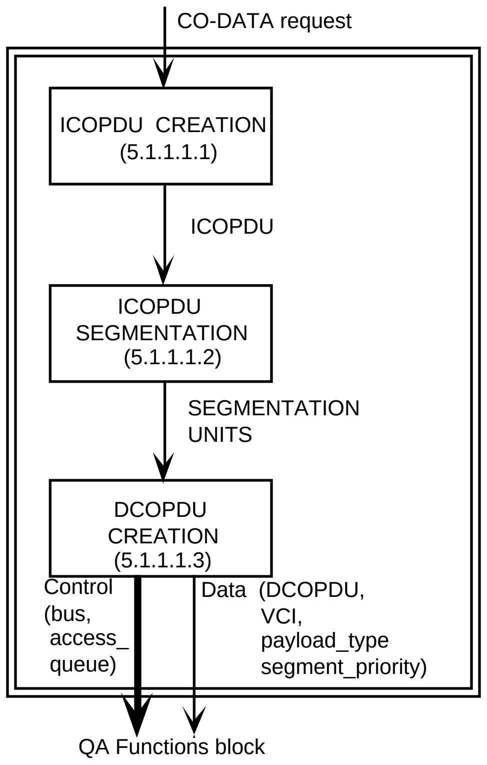

CO-DATA request
ICOPDU CREATION (5.1.1.1.1)
ICOPDU
ICOPDU SEGMENTATION (5.1.1.1.2)
SEGMENTATION UNITS
DCOPDU CREATION (5.1.1.1.3)
Control (bus, access_ queue)
Data (DCOPDU, VCI, payload_type segment_priority)
QA Functions block

**Figure 3—COCF3/4 Transmit functions**

### 5.1.1.1.1 Creation of the ICOPDU

Upon receipt of a CO-DATArequest from the sublayer above DQDB, the COCF3/4 shall perform the following operations to create an ICOPDU from the data parameter contained in the CO-DATArequest, which is the CODU. The format of the ICOPDU is given in 6.1.1.1

Steps a) through c) are for the creation of the Common PDU header of the ICOPDU.

a) Code the Reserved field of the Common PDU header to zero, according to 6.1.1.1.1.

b) Select a Beginning-End tag (BEtag) value and code the BEtag field of the Common PDU header according to 6.1.1.1.2.

   The BEtag field value shall be incremented by one (modulo 256) for sequential ICOPDUs sent by the node.

c) Code the Buffer Allocation size (BAsize) field of the Common PDU header to the number of octets contained in the CODU according to 6.1.1.1.3. If the CODU is not received completely in one CO-DATAindication by the COCF3/4, the BAsize shall be set to the maximum value (65535 octets).

Step d) is for the creation of the INFO field.

d) Code the data parameter (CODU) received in the CO_DATArequest into the INFO field, according to 6.1.1.1.4.

Step e) is for the creation of the PAD field.

e) Create the PAD field such that the length of the INFO field plus the PAD field is an integral multiple of four octets. The number of PAD octets is either 0, 1, 2, or 3, according to the following formula:

Number_PAD_octets = 3 – [(Length of INFO field + 3)(mod 4)]

Each PAD octet is coded as zero, according to 6.1.1.1.5.

Step f) is for the creation of the CRC32 field.

f) If the CRC32_CO_TX_CONTROL flag (see 7.4.1) is set to OFF, then there is no CRC32 field used for this ICOPDU. If the CRC32_CO_TX_CONTROL flag is set to ON, then the CRC32 field is coded according to 6.1.1.1.6. [Note that the fields covered by the CRC32 field are those created in steps d) and e), i.e., the INFO field and the PAD field.]

Steps g) and h) are for the creation of the Common PDU trailer of the ICOPDU.

g) Code the Reserved field of the Common PDU trailer to zero, according to 6.1.1.1.7.

h) Code the BEtag field of the Common PDU trailer with the same value as contained in the BEtag field in the Common PDU header, as per step b) above.

i) Code the Length field of the Common PDU trailer to the number of octets contained in the CODU according to 6.1.1.1.9. This value is lower than or equal to the value in the BAsize field in the Common PDU header, as per step c) above.

##### 5.1.1.1.2 Segmentation of ICOPDUs

The segmentation of the ICOPDUs is like the segmentation of the Initial MAC Protocol Data Units (IMPDUs).

***Replace all instances of "IMPDU" with "ICOPDU" in 5.1.1.1.2 (and all of its subsequent subclauses) of ISO/IEC 8802-6: 1994.***

##### 5.1.1.1.3 Creation of DCOPDUs

The COCF3/4 creates a DCOPDU from each segmentation unit created as described in 5.1.1.1.2. The type of DCOPDU created depends upon the type of the segmentation unit, and is described in items a1) and a2) below. Each DCOPDU is passed to the QA Functions block as described in 5.1.1.1.4. The order of passing DCOPDUs shall preserve the order of octets in the original ICOPDU.

a) DCOPDU from an ICOPDU of one segmentation unit

   If segmentation of the ICOPDU generates a Single Segment Message (SSM) segmentation unit, as described in 5.1.1.1.2, the COCF3/4 shall add the following protocol information to the segmentation unit to create a DCOPDU. The format of the DCOPDU is given in 6.1.1.2.

   1) Code the Segment_Type subfield of the SSM DCOPDU header with the binary value 11 (SSM), according to 6.1.1.2.

   2) Take the current value of the TX_SEQUENCE_NUM counter (see 7.2.6 of ISO/IEC 8802-6: 1994) associated with the Message Identifier (MID) selected in item a3) below and the Virtual Channel Identifier (VCI) selected in 5.1.1.1.4, and code the Sequence_Number subfield of the SSM DCOPDU header with this value according to 6.1.1.2. The value of TX_SEQUENCE_NUM shall then be incremented by one (modulo 16) for use with the next DCOPDU to be sent by the node with the same MID value and using the same VCI.

3) Select an arbitrary MID value of the set allocated to that connection (or subconnection in case of multiplexing; see 6.1.1.2).[6] Code the MID subfield of the DCOPDU header, according to 6.1.1.2.

4) Code the Payload_Length subfield of the DCOPDU trailer with the number of octets of the ICOPDU contained in the SSM segmentation unit, according to 6.1.1.2. The value of this subfield shall be any decimal number that is a multiple of 4 in the range of 12 to 44 inclusive.

5) Calculate the Payload_CRC subfield of the DCOPDU trailer, according to 6.1.1.2.

The SSM DCOPDU is then passed to the QA Functions block as described in 5.1.1.1.4.

b) DCOPDU from an ICOPDU of more than one segmentation unit

If segmentation of the ICOPDU generates more than one segmentation unit, as described in 5.1.1.1.2, the COCF3/4 shall add the following protocol control information to each segmentation unit to create a DCOPDU. The format of the DCOPDU is given in 6.1.1.2.

1) Code the Segment_Type subfield of the DCOPDU header with the appropriate binary value of either 10 for Beginning Of Message (BOM), 00 for Continuation Of Message (COM), or 01 for End Of Message (EOM), according to 6.1.1.2.

2) Take the current value of the TX_SEQUENCE_NUM counter (see 7.2.6 of ISO/IEC 8802-6: 1994) associated with the MID selected in item b3) below and the VCI selected in 5.1.1.1.4, and code the Sequence_Number subfield of the DCOPDU header with this value according to 6.1.1.2. The value of TX_SEQUENCE_NUM shall then be incremented by one (modulo 16) for use with the next DCOPDU to be sent by the node with the same MID value and using the same VCI.

3) Select an arbitrary MID value of the set allocated to that connection (or subconnection in the case of multiplexing; see 6.1.1.2). Code the MID subfield of the DCOPDU header, according to 6.1.1.2 and subject to the following condition:

   The MID value shall be the same for every DCOPDU derived from the same ICOPDU.

4) Code the Payload_Length subfield of the DCOPDU trailer with the number of octets of the ICOPDU contained in the segmentation unit, according to 6.1.1.2. For the BOM DCOPDU and any COM DCOPDU, the value of this subfield shall be decimal 44. For the EOM DCOPDU, the value of this subfield shall be any number that is a multiple of 4 in the range 4 to 44 inclusive.

5) Calculate the Payload_CRC subfield of the DCOPDU trailer according to 6.1.1.2.

The DCOPDU is then passed to the QA Functions block as described in 5.1.1.1.4.

#### 5.1.1.1.4 Transmit interactions between COCF3/4 and the QA Functions block

The interaction between a COCF3/4 and the QA Functions block for the transfer of a QA segment payload is depicted in figure 4.

The TX_BUS_SIGNAL, ACCESS_QUEUE_SIGNAL, VCI_DATA, and SP_DATA are set to the values of TX_BUS, TX_ACCESS_QUEUE_PRIORITY, TX_VCI, AND TX_SEGMENT_PRIORITY, respectively, received in an OPEN_CE_ COCF3/4 action. The PT_DATA shall be set to 00 for all QA segments passed from the COCF3/4 to the QA Functions block.

[6]This message identifier fulfills the functionality of the AAL3/4 multiplexing identification. However, for consistency, only the term Message Identifier (MID) is used in this standard.

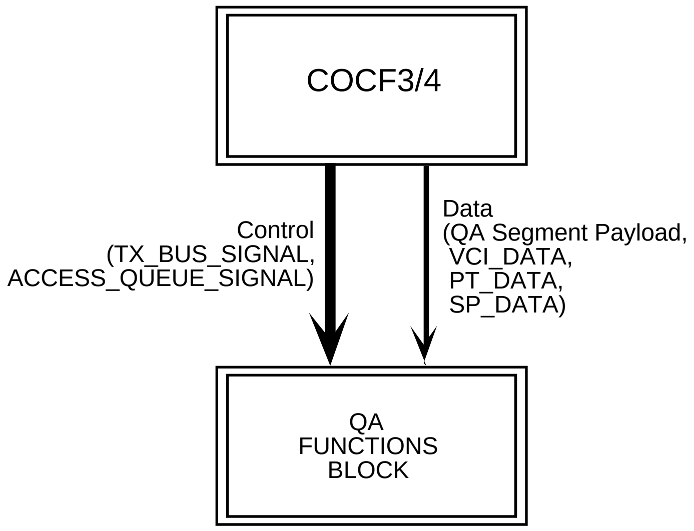


**Figure 4—Transmit interactions between COCF3/4 and the QA Functions block**

### 5.1.1.2 COCF3/4 Receive functions

The process of reassembly is used to reconstruct an ICOPDU from the segmentation units contained in DCOPDUs received by the COCF3/4. Any ICOPDU that has more than one DCOPDU derived from it, and that is currently being received and reassembled by the node, has an instance of the Reassembly State Machine (RSM) (see 8.2) associated with it to control the reassembly process.

As the DCOPDUs from different ICOPDUs destined for the node may arrive in an interspersed manner, the COCF3/4 includes the functions required to support the simultaneous operation of multiple reassembly processes, each controlled by an active instance of the RSM. For descriptive purpose in this standard, the abstract model of reassembly assumes that there is an instance of the RSM for all of the MIDs of every VCI that the COCF3/4 is programmed to receive. Therefore, each RSM is uniquely associated with a MID/VCI pair. However, the number of reassembly processes that can be supported simultaneously in a given node is an implementation choice.

This subclause specifies the functions performed by the COCF3/4 upon the receipt of a stream of DCOPDUs from the QA Functions block, as depicted in figure 5.

Subclause 5.1.1.2.1 specifies how a DCOPDU is passed from the QA Functions block along with data extracted from the header of the QA segment that carried the DCOPDU as payload, and other control information needed to process the DCOPDU. The operation of the COCF3/4 to direct the DCOPDU to the appropriate RSM is specified in 5.1.1.2.2. The function of reassembly of an ICOPDU from received DCOPDUs is specified in 5.1.1.2.3. How a reassembled ICOPDU is validated is specified in 5.1.1.2.4. How the CODU and appropriate parameters are extracted from the ICOPDU and passed by the COCF3/4 in a CO-DATAindication is specified in 5.1.1.2.5.

#### 5.1.1.2.1 Receive interactions between QA Functions block and COCF

The receive interactions between the QA Functions block and the COCF3/4 are as described in 5.3.1.2.2 of ISO/IEC 8802-6: 1994.

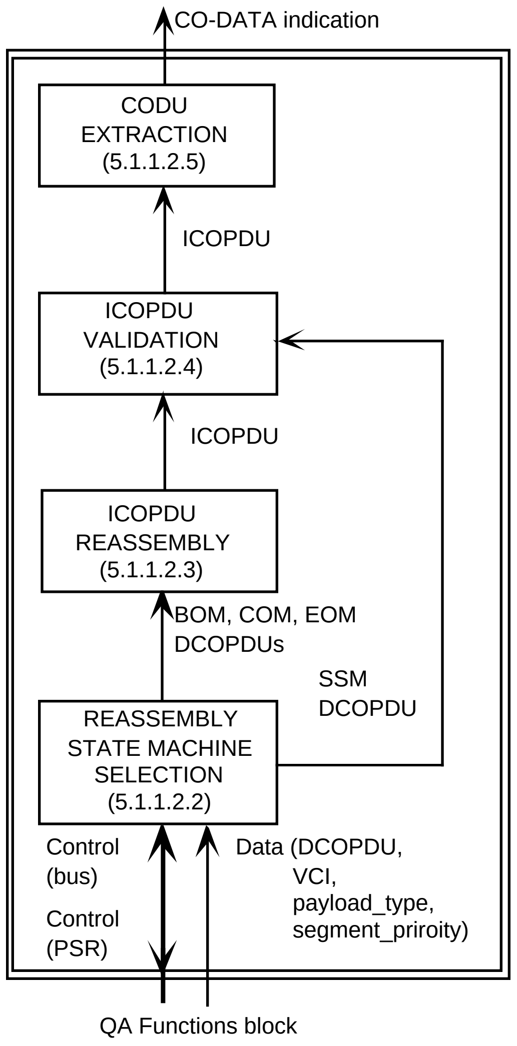

CO-DATA indication
CODU EXTRACTION (5.1.1.2.5)
ICOPDU
ICOPDU VALIDATION (5.1.1.2.4)
ICOPDU
ICOPDU REASSEMBLY (5.1.1.2.3)
BOM, COM, EOM DCOPDUs
SSM DCOPDU
REASSEMBLY STATE MACHINE SELECTION (5.1.1.2.2)
Control (bus)
Control (PSR)
Data (DCOPDU, VCI, payload_type, segment_priroity)
QA Functions block

**Figure 5—COCF3/4 Receive functions**

### 5.1.1.2.2 RSM selection

When the COCF3/4 receives a DCOPDU from the QA Functions block at a VCI that matches the VCI of an established connection, the COCF3/4 validates the correctness of the DCOPDU using the Payload_CRC subfield in the DCOPDU trailer (see 6.1.1.2). Then:

a) If the DCOPDU is valid, then the COCF3/4 shall perform one of the following operations, depending upon the value in the Segment_Type subfield of the DCOPDU header (see 6.1.1.2):

   1) Segment_Type = BOM, COM, or EOM

      If the Segment_Type of the DCOPDU is either BOM, COM, or EOM, then the COCF3/4 forwards the DCOPDU to the RSM associated with the MID/VCI pair corresponding to the VCI_DATA value and the MID contained in the DCOPDU header (see 6.1.1.2). The DCOPDU shall then be processed as described functionally in 5.1.1.2.3 and formally in 8.2.

      If the DCOPDU is destined only to this node, as determined by either condition a1 i) or ii) below, the COCF3/4 shall assert PSR_x_SIGNAL for the Bus x (x = A or B), which equals the value of RX_BUS_SIGNAL that was asserted when the DCOPDU was received by the COCF3/4.

i) If the DCOPDU is a BOM DCOPDU, and the value associated with VCI_DATA matches the VCI of an established point-to-point connection, then PSR_x_SIGNAL shall be asserted.

ii) If the DCOPDU is a COM or EOM DCOPDU, and the value of the MID subfield of the DCOPDU header (see 6.1.1.2) and the value associated with VCI_DATA matches that associated with an active reassembly process for an established point-to-point connection with that MID/VCI_DATA pair, then PSR_x_SIGNAL shall be asserted.

If neither condition a1 i) nor ii) above holds for a valid DCOPDU, then the COCF3/4 shall not assert PSR_x_SIGNAL.

2) Segment_Type = SSM

If the Segment_Type of the DCOPDU is SSM, then the DCOPDU contains a complete ICOPDU and does not need to be forwarded to a reassembly process.

The COCF3/4 shall examine the value associated with the VCI_DATA signal. Then:

i) If no VCI match is made, the SSM DCOPDU shall be discarded and no further action shall be taken.

ii) If a VCI match is made, the COCF3/4 shall extract the ICOPDU from the first N octets [where N is the value of the Payload_Length subfield in the DCOPDU trailer (see 6.1.1.2)] of the SSM segmentation unit, and pass the DCOPDU to the validation process specified in 5.1.1.2.4.

b) If the DCOPDU is not valid, the DCOPDU shall be discarded.

If either Condition A or B below holds, the COCF3/4 shall assert PSR_x_SIGNAL for the Bus x (x = A or B), which equals the value of RX_BUS_SIGNAL that was asserted when the DCOPDU was received by the COCF3/4.

If neither Condition A nor B below holds, the COCF3/4 shall not assert PSR_x_SIGNAL.

*Condition A.* The Segment_Type of the DCOPDU is BOM or SSM, and the value associated with VCI_DATA matches the VCI of an established connection.

*Condition B.* The Segment_Type of the DCOPDU is COM or EOM, the node does not support single bit error correction of the DCOPDU header via the Payload_CRC subfield in the DCOPDU trailer, and the MID value matches that associated with an active reassembly process for an established connection with that MID/VCI_DATA pair.

#### 5.1.1.2.3 Reassembly of the ICOPDU

This subclause specifies the process of reassembly of an ICOPDU from received DCOPDUs, assuming that no errors occur in the transfer of the DCOPDUs. The complete description of the instance of the RSM that controls this process is given in 8.2, and takes precedence over this subclause.

A reassembly process is activated when a BOM DCOPDU is received from the RSM selection function described in 5.1.1.2.2, and the value associated with VCI_DATA matches the VCI of an established connection. The reassembly process is associated with the value of VCI_DATA for the BOM DCOPDU and the MID value in the BOM DCOPDU header (see 6.1.1.2). The reassembly process extracts and stores the BOM segmentation unit.

If the segmentation of the ICOPDU led to more than two segmentation units (as described in 5.1.1.1.2), then the COCF3/4 should receive a series of COM DCOPDUs with the same value of VCI_DATA and the same MID value in each COM DCOPDU header as was in the BOM DCOPDU header. The value of the Sequence Number received for each COM DCOPDU should be larger than the Sequence Number of the previous COM DCOPDU (or BOM DCOPDU in the case of the first COM DCOPDU) by one (modulo 16). The reassembly process extracts the segmentation unit from each such COM DCOPDU and appends it to the previously received BOM segmentation unit and COM segmentation unit(s).

If the segmentation of the ICOPDU leads to more than one segmentation unit (as described in 5.1.1.1.2), then the COCF3/4 should receive an EOM DCOPDU with the same value of VCI_DATA and the same MID value in the EOM DCOPDU header as was in the BOM DCOPDU header. The value of the Sequence Number received for the EOM DCOPDU should be larger than the Sequence Number of the previous COM DCOPDU (or BOM DCOPDU in the case of a two segmentation unit ICOPDU) by one (modulo 16). The reassembly process extracts the number of octets from the EOM segmentation unit that contains ICOPDU data, as indicated by the value of the Payload_Length field in the DCOPDU trailer (see 6.1.1.2), and appends them to the previously received BOM segmentation unit and COM segmentation unit(s). This completes the reassembly process.

The reassembled ICOPDU is passed to the ICOPDU validation process, described in 5.1.1.2.4. The reassembly process is then stopped and disassociated from the VCI_DATA value and MID value.

#### 5.1.1.2.4 Validation of the ICOPDU

Each completely received ICOPDU, either fully contained in an SSM DCOPDU (described in 5.1.1.2.2) or from a completed reassembly process (described in 5.1.1.2.3) is validated by performing the three operations described below.

a) The value in the Length field of the Common PDU trailer (see 6.1.1.1.9) is compared with the number of octets received in the ICOPDU. The number of received ICOPDU octets should equal the value of the Length field. A mismatch shall cause the COCF3/4 to discard the ICOPDU.
b) The value in the BEtag field of the Common PDU header (see 6.1.1.1.3) is compared with the BEtag field of the Common PDU trailer (see 6.1.1.1.8). A mismatch shall cause the COCF3/4 to discard the ICOPDU.
c) If the CRC32_CO_RX_CONTROL flag (see 7.4.2) is set to ON, and if the node is able to evaluate the CRC32 field, then the CRC32 field (see 6.1.1.1.6) is used to validate the ICOPDU. The fields included in the calculation of the 32-bit CRC are the INFO field (see 6.1.1.1.4) and the PAD field (see 6.1.1.1.5). These fields are referred to as the calculation fields.

   If the CRC32 field is present and checked, a mismatch between the value received in the CRC32 field and the value calculated at the receiver shall cause the COCF3/4 to discard the ICOPDU.

If all three operations are successful, the ICOPDU is passed to the CODU extraction described in 5.1.1.2.5.

The ICOPDU validation function shall receive ICOPDUs in the same order in which the final DCOPDU (EOM DCOPDU in the case of a multiple segmentation unit ICOPDU, or SSM DCOPDU in the case of a single segmentation unit ICOPDU) is received by the COCF3/4. The ICOPDU validation function shall forward ICOPDUs to the CODU extraction in the same order as it receives them.

#### 5.1.1.2.5 Extraction of the CODU

The CODU extraction function takes a validated ICOPDU and performs the following function to create the parameters passed in a CO_DATAindication:

a) Extract the CODU contained in the INFO field to create the data parameter.

The CODU extraction function then generates a CO-DATAindication. The CODU extraction function shall generate CO-DATAindication primitives in the same order as it receives the ICOPDUs from the ICOPDU validation function.

### 5.1.2 COCF block using AAL5 (COCF5)

In addition to AAL3/4, AAL5 defines a service for data transfer in B-ISDN [see ITU-T Recommendation I.363 (1993)]. The relevant recommendation is not yet finally approved in ITU-T and hence, the support of AAL5 in DQDB is still under study in the IEEE 802.6 working group. Future editions of this standard will contain a subclause specifying a COCF5.

The PDU of the COCF5 is received, either completely or in segments of 48 octets, as the CODU in a CO-DATArequest, as defined in 3.1. Figure 2 summarizes the Transmit and Receive functions that are required of the COCF5 block and the QA Functions block to transfer a CODU between nodes. The CODU is delivered to each destination in a CO-DATAindication, as defined in 3.2.

### 5.1.3 Stream-Oriented Convergence Function block (COCFx)

The Stream-Oriented Service is intended to be used to transmit video and other material where it is not appropriate to request retransmission of missing or errored data. The structure of the data stream is left to the application to manage. Transmission requests will result in an integral number of slots sent over the network with the last one potentially unfilled, rather than waiting for more data to fill it completely.

Thus, a transmission request for a 188-octet MPEG 2 frame would result in the sending of 4 slots, since each one has an available payload of 48 octets. The last 4 octets would be empty, with a value of 0. Note that the final decision on a mapping scheme will be made when COCFx is specified.

This subclause will describe the functions performed by the DQDB Layer in support of the transfer of a stream-oriented service unit from one DQDB node to one or more peer nodes.

The PDU of the Stream-Oriented Service is received, either completely or in segments of 48 octets, as the CODU in a CO-DATArequest, as defined in 3.1. Figure 2 summarizes the Transmit and Receive functions that are required of the COCFx block and the QA Functions block to transfer a Stream-Oriented Service Data Unit between nodes. The Stream-Oriented Service Data Unit is delivered by each destination in a CO-DATAindication, as defined in 3.2.

## 5.2 Queued Arbitrated (QA) Functions block

***This subclause replaces 5.1.2 through 5.1.2.1.4.5 of ISO/IEC 8802-6: 1994.***

The QA Functions block controls the transfer in QA segments of QA segment payloads generated by Convergence Function blocks, such as the Derived MAC Protocol Data Units (DMPDUs) generated by the MAC Convergence Function (MCF) block and the DCOPDUs generated by the COCF blocks.

The relationship between the QA Functions block and the MCF block for the default connectionless VCI is defined in this standard, and requires that all nodes conforming to this standard shall be able to transmit and receive DMPDUs at the default connectionless VCI (see 6.3.1.1.1.1 of ISO/IEC 8802-6: 1994). This support shall include transmission and reception of DMPDUs on both buses with the VCI, PAYLOAD_TYPE, and SEGMENT_PRIORITY fields of the QA segment header set to the default values defined in 6.1.1 of ISO/IEC 8802-6: 1994.

The relationship between the QA Functions block and a connectionless Convergence Function block, including the MCF, for other VCI values is established by the optional DQDB Layer management CL_VCI_ADD and CL_VCI_DELETE actions (see 9.2.1 and 9.2.2 of ISO/IEC 8802-6: 1994). These actions establish the conditions under which a particular connectionless VCI may be used.

The relationship between the QA Functions block and a COCF block is established by DQDB Layer management actions. These actions establish the bus and VCI to be used to transmit and receive QA segment payloads for a COCF. The action also establishes the SEGMENT_PRIORITY and ACCESS_QUEUE_PRIORITY to be used to transmit QA segment payloads.

### 5.2.1 QA Transmit functions

This subclause specifies the functions performed by the QA Functions block upon the receipt of a single QA segment payload from a convergence function, as depicted in figure 6. How the QA segment payload is passed by the Convergence Function block to the QA Function block is specified in 5.2.1.1, along with data needed by the QA Functions block to generate the QA segment header, and other control information needed to transfer the QA segment payload. The creation of the QA segment by the addition of the protocol control information to the QA segment payload is specified in 5.2.1.2. The function of First-In-First-Out (FIFO) queueing of QA segments that cannot yet be placed in the Distributed Queue is specified in 5.2.1.3. The functional operation of the enhanced Distributed Queue is specified in 5.2.1.4. How a QA segment octet is passed to the Common Functions block is specified in 5.2.1.5.

#### 5.2.1.1 Transmit interactions between Convergence Function block and QA Functions block

The interaction between a Convergence Function block and the QA Functions block is depicted in figure 7. The specific cases for the different Convergence Function blocks (e.g., MCF block, COCF3/4) are described in the relevant subclauses of the Convergence Function block descriptions.

a) The TX_BUS_SIGNAL control indicates on which bus or buses the QA segment payload should be sent. The values that can be associated with TX_BUS_SIGNAL are:

   BUS_A, or
   BUS_B, or
   BOTH

b) The ACCESS_QUEUE_SIGNAL control indicates in which priority level access queue the QA segment used to transfer the QA segment payload should be placed. The values that can be associated with ACCESS_QUEUE_SIGNAL are 0, 1, or 2.

c) The VCI_DATA contains the value of the VCI to be placed in the header of the QA segment used to transfer the QA segment payload.

d) The PT_DATA contains the value of PAYLOAD_TYPE to be placed in the header of the QA segment used to transfer the QA segment payload.

e) The SP_DATA contains the value of SEGMENT_PRIORITY to be placed in the header of the QA segment used to transfer the QA segment payload.

Note that the interface data unit between a Convergence Function block and the QA Functions block has a length of 48 octets. This makes the QA functions independent of the Protocol Control Information (PCI) added in a convergence function.

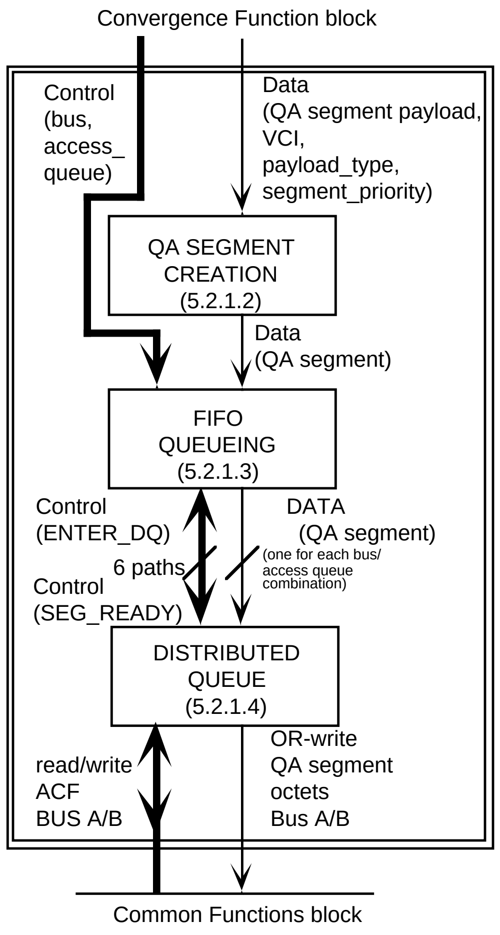

Convergence Function block
Control (bus, access_ queue)
Data (QA segment payload, VCI, payload_type, segment_priority)
QA SEGMENT CREATION (5.2.1.2)
Data (QA segment)
FIFO QUEUEING (5.2.1.3)
Control (ENTER_DQ)
DATA (QA segment)
6 paths
(one for each bus/ access queue combination)
Control (SEG_READY)
DISTRIBUTED QUEUE (5.2.1.4)
read/write ACF BUS A/B
OR-write QA segment octets Bus A/B
Common Functions block

**Figure 6—QA Functions block Transmit functions**

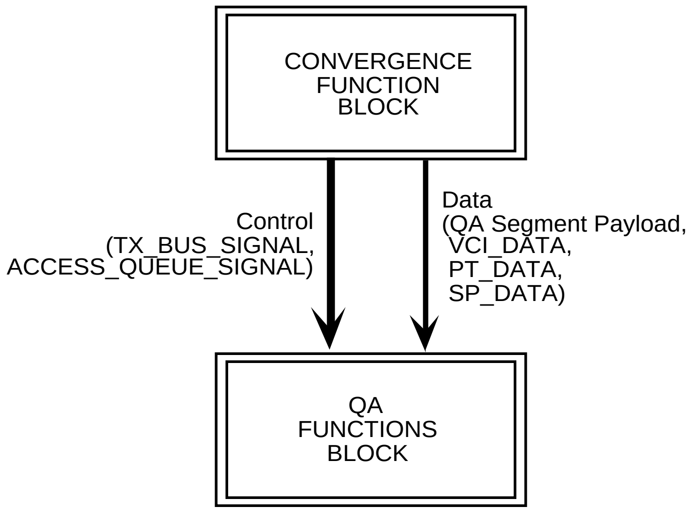

CONVERGENCE FUNCTION BLOCK
Control (TX_BUS_SIGNAL, ACCESS_QUEUE_SIGNAL)
Data (QA Segment Payload, VCI_DATA, PT_DATA, SP_DATA)
QA FUNCTIONS BLOCK

**Figure 7—Transmit interactions between Convergence Function block and QA Functions block**

### 5.2.1.2 QA segment creation

The QA segment creation is as described in 5.1.2.1.2 of ISO/IEC 8802-6: 1994.

### 5.2.1.3 FIFO queueing of QA segments

Several QA segments can be queued within the Distributed Queue for access at each node. However, the QA Functions block can accept more QA segments that cannot be placed in the Distributed Queue and are waiting for access to any given bus at any given access queue priority level. This is done by maintaining six local FIFO queues of QA segments, one for each combination of bus and access queue priority level.

Each segment created as described in 5.2.1.2 is placed in the FIFO queue associated with the value of TX_BUS_SIGNAL and ACCESS_QUEUE_SIGNAL asserted when the QA segment payload is received by the QA Functions block. If TX_BUS_SIGNAL is asserted to BOTH, then the QA segment is placed in both queues for the ACCESS_QUEUE_SIGNAL.

Whenever a QA segment is at the head of a FIFO queue (for Bus x at priority level I), the SEG_READY_I_x control signal shall be sent to the Distributed Queue as shown in figure 6. The QA segment at the head of a FIFO queue (for Bus x at priority level I) is placed in the Distributed Queue for that combination of bus and access queue priority level when the ENTER_DQ_I_x control signal shown in figure 6 indicates that a QA segment for that particular bus and priority level combination is allowed to enter the Distributed Queue.

NOTE—This standard does not enforce a specific implementation. Thus, a QA segment need not be transported physically into the Distributed Queue when accepted there. An implementation may, for example, maintain a counter that indicates how many QA segments are already accepted by the Distributed Queue, and serve as an index to the head of the FIFO queue within the internal data structures.

### 5.2.1.4 Distributed queueing of QA segments

The Distributed Queue performs the Medium Access Control (MAC) procedure for the write access of QA segments into empty QA slots. The Distributed Queue operates request counters, countdown counters, local request queue counters, and local reservation queues for each combination of bus and access queue priority level. The Distributed Queue also operates a bandwidth balancing counter for each bus.

The access of segments to the Distributed Queue is controlled by the Traffic Shaping State Machine (TSSM). The operation on its counters involves interactions with the Distributed Queue State Machine (DQSM) for priority level 0 and read operations on the bus for priority levels 1 and 2. For each combination of Bus x and access queue priority level I (I = 0,1,2), there is an instance of the TSSM, which controls the rate control counters for Bus x and access queue priority level I, denoted MEAN_CNTR_I_x and PEAK_CNTR_I_x, respectively.

Operation of the request and countdown counters for Bus x (x = A or B) involves read operations on the BUSY bit and SL_TYPE bit in the Access Control Field (ACF) (see 6.2.1 of ISO/IEC 8802-6: 1994) for all slots on Bus x and read operations on the REQUEST field in the ACF of all slots on the opposite bus, Bus y (y = B or A, respectively). For each combination of Bus x and access queue priority level I (I = 0,1,2), there is an instance of the DQSM, which controls the request and countdown counters and local reservation queues for that Bus x and access queue priority level I, denoted REQ_I_CNTR_x, CD_I_CNTR_x, and LOCRES_I_x, respectively.

A request for access to Bus x at priority level I is signalled to other nodes by a write operation performed on the REQ_I bit in the ACF of each slot passing on the opposite bus, Bus y. The write operation stops after it sets to one the first zero REQ_I bit on the opposite bus. For each instance of the DQSM there is an associated instance of the REQ Queue Machine (RQM), which operates a local request queue counter, REQ_I_Q_y, to control the sending of requests at a given access queue priority level I on the opposite bus, Bus y.

For each Bus x, there is an instance of the Bandwidth Balancing Machine (BWBM), which controls the bandwidth balancing counter for that bus, denoted BWB_CNTR_x.

This subclause gives the functional description of the Distributed Queue. The complete description of an instance of the DQSM and the associated instances of the TSSM and the RQM are given in 8.1.1, 8.1.2, and 8.1.3, respectively, and take precedence over this subclause. The description of an instance of the BWBM is given in 8.1.4 and takes precedence over this subclause.

#### 5.2.1.4.1 Node not queued to send

When no QA segment is queued for access to Bus x at priority level I, then the associated instance of the DQSM is in the IDLE state (DQ1). While the DQSM is in this state, the Distributed Queue maintains the value of the associated request counters, REQ_I_CNTR_x, at the number of the requests received at the node as unsatisfied for access by QA segments downstream on Bus x at access queue priority levels equal to or higher than I, by:

- Incrementing REQ_I_CNTR_x for any REQ_J bit set to one on the opposite bus at an access queue priority level J equal to or higher than priority level I
- Incrementing REQ_I_CNTR_x for each request made by the node itself to access Bus x at an access queue priority level higher than priority level I
- Decrementing REQ_I_CNTR_x for each slot that passes on Bus x with the BUSY bit and SL_TYPE bit set to zero, i.e., an empty QA slot
- Incrementing REQ_I_CNTR_x when BWB_CNTR_x is reset to zero

#### 5.2.1.4.2 Node queueing to send

When a TSSM is in the GO state, the Distributed Queue can accept QA segments from the FIFO queue associated with the same bus and access queue priority level (see 5.2.1.3). If there is at least one segment in that FIFO queue, then the Distributed Queue accepts the QA segment at the head of the FIFO queue.

There are two different cases:

- If the associated DQSM is in the IDLE state, i.e., there is no QA segment queued for that bus and access queue priority level, then the DQSM transfers the value of REQ_I_CNTR_x to CD_I_CNTR_x; sets REQ_I_CNTR_x to zero; increments the value of REQ_I_Q_y, the local request queue counter at access queue priority level I for the opposite bus, Bus y; and enters the ACTIVE state.
- If the associated DQSM is in the ACTIVE state, i.e., there are QA segment(s) queued for that bus and access queue priority level, then the DQSM appends the value of REQ_I_CNTR_x to the LOCRES_I_x queue; sets REQ_I_CNTR_x to zero; and increments the value of REQ_I_Q_y, the local request queue counter at access queue priority level I for the opposite bus, Bus y.

#### 5.2.1.4.3 Sending the request

The sending of the request is as described in 5.1.2.1.4.3 of ISO/IEC 8802-6: 1994.

### 5.2.1.4.4 Node queued to send

When one or more QA segments are queued for access to Bus x at priority level I, then the associated instance of the DQSM is in the ACTIVE state (DQ2). While the DQSM is in this state, the Distributed Queue maintains the value of the associated request counters, CD_I_CNTR_x, at the number of the requests for access by QA segments downstream on Bus x that have to be satisfied before the QA segment queued locally for access to Bus x at access queue priority levels can gain access. It does this by:

- Incrementing CD_I_CNTR_x for any REQ_J bit set to one on the opposite bus at an access queue priority level J higher than priority level I
- Incrementing CD_I_CNTR_x for each request made by the node itself to access Bus x at an access queue priority level higher than priority level I
- Decrementing CD_I_CNTR_x for each slot that passes on Bus x with the BUSY bit and SL_TYPE bit set to zero, i.e., an empty QA slot
- Incrementing CD_I_CNTR_x when BWB_CNTR_x is reset to zero

While in this state, the Distributed Queue also maintains the value of the associated request counters, REQ_I_CNTR_x, at the number of requests for access by QA segments downstream on Bus x at access queue priority level I that were made after the local QA segment was queued for access to Bus x at access queue priority level I. It does this by:

- Incrementing REQ_I_CNTR_x for any REQ_J bit set to one on the opposite bus at an access queue priority level J equal to priority level I

### 5.2.1.4.5 Node sending a QA segment

When the value of the CD_I_CNTR_x associated with a QA segment queued for access to Bus x at access queue priority level I equals zero, and the Distributed Queue receives an ACF on Bus x that has both the BUSY bit and the SL_TYPE bit set to zero (i.e., an empty QA slot), then the Distributed Queue function shall set the BUSY bit to one and shall OR-write the QA segment octets with the octets of the segment field of the QA slot as they pass along the bus in the Common Functions block.

If the access queue level is zero, then the following treatment of the BWBM applies. If the value of BWB_MOD is zero, then the bandwidth balancing operation is disabled. Otherwise, when the Distributed Queue changes the BUSY bit to one, it shall also increment BWB_CNTR_x, provided that the counter does not have a value of (BWB_MOD – 1), where BWB_MOD is the value of the bandwidth balancing modulus. Otherwise, if BWB_CNTR_x does have a value of (BWB_MOD – 1), then it shall be reset to zero. If BWB_MOD has a value of one, then BWB_CNTR_x shall remain at zero.

NOTE—The DQSM is allowed to enter the IDLE state immediately upon changing the BUSY bit from zero to one, i.e., another QA segment from the FIFO queue associated with the same bus and priority level 0 is allowed to enter the Distributed Queue. This is done by issuing a payment control signal for priority level 0 to the TSSM which issues an ENTER_DQ control signal in that case. The ENTER_DQ control signal allows another QA segment to enter the Distributed Queue at priority level 0.

There are two different cases:

- If there is only one segment queued in the Distributed Queue, i.e., EMPTY(LOCRES_I_x) equals TRUE, then the DQSM shall change to the IDLE state.
- If there are more segments queued in the Distributed Queue, i.e., EMPTY(LOCRES_I_x) equals FALSE, then the DQSM shall set CD_I_CNTR_x equal to POP(LOCRES_I_x) and remain in the ACTIVE state.

#### 5.2.1.5 Transmit interactions between QA Functions block and Common Functions block

The transmit interactions between the QA Functions block and the Common Functions block are as described in 5.1.2.1.5 of ISO/IEC 8802-6: 1994.

### 5.2.2 QA Receive functions

The QA Receive functions are as described in 5.1.2.2 (and all of its subsequent subclauses) of ISO/IEC 8802-6: 1994.

## 5.3 Common functions

The Common functions in support of connection-oriented service are as described in 5.4 of ISO/IEC 8802-6: 1994.

# 6. DQDB Layer Protocol Data Units (PDUs) in support of connection-oriented service

***Add the following subclauses to ISO/IEC 8802-6: 1994.***

This clause describes the PDUs to be used for connection-oriented service.

## 6.1 DQDB Layer PDUs in support of connection-oriented service using AAL3/4

### 6.1.1 Transfer of CODUs using AAL3/4

A CODU is transferred by the connection-oriented service using AAL3/4 within an ICOPDU. An ICOPDU is transferred between a peer COCF using DCOPDUs. ICOPDUs and DCOPDUs are described below.

The hierarchy of DQDB Layer PDUs to support the transfer of a CODU in an ICOPDU is summarized in diagrammatic form in 6.3.

#### 6.1.1.1 ICOPDU

An ICOPDU is constructed by adding PCI, zero to three PAD octets, and zero or four CRC32 octets to the variable length CODU. The PCI consists of a Common PDU header and a Common PDU trailer. The CODU is put into the INFO field of the ICOPDU. The format of an ICOPDU is given in table 1. The length of each field is shown in octets.

**Table 1—ICOPDU format**

| **Common PDU header** | **INFO** | **PAD** | **CRC32** | **Common PDU trailer** |
|---|---|---|---|---|
| (4 octets) | (*) | (#) | (§) | (4 octets) |
| (*) The INFO field contains the CODU. The length of the INFO field will vary between CODUs, but this standard requires all nodes to receive CODUs up to and including 65535 octets.<br>(#) The PAD field consists of either 0, 1, 2, or 3 octets. The number of PAD octets is such that the total length of the INFO plus PAD fields is an integral multiple of 4 octets.<br>(§) The CRC32 field consists of 4 octets if the crc32-option is used; otherwise, it consists of 0 octets. | | | | |

The fields contained in the Common PDU header are given in table 2; the fields contained in the Common PDU trailer are given in table 3.

**Table 2—Common PDU header format**

| Reserved | BEtag | BAsize |
|---|---|---|
| (1 octet) | (1 octet) | (2 octets) |

**Table 3—Common PDU trailer format**

| Reserved | BEtag | Length |
|---|---|---|
| (1 octet) | (1 octet) | (2 octets) |

#### 6.1.1.1.1 Reserved field of the Common PDU header

The Reserved field of the Common PDU header is coded as described in 6.5.1.1.1 of ISO/IEC 8802-6: 1994.

#### 6.1.1.1.2 Beginning-End tag (BEtag) field of the Common PDU header

The BEtag field of the Common PDU header is coded as described in 6.5.1.1.2 of ISO/IEC 8802-6: 1994.

#### 6.1.1.1.3 Buffer allocation size (BAsize) fields of the Common PDU header

The BAsize fields shall be set by the COCF3/4 to the length, in octets, of the INFO field for the transfer of an ICOPDU. If the CODU is not received completely in one CO-DATA indication by the COCF3/4, the BAsize shall be set to the maximum value (65535 octets). This standard requires all nodes to be able to receive INFO fields of length up to and including 65535 octets.

#### 6.1.1.1.4 Information (INFO) field

The INFO field contains the CODU. The length of the INFO field will vary, but this standard requires all nodes to receive CODUs up to and including 65535 octets.[7]

#### 6.1.1.1.5 PAD field

The PAD field is as described in 6.5.1.5 of ISO/IEC 8802-6: 1994.

#### 6.1.1.1.6 CRC32 field

The CRC32 field provides the capability for including a 32-bit CRC, calculated over the INFO field (see 6.1.1.1.4) and the PAD field (see 6.1.1.1.5). These are referred to below as the calculation fields.

The CRC32 is calculated using the following standard generator polynomial of degree 32:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

The CRC32 shall be the one’s complement of the sum (modulo 2) of the following:

— The remainder of $x^k * (x^{31} + x^{30} + ... + x^2 + x + 1)$ divided (modulo 2) by G(x), where k is the number of bits in the calculation fields, with

[7]The size of 65535 octets was chosen for compatibility with AAL 3/4 [ITU-T Recommendation I.363 (1993)].

— The remainder, after multiplication of the contents (treated as a polynomial) of the calculation fields by $x^{32}$, and then division (modulo 2) by G(x).

The CRC32 shall contain the coefficient in the highest term in the leftmost bit.

As a typical implementation, at the source node, the initial remainder of the division is preset to all ones and is then modified by division of the calculation fields by the generator polynomial, G(x). The one's complement of this remainder is inserted in the CRC32 field, with the most significant bit inserted first.

At the destination node(s), the initial remainder is preset to all ones. The division of the received calculation fields by the generator polynomial, G(x), results, in the absence of transmission errors, in a unique nonzero remainder value, which is the polynomial:

$$G(x) = x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^{8} + x^{6} + x^{5} + x^{4} + x^{3} + x + 1$$

#### 6.1.1.1.7 Reserved field of the Common PDU trailer

The Reserved field of the Common PDU trailer is coded as described in 6.5.1.7.1 of ISO/IEC 8802-6: 1994.

#### 6.1.1.1.8 Beginning-End tag (BEtag) field of the Common PDU trailer

The BEtag field of the Common PDU trailer is coded as described in 6.5.1.7.2 of ISO/IEC 8802-6: 1994.

#### 6.1.1.1.9 Length field of the Common PDU trailer

The Length field of the Common PDU trailer shall be set by the COCF3/4 to the length, in octets, of the INFO field. This value is less than or equal to the value inserted in the BAsize field in the Common PDU header (see 6.1.1.1.3).

### 6.1.1.2 DCOPDU

The DCOPDU uses nearly the same fields as the DMPDU.

***Replace all instances of "DMPDU" with "DCOPDU" in 6.5.2 (and all of its subsequent subclauses) of ISO/IEC 8802-6: 1994.***

***Replace the text of 6.5.2.1.3, Message Identifier (MID) subfield, of ISO/IEC 8802-6: 1994 with the following text:***

The ten-bit MID subfield is used to

— Reassemble the DCOPDU segmentation unit(s) into an ICOPDU. The source shall send the same MID in all DCOPDUs derived from a given ICOPDU.

— Distinguish the ICOPDUs of several connections multiplexed on the same VCI connection. In case of multiplexing, a connection of the set of connections multiplexed on one VCI connection (i.e., one of the connections using the same VCI) is called subconnection in this standard. The set of MIDs allocated to a subconnection is determined at connection setup. If no multiplexing is used, the default MID shall be zero.

## 6.1.2 PDU hierarchy for connection-oriented service using AAL3/4

Figures 8 a) and 8 b) give a diagrammatic representation of the relationship between the set of DQDB Layer PDUs required to transfer a CODU as far as differences to the MAC service PDU hierarchy (see 6.6 of ISO/IEC 8802-6: 1994) occur. Note that all numbers in brackets are octets. Field lengths that are given in bits are indicated specifically.

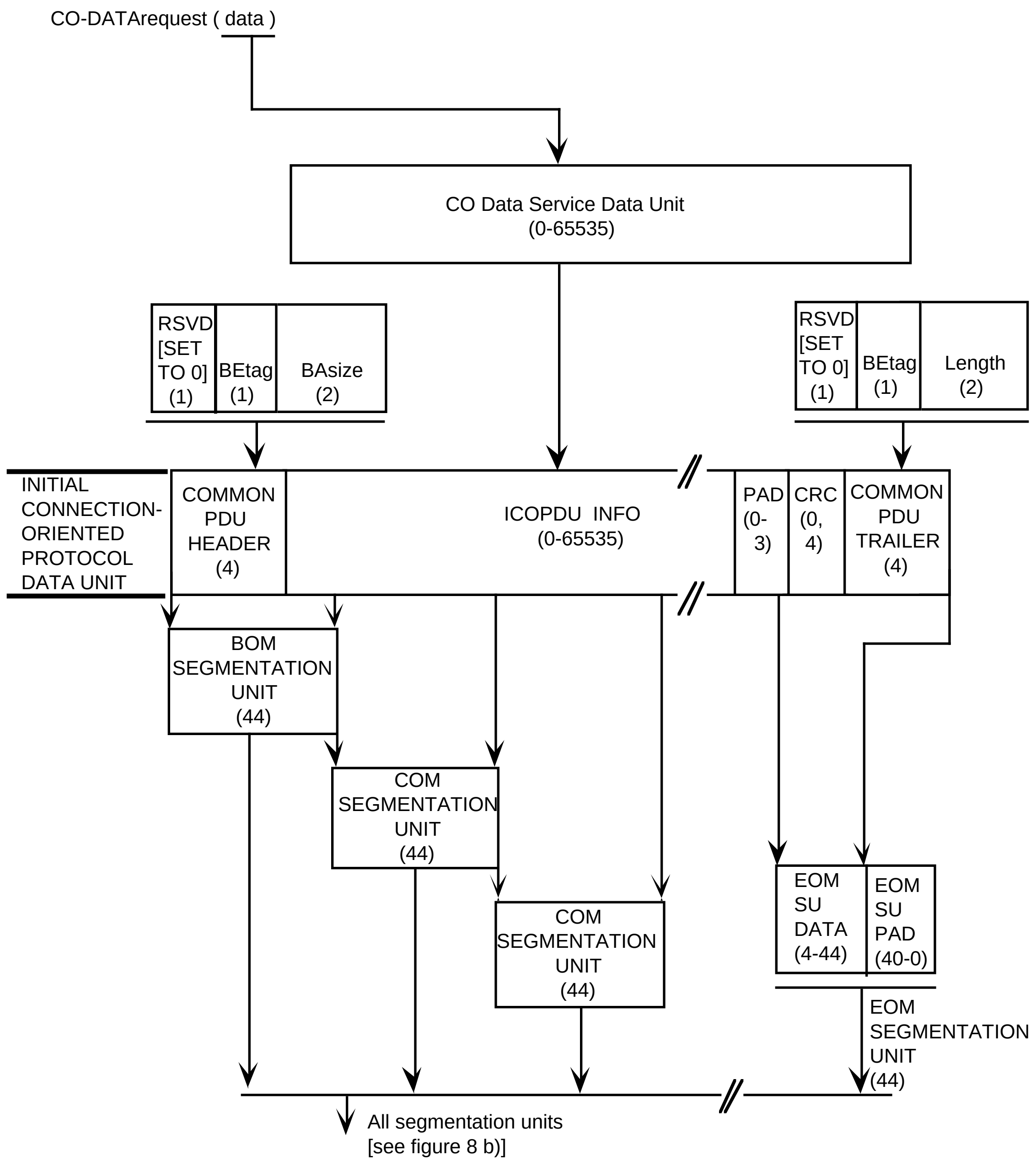


**Figure 8 a)—Connection-oriented service protocol data unit hierarchy**

## 6.2 DQDB Layer PDUs in support of connection-oriented service using AAL5

This subclause shall specify the PDUs in support of connection-oriented service using AAL5.

## 6.3 DQDB Layer PDUs in support of COCFx

This subclause shall specify the PDUs in support of stream-oriented service.

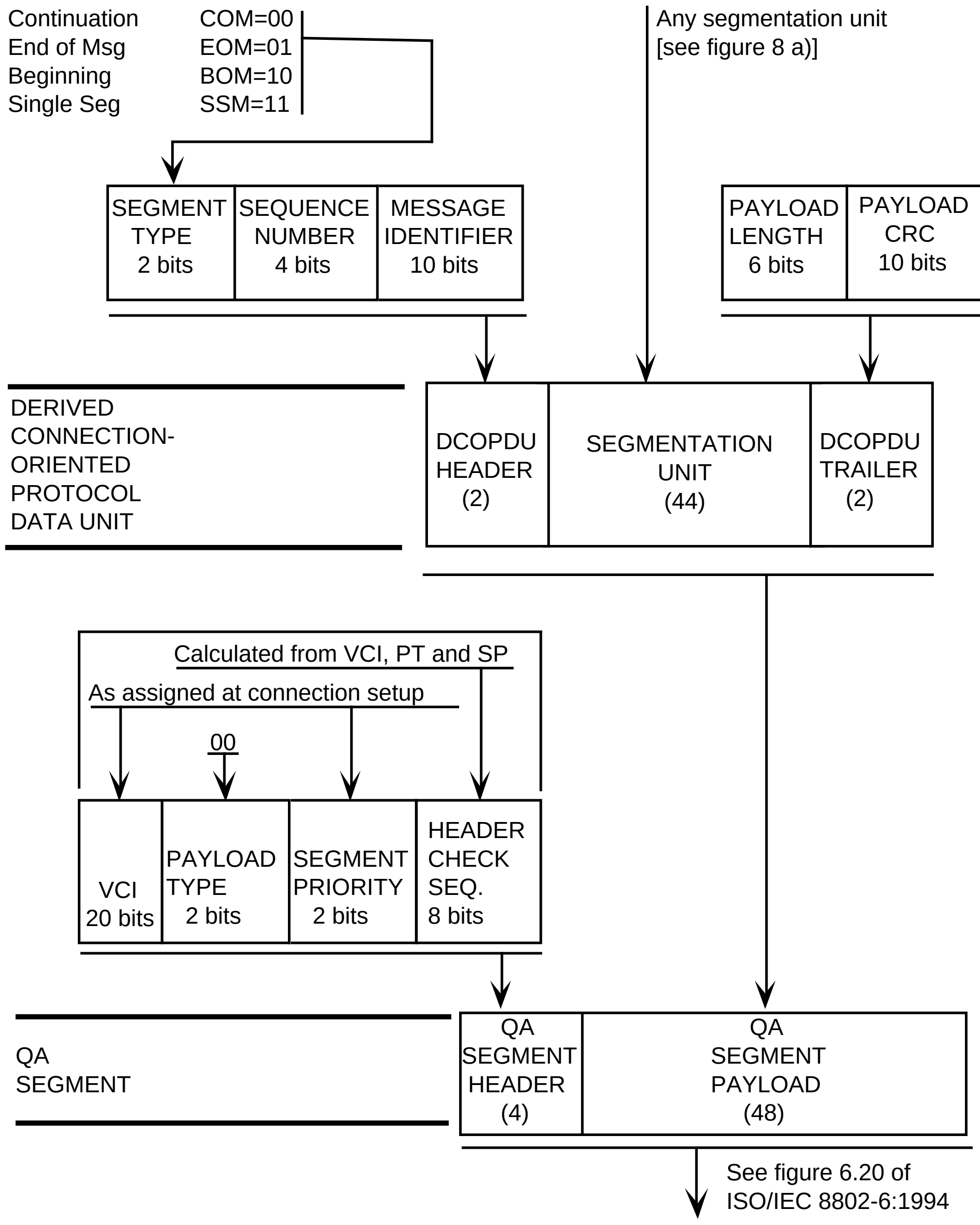


**Figure 8 b)—Connection-oriented service protocol data unit hierarchy (cont.)**

# 7. DQDB Layer facilities in support of connection-oriented service

***Add the following subclauses to clause 7 of ISO/IEC 8802-6: 1994.***

All of the DQDB Layer facilities described may be operated upon using certain DQDB Layer management interface operations described in clause 9.

The following conventions apply to the suffixes used for facilities in this clause. If the suffix is not used, then all facilities in the node which have that name are being referenced.

| | |
|---|---|
| x, | where x = A or B, refers to the forward bus at the node for the facility being described |
| y, | where x = A or B, is used in combination with x to refer to the opposite bus for the facility being described (e.g., if the forward bus is Bus A, then the opposite bus is Bus B) |
| I, | where I = 0,1,2 is used to refer to a Distributed Queue priority level |

## 7.1 Counters

### 7.1.1 Peak rate counter (PEAK_CNTR_I_x)

The QA Functions block maintains six independent counters, two at each priority level I (I = 0,1,2): one for Bus A (PEAK_CNTR_I_A) and one for Bus B (PEAK_CNTR_I_B). The PEAK_CNTR_I_x (x = A or B) counters are used by the TSSM (see 8.1.1), which performs the access control function for QA segments. Each PEAK_CNTR_I_x is associated with a PEAK_INCR_I_x, PEAK_COST_I_x, and PEAK_MAX_I_x parameter.

Each PEAK_CNTR_I_x counter operates at all times, and is used by the traffic shaping unit in queueing QA segments at priority level I in the Distributed Queue for Bus x. PEAK_CNTR_I_x is incremented at the event PAYMENT (see 8.1.2) by PEAK_INCR_I_x up to a maximum value of PEAK_MAX_I_x.

Each PEAK_CNTR_I_x counter has a minimum value of zero and a maximum value of ($2^{32}$–1). Changes that would take the PEAK_CNTR_I_x counter beyond the minimum or maximum value leave the counter at its minimum or maximum value, respectively. The PEAK_CNTR_I_x counter is initialized to one when a node is powered up.

### 7.1.2 Mean rate counter (MEAN_CNTR_I_x)

The QA Functions block maintains four independent counters, two at each priority level I (I = 1,2): one for Bus A (MEAN_CNTR_I_A) and one for Bus B (MEAN_CNTR_I_B). The MEAN_CNTR_I_x (x = A or B) counters are used by the TSSM (see 8.1.1), which performs the access control function for QA segments. Each MEAN_CNTR_I_x is associated with a MEAN_INCR_I_x, MEAN_COST_I_x, and MEAN_MAX_I_x parameter.

Each MEAN_CNTR_I_x counter operates at all times and is used by the traffic shaping unit in queueing QA segments at priority level I in the Distributed Queue for Bus x. MEAN_CNTR_I_x is incremented at the event PAYMENT (see 8.1.2) by MEAN_INCR_I_x up to a maximum value of MEAN_MAX_I_x.

Each MEAN_CNTR_I_x counter has a minimum value of zero and a maximum value of ($2^{32}$–1). Changes that would take the MEAN_CNTR_I_x counter beyond the minimum or maximum value leave the counter at its minimum or maximum value, respectively. The MEAN_CNTR_I_x counter is initialized to zero when a node is powered up.

### 7.1.3 Transmit sequence number counter (TX_SEQUENCE_NUM)

The MCF block maintains one transmit sequence number counter (TX_SEQUENCE_NUM) for each combination of MID value that is contained in the node MID page list (see 5.4.4.2.1 of ISO/IEC 8802-6: 1994) and VCI value which the node is programmed to receive on behalf of the MCF block (see 5.4.4.2.1 of ISO/IEC 8802-6: 1994). The TX_SEQUENCE_NUM counter associated with a MID/VCI pair contains the value to be inserted in the Sequence_Number subfield of the next DMPDU sent by the node that contains the associated MID and VCI values.

The COCF3/4 block maintains one transmit sequence number counter (TX_SEQUENCE_NUM) for each combination of MID value and VCI value used. The TX_SEQUENCE_NUM counter associated with a MID/VCI pair contains the value to be inserted in the Sequence_Number subfield of the DCOPDU sent by the node that contains the associated MID and VCI values.

Each TX_SEQUENCE_NUM counter has a minimum value of zero and a maximum value of ($2^4$–1). The value of the TX_SEQUENCE_NUM counter associated with a MID is set to zero (for all VCIs that the node is programmed to receive on behalf of the MCF block) when that MID is obtained by the Get Page State Machine (see 10.3.6 of ISO/IEC 8802-6: 1994) or the first time it is used by the COCF3/4. The TX_SEQUENCE_NUM counter is incremented by one (modulo 16) whenever a DMPDU or DCOPDU containing the associated MID/VCI pair is sent by the node.

## 7.2 Local reservation queue (LOCRES_I_x)

The QA Function block maintains two independent local reservation FIFO queues at each priority level I (I = 0,1,2): one for Bus A (LOCRES_I_A) and one for Bus B (LOCRES_I_B). The LOCRES_I_x queues are used by the DQSM (see 8.2.1), which performs the access control functions for QA segments. Each LOCRES_I_x is associated with a REQ_I_CNTR_x and a CD_I_CNTR_x counter.

The LOCRES_I_x queue keeps track of the position of its own segments of priority I on Bus x in the global request queue. Each LOCRES_I_x queue operates when a segment is allowed to access the global request queue (see 8.2.1) and when it already has a segment waiting for access to Bus x at priority I. The REQ_I_CNTR_x is appended to the LOCRES_I_x queue and set to zero. When a segment is sent and the LOCRES_I_x queue is not empty, the head of that queue is moved into CD_I_CNTR_x. The number of elements in the queue plus one corresponds to the number of its own segments of priority I for Bus x that are accepted by the TSSM but not sent yet.

On theses queues, the following operations are defined: APPEND, GET, and EMPTY. APPEND_I_x (N) appends an element containing the value N to the tail of the queue LOCRES_I_x. GET_I_x delivers the value of the head element of the LOCRES_I_x queue and removes the head element from the queue. EMPTY_I_x delivers a status of the LOCRES_I_x queue; TRUE indicates that there are no elements in the LOCRES_I_x queue, and FALSE indicates that there are elements in the LOCRES_I_x queue.

For descriptive purposes in this standard, the abstract model of the Distributed Queue assumes that there is a place for each QA segment that is allowed to enter the Distributed Queue. Therefore, the number of places in the local reservation queue is assumed to be unlimited. However, the number of places in the local reservation queue in a given node is an implementation choice.

All LOCRES_I_x queues are initialized to empty when the node is powered up.

## 7.3 System parameters

### 7.3.1 Income of peak rate control (PEAK_INCR_I_x)

The system parameter PEAK_INCR_I_x is associated with the access control of QA segments to the Distributed Queue. It is used as the increment for the PEAK_CNTR_I_x counter. The value of the parameter shall be in the range of zero to a Maximum Value (MV). For a bus rate of 155.52 Mbit/s, it is recommended to use an MV of 2100.[8] The value zero implies that priority level I is not used. A nonzero value of PEAK_INCR_I_x = n means that the node can use a fraction (n / PEAK_COST_I_x) of the QA slots on Bus x.

The value for PEAK_INCR_0_x shall be 1; the default value for PEAK_INCR_I_x (I = 1,2) shall be 0.

### 7.3.2 Slotcost of peak rate control (PEAK_COST_I_x)

The system parameter PEAK_COST_I_x is associated with the access control of QA segments to the Distributed Queue. It is used as a decrement for PEAK_CNTR_I_x when the station queues a segment at priority level I. The value of the parameter shall be in the range of 1 to MV (see 7.3.1). A value of PEAK_COST_I_x = n means that the node can use a fraction (PEAK_INCR_I_x / n) of the QA slots on Bus x.

The value for PEAK_COST_0_x shall be 1; the default value for PEAK_COST_I_x (I = 1 or 2) shall be MV.

### 7.3.3 Creditmax of peak rate control (PEAK_MAX_I_x)

The system parameter PEAK_MAX_I is associated with the access control of QA segments to the Distributed Queue. It is used as the maximum value for PEAK_CNTR_I_x after the node has checked its transmit queue for priority level I. The value of the parameter shall be in the range of 1 to 10·MV inclusive (see 7.3.1).[9] A value of PEAK_MAX_I_x = n means that the node can queue (n / PEAK_COST_I_x) segments for Bus x immediately if credit is available.

The value of PEAK_MAX_0_x shall be 1. The PEAK_MAX_I_x shall not be set below the following bound since a node can not get the fraction (PEAK_COST_I_x / PEAK_INCR_I_x) of the QA slots on Bus x otherwise.

$$\text{PEAK\_MAX\_I\_x} \geq (\text{PEAK\_COST\_I\_x} / \text{PEAK\_INCR\_I\_x}) \cdot \text{PEAK\_INCR\_I\_x}$$

### 7.3.4 Income of mean rate control (MEAN_INCR_I_x)

The system parameter MEAN_INCR_I_x is associated with the access control of QA segments to the Distributed Queue. It is used as the increment for the MEAN_CNTR_I_x counter. The value of the parameter shall be in the range of zero to MV inclusive (see also 7.3.1). The value zero implies that priority level I is not used. A nonzero value of MEAN_INCR_I_x = n means that the node can use a fraction (n/ MEAN_COST_I_x) of the QA slots on Bus x.

The default value for MEAN_INCR_I_x (I = 1,2) shall be zero. The value of (MEAN_INCR_I_x / MEAN_COST_I_x) shall be lower than or equal to the value of (PEAK_INCR_I_x / PEAK_COST_I_x). If mean rate control is not used, the value of MEAN_INCR_I_x shall be set to the value of PEAK_INCR_I_x.

[8]The net data rate of a 155.52 Mbit/s bus is 135.6 Mbit/s after reduction of Physical Layer Convergence Procedure (PLCP) overhead and slot overhead (5/53). The range of zero to MV was chosen to be able to vary bandwidth allocation from 0–100% of the total bandwidth; see also 7.3.2. The minimum granularity (a fraction of 1/2100 of the net data rate) equals 64 kbit/s.

[9]The value 10·MV is such that a maximum of 10 segments may be sent continuously.

### 7.3.5 Slotcost of mean rate control (MEAN_COST_I_x)

The system parameter MEAN_COST_I_x is associated with the access control of QA segments to the Distributed Queue. It is used as a decrement for MEAN_CNTR_I_x when the station queues a segment at priority level I. The value of the parameter shall be in the range of 1 to MV inclusive (see 7.3.1). A value of MEAN_COST_I_x = n means that the node can use a fraction (MEAN_INCR_I_x / n) of the QA slots on Bus x.

The default value for MEAN_COST_I_x (I = 1 or 2) shall be MV (see also 7.3.1). If mean rate control is not used, the value of MEAN_COST_I_x shall be set to the value of PEAK_COST_I_x. For selection of this value, see the condition for MEAN_INCR_I_x (7.3.4).

### 7.3.6 Creditmax of mean rate control (MEAN_MAX_I_x)

The system parameter MEAN_MAX_I is associated with the access control of QA segments to the Distributed Queue. It is used as the maximum value for MEAN_CNTR_I_x after the node has checked its transmit queue for priority level I. The value of the parameter shall be in the range of 1 to 500·MV inclusive (see also 7.3.1).

The MEAN_MAX_I_x shall not be set below the following bound since a node cannot get the fraction (MEAN_COST_I_x / PEAK_INCR_I_x) of the QA slots on Bus x otherwise.

$$\text{MEAN\_MAX\_I\_x} \geq (\text{MEAN\_COST\_I\_x} / \text{MEAN\_INCR\_I\_x}) \cdot \text{MEAN\_INCR\_I\_x}$$

MEAN_MAX_I_x/MEAN_COST_I_x is an expression for the observed interval at mean rate policing.

If mean rate control is not used, the value of MEAN_MAX_I_x shall be set to the value of PEAK_ MAX_I_x.

## 7.4 Flags

### 7.4.1 CRC32 transmission flag (CRC32_CO_TX_CONTROL)

The COCF3/4 maintains a CRC32 transmission flag (CRC32_CO_TX_CONTROL) for each active connection. The CRC32_CO_TX_CONTROL flag is used to control the operation of the CRC32 generation function in the COCF3/4. The CRC32 transmission flag shall contain one of the following values: ON or OFF.

The value ON directs the COCF3/4 to generate the CRC32 field in the ICOPDUs of that connection, as described in 5.1.1.1.1. The value OFF directs the COCF3/4 to not create a CRC32 field in the ICOPDUs of that connection.

The value of CRC32_CO_TX_CONTROL is determined when opening a connection by means of a DQDB Layer management action.

### 7.4.2 CRC32 receive flag (CRC32_CO_RX_CONTROL)

The COCF3/4 maintains a CRC32 receive flag (CRC32_CO_RX_CONTROL) for each active connection. The CRC32_CO_RX_CONTROL flag is used to control the validation of the CRC32 field in the COCF3/4.

It indicated whether the received ICOPDU contains a CRC32 field or not. The CRC32 receive flag shall contain one of the following values: ON or OFF.

The value ON directs the COCF3/4 to assume a CRC32 field in the ICOPDUs of that connection. The value OFF directs the COCF3/4 to not assume a CRC32 field in the ICOPDUs. The value of CRC32_CO_RX_CONTROL influences the behavior of the ICOPDU validation, as described in 5.1.1.2.4.

The value of CRC32_CO_RX_CONTROL is determined when opening a connection by means of a DQDB Layer management action.

# 8. DQDB Layer operations

This clause provides the description of the protocol machines used by the DQDB Layer functions to support the connection-oriented service. The functional description of the connection-oriented service functions are given in clause 5.

## 8.1 Distributed Queue operation

***This subclause replaces 8.1 of ISO/IEC 8802-6: 1994.***

The QA Functions block includes the functions required to support operation of the Distributed Queue.

The Distributed Queue operates the request, REQ_I_CNTR_x, and countdown, CD_I_CNTR_x, counters and local reservation queues, LOCRES_I_x, for each combination of priority level I (I = 0,1,2) and Bus x (x = A or B). It also operates the bandwidth balancing counter, BWB_CNTR_x, for each Bus x. The counters and queues (see 7.2.1, 7.2.2, and 7.2.5 of ISO/IEC 8802-6: 1994) are used to control the write access of QA segments to empty QA slots on the respective bus.

Operation of the request and countdown counters for Bus x involves read operations on the BUSY bit and SL_TYPE bit in the ACF (see 6.2.1 of ISO/IEC 8802-6: 1994) for all slots on Bus x and read operations on the REQUEST field in the ACF of all slots on the opposite bus, Bus y (y = B or A, respectively). This operation is controlled by the DQSM and is described in 8.2.1.

A QA segment is placed in the queue(s) indicated by the combination of the TX_BUS_SIGNAL and ACCESS_QUEUE_SIGNAL values asserted when the QA segment payload is received by the QA Functions block (see 5.2.1.3 and 5.2.1.4). Request for Bus x is signalled to other nodes by a write operation performed on the REQUEST field of slots passing on the opposite bus, Bus y. This operation is controlled by the RQM, which operates the local request queue counter, REQ_I_Q_y, and is described in 8.1.3.

Operation of the bandwidth balancing counter for Bus x is related to the sending of QA segments on Bus x and affects the value of the request and countdown counters for Bus x. This operation is controlled by the BWBM and is described in 8.1.4. Note that the bandwidth balancing mechanism applies to priority level 0 only.

### 8.1.1 Distributed Queue State Machine (DQSM)

***This subclause replaces 8.1.1 of ISO/IEC 8802-6: 1994, except for the text of Transitions 11a–c, 12, and 22a–d.***

There are six instances of the DQSM at each node: one for each possible combination of Bus x (x = A or B) and priority level I (I = 0,1,2) that can be requested by the TX_BUS_SIGNAL and ACCESS_QUEUE_SIGNAL values asserted when the QA Functions block receives the QA segment payload (priority level 2 is the highest priority level). All instances of the DQSM for a particular bus pass information between themselves about requests for access made by the node itself, called self-requests.

The generic DQSM for QA access control to Bus x at priority level I is specified in figure 9. It describes the control of the access for a QA segment to one Bus x (x = A or B), for one priority level I (I = 0,1,2). The other bus is defined as Bus y (y = B or A, respectively). There is a request counter, REQ_I_CNTR_x, a countdown counter, CD_I_CNTR_x, and a local reservation queue, LOCRES_I_x, controlled by each DQSM.

Several QA segments can be queued within the Distributed Queue for access at each node. However, the QA Functions block can accept more QA segment payloads waiting for access to Bus x at priority level I, by maintaining a local queue of QA segments that cannot be placed in the Distributed Queue. The order of QA segments waiting for access to Bus x at priority level I shall be maintained by the QA Functions block.

*Operation of the DQSM*

The DQSM for access at priority level I (I = 0,1,2) to Bus x (x = A or B) can be in one of the following states: IDLE or ACTIVE.

In both states, the DQSM shall observe Bus y for requests generated by nodes downstream on Bus x. A request at priority level J is indicated by the REQ_J bit in the ACF being set to one. The DQSM shall observe Bus x for empty QA slots, which are indicated by the BUSY bit and SL_TYPE bit of ACF both being set to zero.

NOTE—Slots are considered to be empty QA slots if they are received at the node with the BUSY bit and SL_TYPE bit being set to zero. This ensures that the correct request and countdown counter values are maintained if the highest priority segment at the node gains access to the empty QA slot and the node marks the slot busy.

The DQSM is in the IDLE state when it has no QA segment to be transferred at priority level I on Bus x. While in this state, the DQSM maintains a count of the requests generated by nodes downstream on Bus x at priority level I or higher that are still unsatisfied. The count also includes requests from the node itself for access to Bus x at higher priorities than priority I. This count is maintained in REQ_I_CNTR_x, the value of which is denoted as RQ_I in figure 9.

The DQSM is in the ACTIVE state when it has QA segments queued for transfer at priority level I on Bus x and is waiting for access to empty QA slots received on Bus x. The permission for access is controlled by a counter called CD_I_CNTR_x, the value of which is denoted as CD_I in figure 9.

**State DQ1: IDLE**

The DQSM remains in the IDLE state until it is requested to queue a QA segment for transfer on Bus x at priority level I (Transition 12). While in the IDLE state, the value of REQ_I_CNTR_x shall be maintained by actions in Transitions 11a through 11d.

*Transitions 11a–11c*

As described in 8.1.1 of ISO/IEC 8802-6: 1994.

*Transition 11d*

If a BWB_reset_x signal is received from the BWBM for Bus x, REQ_I_CNTR_x is not equal to its maximum value, and the DQSM is responsible for priority level 0, then REQ_I_CNTR_x shall be decremented by 1.

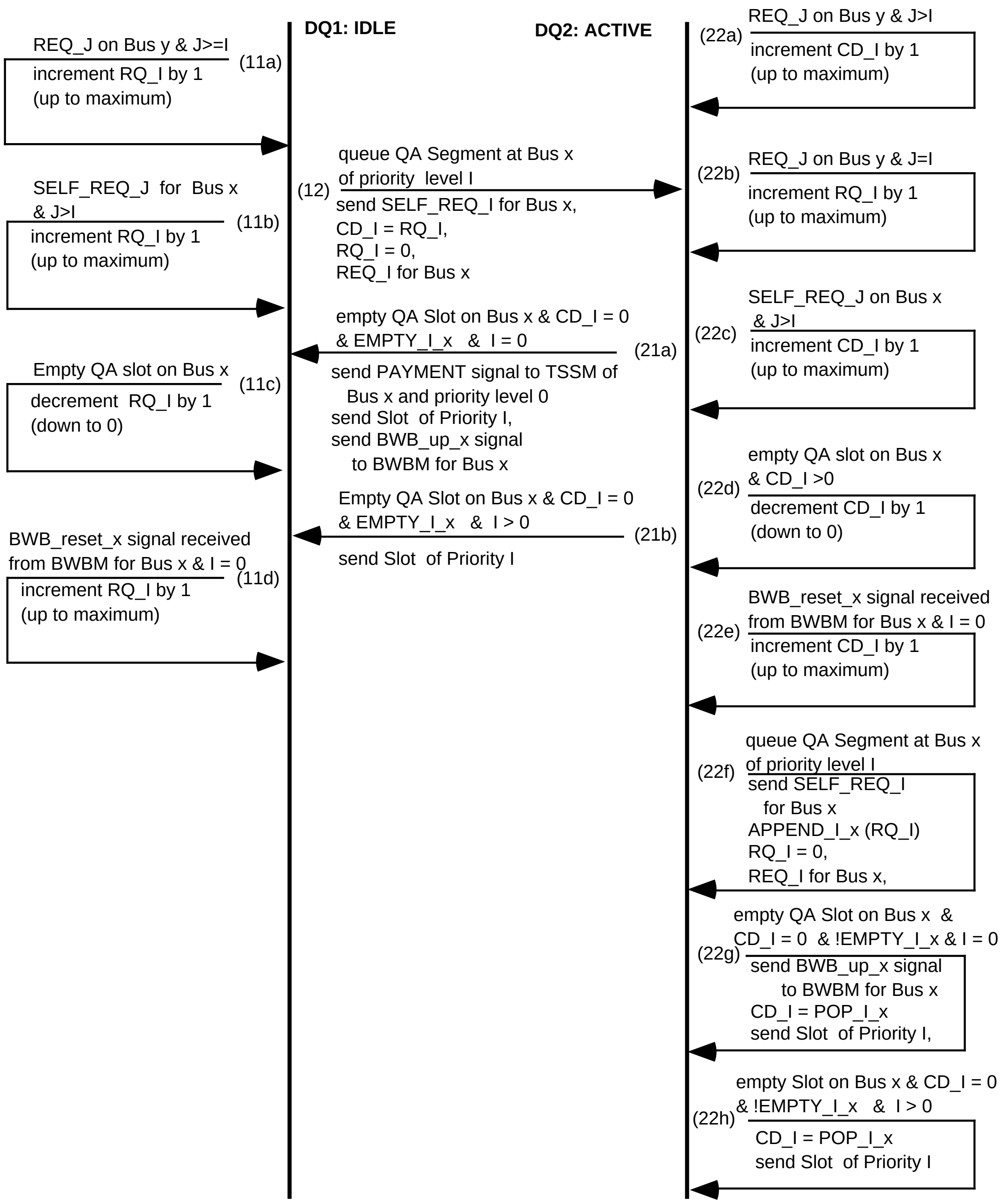


**Figure 9—DQSM for bus x at priority level I**

*Transition 12*

As described in 8.1.1 of ISO/IEC 8802-6: 1994.

**State DQ2: ACTIVE**

The DQSM remains in the ACTIVE state until the last active QA segment queued for transfer on Bus x is written to an empty QA slot (Transition 21a or 21b, respectively). While in the ACTIVE state, the value of the CD_I_CNTR_x shall be maintained by actions in Transitions 22a, 22c, 22d, 22e, 22g, and 22h and the value of the REQ_I_CNTR_x shall be maintained by the actions in Transitions 22b and 22f.

*Transition 21a*

If an empty QA slot is received on Bus x, CD_I_CNTR_x equals zero, EMPTY(LOCRES_I_x) is true, and the DQSM is responsible for priority level 0, then the DQSM shall:

a) Mark the QA slot as busy by setting the BUSY bit to one.
b) Send a PAYMENT signal to TSSM of Bus x and priority level 0.
c) Send a BWB_up_x signal to the BWBM for Bus x.
d) Start writing the QA segment into the QA slot.
e) Enter the IDLE state.

*Transition 21b*

If an empty QA slot is received on Bus x, CD_I_CNTR_x equals zero, EMPTY(LOCRES_I_x) is true, and the DQSM is responsible for priority level I (I = 1,2), then the DQSM shall:

a) Mark the QA slot as busy by setting the BUSY bit to one.
b) Start writing the QA segment into the QA slot.
c) Enter the IDLE state.

*Transitions 22a–22d*

As described in 8.1.1 of ISO/IEC 8802-6: 1994.

*Transition 22e*

If a BWB_reset_x signal is received from the BWBM for Bus x, the DQSM is responsible for priority level 0, and the CD_I_CNTR_x is not equal to its maximum value, then CD_I_CNTR_x shall be incremented by 1.

*Transition 22f*

If the DQSM is in the ACTIVE state and receives a request to queue a QA segment for transfer on Bus x at priority level I, then it shall:

a) Inform the other DQSMs within the node associated with Bus x of the self-request.
b) Call APPEND_I_x (REQ_I_CNTR_x).
c) Set the value of REQ_I_CNTR_x to zero.
d) Indicate to the associated RQM that a segment is to be sent at priority level I on Bus y.

*Transition 22g*

If an empty QA slot is received on Bus x, CD_I_CNTR_x equals zero, EMPTY(LOCRES_I_x) is false, and the DQSM is responsible for priority level 0, then the DQSM shall:

a) Mark the QA slot as busy by setting the BUSY bit to one.
b) Send a BWB_up_x signal to the BWBM for Bus x.
c) Start writing the QA segment into the QA slot.
d) Set CD_CNTR_I_x to POP(LOCRES_I_x).

*Transition 22h*

If an empty QA slot is received on Bus x, CD_I_CNTR_x equals zero, EMPTY(LOCRES_I_x) is false, and the DQSM is responsible for priority level I (I = 1,2), then the DQSM shall:

a) Mark the QA slot as busy by setting the BUSY bit to one.
b) Start writing the QA segment into the QA slot.
c) Set CD_CNTR_I_x to POP(LOCRES_I_x).

### 8.1.2 Traffic Shaping State Machine (TSSM)

***Add this subclause to 8.1 of ISO/IEC 8802-6: 1994.***

Traffic shaping is a technique to enforce certain traffic parameters. Segments violating these parameters (e.g., peak rate) are delayed but not dropped. In the case of peak rate shaping, segments are delayed until the minimum intersegment time is over. Those segments will only be dropped if no buffer space is available. For operation of the GBW protocol, peak rate shaping is necessary to guarantee each node the fraction of the bandwidth agreed upon at connection setup. Thus, the use of peak rate shaping is mandatory. Mean rate shaping is an optional feature that allows for easier handling of statistical multiplexing.

There are six instances of the TSSM at each node: one for each possible combination of Bus x (x = A or B) and priority level I (I = 0,1,2) that can be requested by the TX_BUS_SIGNAL and ACCESS_QUEUE_SIGNAL values asserted when the QA Functions block receives the QA segment payload.

There are two different types of TSSMs, one for priority level 0 and the other one for priority levels 1 and 2. The TSSM for priority level 0 uses a peak rate shaping only and keeps the behavior of the Distributed Queue for priority level 0 compatible with the specification in ISO/IEC 8802-6: 1994. The TSSM for priority levels 1 and 2 supports peak and mean rate shaping. Note that the traffic shaping for priority level 0 guarantees that DQDB stations with QA functions according to ISO/IEC 8802-6: 1994 and DQDB stations with enhanced QA functions operate in the same way in priority level 0.

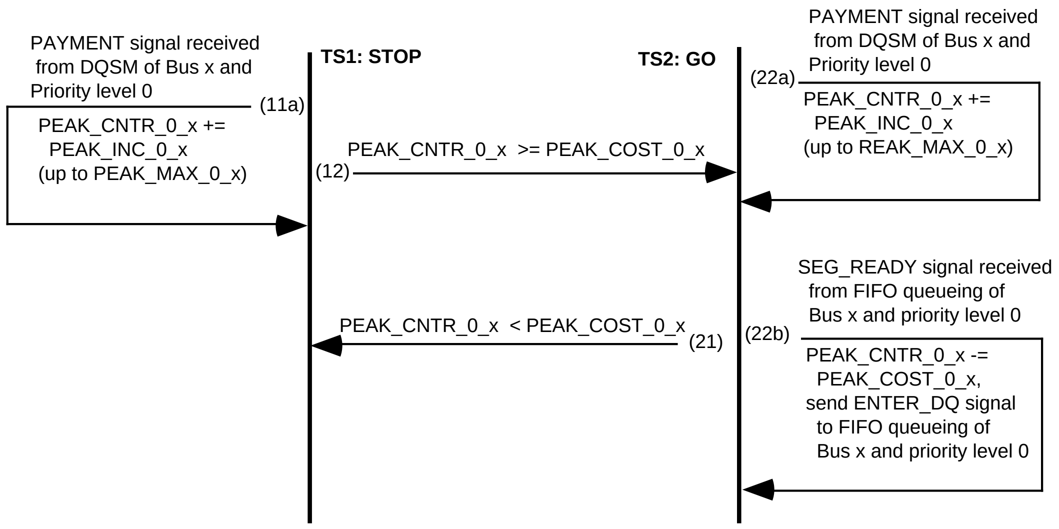
#### 8.1.2.1 Traffic shaping for priority level 0

The generic TSSM for peak rate shaping of segments being sent to Bus x at priority level 0 is specified in figure 10. It describes the control of the access for a QA segment to one Bus x (x = A or B), priority level 0. The other bus is defined as Bus y (y = B or A, respectively). There is one credit counter, PEAK_CNTR_0_x, for peak rate control, controlled by each TSSM.

Whenever a segment is stored in the FIFO queue, a SEG_READY control signal is asserted by the FIFO queue. The time until this QA segment is allowed to access the Distributed Queue is controlled by the TSSM. If the TSSM for Bus x and priority level I is in the GO state, the ENTER_DQ signal is issued and the QA segment shall be transmitted into the Distributed Queue. Otherwise, if the TSSM is in the STOP state, the QA segment that asked for access to the Distributed Queue must wait until the TSSM is in the GO state again. By this procedure, several QA segments may be queued within the Distributed Queue for access at each node. This is because the Distributed Queue may not be able to send a QA segment immediately after a segment was received from the FIFO queue and other segments may arrive from the FIFO queue before the first segment was sent.

*Operation of the TSSM*

The TSSM for access at priority level I (I = 0,1,2) to Bus x (x = A or B) can be in one of the following states: STOP or GO. In both states, the TSSM (associated with Bus x and priority level I) shall observe the transitions of the associated DQSM (priority level 0) or observe Bus y for slots arriving there (priority levels 1 and 2, respectively), and maintain the credit counters, PEAK_CNTR_0_x. If the TSSM is in the GO state, QA segments may be stored in the Distributed Queue of Bus x and priority level I. If the TSSM is in the STOP state, QA segments shall wait until the TSSM is back in the GO state.


PAYMENT signal received from DQSM of Bus x and Priority level 0
(11a)
PEAK_CNTR_0_x += PEAK_INC_0_x (up to PEAK_MAX_0_x)
TS1: STOP
TS2: GO
PEAK_CNTR_0_x >= PEAK_COST_0_x
(12)
PAYMENT signal received from DQSM of Bus x and Priority level 0
(22a)
PEAK_CNTR_0_x += PEAK_INC_0_x (up to REAK_MAX_0_x)
SEG_READY signal received from FIFO queueing of Bus x and priority level 0
PEAK_CNTR_0_x < PEAK_COST_0_x
(21)
(22b)
PEAK_CNTR_0_x -= PEAK_COST_0_x, send ENTER_DQ signal to FIFO queueing of Bus x and priority level 0

"a+=b" DENOTES THE ACTION: INCREMENT a by b.

**Figure 10—TSSM for bus x at priority level 0**

**State TS1: STOP**

The TSSM remains in the STOP state until PEAK_CNTR_0_x is greater than or equal to the value of PEAK_COST_0_x (Transition 12). While in the STOP state, the value of PEAK_CNTR_0_x shall be maintained by actions in Transition 11.

*Transition 11*

If a payment control signal arrives, PEAK_CNTR_0_x shall be incremented by PEAK_INCR_0_x. If PEAK_CNTR_0_x was incremented above PEAK_MAX_0_x, PEAK_CNTR_0_x shall be reset to PEAK_MAX_0_x.

**State TS2: GO**

The TSSM remains in the GO state until the value of PEAK_CNTR_0_x is less than PEAK_COST_0_x (Transition 21). While in the GO state, the value of PEAK_CNTR_0_x shall be maintained by actions in Transitions 22a and 22b.

*Transition 21*

If the value of PEAK_CNTR_0_x is less than PEAK_COST_0_x, the DQSM shall enter the STOP state.

*Transition 22a*

If a payment control signal arrives, PEAK_CNTR_0_x shall be incremented by PEAK_INCR_0_x. If PEAK_CNTR_0_x is incremented above PEAK_MAX_0_x, PEAK_CNTR_0_x shall be reset to PEAK_ MAX_0_x.

*Transition 22b*

If a SEG_READY control signal arrives from the FIFO queue associated with Bus x and priority level 0, PEAK_CNTR_0_x shall be decremented by PEAK_COST_0_x. The TSSM shall send a ENTER_DQ control signal to the FIFO queue associated with Bus x and priority level I.

### 8.1.2.2 Traffic shaping for priority levels 1 and 2

The generic TSSM for peak rate shaping and mean rate shaping of segments being sent to Bus x at priority level I is specified in figure 11. It describes the control of the access for a QA segment to one Bus x (x = A or B), for one priority level I (I = 1,2). The other bus is defined as Bus y (y = B or A, respectively). There are two credit counters, PEAK_CNTR_I_x for peak rate control, and MEAN_CNTR_I_x for mean rate control, controlled by each TSSM.

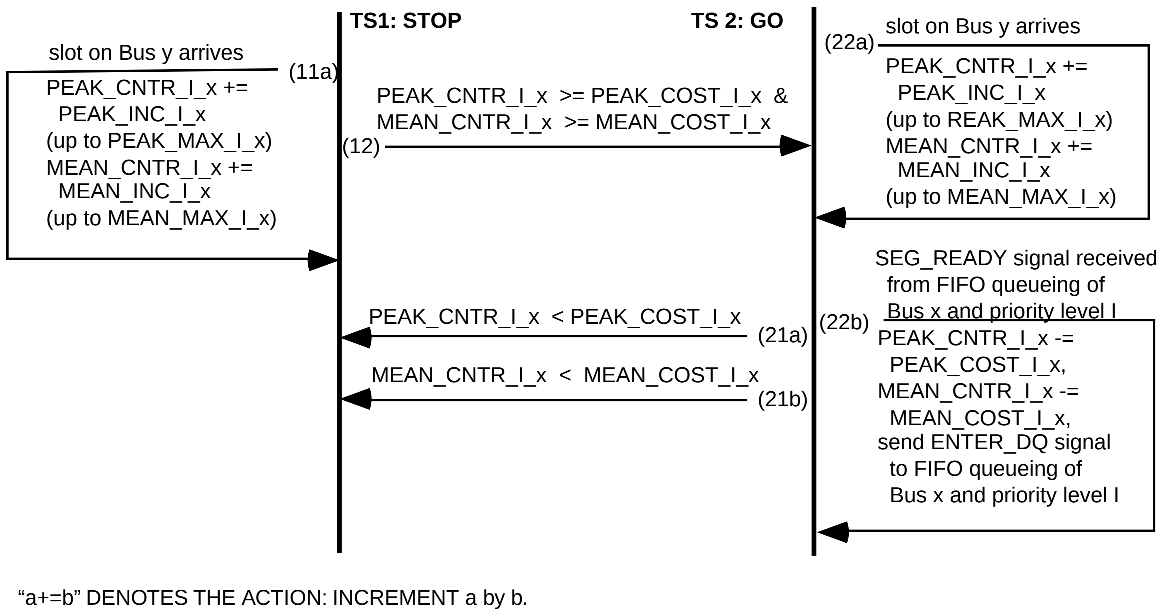

TS1: STOP
TS 2: GO
slot on Bus y arrives
(11a)
PEAK_CNTR_I_x += PEAK_INC_I_x (up to PEAK_MAX_I_x)
MEAN_CNTR_I_x += MEAN_INC_I_x (up to MEAN_MAX_I_x)
PEAK_CNTR_I_x >= PEAK_COST_I_x & MEAN_CNTR_I_x >= MEAN_COST_I_x
(12)
(22a)
slot on Bus y arrives
PEAK_CNTR_I_x += PEAK_INC_I_x (up to REAK_MAX_I_x)
MEAN_CNTR_I_x += MEAN_INC_I_x (up to MEAN_MAX_I_x)
SEG_READY signal received from FIFO queueing of Bus x and priority level I
(22b)
PEAK_CNTR_I_x -= PEAK_COST_I_x, MEAN_CNTR_I_x -= MEAN_COST_I_x, send ENTER_DQ signal to FIFO queueing of Bus x and priority level I
PEAK_CNTR_I_x < PEAK_COST_I_x
(21a)
MEAN_CNTR_I_x < MEAN_COST_I_x
(21b)
"a+=b" DENOTES THE ACTION: INCREMENT a by b.

**Figure 11—TSSM for bus x at priority level I (I = 1,2)**

Whenever a segment is stored in the FIFO queue, a SEG_READY control signal is asserted by the FIFO queue. The time until this QA segment is allowed to access the Distributed Queue is controlled by the TSSM. If the TSSM for Bus x and priority level I is in the GO state, the ENTER_DQ signal is issued and the QA segment shall be transmitted into the Distributed Queue. Otherwise, if the TSSM is in the STOP state, the QA segment that asked for access to the Distributed Queue must wait until the TSSM is in the GO state again. By this procedure, several QA segments may be queued within the Distributed Queue for access at each node. This is because the Distributed Queue may not be able to send a QA segment immediately after a segment was received from the FIFO queue and other segments may arrive from the FIFO queue before the first segment was sent.

*Operation of the TSSM*

The TSSM for access at priority level I (I = 1,2) to Bus x (x = A or B) can be in one of the following states: STOP or GO. In both states, the TSSM (associated with Bus x and priority level I) shall observe Bus y for slots arriving there (priority levels 1 and 2, respectively), and maintain the credit counters, PEAK_ CNTR_I_x and MEAN_CNTR_I_x. If the TSSM is in the GO state, QA segments may be stored in the Distributed Queue of Bus x and priority level I. If the TSSM is in the STOP state, QA segments shall wait until the TSSM is back in the GO state.

**State TS1: STOP**

The TSSM remains in the STOP state until PEAK_CNTR_I_x is greater than or equal to the value of PEAK_COST_I_x and MEAN_CNTR_I_x is greater than or equal to the value of MEAN_COST_I_x (Transition 12). While in the STOP state, the value of PEAK_CNTR_I_x and MEAN_CNTR_I_x shall be maintained by actions in Transition 11.

*Transition 11*

If a payment control signal arrives, PEAK_CNTR_I_x shall be incremented by PEAK_INCR_I_x and MEAN_CNTR_I_x shall be incremented by MEAN_INCR_I_x. If PEAK_CNTR_I_x was incremented above PEAK_MAX_I_x, PEAK_CNTR_I_x shall be reset to PEAK_MAX_I_x. If MEAN_CNTR_I_x was incremented above MEAN_MAX_I_x, MEAN_CNTR_I_x shall be reset to MEAN_MAX_I_x.

**State TS2: GO**

The TSSM remains in the GO state until the value of PEAK_CNTR_I_x is less than PEAK_COST_I_x (Transition 21a) or the value of MEAN_CNTR_I_x is less than MEAN_COST_I_x (Transition 21b). While in the GO state, the values of PEAK_CNTR_I_x and MEAN_CNTR_I_x shall be maintained by actions in Transitions 22a and 22b.

*Transition 21a*

If the value of PEAK_CNTR_I_x is less than PEAK_COST_I_x, the DQSM shall enter the STOP state.

*Transition 21b*

If the value of MEAN_CNTR_I_x is less than MEAN_COST_I_x, the DQSM shall enter the STOP state.

*Transition 22a*

If a payment control signal arrives, PEAK_CNTR_I_x shall be incremented by PEAK_INCR_I_x and MEAN_CNTR_I_x shall be incremented by MEAN_INCR_I_x. If PEAK_CNTR_I_x was incremented above PEAK_MAX_I_x, PEAK_CNTR_I_x shall be reset to PEAK_MAX_I_x. If MEAN_CNTR_I_x was incremented above MEAN_MAX_I_x, MEAN_CNTR_I_x shall be reset to MEAN_MAX_I_x.

*Transition 22b*

If a SEG_READY control signal arrives from the FIFO queue associated with Bus x and priority level I, PEAK_CNTR_I_x shall be decremented by PEAK_COST_I_x and MEAN_CNTR_I_x shall be decremented by MEAN_COST_I_x. The TSSM shall send a ENTER_DQ control signal to the FIFO queue associated with Bus x and priority level I.

### 8.1.3 REQ Queue Machine (RQM)

The RQM remains as described in 8.1.2 of ISO/IEC 8802-6: 1994.

### 8.1.4 Bandwidth Balancing Machine (BWBM)

The BWBM remains as described in 8.1.3 of ISO/IEC 8802-6: 1994.

## 8.2 Reassembly operations

***Add the following subclauses to 8.2 of ISO/IEC 8802-6: 1994.***

### 8.2.1 Reassembly operation of the COCF3/4

The reassembly operation of the COCF3/4 is similar to the reassembly operation for the MCF Function block as described in 8.2 of ISO/IEC 8802-6: 1994.

***Replace all instances of "DMPDU" with "DCOPDU" in 8.2 (and all of its subsequent subclauses) of ISO/IEC 8802-6: 1994.***

### 8.2.2 Reassembly operation of the COCF5

In future editions of this standard, this subclause will contain a specification of the reassembly operation performed by the COCF5. This operation is similar to the reassembly operation for the MCF Function block as described in 8.2 of ISO/IEC 8802-6: 1994.

# 9. DQDB Layer management

The layer management interface, the layer management procedures, and the definition of the relating managed objects are currently under development under an approved IEEE project within the IEEE 802.6 Working Group. Future editions of this standard will contain these specifications.

# Annex A[10]

(normative)

# Protocol Implementation Conformance Statement (PICS) proforma

## A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to IEEE Std 802.6j-1995 shall complete the following PICS proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use

a) By the protocol implementor, as a checklist to reduce the risk of failure to conform to the standard through oversight

b) By the supplier and acquirer, or potential acquirer, of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma

c) By the user, or potential user, of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICS proformas)

d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation

## A.2 Instructions for completing the PICS proforma

### A.2.1 Status symbols

In this PICS proforma, the following abbreviations are used in defining the status type of a feature, parameter, or capability:

| | |
|---|---|
| M | is mandatory field/function |
| O | is optional field/function |
| O.<n> | is optional field/function, but support of at least one of the group of options labeled by the same numeral <n> is required |
| C | is conditional |
| N/A | is not applicable |
| PR | is prohibited |

[10]*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

### A.2.2 Other symbols

When used in the column labeled Value, the following symbol is used:

| | |
|---|---|
| xx-yy | is from number xx to number yy inclusive |

### A.2.3 General structure of the PICS proforma

The first part of the PICS proforma, Implementation Identification and Protocol Summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire that is divided into subclauses, each containing a group of individual items. Answers to the questionnaire items are to be provided in the right-most column, either by simply marking an answer to indicate a restricted choice (usually Yes, No, or Not Applicable), or by entering a value or a set or range of values. (Note that there are some items in which two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the reason to be answered. The third column contains the reference(s) to the material that specifies the item in the main body of the standard. The remaining columns record the status of the item, i.e., whether support is mandatory, optional, prohibited, or conditional, and provide the space for the answers (see also A.2.6).

A supplier may also provide, or be required to provide, further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labelled A<i> or X<i>, respectively, for cross-referencing purposes. Where <i> is any unambiguous identification for the item (e.g., simply a numeral), there are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformance Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, according to the items listed under Major Capabilities (see A.4), a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, if this makes for easier and clearer presentation of the information. An example might be for a node that can be configured to support the default configuration control (CC_D) function.

### A.2.4 Additional information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which an (single) implementation can be set up to operate in a variety of environments and configurations; or a brief rationale, based perhaps on specific application needs, for the exclusion of features that, although optional, are nonetheless commonly present in implementations of the DQDB protocol.

References to items of Additional Information may be entered next to any answer in the questionnaire, and may be included in terms of Exception Information.

### A.2.5 Exception information

It may happen occasionally that a supplier will wish to answer an item with mandatory or prohibited status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this. Instead, the supplier shall write the missing answer into the Support column, together with an X<i> reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to IEEE Std 802.6j-1995.

NOTE—A possible reason for the situation described above is that a defect in the standard has been reported, a correction for which is expected to change the requirement not met by the standard.

### A.2.6 Conditional status

#### A.2.6.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which the status (mandatory, optional, or prohibited) that applies is dependent upon whether or not certain other items are supported, or upon the values supported for other items.

In many cases, whether or not the item applies at all is conditional in this way, as well as the status when the item does apply.

Individual conditional items are indicated by one or more conditional symbols (on separate lines) in the Status column.

A conditional item is indicated with “C:<s>” in the Status column when “<s>” is one of M, O, or O.<n> as described in A.2.1. The Predicate column will contain a predicate, “<pred>,” as described in A.2.6.2.

If the value of the predicate in any line of a conditional items is true (see A.2.6.2), the conditional item is applicable, and its status is that indicated by the status symbol following the predicate; the answer column is to be marked in the usual way. If the value of a predicate is false, no answer is required.

#### A.2.6.2 Predicates

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma: the value of the predicate is true if the item is marked as supported, and is false otherwise.

b) The logical negation symbol “NOT” prefixed to an item-reference. The value of the predicate is true if the value of the predicate formed by omitting the “NOT” symbol is false, and vice versa.

c) A BOOLEAN expression constructed by combining simple predicates using the BOOLEAN operators, AND, OR, and NOT, and parenthesis, in the usual way. The value of such a predicate is true if the BOOLEAN expression evaluates to true when the simple predicates are interpreted as described above.

## A.3 Identification

### A.3.1 Implementation identification

| | |
|---|---|
| Supplier | |
| Contact point for queries about the PICS | |
| Implementation name(s) and version(s) | |
| Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating systems; System name(s) | |
| NOTES<br>1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification.<br>2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model) | |

### A.3.2 Protocol summary, IEEE Std 802.6j-1995

| | |
|---|---|
| Identification of protocol specification | |
| Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS | |
| Protocol version(s) supported | Yes [ ] |
| Have any Exception items been required (see A.2.5)?<br>(The answer "Yes" means that the implementation does not conform to IEEE Std 802.6j-1995.) | No [ ] Yes [ ] |

| | |
|---|---|
| Date of statement: (dd/mm/yy) | |

### A.3.3 Claimed conformance to IEEE Std 802.6j-1995

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 3.1 | Are all mandatory features implemented? (The answer "No" means that the implementation does not conform to IEEE Std 802.6j-1995.) | | M | | | Yes [ ] |

## A.4 Major capabilities and features commonly used as predicates

This clause contains the conformance requirements for the major capabilities of any implementation for which compliance with this standard is claimed.

***Replace the last row of A.4.1 of ISO/IEC 8802-6: 1994 with the following five rows:***

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| CO | Implementation of a COCF | 6.1 | M | | | Y [ ] N [ ] |
| COCF3/4 | Connection-oriented service using AAL3/4 | 3, 5.1.1, 6.1 | O | | | Y [ ] N [ ] |
| COCF5 | Connection-oriented service using AAL5 | 3, 5.1.2, 6.2 | O | | | Y [ ] N [ ] |
| COCFx | Stream-oriented service | 3, 5.1.3, 6.3 | O | | | Y [ ] N [ ] |
| QA | Enhanced QA functions | 5.2, 8.1 | M | | | Y [ ] N [ ] |

To be conformant with this standard, at least one COCF block must be implemented.

NOTE—The COCF5 and COCFx will be fully specified in future editions of this standard.

## A.5 Functional architecture of a DQDB node supporting connection-oriented service

An implementation is required to conform to the functional models defined in the standard in regard to all of the externally observable effects.

***The following subclauses replace A.5.3.1 of ISO/IEC 8802-6: 1994.***

### A.5.1 Connection-Oriented Convergence Function (COCF) blocks

#### A.5.1.1 COCF block using AAL3/4 (COCF3/4)

##### A.5.1.1.1 COCF3/4 Transmit functions

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 5.1 | COCF3/4 Transmit functions | 5.1.1.1, figure 3 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 5.2 | Creation of the ICOPDU | 5.1.1.1.1, 6.1.1.1 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 5.3 | Segmentation of the ICOPDU | 5.1.1.1.2 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 5.4 | Creation of the DCOPDU | 5.1.1.1.3, 6.1.1.2 | C:M | COCF3/4 | | Y [ ] N [ ] |

#### A.5.1.1.2 COCF3/4 Receive functions

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 5.5 | COCF3/4 Receive functions | 5.1.1.2, figure 5 | C:M | COCF3/4 |  | Y [ ]   N [ ] |
| 5.6 | RSM selection | 5.1.1.2.2 | C:M | COCF3/4 |  | Y [ ]   N [ ] |
| 5.7 | Reassembly of the ICOPDU | 5.1.1.2.3, 8.2.1 | C:M | COCF3/4 |  | Y [ ]   N [ ] |
| 5.8 | Validation of the ICOPDU | 5.1.1.2.4 | C:M | COCF3/4 |  | Y [ ]   N [ ] |
| 5.9 | Extraction of the CODU | 5.1.1.2.5 | C:M | COCF3/4 |  | Y [ ]   N [ ] |

## A.5.2 Queued Arbitrated (QA) Functions Block

***The following subclause replaces A.5.1.2 of ISO/IEC 8802-6: 1994 and its subclauses.***

### A.5.2.1 QA Transmit functions

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 5.10 | QA Transmit functions | 5.2.1, figure 6 | M |  |  | Y [ ]   N [ ] |
| 5.11 | QA segment creation | 5.2.1.2 | M |  |  | Y [ ]   N [ ] |
| 5.12 | FIFO queueing of QA segments | 5.2.1.3 | M |  |  | Y [ ]   N [ ] |
| 5.13 | Distributed queueing of QA segments | 5.2.1.4, 8.1 | M |  |  | Y [ ]   N [ ] |

### A.5.2.2 QA Receive functions

The PICS for this part is as specified in A.5.1.2.2 of ISO/IEC 8802-6: 1994.

## A.5.3 Common Functions block

The PICS for this part is as specified in A.5.4 of ISO/IEC 8802-6: 1994.

## A.6 DQDB Layer PDUs used in support of connection-oriented service

***Add the following clause to clause A.6 of ISO/IEC 8802-6: 1994.***

This clause specifies the formats of the data units as a sequence of fields.

## A.6.1 Transfer of CODU using AAL3/4

### A.6.1.1 Initial Connection-Oriented Service Protocol Data Unit (ICOPDU)

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 6.1 | ICOPDU format | 6.1.1.1, Tables1, 2, and 3 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.2 | Set Reserved field of Common PDU header | 5.1.1.1.1, 6.1.1.1.1 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.3 | Interpret Reserved field of Common PDU header | 6.1.1.1.1 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.4 | Set BEtag field | 5.1.1.1.1, 6.1.1.1.2 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.5 | Interpret BEtag field | 6.1.1.1.2 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.6 | Set BAsize field | 5.1.1.1.1, 6.1.1.1.3 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.7 | Interpret BAsize field | 6.1.1.1.3 | C:O | COCF3/4 | | Y [ ] N [ ] |
| 6.8 | Node must be able to receive INFO fields up to 65535 octets | 6.1.1.1.4 | C:M | COCF3/4 | 65535 octets | Y [ ] N [ ] |
| 6.9 | Set up PAD field | 5.1.1.1.1, 6.1.1.1.5 | C:M | COCF3/4 | all 0’s | Y [ ] N [ ] |
| 6.10 | Recognize PAD field | 6.1.1.1.5 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.11 | Interpret PAD field | 6.1.1.1.5 | C:O | COCF3/4 | | Y [ ] N [ ] |
| 6.12 | Set CRC32 field according to CRC32_CO_TX flag | 5.1.1.1.1, 6.1.1.1.6 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.13 | Recognize CRC32 field depending on value of CRC32_CO_RX flag | 6.1.1.1.6 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.14 | Validate CRC32 field | 6.1.1.1.6 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.15 | Set Reserved field of Common PDU trailer | 5.1.1.1.1, 6.1.1.1.7 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.16 | Interpret Reserved field of Common PDU trailer | 6.1.1.1.7 | C:O | COCF3/4 | | Y [ ] N [ ] |
| 6.17 | Set BEtag field of Common PDU trailer | 5.1.1.1.1, 6.1.1.1.8 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.18 | Interpret BEtag field of Common PDU trailer | 6.1.1.1.8 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.19 | Set Length field | 5.1.1.1.1, 6.1.1.1.3 | C:M | COCF3/4 | | Y [ ] N [ ] |
| 6.20 | Interpret Length field | 6.1.1.1.3 | C:M | COCF3/4 | | Y [ ] N [ ] |

### A.6.1.2 Derived Connection-Oriented Service Protocol Data Unit (DCOPDU)

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 6.21 | DCOPDU format | 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.22 | Set Segment_Type | 5.1.1.1.3, 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.23 | Interpret Segment_Type | 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.24 | Set Sequence_ Number | 5.1.1.1.3, 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.25 | Interpret Sequence_ Number | 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.26 | Set Message_ Identifier | 5.1.1.1.3, 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.27 | Interpret Message_ Identifier | 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.28 | Payload_length | 5.1.1.1.3, 5.1.1.2.3, 6.1.1.2 | C:M | COCF3/4 | 4-44 | Y [ ]   N [ ] |
| 6.29 | Set Payload_CRC at source | 5.1.1.1.3, 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.30 | Use Payload_CRC for error detection | 6.1.1.2 | C:M | COCF3/4 | | Y [ ]   N [ ] |
| 6.31 | Use Payload_CRC for error correction of DCOPDU header | 6.1.1.2 | C:O | COCF3/4 | | Y [ ]   N [ ] |
| 6.32 | Use Payload_CRC for error correction of DCOPDU segmentation unit or trailer | 6.1.1.2 | C:PR | COCF3/4 | | Y [ ]   N [ ] |

## A.7 DQDB Layer facilities in support of connection-oriented service

***Add the following clause to clause A.7 of ISO/IEC 8802-6: 1994.***

### A.7.1 Counters

#### A.7.1.1 Peak rate counter

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 7.1 | 6x PEAK_CNTR_I_x (I = 0,1,2 x = A,B) | 7.1.1, 8.1.2 | M | | 0–($2^{32}$ –1) | Y [ ] N [ ] |
| 7.2 | Initialize counters | 7.1.1 | M | | 0 | Y [ ] N [ ] |

#### A.7.1.2 Mean rate counter

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 7.3 | 4x MEAN_CNTR_ I_x (I = 1, 2 x = A,B) | 7.1.2, 8.1.2 | M | | 0–($2^{32}$–1) | Y [ ] N [ ] |
| 7.4 | Initialize counters | 7.1.2 | M | | 0 | Y [ ] N [ ] |

#### A.7.1.3 Transmit-sequence number counter

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 7.5 | TX_SEQUENCE_NUM | 7.1.3 | C:M | COCF3/4 | 0–15 | Y [ ] N [ ] |
| 7.6 | Reset TX_SEQUENCE_ NUM | 7.1.3 | C:M | COCF3/4 | 0 | Y [ ] N [ ] |
| 7.7 | Initialize TX_ SEQUENCE_NUM | 7.1.3 | C:M | COCF3/4 | 0 | Y [ ] N [ ] |

### A.7.2 Local reservation queue

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 7.8 | 6x LOCRES_I_x (I = 0,1,2 x = A,B) | 7.2, 8.1.1 | M | | | Y [ ] N [ ] |
| 7.9 | Functions on Queues: PUSH, POP, EMPTY | 7.2 | M | | | Y [ ] N [ ] |
| 7.10 | Initialize list | 7.2 | M | | EMPTY=true | Y [ ] N [ ] |

### A.7.3 System parameter

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 7.11 | 2x PEAK_INCR_0_x (x = A,B) | 7.3.1, 8.1.2.1 | M | | 1 | Y [ ] N [ ] |
| 7.12 | 4x PEAK_INCR_I_x (I = 1,2 x = A,B) | 7.3.1, 8.1.2.2 | M | | 0–MV, default=0 | Y [ ] N [ ] |
| 7.13 | 2x PEAK_COST_0_x (x= A,B) | 7.3.2, 8.1.2.1 | M | | 1 | Y [ ] N [ ] |
| 7.14 | 4x PEAK_COST_I_x (I = 1,2 x = A,B) | 7.3.2, 8.1.2.2 | M | | 1–MV, default=MV | Y [ ] N [ ] |
| 7.15 | 2x PEAK_MAX_0_x (x = A,B) | 7.3.3, 8.1.2.1 | M | | 1 | Y [ ] N [ ] |
| 7.16 | 4x PEAK_MAX_I_x (I = 1,2 x = A,B) | 7.3.3, 8.1.2.2 | M | | 1–(10·MV) | Y [ ] N [ ] |
| 7.17 | 4x MEAN_INCR_I_x (I = 1,2 x = A,B) | 7.3.4, 8.1.2.2 | M | | 0–MV, default=0 | Y [ ] N [ ] |
| 7.18 | 4x MEAN_COST_I_x (I = 1,2 x = A,B) | 7.3.5, 8.1.2.2 | M | | 1–MV, default=MV | Y [ ] N [ ] |
| 7.19 | 4x MEAN_MAX_I_x (I = 1,2 x = A,B) | 7.3.6, 8.1.2.2 | M | | 1–(100·MV) | Y [ ] N [ ] |

### A.7.4 Flags

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 7.20 | CRC32_CO_TX_ CONTROL | 7.4.1 | C:M | COCF3/4 | ON/OFF | Y [ ] N [ ] |
| 7.21 | CRC32_CO_RX_ CONTROL | 7.4.2 | C:M | COCF3/4 | ON/OFF | Y [ ] N [ ] |

## A.8 DQDB Layer operation

### A.8.1 Distributed Queue operation

#### A.8.1.1 Distributed Queue State Machine (DQSM)

***This subclause replaces A.8.1.1 of ISO/IEC 8802-6: 1994.***

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 8.1 | 6x DQSM_x_I<br>(x = A,B I = 0,1,2) | 8.1.1 | M | | | Y [ ] N [ ] |
| 8.2 | Detect REQ_x_1 | 8.1.1 | M | | | Y [ ] N [ ] |
| 8.3 | Set REQ_x_I | 8.1.1 | M | | | Y [ ] N [ ] |
| 8.4 | Detect BUSY_x_I | 8.1.1 | M | | | Y [ ] N [ ] |
| 8.5 | Set BUSY_x | 8.1.1 | M | | | Y [ ] N [ ] |

#### A.8.1.2 Traffic Shaping State Machine (TSSM)

***Add this subclause to A.8.1 of ISO/IEC 8802-6: 1994.***

| Item | Feature | Reference | Status | Predicate | Value | Support |
|---|---|---|---|---|---|---|
| 8.6 | 2x TSSM_x_0<br>(x = A,B) | 8.1.2.1 | M | | | Y [ ] N [ ] |
| 8.7 | 4x TSSM_x_I<br>(x = A,B I = 1,2) | 8.1.2.2 | M | | | Y [ ] N [ ] |
| 8.8 | Detect slot_x | 8.1.2.2 | M | | | Y [ ] N [ ] |

#### A.8.1.3 REQ Queue State Machine (RQM)

The PICS for this part is as specified in A.8.1.2 of ISO/IEC 8802-6: 1994.

#### A.8.1.4 Bandwidth Balancing Machine (BWBM)

The PICS for this part is as specified in A.8.1.3 of ISO/IEC 8802-6: 1994.

### A.8.2 Reassembly operation

#### A.8.2.1 Reassembly State Machine (RSM) of the COCF3/4

The PICS for this part is as specified in A.8.2.1 of ISO/IEC 8802-6: 1994.

# Annex B

(informative)

# Recommended parameters

***Add the following annex to ISO/IEC 8802-6: 1994.***

## B.1 Usage of priority levels

When establishing a connection (by means of a layer management action), the relationship between a Convergence Function block and the QA Functions block is established. The quality of service provided by the QA functions depends on the priority level assigned at the connection setup. The following service characteristics are provided by the different priority levels.

Priority level 2

This priority level guarantees its user a certain throughput and a maximum delay. The achievable throughput is according to the traffic shaper parameters PEAK_INCR_2_x and PEAK_COST_2_x. For sending on Bus x, this node is guaranteed a fraction (PEAK_INCR_2_x / PEAK_COST_2_x) of the total bandwidth on Bus x, provided that the total bandwidth allocated on Bus x for priority level 2 does not exceed 100%. If mean rate shaping is used, then the Guaranteed Bandwidth (GBW) of this node equals a fraction (MEAN_ INCR_2_x / MEAN_COST_2_x) of the total bandwidth on bus x. The maximum delay can be calculated using the formula given in B.2.

Priority level 1

This priority level guarantees its user a certain throughput. The achievable throughput is according to the traffic shaper parameters PEAK_INCR_1_x and PEAK_COST_1_x. For sending on Bus x, this node is guaranteed a fraction (PEAK_INCR_1_x / PEAK_COST_1_x) of the total bandwidth on Bus x, provided that the total bandwidth allocated on Bus x for priority levels 1 and 2 does not exceed 100%. If mean rate shaping is used, then the GBW of this node equals a fraction (MEAN_INCR_1_x / MEAN_COST_1_x) of the total bandwidth on Bus x.

Priority level 0

This priority level should be used for connectionless traffic. There are no throughput guarantees for this priority level.

If mean rate shaping is used in all nodes sending on Bus x at higher priority levels, the following throughput guarantee for priority level 0 can be given. All nodes sending on Bus x at priority level 0 are guaranteed the difference of the total bandwidth on Bus x and the guaranteed mean rates for higher priority levels of all other nodes on Bus x. Thus, a node may achieve at least a throughput of:

(total bandwidth) – (guaranteed mean rates)/(number of nodes)

## B.2 Maximum access delay of a segment

This clause provides a formula for calculating the maximum segment delay on a DQDB bus using the enhanced QA functions. The segment delay is calculated from the moment a segment enters the Distributed Queue until it is completely put into an empty slot.

Note that the *total delay* of a segment, i.e., the end-to-end delay, includes processing time, the FIFO delay, the segment delay, and the propagation delay. The processing time depends on the implementation of the node but should be negligible. The FIFO delay is the time a slot has to wait in the FIFO queue (see 5.2.1.3). This time is zero if segments arrive regularly, i.e., at the agreed peak rate. The segment delay depends on the traffic on the bus and may be calculated using the formula below. The propagation delay from the source to the destination is fixed for a certain connection, i.e., is a function of the distance between the source and the destination.

The formula is valid under the following conditions:

— Priority level 2 is used for transmission
— The sum of guarantees for priority level 2 on Bus x does not exceed the total bandwidth of Bus x
— No PA slots are used on Bus x

The formula is independent of:

— Traffic sent at priority levels 0 and 1
— Usage of mean rate shaping

The following parameters are used in the formula:

— $N_0$ : Head-of-Bus of Bus x, $N_1$,..., $N_n$ : Nodes on Bus x
— L(i, j) : Propagation delay from $N_i$ to $N_j$ (in slottimes)
— PEAK_INCR_2_x (i), PEAK_COST_2_x (i), PEAK_MAX_2_x (i) : traffic shaper parameters of Node $N_i$
— $c_i$ = PEAK_MAX_2_x (i) / PEAK_COST_2_x (i) : Maximum number of segments $N_i$ is allowed to send back-to-back on Bus x at priority level 2
— slottime = slotlength [bit] / data rate [bit/s]

The maximum segment delay $t_i$ (j) of Node $N_i$ caused by $N_j$ through $N_{i-1}$ (1_j < i) for sending on Bus x may be calculated as follows:

$t_i$ (i) = 0, and

$t_i$ (j) = $t_i$ (j+1) + [$c_j$ + $t_i$ (j+1) · PEAK_INCR_2_x (j)/PEAK_COST_2_x (j)] (for 1_j < i)

The maximum segment delay D(i) at $N_i$ sending on bus x (in slottimes) is then:

$$D(\mathrm{i}) = t_i(1) + 2 \times L(1, \mathrm{i}) + 2$$

The use of the formula above is illustrated in the following example. Assume a bus of 12 km length with 12 nodes equidistantly placed on the bus. Each node except for the last one is guaranteed a data rate of 10 Mbit/s.

The second, third, and fourth columns of table B.1 show the GBW parameter of the scenario. The fifth column shows the recursive calculation of $t_{11}$(i). It starts with $t_{11}$(11)=0. Then $t_{11}$(10), $t_{11}$(9), etc., are calculated. Then the maximum delay of a segment sent by node 11 on Bus x max be calculated as

$$D(11) = t_{11}(1) + 2 \times L(1, 11) + 2 = 52$$

This equals 0.147 ms. The sixth and seventh columns show the calculation of $t_6$(i) and $t_2$(i), respectively.

**Table B.1—Access delay computation**

| GBW parameter | | | | $t_{11}$(i) | $t_6$(i) | $t_2$(i) | [slottimes] | [ms] |
|---|---|---|---|---|---|---|---|---|
| **Number of station** | **Number of PEAK_ INC_2_x** | **PEAK_ COST_2_x** | **PEAK_ MAX_2_x** | | | | | |
| 1 | 10 | 150 | 150 | 10 | 5 | 2 | | |
| 2 | 10 | 150 | 150 | 9 | 4 | 1 | D(2)= 8 | 0.0226133 |
| 3 | 10 | 150 | 150 | 8 | 3 | | | |
| 4 | 10 | 150 | 150 | 7 | 2 | | | |
| 5 | 10 | 150 | 150 | 6 | 1 | | | |
| 6 | 10 | 150 | 150 | 5 | 0 | | D(6)=27 | 0.07632 |
| 7 | 10 | 150 | 150 | 4 | | | | |
| 8 | 10 | 150 | 150 | 3 | | | | |
| 9 | 10 | 150 | 150 | 2 | | | | |
| 10 | 10 | 150 | 150 | 1 | | | | |
| 11 | 10 | 150 | 150 | 0 | | | D(11)=52 | 0.1469867 |

## B.3 Recommended use of GBW parameters

The GBW protocol uses several parameters (PEAK_INCR_I_x, PEAK_COST_I_x, and PEAK_MAX_I_x) for defining the bandwidth allocation. More specifically, the ratio of PEAK_INCR_I_x/PEAK_COST_I_x determines the bandwidth assigned to a node at priority level I on Bus x. Hence, there are two parameters that must be set by network management. It is more practical to manage only one parameter instead of several. Therefore, it is recommended to set PEAK_COST_I_x to a multiple of the physical data rate and keep it fixed. Then only PEAK_INCR_I_x determines the bandwidth allocation. For example, if the net data rate is 135 Mbit/s and PEAK_COST_I_x is set to 2400 (MV, see 7.3.1), then the smallest granularity is 64 kbit/s, i.e., the bandwidth allocation can be incremented in steps of 64 kbit/s.

Of course, PEAK_MAX_I_x must be modified together with PEAK_INCR_I_x, but it can be derived from PEAK_INCR_I_x (according to the formula given in 7.3.3), and its modification does not interfere with the bandwidth allocation.