93. Multipoint MAC Control

Editors' Note #1 (to be removed prior to release): This amendment is based on the current edition of IEEE P802.3ay (D2.2). The editing instructions define how to merge the material contained in this amendment into the base document set to form the new comprehensive standard as created by the addition of IEEE P802.3av.

This amendment is new material to be added to IEEE P802.3ay (D2.2). The material contained in this amendment forms the new comprehensive standard as created by the addition of IEEE P802.3av.
External cross references are marked with double "@" signs (for example Clause @@1.1.1@@) and will be converted to hyperlinks in the later release of the draft.

Editors' Note #2 (to be removed prior to release): Draft revision history for Clause 91

Draft	***Date***	***Comment***
D1.1	*Feb 9, 2008*	*Draft for Task Force review with comment resolution from January 2008 meeting* *First version of Clause 93 draft available*
D1.2	*Apr 1, 2008*	*Draft for Task Force review with comment resolution from March 2008 meeting*
D1.3	*Apr XX, 2008*	*Draft for Task Force review with comment resolution from April 2008 meeting*

93.1 Overview

This clause deals with the mechanism and control protocols required in order to reconcile the P2MP topology into the Ethernet framework. The P2MP medium is a passive optical network (PON), an optical network with no active elements in the signal's paths from source to destination. The only interior elements used in a PON are passive optical components, such as optical fiber, splices, and splitters. When combined with the Ethernet protocol, such a network is referred to as Ethernet passive optical network (EPON).

P2MP is an asymmetrical medium based on a tree (or tree–and–branch) topology. The DTE connected to the trunk of the tree is called optical line terminal (OLT) and the DTEs connected at the branches of the tree are called optical network units (ONU). The OLT typically resides at the service provider's facility, while the ONUs are located at the subscriber premises.

In the downstream direction (from the OLT to an ONU), signals transmitted by the OLT pass through a 1:N passive splitter (or cascade of splitters) and reach each ONU. In the upstream direction (from the ONUs to the OLT), the signal transmitted by an ONU would only reach the OLT, but not other ONUs. To avoid data collisions and increase the efficiency of the subscriber access network, ONU's transmissions are arbitrated. This arbitration is achieved by allocating a transmission window (grant) to each ONU. An ONU defers transmission until its grant arrives. When the grant arrives, the ONU transmits frames at wire speed during its assigned time slot.

A simplified P2MP topology example is depicted in Figure 93–1. Clause 67 provides additional examples of P2MP topologies.

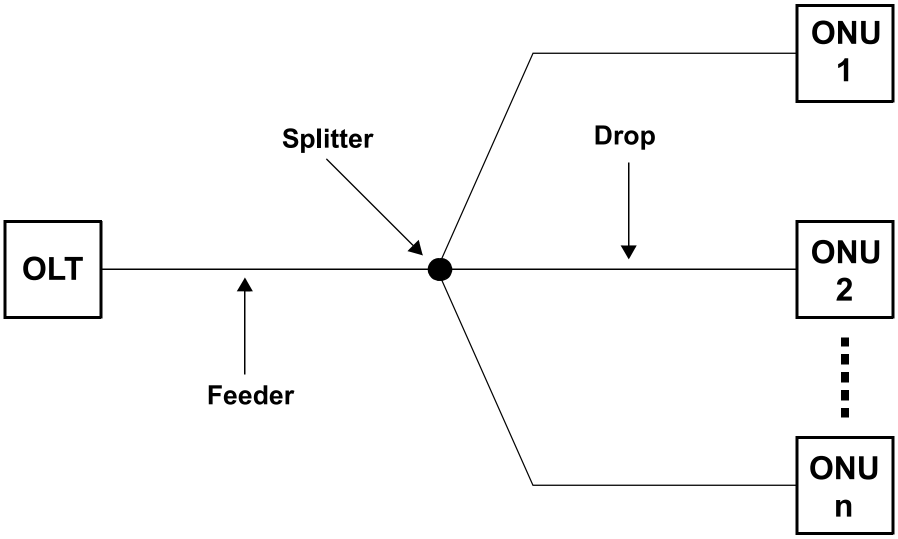


Figure 93–1—PON topology example

Topics dealt with in this clause include allocation of upstream transmission resources to different ONUs, discovery and registration of ONUs into the network, and reporting of congestion to higher layers to allow for dynamic bandwidth allocation schemes and statistical multiplexing across the PON.

This clause does not deal with topics including bandwidth allocation strategies, authentication of end–devices, quality–of–service definition, provisioning, or management.

This clause specifies the multipoint control protocol (MPCP) to operate an optical multipoint network by defining a Multipoint MAC Control sublayer as an extension of the MAC Control sublayer defined in Clause 31, and supporting current and future operations as defined in Clause 31 and annexes.

Each PON consists of a node located at the root of the tree assuming the role of OLT, and multiple nodes located at the tree leaves assuming roles of ONUs. The network operates by allowing only a single ONU to transmit in the upstream direction at a time. The MPCP located at the OLT is responsible for timing the different transmissions. Reporting of congestion by the different ONUs may assist in optimally allocating the bandwidth across the PON.

Automatic discovery of end stations is performed, culminating in registration through binding of an ONU to an OLT port by allocation of a Logical Link ID (see LLID in Sublause 65.1.3.3.2), and dynamic binding to a MAC connected to the OLT.

The Multipoint MAC Control functionality shall be implemented for subscriber access devices containing point–to–multipoint Physical Layer devices defined in Clause 60.

93.1.1 Goals and objectives

The goals and objectives of this clause are the definition of a point–to–multipoint Ethernet network utilizing an optical medium.

Specific objectives met include:

a) Support of Point–to–Point Emulation (P2PE) as specified
b) Support multiple LLIDs and MAC Clients at the OLT

c) Support a single LLID per ONU
d) Support a mechanism for single copy broadcast
e) Flexible architecture allowing dynamic allocation of bandwidth
f) Use of 32 bit timestamp for timing distribution
g) MAC Control based architecture
h) Ranging of discovered devices for improved network performance
i) Continuous ranging for compensating round trip time variation

93.1.2 Position of Multipoint MAC Control within the IEEE 802.3 hierarchy

Multipoint MAC Control defines the MAC control operation for optical point–to–multipoint networks. Figure 93–2 depicts the architectural positioning of the Multipoint MAC Control sublayer with respect to the MAC and the MAC Control client. The Multipoint MAC Control sublayer takes the place of the MAC Control sublayer to extend it to support multiple clients and additional MAC control functionality.

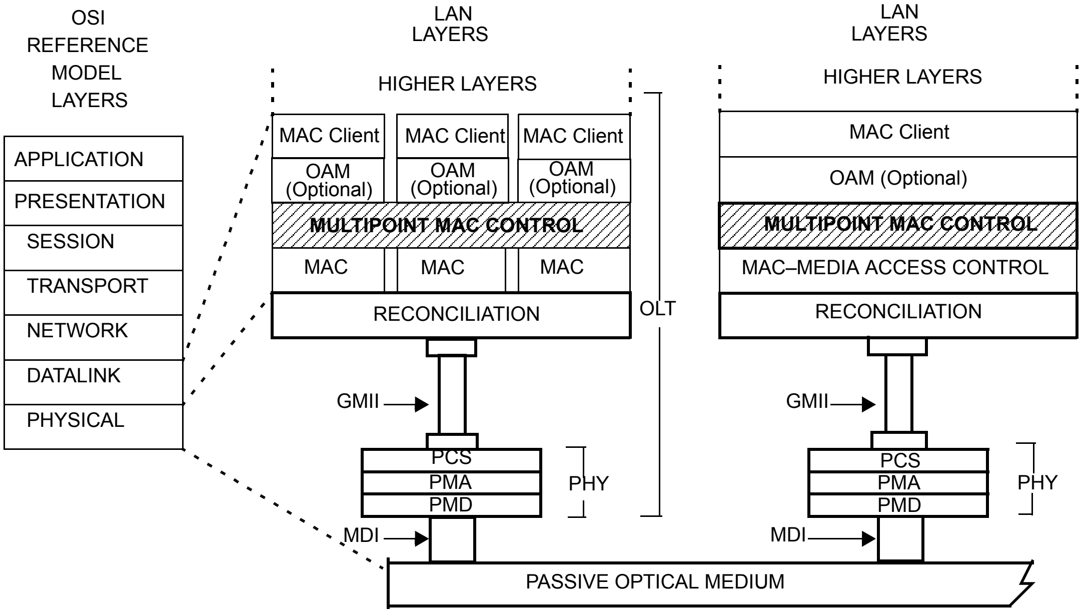
Multipoint MAC Control is defined using the mechanisms and precedents of the MAC Control sublayer. The MAC Control sublayer has extensive functionality designed to manage the real–time control and manipulation of MAC sublayer operation. This clause specifies the extension of the MAC Control mechanism to manipulate multiple underlying MACs simultaneously. This clause also specifies a specific protocol implementation for MAC Control.

The Multipoint MAC Control sublayer is specified such that it can support new functions to be implemented and added to this standard in the future. MultiPoint Control Protocol (MPCP), the management protocol for P2MP is one of these protocols. Non–real–time, or quasi–static control (e.g., configuration of MAC operational parameters) is provided by Layer Management. Operation of the Multipoint MAC Control sublayer is transparent to the MAC.

As depicted in Figure 93–2, the layered system instantiates multiple MAC entities, using a single Physical Layer. The individual MAC instances offer a Point-to-point emulation service between the OLT and the ONU. An additional MAC is instantiated to communicate to all 10 Gb/s downstream ONUs at once. This instance takes maximum advantage of the broadcast nature of the downstream channel by sending a single copy of a frame that is received by all ONUs. This MAC instance is referred to as Single Copy Broadcast (SCB).

The ONU only requires one MAC instance since frame filtering operations are done at the RS layer before reaching the MAC. Therefore, MAC and layers above are emulation–agnostic at the ONU (see Subclause 65.1.3.3).

Although Figure 93–2 and supporting text describe multiple MACs within the OLT, a single unicast MAC address may be used by the OLT. Within the EPON Network, MACs are uniquely identified by their LLID which is dynamically assigned by the registration process.



GMII = GIGABIT MEDIA INDEPENDENT INTERFACE
MDI = MEDIUM DEPENDENT INTERFACE
OAM = OPERATIONS, ADMINISTRATION & MAINTENANCE
OLT = OPTICAL LINE TERMINAL

ONU = OPTICAL NETWORK UNIT
PCS = PHYSICAL CODING SUBLAYER
PHY = PHYSICAL LAYER DEVICE
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT

Figure 93–2—Relationship of Multipoint MAC Control and the OSI protocol stack

93.1.3 Functional block diagram

Figure 93–3 provides a functional block diagram of the Multipoint MAC Control architecture.

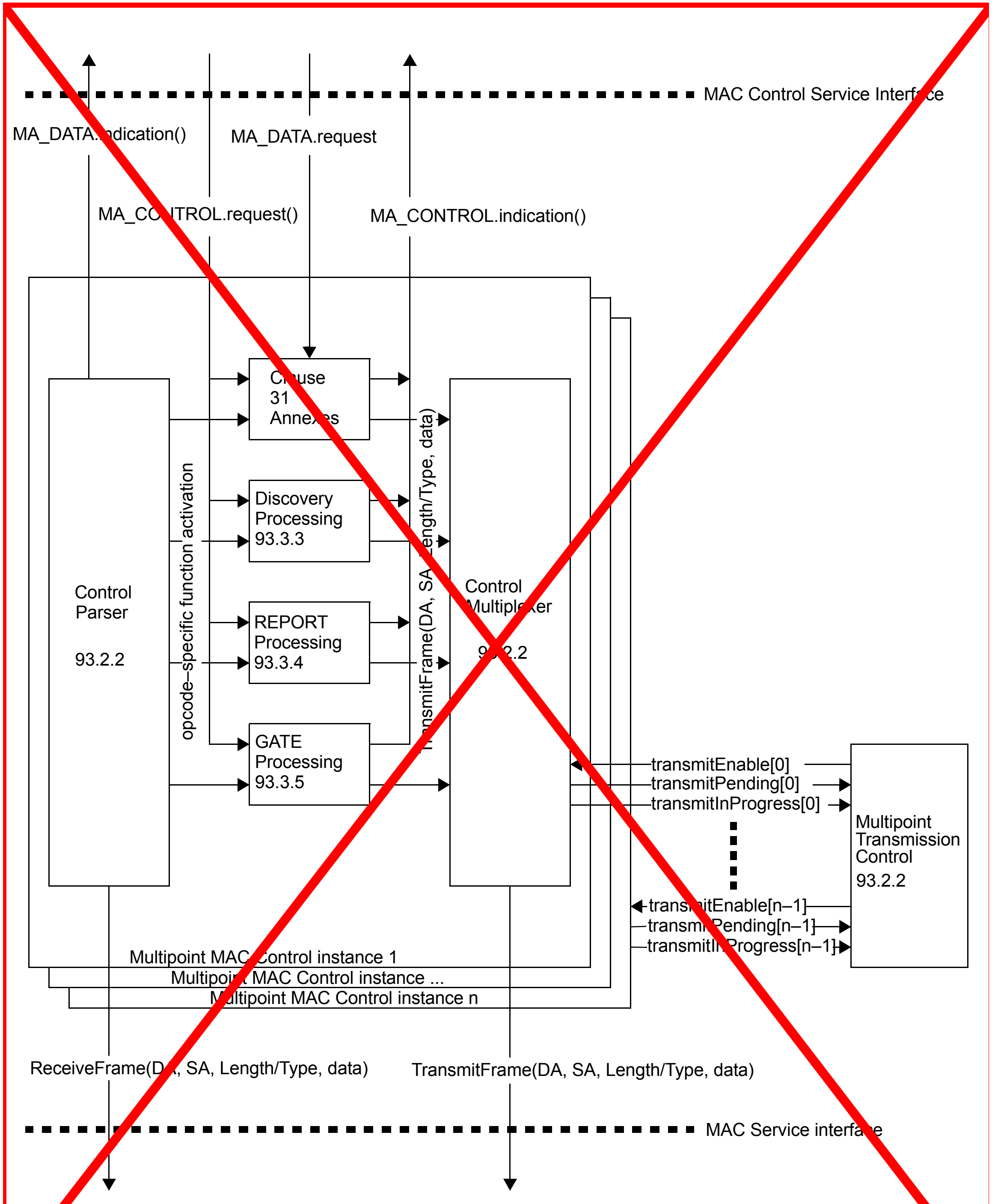


Figure 93–3—Multipoint MAC Control Functional Block Diagram

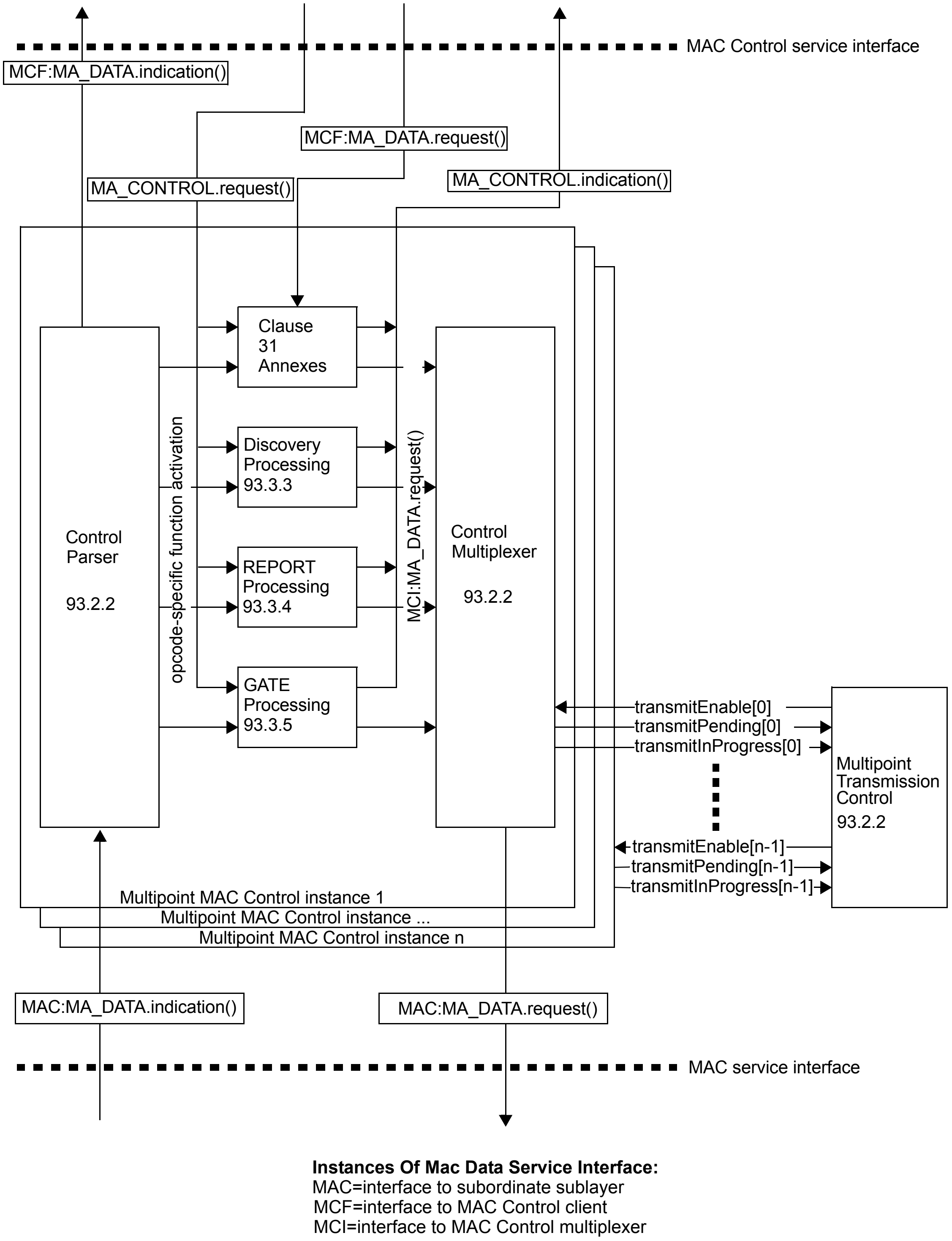


Figure 93–3—Multipoint MAC Control functional block diagram

93.1.4 Service interfaces

The MAC Client communicates with the Control Multiplexer using the standard service interface specified in Subclause 2.3. Multipoint MAC Control communicates with the underlying MAC sublayer using the standard service interface specified in Annex 4A.3.2. Similarly, Multipoint MAC Control communicates internally using primitives and interfaces consistent with definitions in Clause 31.

93.1.5 State diagram conventions

The body of this standard comprises state diagrams, including the associated definitions of variables, constants, and functions. Should there be a discrepancy between a state diagram and descriptive text, the state diagram prevails.

The notation used in the state diagrams follows the conventions of Subclause 21.5. State diagram timers follow the conventions of Subclause 14.2.3.2 augmented as follows:

a) [start x_timer, y] sets expiration of *y* to timer *x_timer*.
b) [stop x_timer] aborts the timer operation for *x_timer* asserting *x_timer_not_done* indefinitely.

The state diagrams use an abbreviation MACR as a shorthand form for MA_CONTROL.request and MACI as a shorthand form for MA_CONTROL.indication.

The vector notations used in the state diagrams for bit vector use 0 to mark the first received bit and so on (for example data[0:15]), following the conventions of Subclause 3.1 for bit ordering. When referring to an octet vector, 0 is used to mark the first received octet and so on (for example m_sdu[0..1]).

a < *b:* A function that is used to compare two (cyclic) time values. Returned value is true when *b* is larger than *a* allowing for wrap around of *a* and *b*. The comparison is made by subtracting *b* from *a* and testing the MSB. When MSB(*a*–*b*) = 1 the value true is returned, else false is returned. In addition, the following functions are defined in terms of *a* < *b*:
a > *b* is equivalent to !(*a* < *b* or *a* = *b*)
a ≥ *b* is equivalent to !(*a* < *b*)
a ≤ *b* is equivalent to !(*a* > *b*)

93.2 Multipoint MAC Control operation

As depicted in Figure 93–3, the Multipoint MAC Control functional block comprises the following functions:

a) *Multipoint Transmission Control*. This block is responsible for synchronizing Multipoint MAC Control instances associated with the Multipoint MAC Control. This block maintains the Multipoint MAC Control state and controls the multiplexing functions of the instantiated MACs.
b) *Multipoint MAC Control Instance n*. This block is instantiated for each MAC and respective MAC and MAC Control clients associated with the Multipoint MAC Control. It holds all the variables and state associated with operating all MAC Control protocols for the instance.
c) *Control Parser*. This block is responsible for parsing MAC Control frames, and interfacing with Clause 31 entities, the opcode specific blocks, and the MAC Client.
d) *Control Multiplexer*. This block is responsible for selecting the source of the forwarded frames.
e) *Clause 31 Annexes*. This block holds MAC Control actions as defined in Clause 31 annexes for support of legacy and future services.
f) *Discovery, Report and Gate Processing*. These blocks are responsible for handling the MPCP in the context of the MAC.

93.2.1 Principles of Multipoint MAC Control

As depicted in Figure 93–3, Multipoint MAC Control sublayer may instantiate multiple Multipoint MAC Control instances in order to interface multiple MAC and MAC Control clients above with multiple MACs below. A unique unicast MAC instance is used at the OLT to communicate with each ONU. The individual

MAC instances utilize the point–to–point emulation service between the OLT and the ONU as defined in Subclause 65.1.

At the ONU, a single MAC instance is used to communicate with a MAC instance at the OLT. In that case, the Multipoint MAC Control contains only a single instance of the Control Parser/Multiplexer function.

Multipoint MAC Control protocol supports several MAC and client interfaces. Only a single MAC interface and Client interface is enabled for transmission at a time. There is a tight mapping between a MAC service interface and a Client service interface. In particular, the assertion of the ~~ReceiveFrame interface~~ MAC:MA_DATA.indication primitive in MAC *j* ~~enables~~ leads to the assertion of the MCF:MA_DATA.indication ~~interface of~~ primitive to Client *j*. Conversely, the assertion of the request service interface in Client *i* ~~enables~~ leads to the assertion of the ~~TransmitFrame interface~~ MAC:MA_DATA.request primitive of MAC *i*. Note that the Multipoint MAC sublayer need not receive and transmit packets associated with the same interface at the same time. Thus the Multipoint MAC Control acts like multiple MAC Controls bound together with common elements.

The scheduling algorithm is implementation dependent, and is not specified for the case where multiple transmit requests happen at the same time.

The reception operation is as follows. The Multipoint MAC Control instances generate ~~ReceiveFrame function calls~~ MAC:MA_DATA.indication service primitives continuously to the underlying MAC instances. Since these MACs are receiving frames from a single PHY only one frame is passed from the MAC instances to Multipoint MAC Control. The MAC instance responding to the ~~ReceiveFrame~~ MAC:MA_DATA.indication is referred to as the enabled MAC, and its service interface is referred to as the enabled MAC interface. The MAC passes to the Multipoint MAC Control sublayer all valid frames. Invalid frames, as specified in Subclause 3.4, are not passed to the Multipoint MAC Control sublayer in response to a ~~ReceiveFrame function call~~MAC:MA_DATA.indication service primitive.

The enabling of a transmit service interface is performed by the Multipoint MAC Control instance in collaboration with the Multipoint Transmission Control. Frames generated in the MAC Control are given priority over MAC Client frames, in effect, prioritizing the MA_CONTROL primitive over the MCF:MA_DATA primitive, and for this purpose MCF:MA_DATA.request primitives may be delayed, discarded or modified in order to perform the requested MAC Control function. For the transmission of this frame, the Multipoint MAC Control instance enables forwarding by the MAC Control functions, but the MAC Client interface is not enabled. The reception of a frame in a MAC ~~enables the ReceiveFrame interface~~ results in generation of the MAC:MA_DATA.indication primitive on that MAC's interface. Only one receive MAC interface will be enabled at any given time since there is only one PHY interface.

The information of the enabled interfaces is stored in the controller state variables, and accessed by the Multiplexing Control block.

The Multipoint MAC Control sublayer uses the services of the underlying MAC sublayer to exchange both data and control frames.

Receive operation (MAC:MA_DATA.indication) at each instance:

a) A frame is received from the underlying MAC.
b) The frame is parsed according to Length/Type field
c) MAC Control frames are demultiplexed according to opcode and forwarded to the relevant processing functions
d) Data frames (see 31.5.1) are forwarded to the MAC Client by asserting MCF:MA_DATA.indication primitives

Transmit operation (MAC:MA_DATA.request) at each instance:

e) The MAC Client signals a frame transmission by asserting MCF:MA_DATA.request, or
f) A protocol processing block attempts to issue a frame, as a result of a previous MA_CONTROL.request or as a result of an MPCP event that generates a frame.

g) When allowed to transmit by the Multipoint Transmission Control block, the frame is forwarded.

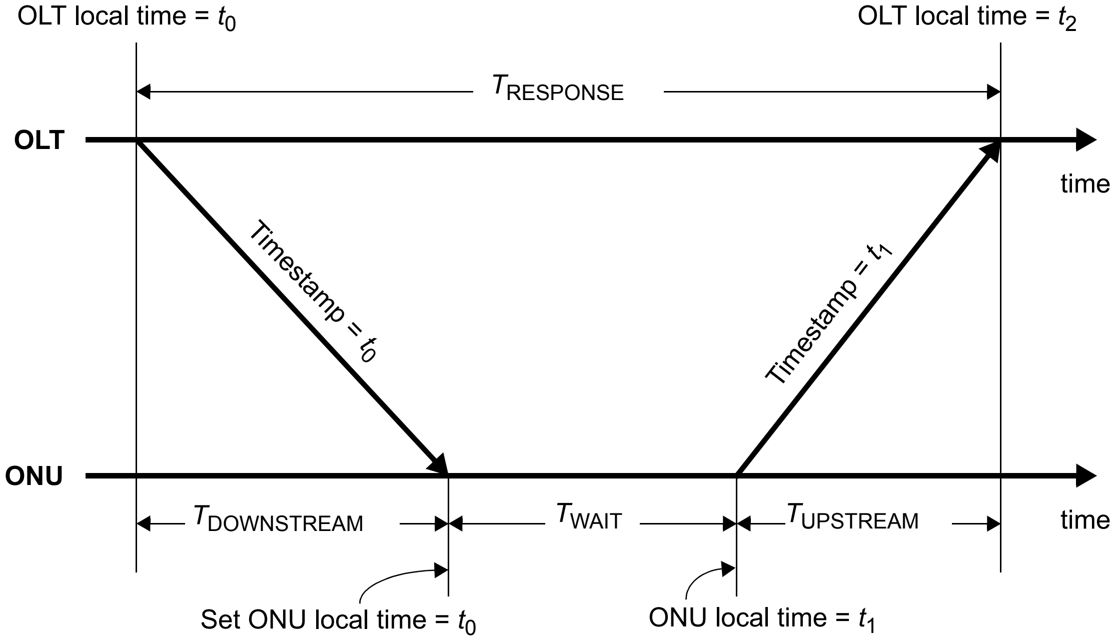
93.2.1.1 Ranging and Timing Process

Both the OLT and the ONU have 32–bit counters that increment every 16 ns. These counters provide a local time stamp. When either device transmits an MPCPDU, it maps its counter value into the timestamp field. The time of transmission of the first octet of the MPCPDU frame from the MAC Control to the MAC is taken as the reference time used for setting the timestamp value.

When the ONU receives MPCPDUs, it sets its counter according to the value in the timestamp field in the received MPCPDU.

When the OLT receives MPCPDUs, it uses the received timestamp value to calculate or verify a round trip time between the OLT and the ONU. The RTT is equal to the difference between the timer value and the value in the timestamp field. The calculated RTT is notified to the client via the MA_CONTROL.indication primitive. The client can use this RTT for the ranging process.

A condition of *timestamp drift error* occurs when the difference between OLT's and ONU's clocks exceeds some predefined threshold. This condition can be independently detected by the OLT or an ONU. The OLT detects this condition when an absolute difference between new and old RTT values measured for a given ONU exceeds the value of guardThresholdOLT (see Subclause 93.2.2.1), as shown in Figure 93–10. An ONU detects the timestamp drift error condition when absolute difference between a timestamp received in an MPCPDU and the localTime counter exceeds guardThresholdONU (see Subclause 93.2.2.1), as is shown in Figure 93–11.



$T_{DOWNSTREAM}$ = downstream propagation delay

$T_{UPSTREAM}$ = upstream propagation delay

T_{WAIT} = wait time at ONU = $t_1 - t_0$

$T_{RESPONSE}$ = response time at OLT = $t_2 - t_0$

$$RTT = T_{DOWNSTREAM} + T_{UPSTREAM} = T_{RESPONSE} - T_{WAIT} = (t_2 - t_0) - (t_1 - t_0) = t_2 - t_1$$

Figure 93–4—Round trip time calculation

93.2.2 Multipoint transmission control, Control Parser, and Control Multiplexer

The purpose of the multipoint transmission control is to allow only one of the multiple MAC clients to transmit to its associated MAC and subsequently to the RS layer at one time by only asserting one transmitEnable signal at a time.

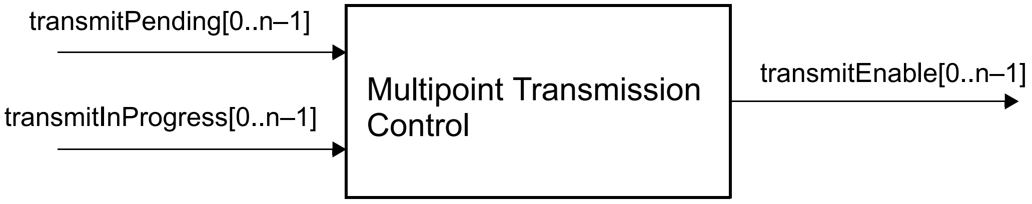


Figure 93–5—Multipoint Transmission Control Service Interfaces

Multipoint MAC Control Instance n function block communicates with the Multipoint Transmission Control using transmitEnable[n], transmitPending[n], and transmitInProgress[n] state variables (see Figure 93–3).

The Control Parser is responsible for opcode independent parsing of MAC frames in the reception path. By identifying MAC Control frames, demultiplexing into multiple entities for event handling is possible. Interfaces are provided to existing Clause 31 entities, functional blocks associated with MPCP, and the MAC Client.

The Control Multiplexer is responsible for forwarding frames from the MAC Control opcode–specific functions and the MAC Client to the MAC. Multiplexing is performed in the transmission direction. Given multiple MCF:MA_DATA.request primitives from the MAC Client, and MA_CONTROL.request primitives from the MAC Control Clients, a single ~~TransmitFrame~~ MAC:MA_DATA.request service primitive is generated for transmission. At the OLT, multiple MAC instances share the same Multipoint MAC Control, as a result, the transmit block is enabled based on an external control signal housed in Multipoint Transmission Control for transmission overlap avoidance. At the ONU the Gate Processing functional block interfaces for upstream transmission administration.

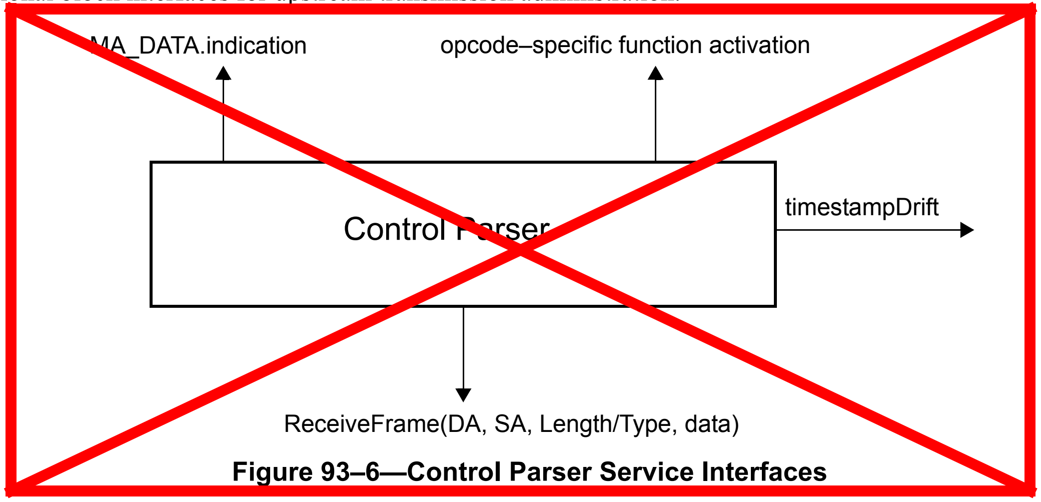
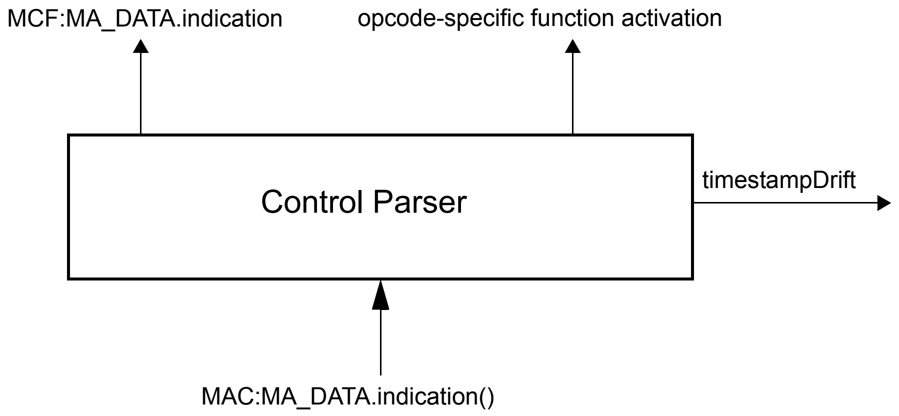
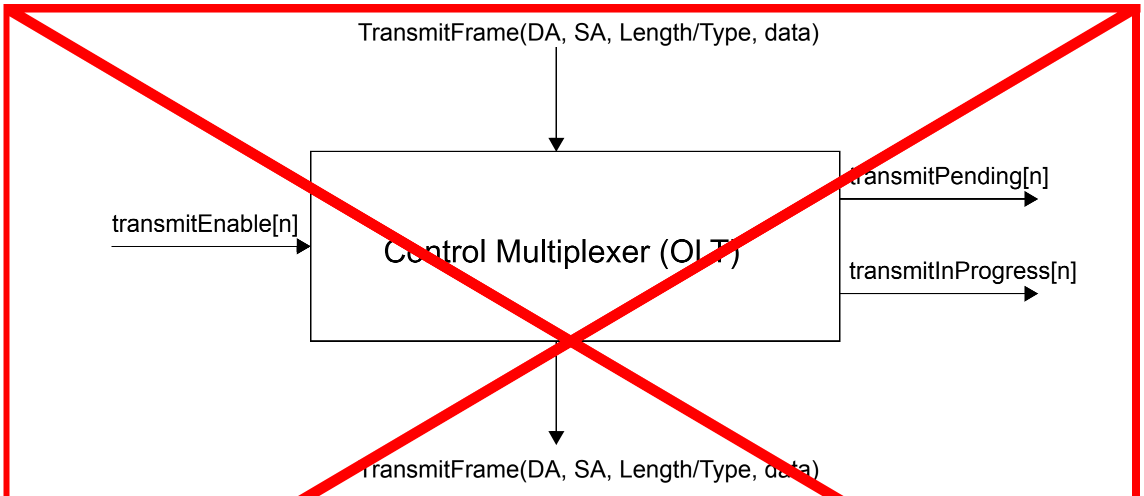

MA_DATA.indication
opcode–specific function activation
Control Parser
timestampDrift
ReceiveFrame(DA, SA, Length/Type, data)

Figure 93–6—Control Parser Service Interfaces



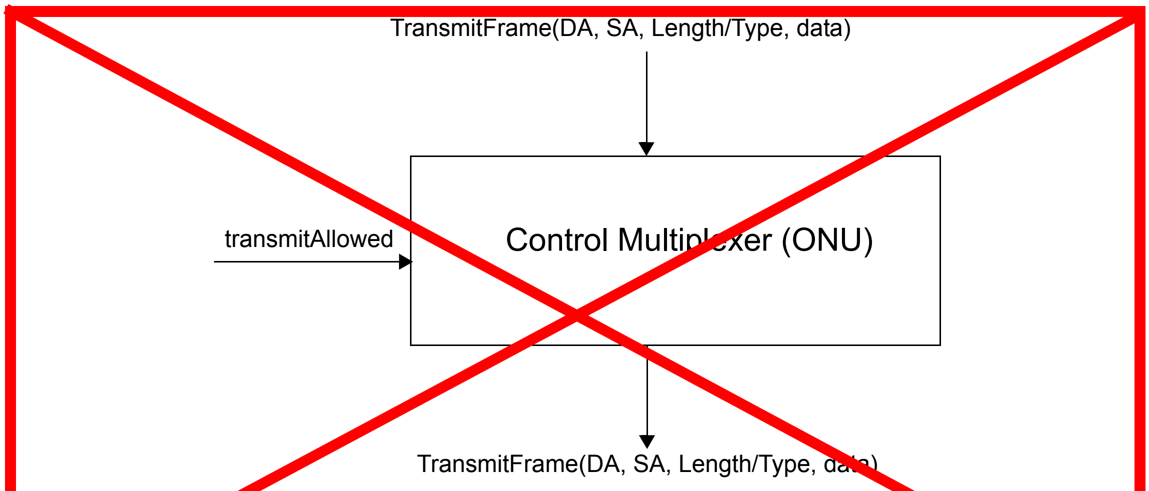
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

Figure 93–6—Control Parser service interfaces


TransmitFrame(DA, SA, Length/Type, data)
transmitEnable[n]
Control Multiplexer (OLT)
transmitPending[n]
transmitInProgress[n]
TransmitFrame(DA, SA, Length/Type, data)

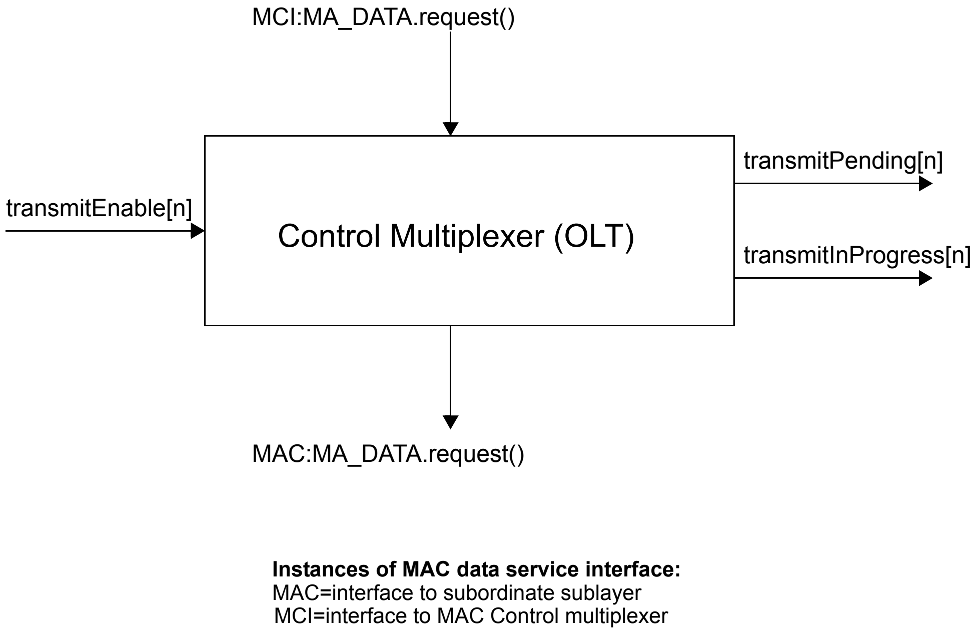
NOTE—TransmitFrame primitive may be issued from multiple MAC Control processing blocks.

Figure 93–7—OLT Control Multiplexer Service Interfaces


TransmitFrame(DA, SA, Length/Type, data)
transmitAllowed
Control Multiplexer (ONU)
TransmitFrame(DA, SA, Length/Type, data)

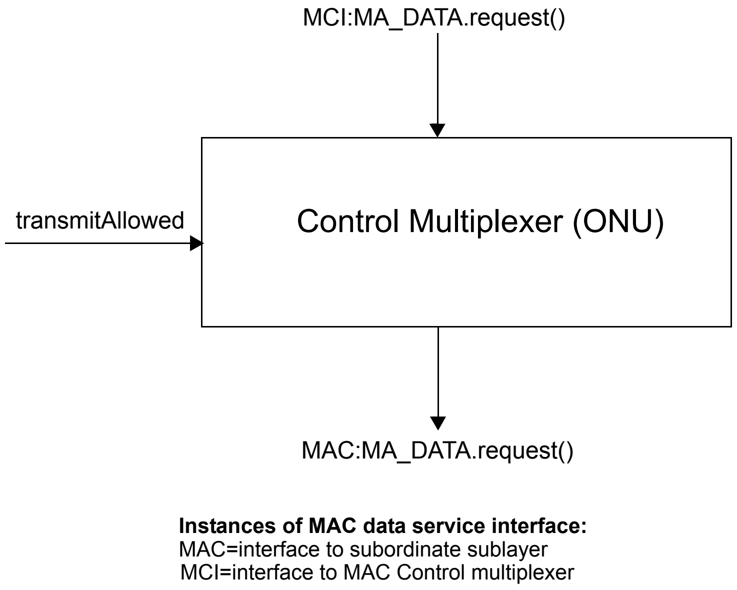
NOTE—TransmitFrame primitive may be issued from multiple MAC Control processing blocks.

Figure 93–8—ONU Control Multiplexer Service Interfaces


MCI:MA_DATA.request()
transmitEnable[n]
Control Multiplexer (OLT)
transmitPending[n]
transmitInProgress[n]
MAC:MA_DATA.request()
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

NOTE—MAC:MA_DATA.request primitive may be issued from multiple MAC Control processing blocks.

Figure 93–7—OLT Control Multiplexer service interfaces


MCI:MA_DATA.request()
transmitAllowed
Control Multiplexer (ONU)
MAC:MA_DATA.request()
Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

NOTE—MAC:MA_DATA.request primitive may be issued from multiple MAC Control processing blocks.

Figure 93–8—ONU Control Multiplexer service interfaces

93.2.2.1 Constants

defaultOverhead

This constant holds the size of packet transmission overhead. This overhead is measured in units of time quanta.

TYPE: integer
VALUE: 6

guardThresholdOLT

This constant holds the maximal amount of drift allowed for a timestamp received at the OLT. This value is measured in units of time_quantum.

TYPE: integer
VALUE: 12

guardThresholdONU

This constant holds the maximal amount of drift allowed for a timestamp received at the ONU. This value is measured in units of time_quantum.

TYPE: integer
VALUE: 8

MAC_Control_type

The value of the Length/Type field as defined in Subclause 31.4.1.3.

TYPE: integer
VALUE: 0x8808

tailGuard

This constant holds the value used to reserve space at the end of the upstream transmission at the ONU in addition to the ~~sie~~ size of last MAC service data unit (m_sdu) in units of octets

. ~~-~~Space is reserved for the MAC overheads including: preamble, SFD, DA, SA, Length/Type, FCS, and minimum inter-packet gap. The sizes of the above listed MAC overhead items are described in Subcause 3.1.1. The size of the minimum IPG is described in ~~Annex 4A~~Clause 36.2.4.~~2~~14.

TYPE: integer
VALUE: 42

time_quantum

The unit of time_quantum is used by all mechanisms synchronized to the advancement of the localTime variable. All variables that represent counters and time intervals are defined using time_quantum. Each time_quantum is 16 ns.

TYPE: integer
VALUE: 16

tqSize

This constant represents time_quantum in octet transmission times.

TYPE: integer
VALUE: 20

93.2.2.2 Counters

localTime

This variable holds the value of the local timer used to control MPCP operation. This variable is advanced by a timer at 62.5MHz, and counts in time_quanta. At the OLT the counter shall track the transmit clock, while at the ONU the counter shall track the receive clock. For accuracy of receive clock see Subclause 65.3.1.2. It is reloaded with the received timestamp value (from the OLT) by the Control Parser (see Figure 93–11). Changing the value of this variable while running using Layer Management is highly undesirable and is unspecified.

TYPE: 32 bit unsigned

93.2.2.3 Variables

BEGIN

This variable is used when initiating operation of the functional block state diagram. It is set to true following initialization and every reset.

TYPE: Boolean

data_rx

This variable represents a 0–based bit array corresponding to the payload of a received MPCPDU. This variable is used to parse incoming MPCPDU frames.

TYPE: bit array

data_tx

This variable represents a 0–based bit array corresponding to the payload of an MPCPDU being transmitted. This variable is used to access payload of outgoing MPCPDU frames, for example to set the timestamp value.

TYPE: bit array

fecEnabled

This variable represents whether the FEC function is enabled. If FEC function is enabled, this variable equals true, otherwise it equals false.

TYPE:Boolean

newRTT

This variable temporary holds a newly–measured Round Trip Time to the ONU. The new RTT value is represented in units of time_quanta.

TYPE: 16 bit unsigned

nextTxTime

This variable represents a total transmission time of next packet and is used to check whether the next packet fits in the remainder of ONU's transmission window. The value of nextTxTime includes packet transmission time, tailGuard defined in Subclause 93.2.2.1, and FEC parity data overhead, if FEC is enabled. This variable is measured in units of time quanta.

TYPE: 16 bit unsigned

opcode_rx

This variable holds an opcode of the last received MPCPDU.

TYPE: 16 bit unsigned

opcode_tx

This variable holds an opcode of an outgoing MPCPDU.

TYPE: 16 bit unsigned

packet_initiate_delay

This variable is used to set the time–out interval for packet_initiate_timer defined in 93.2.2.5. The packet_initiate_delay value is represented in units of time_quanta.

TYPE: 16 bit unsigned

payloadBalance

This variable stores the number of time quanta from the previous transmission that have not yet been compensated for.

TYPE: 16 bit signed

prevTime

This variable stores the value of localTime when a frame is transmitted. -It is used by the FEC_Overhead function to determine how much time has elapsed between subsequent frames.

TYPE: 32 bit unsigned

RTT

This variable holds the measured Round Trip Time to the ONU. The RTT value is represented in units of time_quanta.

TYPE: 16 bit unsigned

stopTime

This variable holds the value of the localTime counter corresponding to the end of the nearest grant. This value is set by the Gate Processing function as described in Subclause 93.3.5.

TYPE: 32 bit unsigned

timestamp

This variable holds the value of timestamp of the last received MPCPDU frame.

TYPE: 32 bit unsigned

timestampDrift

This variable is used to indicate whether an error is signaled as a result of uncorrectable timestamp drift.

TYPE: Boolean

transmitAllowed

This variable is used to control PDU transmission at the ONU. It is set to true when the transmit path is enabled, and is set to false when the transmit path is being shut down. transmitAllowed changes its value according to the state of the Gate Processing functional block.

TYPE: Boolean

transmitEnable[j]

These variables are used to control the transmit path in a Multipoint MAC Control instance at the OLT. Setting them to on indicates that the selected instance is permitted to transmit a frame. Setting it to off inhibits the transmission of frames in the selected instance. Only one of transmitEnable[j] should be set to on at a time.

TYPE: Boolean

transmitInProgress[j]

This variable indicates that the Multipoint MAC Control instance *j* is in a process of transmitting a frame.

TYPE: Boolean

transmitPending[j]

This variable indicates that the Multipoint MAC Control instance *j* is ready to transmit a frame.

TYPE: Boolean

93.2.2.4 Functions

10G_PCS_Overhead(length)

This function calculates the size of additional overhead to be added by the PCS and FEC encoders while transmitting a frame of size length. Parameter length represents the size of an entire frame including preamble, SFD, DA, SA Length/Type, and FCS. If the frame does not occupy an entire FEC block, the function result rounds up to the size of a complete FEC block. As well, the burst preamble is included in the overhead.

As specified in Subclause 49.2.4, the 66bit encoder adds 2 control bits for every 8 octets of data. As specified in @@Subclause 92.2.3.4@@, the FEC encoder adds 264 bits of parity and related control for every 216 data octets. The function returns the value of FEC overhead in units of time quanta.

The following formula is used to calculate the overhead:

abs(n)

This function returns the absolute value of the parameter n.

$$\text{10G\_PCS\_Overhead} = \lceil((\lceil(length)/216\rceil \cdot 318) + preambleBits)/(BitTimesPerTQ)\rceil$$

where PreambleBits == 66 and BitTimesPerTQ == 165

NOTE–The notation $\lceil x \rceil$ represents a *ceiling* function, which returns the value of its argument x rounded up to the nearest integer.

Opcode–specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

FEC_Overhead(length)

Editors' Note #3 (to be removed prior to release): Function FEC_Overhead is currently incorrect and expected to be corrected during the interim meeting in Munich, Germany in May 2008.

This function calculates the size of additional overhead to be added by the FEC encoder while encoding a frame of size length. Parameter length represents the size of an entire frame including preamble, SFD, DA, SA, Length/Type, and FCS. The function maintains and updates the values of prevTime and payloadBalance. As described in @@Subclause 92.2.3@@, FEC encoder adds 32 parity octets for each block of 216 data or control octets. This is equivalent to an overhead of 4 time_quanta for every 27 time_quanta transmitted. The following formula is used to calculate the overhead:

$$\begin{aligned} \text{payloadBalance} &= \text{payloadBalance} + \text{length} + \text{localTime} - \text{prevTime} \\ \text{FEC\_Overhead} &= \lceil \text{payloadBalance}/27 \rceil \cdot 4 \\ \text{payloadBalance} &= \text{payloadBalance} \div 27 \\ \text{prevTime} &= \text{localTime} + \text{FEC\_Overhead} \end{aligned}$$

NOTE–The notation $\lceil x \rceil$ represents a *ceiling* function, which returns the value of its argument x rounded up to the nearest integer. The notation $a \div b$ represents a *modulo* division of two numbers a and b.

~~ReceiveFrame(DA, SA, Length/Type, data)~~

~~The MAC Sublayer function that is called to receive a frame with the specified parameters.~~

select

This function selects the next Multipoint MAC Control instance allowed to initiate transmission of a frame. The function returns an index to the transmitPending array for which the value is not false. The selection criteria in the presence of multiple active elements in the list is implementation dependent.

SelectFrame()

This function enables the interface, which has a pending frame. If multiple interfaces have frames waiting at the same time, only one interface will be enabled. The selection criteria is not specified, except for the case when some of the pending frames have Length/Type = MAC_Control. In this case, one of the interfaces with a pending MAC Control frame shall be enabled.

sizeof(sdu)

This function returns the size of the sdu in octets.

transmissionPending()

This function returns true if any of the Multipoint MAC Control instances has a frame waiting to be transmitted. The function can be represented as

```
transmissionPending() =   transmitPending[0] +
                          transmitPending[1] +
                          ... +
                          transmitPending[n–1]
```

where n is the total number of Multipoint MAC Control instances.

~~TransmitFrame(DA, SA, Length/Type, data)~~

~~The MAC Sublayer function that is called to transmit a frame with the specified parameters.~~

93.2.2.5 Timers

packet_initiate_timer

This timer is used to delay frame transmission from MAC Control to avoid variable MAC delay while MAC enforces IPG after a previous frame. In addition, when FEC is enabled, this timer increases interframe spacing just enough to accommodate the extra parity data to be added by the FEC encoder.

93.2.2.6 Messages

MA_DATA.indication(DA, SA, m_sdu, receiveStatus)

The service primitive is defined in 2.3.2.

MA_DATA.~~indication~~request (DA, SA, m_sdu~~, receiveStatus~~)

The service primitive is defined in ~~Subcause~~ 2.3.2.

93.2.2.7 State Diagrams

The Multipoint transmission control function in the OLT shall implement state diagram shown in Figure 93–9. Control parser function in the OLT shall implement state diagram shown in Figure 93–10. Control parser function in the ONU shall implement state diagram shown in Figure 93–11. Control multiplexer function in the OLT shall implement state diagram shown in Figure 93–12. Control multiplexer function in the ONU shall implement state diagram shown in Figure 93–13.

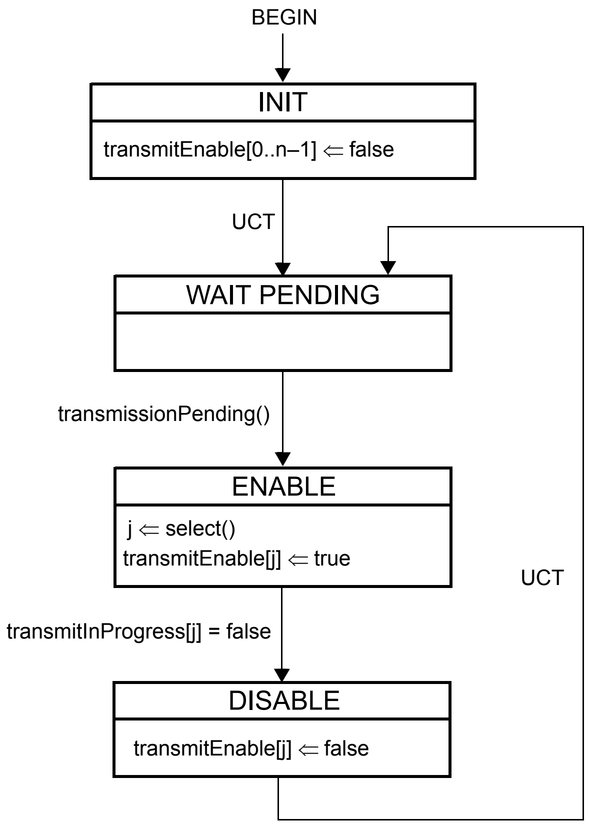
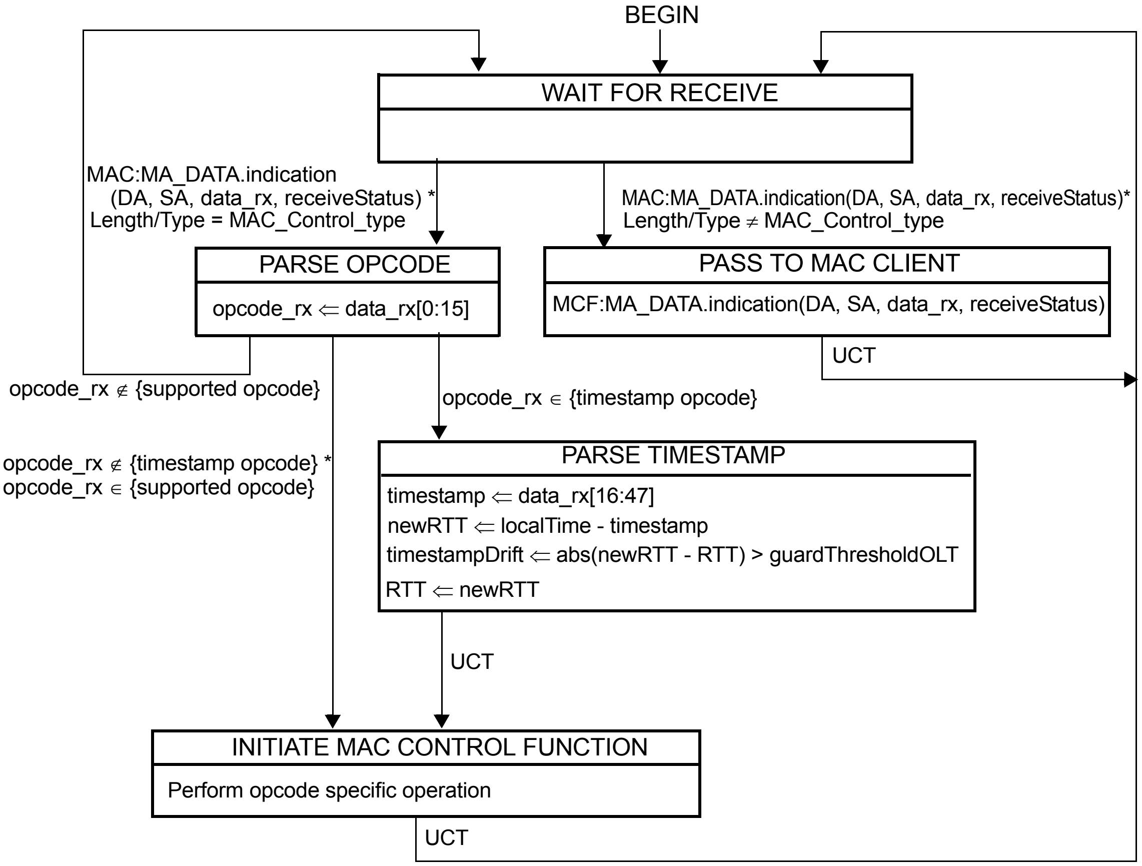


Figure 93–9—OLT Multipoint Transmission Control state diagram

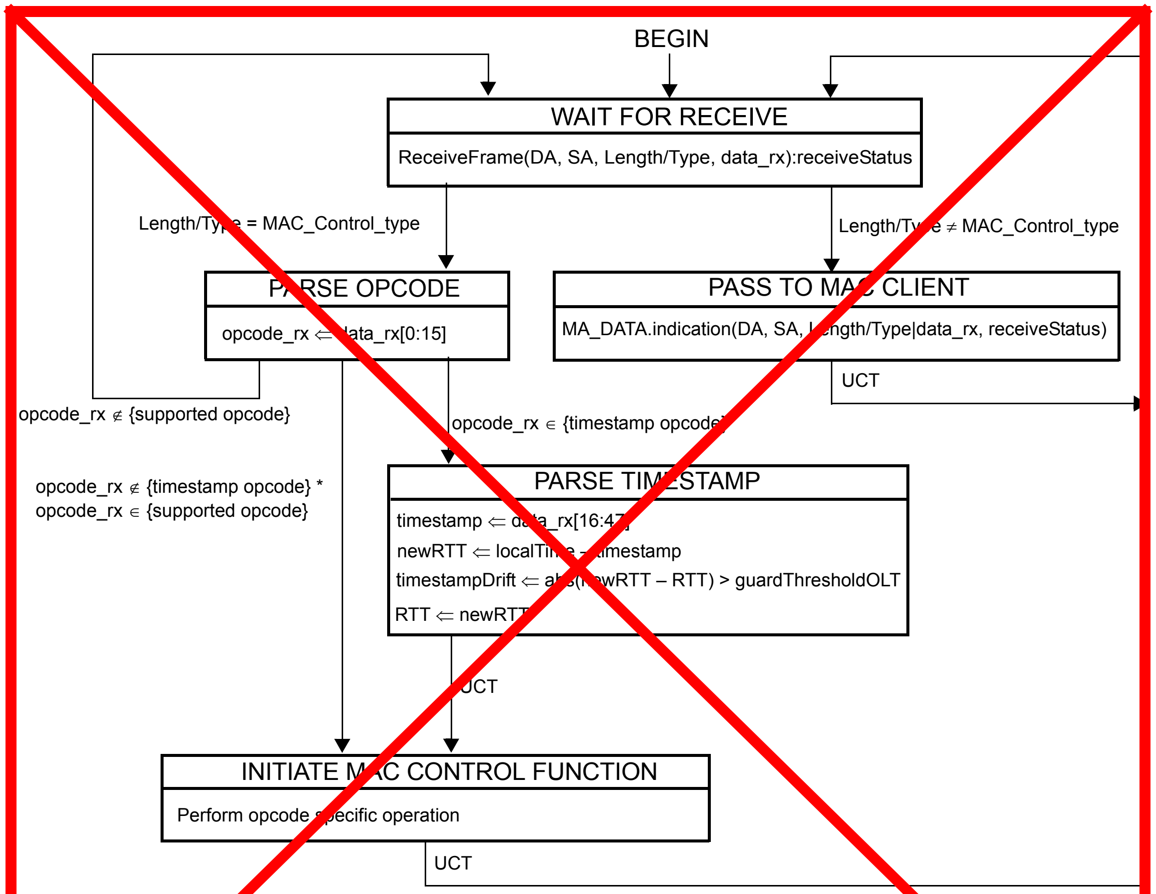

BEGIN
WAIT FOR RECEIVE
MAC:MA_DATA.indication
(DA, SA, data_rx, receiveStatus) *
Length/Type = MAC_Control_type
MAC:MA_DATA.indication(DA, SA, data_rx, receiveStatus)*
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_rx ⇐ data_rx[0:15]
PASS TO MAC CLIENT
MCF:MA_DATA.indication(DA, SA, data_rx, receiveStatus)
UCT
opcode_rx ∉ {supported opcode}
opcode_rx ∈ {timestamp opcode}
opcode_rx ∉ {timestamp opcode} *
opcode_rx ∈ {supported opcode}
PARSE TIMESTAMP
timestamp ⇐ data_rx[16:47]
newRTT ⇐ localTime - timestamp
timestampDrift ⇐ abs(newRTT - RTT) > guardThresholdOLT
RTT ⇐ newRTT
UCT
INITIATE MAC CONTROL FUNCTION
Perform opcode specific operation
UCT

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

NOTE—The opcode-specific operation is launched as a parallel process by the MAC Control sublayer, and not as a synchronous function. Progress of the generic MAC Control Receive state diagram (as shown in this figure) is not implicitly impeded by the launching of the opcode specific function.

Refer to Annex 31A for list of supported opcodes and timestamp opcodes.

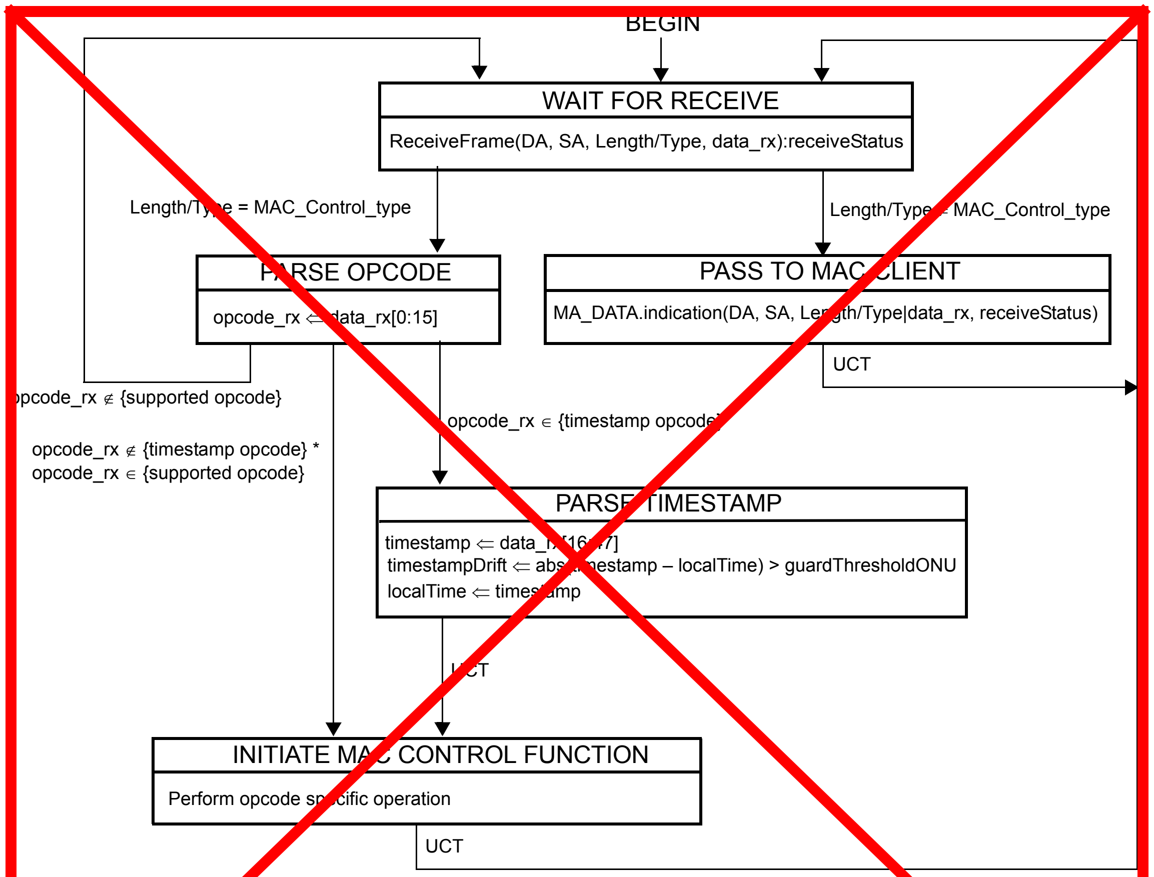
Figure 93–10—OLT Control Parser state diagram


BEGIN
WAIT FOR RECEIVE
ReceiveFrame(DA, SA, Length/Type, data_rx):receiveStatus
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_rx ⇐ data_rx[0:15]
PASS TO MAC CLIENT
MA_DATA.indication(DA, SA, Length/Type|data_rx, receiveStatus)
UCT
opcode_rx ∉ {supported opcode}
opcode_rx ∈ {timestamp opcode}
opcode_rx ∉ {timestamp opcode} *
opcode_rx ∈ {supported opcode}
PARSE TIMESTAMP
timestamp ⇐ data_rx[16:47]
newRTT ⇐ localTime – timestamp
timestampDrift ⇐ abs(newRTT – RTT) > guardThresholdOLT
RTT ⇐ newRTT
UCT
INITIATE MAC CONTROL FUNCTION
Perform opcode specific operation
UCT

~~NOTE–The opcode-specific operation is launched as a parallel process by the MAC Control sublayer, and not as a synchronous function. Progress of the generic MAC Control Receive state diagram (as shown in this figure) is not implicitly impeded by the launching of the opcode specific function.~~

~~Refer to Annex 31A for list of supported opcodes and timestamp opcodes.~~

~~**Figure 93–10—OLT Control Parser state diagram**~~


BEGIN
WAIT FOR RECEIVE
ReceiveFrame(DA, SA, Length/Type, data_rx):receiveStatus
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_rx ⇐ data_rx[0:15]
PASS TO MAC CLIENT
MA_DATA.indication(DA, SA, Length/Type|data_rx, receiveStatus)
UCT
opcode_rx ∉ {supported opcode}
opcode_rx ∈ {timestamp opcode}
opcode_rx ∉ {timestamp opcode} *
opcode_rx ∈ {supported opcode}
PARSE TIMESTAMP
timestamp ⇐ data_rx[16:47]
timestampDrift ⇐ abs(timestamp – localTime) > guardThresholdONU
localTime ⇐ timestamp
UCT
INITIATE MAC CONTROL FUNCTION
Perform opcode specific operation
UCT

NOTE– The opcode-specific operation is launched as a parallel process by the MAC Control sublayer, and not as a synchronous function. Progress of the generic MAC Control Receive state diagram (as shown in this figure) is not implicitly impeded by the launching of the opcode specific function.

Refer to Annex 31A for list of supported opcodes and timestamp opcodes.

Figure 93–11—ONU Control Parser state diagram

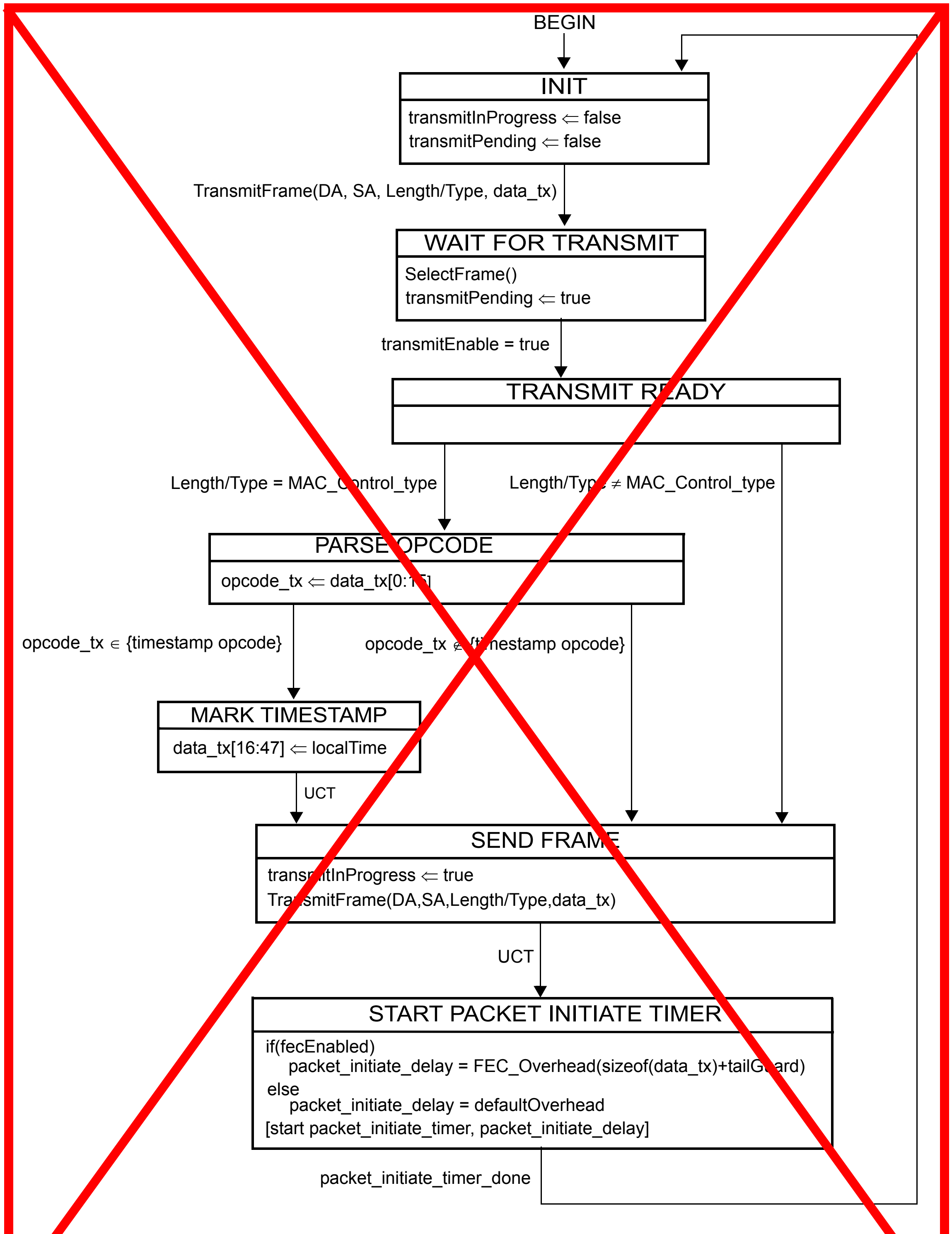

BEGIN
INIT
transmitInProgress ⇐ false
transmitPending ⇐ false
TransmitFrame(DA, SA, Length/Type, data_tx)
WAIT FOR TRANSMIT
SelectFrame()
transmitPending ⇐ true
transmitEnable = true
TRANSMIT READY
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_tx ⇐ data_tx[0:15]
opcode_tx ∈ {timestamp opcode}
opcode_tx ∉ {timestamp opcode}
MARK TIMESTAMP
data_tx[16:47] ⇐ localTime
UCT
SEND FRAME
transmitInProgress ⇐ true
TransmitFrame(DA,SA,Length/Type,data_tx)
UCT
START PACKET INITIATE TIMER
if(fecEnabled)
packet_initiate_delay = FEC_Overhead(sizeof(data_tx)+tailGuard)
else
packet_initiate_delay = defaultOverhead
[start packet_initiate_timer, packet_initiate_delay]
packet_initiate_timer_done

Figure 93–12—OLT Control Multiplexer state diagram

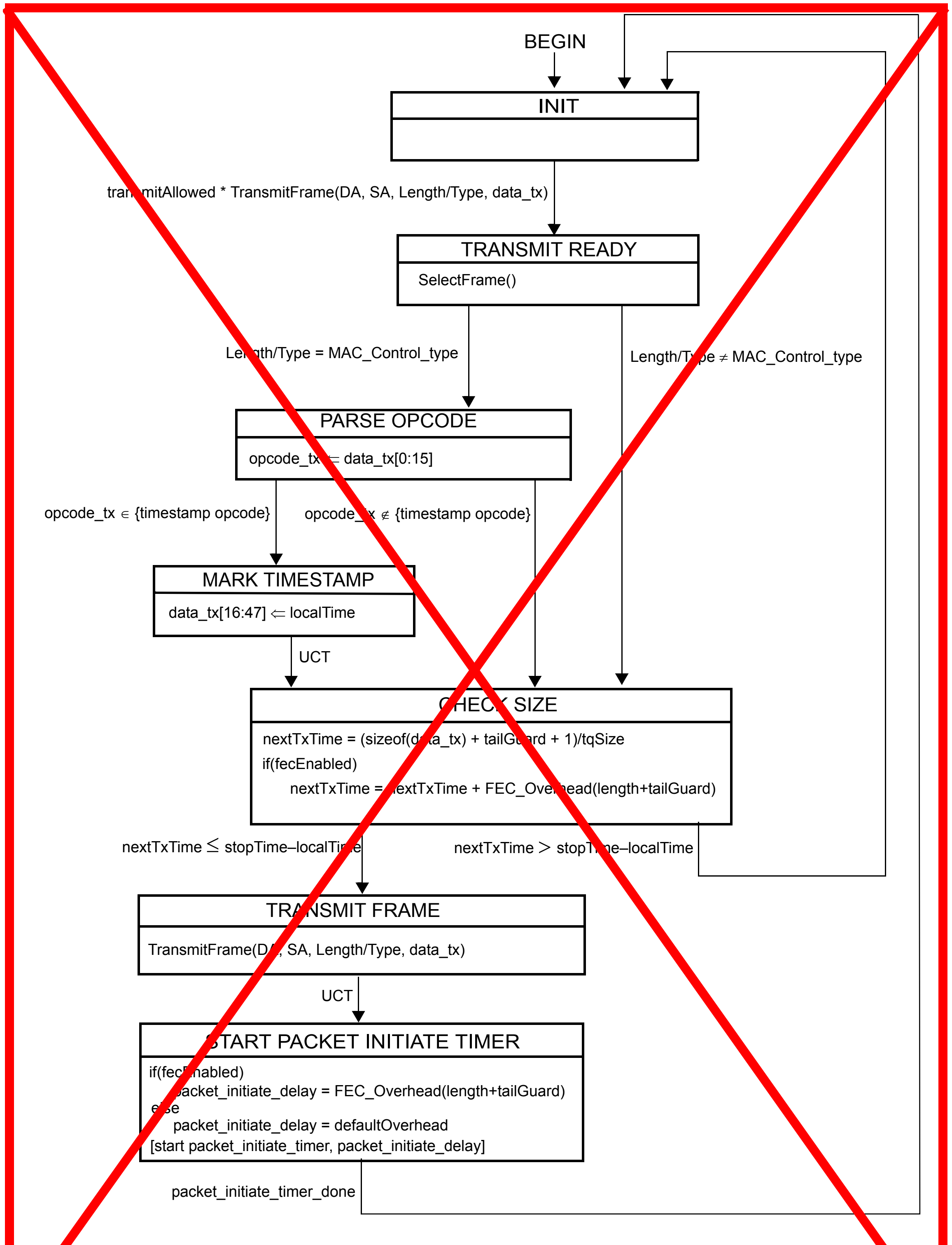
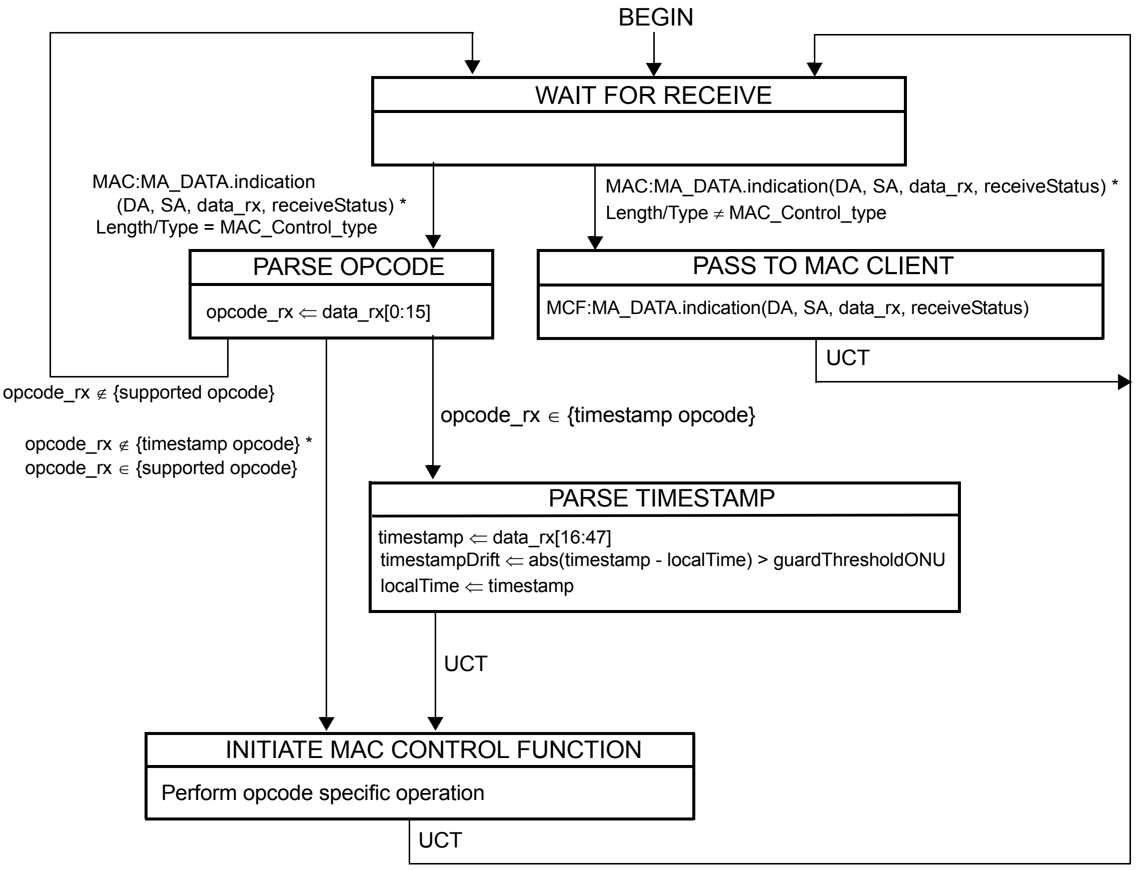

BEGIN
INIT
transmitAllowed * TransmitFrame(DA, SA, Length/Type, data_tx)
TRANSMIT READY
SelectFrame()
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_tx ⇐ data_tx[0:15]
opcode_tx ∈ {timestamp opcode}
opcode_tx ∉ {timestamp opcode}
MARK TIMESTAMP
data_tx[16:47] ⇐ localTime
UCT
CHECK SIZE
nextTxTime = (sizeof(data_tx) + tailGuard + 1)/tqSize
if(fecEnabled)
nextTxTime = nextTxTime + FEC_Overhead(length+tailGuard)
nextTxTime ≤ stopTime–localTime
nextTxTime > stopTime–localTime
TRANSMIT FRAME
TransmitFrame(DA, SA, Length/Type, data_tx)
UCT
START PACKET INITIATE TIMER
if(fecEnabled)
packet_initiate_delay = FEC_Overhead(length+tailGuard)
else
packet_initiate_delay = defaultOverhead
[start packet_initiate_timer, packet_initiate_delay]
packet_initiate_timer_done

Figure 93–13—ONU Control Multiplexer state diagram

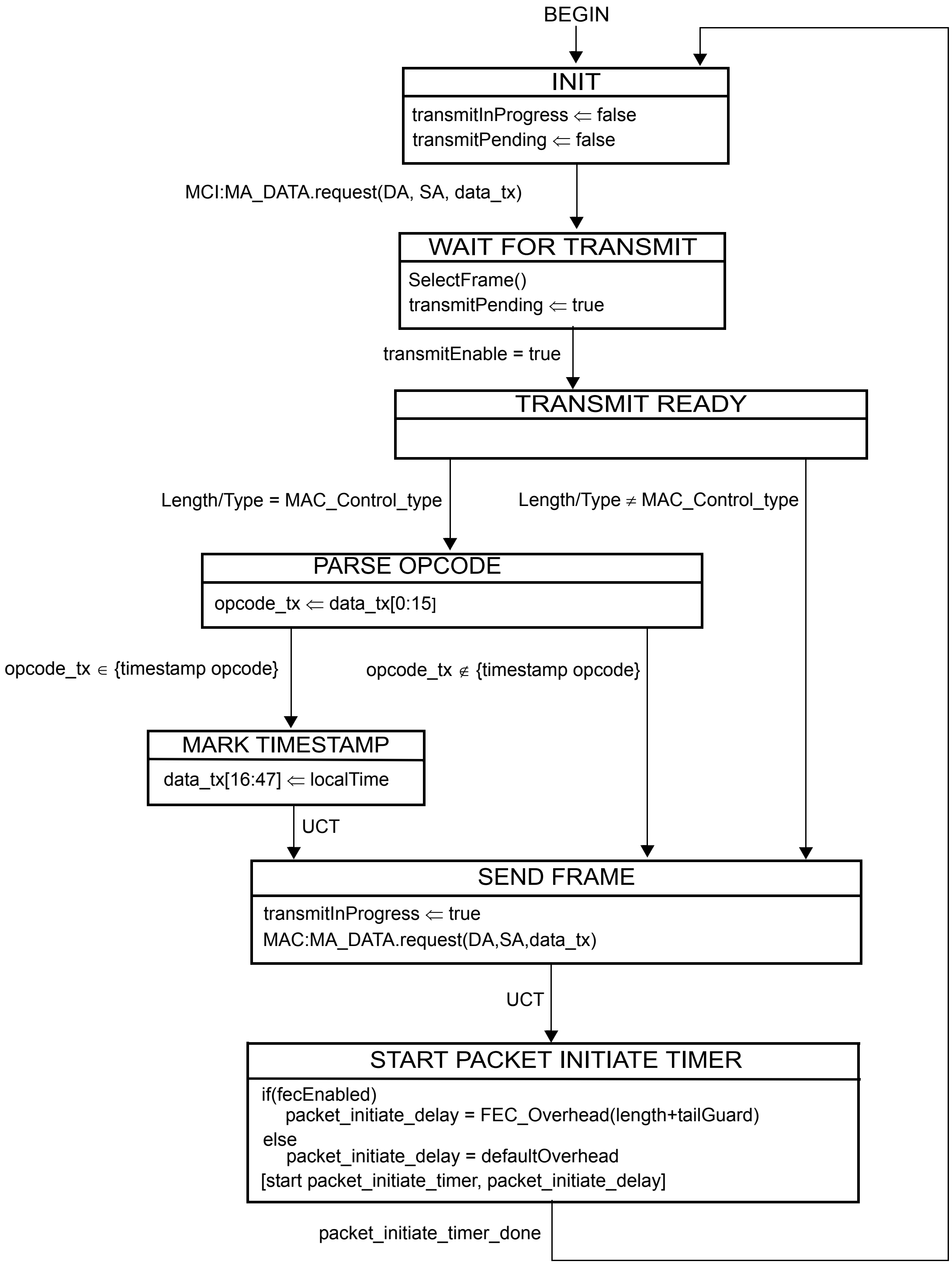

BEGIN
WAIT FOR RECEIVE
MAC:MA_DATA.indication
(DA, SA, data_rx, receiveStatus) *
Length/Type = MAC_Control_type
MAC:MA_DATA.indication(DA, SA, data_rx, receiveStatus) *
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_rx ⇐ data_rx[0:15]
PASS TO MAC CLIENT
MCF:MA_DATA.indication(DA, SA, data_rx, receiveStatus)
UCT
opcode_rx ∉ {supported opcode}
opcode_rx ∈ {timestamp opcode}
opcode_rx ∉ {timestamp opcode} *
opcode_rx ∈ {supported opcode}
PARSE TIMESTAMP
timestamp ⇐ data_rx[16:47]
timestampDrift ⇐ abs(timestamp - localTime) > guardThresholdONU
localTime ⇐ timestamp
UCT
INITIATE MAC CONTROL FUNCTION
Perform opcode specific operation
UCT

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

NOTE—The opcode-specific operation is launched as a parallel process by the MAC Control sublayer, and not as a synchronous function. Progress of the generic MAC Control Receive state diagram (as shown in this figure) is not implicitly impeded by the launching of the opcode specific function.

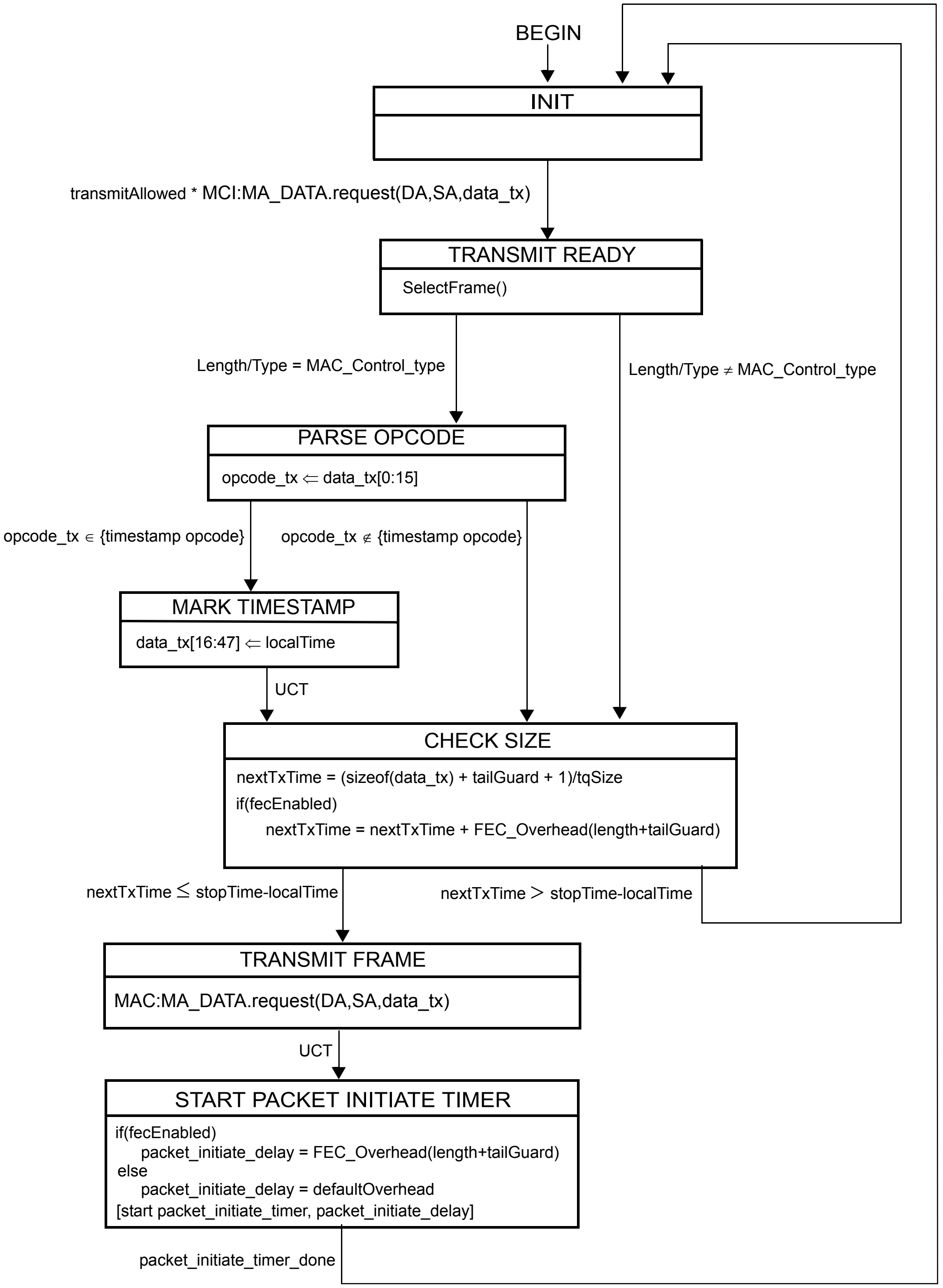
Refer to Annex 31A for list of supported opcodes and timestamp opcodes.

Figure 93–11—ONU Control Parser state diagram


BEGIN
INIT
transmitInProgress ⇐ false
transmitPending ⇐ false
MCI:MA_DATA.request(DA, SA, data_tx)
WAIT FOR TRANSMIT
SelectFrame()
transmitPending ⇐ true
transmitEnable = true
TRANSMIT READY
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_tx ⇐ data_tx[0:15]
opcode_tx ∈ {timestamp opcode}
opcode_tx ∉ {timestamp opcode}
MARK TIMESTAMP
data_tx[16:47] ⇐ localTime
UCT
SEND FRAME
transmitInProgress ⇐ true
MAC:MA_DATA.request(DA,SA,data_tx)
UCT
START PACKET INITIATE TIMER
if(fecEnabled)
packet_initiate_delay = FEC_Overhead(length+tailGuard)
else
packet_initiate_delay = defaultOverhead
[start packet_initiate_timer, packet_initiate_delay]
packet_initiate_timer_done

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

Figure 93–12—OLT Control Multiplexer state diagram


BEGIN
INIT
transmitAllowed * MCI:MA_DATA.request(DA,SA,data_tx)
TRANSMIT READY
SelectFrame()
Length/Type = MAC_Control_type
Length/Type ≠ MAC_Control_type
PARSE OPCODE
opcode_tx ⇐ data_tx[0:15]
opcode_tx ∈ {timestamp opcode}
opcode_tx ∉ {timestamp opcode}
MARK TIMESTAMP
data_tx[16:47] ⇐ localTime
UCT
CHECK SIZE
nextTxTime = (sizeof(data_tx) + tailGuard + 1)/tqSize
if(fecEnabled)
nextTxTime = nextTxTime + FEC_Overhead(length+tailGuard)
nextTxTime ≤ stopTime-localTime
nextTxTime > stopTime-localTime
TRANSMIT FRAME
MAC:MA_DATA.request(DA,SA,data_tx)
UCT
START PACKET INITIATE TIMER
if(fecEnabled)
packet_initiate_delay = FEC_Overhead(length+tailGuard)
else
packet_initiate_delay = defaultOverhead
[start packet_initiate_timer, packet_initiate_delay]
packet_initiate_timer_done

Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

Figure 93–13—ONU Control Multiplexer state diagram

93.3 Multipoint Control Protocol (MPCP)

As depicted in Figure 93–3, the Multipoint MAC Control functional block comprises the following functions:

a) *Discovery Processing*. This block manages the discovery process, through which an ONU is discovered and registered with the network while compensating for RTT.
b) *Report Processing*. This block manages the generation and collection of report messages, through which bandwidth requirements are sent upstream from the ONU to the OLT.
c) *Gate Processing*. This block manages the generation and collection of gate messages, through which multiplexing of multiple transmitters is achieved.

As depicted in Figure 93–3, the layered system may instantiate multiple MAC entities, using a single Physical Layer. Each instantiated MAC communicates with an instance of the opcode specific functional blocks through the Multipoint MAC Control. In addition some global variables are shared across the multiple instances. Common state control is used to synchronize the multiple MACs using MPCP procedures. Operation of the common state control is generally considered outside the scope of this document.

93.3.1 Principles of Multipoint Control Protocol

Multipoint MAC Control enables a MAC Client to participate in a point–to–multipoint optical network by allowing it to transmit and receive frames as if it was connected to a dedicated link. In doing so, it employs the following principles and concepts:

a) A MAC client transmits and receives frames through the Multipoint MAC Control sublayer.
b) The Multipoint MAC Control decides when to allow a frame to be transmitted using the client interface Control Multiplexer.
c) Given a transmission opportunity, the MAC Control may generate control frames that would be transmitted in advance of the MAC Client's frames, utilizing the inherent ability to provide higher priority transmission of MAC Control frames over MAC Client frames.
d) Multiple MACs operate on a shared medium by allowing only a single MAC to transmit upstream at any given time across the network using a time–division multiple access (TDMA) method.
e) Such gating of transmission is orchestrated through the Gate Processing function.
f) New devices are discovered in the network and allowed transmission through the Discovery Processing function.
g) Fine control of the network bandwidth distribution can be achieved using feedback mechanisms supported in the Report Processing function.
h) The operation of P2MP network is asymmetrical, with the OLT assuming the role of master, and the ONU assuming the role of slave.

93.3.2 Compatibility considerations

93.3.2.1 PAUSE operation

Even though MPCP is compatible with flow control, optional use of flow control may not be efficient in the case of large propagation delay. If flow control is implemented, then the timing constraints in Clause 31B supplement the constraints found at Subclause 93.3.2.4.

NOTE—MAC at an ONU can receive frames from unicast channel and SCB channel. If the SCB channel is used to broadcast data frames to multiple ONUs, the ONU's MAC may continue receiving data frames from SCB channel even after the ONU has issued a PAUSE request to its unicast remote–end.

93.3.2.2 Optional Shared LAN Emulation

By combining P2PE, suitable filtering rules at the ONU, and suitable filtering and forwarding rules at the OLT, it is possible to emulate an efficient shared LAN. Support for shared LAN emulation is optional, and requires an additional layer above the MAC, which is out of scope for this document. Thus, shared LAN emulation is introduced here for informational purposes only.

Specific behaviour of the filtering layer at the RS is specified in Subclause 65.1.3.3.2.

93.3.2.3 Multicast and single copy broadcast support

In the downstream direction, the PON is a broadcast medium. In order to make use of this capability for forwarding broadcast frames from the OLT to multiple recipients without frame duplication for each ONU, the SCB support is introduced.

The OLT has at least one MAC associated with every ONU. In addition one more MAC at the OLT is marked as the SCB MAC. The SCB MAC handles all downstream broadcast traffic, but is never used in the upstream direction for client traffic, except for client registration. Optional higher layers may be implemented to perform selective broadcast of frames. Such layers may require additional MACs (multicast MACs) to be instantiated in the OLT for some or all ONUs increasing the total number of MACs beyond the number of ONUs + 1.

When connecting the SCB MAC to an 802.1D bridge port it is possible that loops may be formed due to the broadcast nature. Thus it is recommended that this MAC not be connected to an 802.1D bridge port.

Configuration of SCB channels as well as filtering and marking of frames for support of SCB is defined in Clause @@92.1.2.3.3.2@@ for 10 Gb/s EPON compliant Reconciliation Sublayers.

93.3.2.4 Delay requirements

The MPCP protocol relies on strict timing based on distribution of timestamps. A compliant implementation needs to guarantee a constant delay through the MAC and PHY in order to maintain the correctness of the timestamping mechanism. The actual delay is implementation dependent, however, a complying implementation shall maintain a delay variation of no more than 1 time_quantum through the implemented MAC stack.

The OLT shall not grant less than 1024 time_quanta into the future, in order to allow the ONU processing time when it receives a gate message. The ONU shall process all messages in less than this period. The OLT shall not issue more than one message every 1024 time_quanta to a single ONU. The unit of time_quantum is defined as 16 ns.

93.3.3 Discovery Processing

Discovery is the process whereby newly connected or off–line ONUs are provided access to the PON. The process is driven by the OLT, which periodically makes available Discovery Time Windows during which off–line ONUs are given the opportunity to make themselves known to the OLT. The periodicity of these windows is unspecified and left up to the implementor. The OLT signifies that a discovery period is occurring by broadcasting a discovery GATE MPCPDU, which includes the starting time and length of the discovery window, along with the Discovery Information flag field, as defined in Clause 93.3.6.1. With the appropriate settings of individual flags contained in this 16 bit wide field, the OLT notifies all the ONUs about its upstream and downstream channel transmission capabilities. Note that the OLT may simultaneously support more than one data rate in the given transmission direction.

Off–line ONUs, upon receiving a Discovery GATE MPCPDU, wait for the period to begin and then transmit a REGISTER_REQ MPCPDU to the OLT. Discovery windows are unique in that they are the only times where multiple ONUs can access the PON simultaneously, and transmission overlap can occur. In order to reduce transmission overlaps, a contention algorithm is used by all ONUs. Measures are taken to reduce the probability for overlaps by artificially simulating a random distribution of distances from the OLT. Each ONU shall wait a random amount of time before transmitting the REGISTER_REQ MPCPDU that is shorter than the length of the discovery time window. It should be noted that multiple valid REGISTER_REQ MPCPDUs can be received by the OLT during a single discovery time period. Included in the REGISTER_REQ MPCPDU is the ONU's MAC address and number of maximum pending grants. Additionally, a registering ONU notifies the OLT on its transmission capabilities in the upstream and downstream channels by setting appropriately the flags in the Discovery Information field, as specified in Subclause 93.3.6.3.

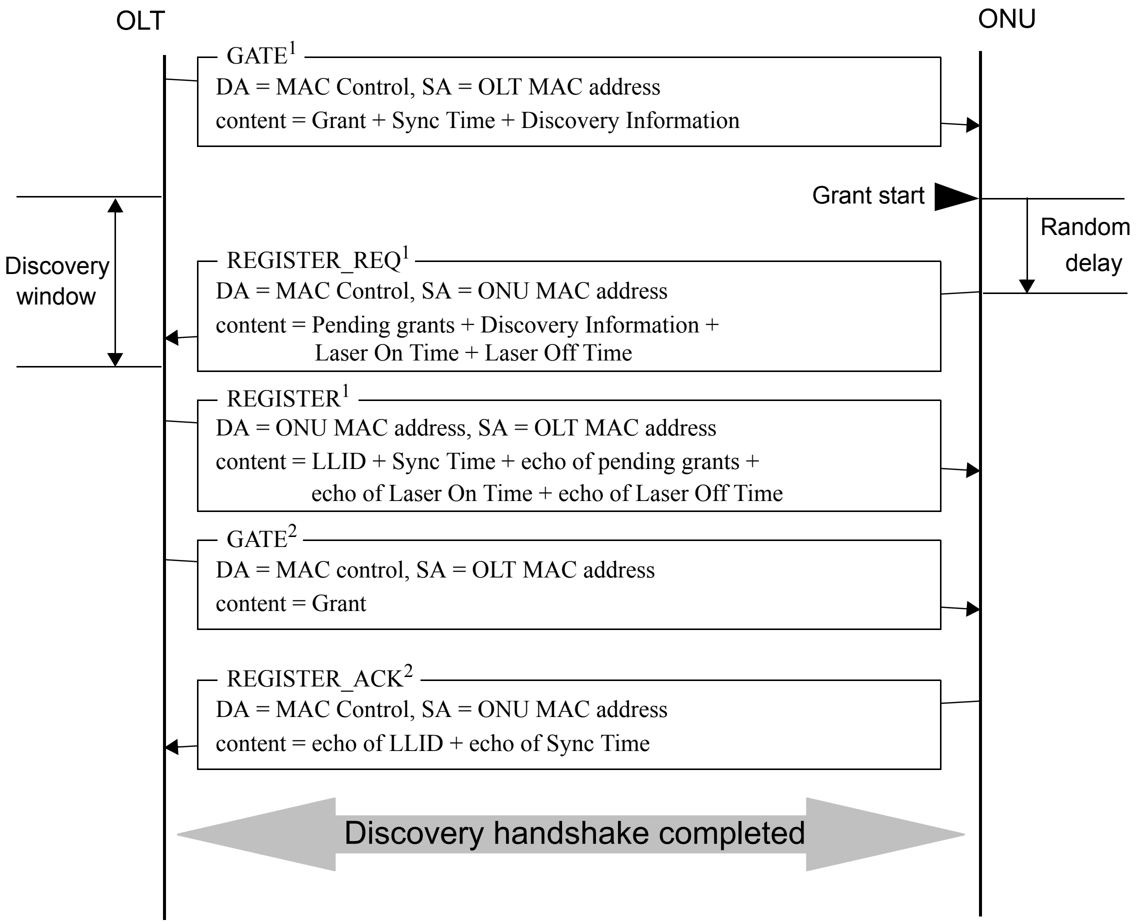
Note that even though a compliant ONU is not prohibited from supporting more than one data rate in any transmission channel, it is expected that a single supported data rate for upstream and downstream channel will be indicated in the Discovery Information field. Moreover, in order to assure maximum utilization of the upstream channel and to decrease the required size of the guard band between individual data bursts, the registering ONU shall notify the OLT of the laser on / off times, by setting appropriate values in the Laser On Time and Laser Off Time fields, where both values are expressed in the units of time_quanta.

Upon receipt of a valid REGISTER_REQ MPCPDU, the OLT registers the ONU, allocating and assigning new port identities (LLIDs), and bonding corresponding MACs to the LLIDs.

The next step in the process is for the OLT to transmit a REGISTER MPCPDU to the newly discovered ONU, which contains the ONU's LLID, and the OLT's required synchronization time. Also, the OLT echoes the maximum number of pending grants, laser on time and laser off time. Note that the echoed parameter values i.e. required OLT synchronization time and laser on/off times are delivered to the registering ONU for confirmation purposes only and their utilization is not prescribed in this specification.

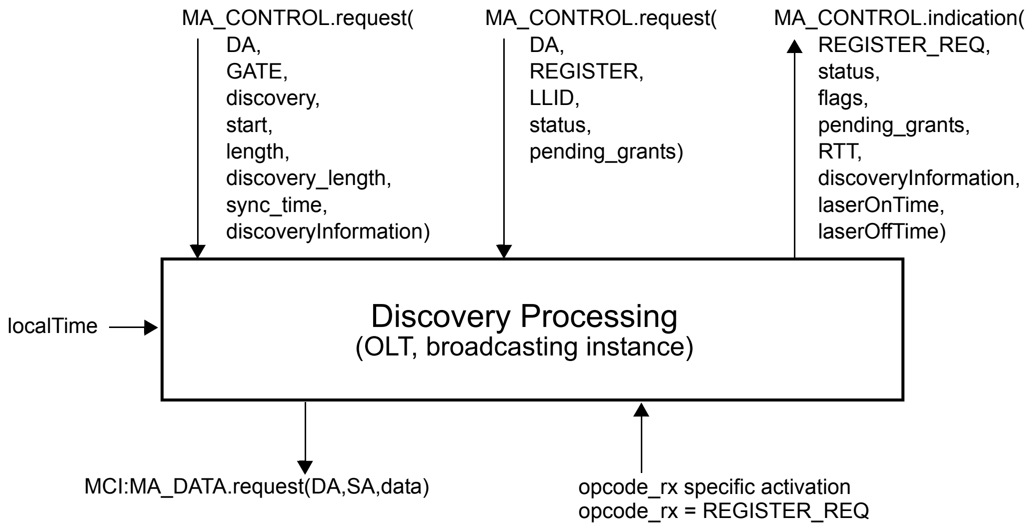
The OLT now has enough information to schedule the ONU for access to the PON and transmits a standard GATE message allowing the ONU to transmit a REGISTER_ACK. Upon receipt of the REGISTER_ACK, the discovery process for that ONU is complete, the ONU is registered and normal message traffic can begin. It is the responsibility of Layer Management to perform the MAC bonding, and start transmission from/to the newly registered ONU. The discovery message exchange is illustrated in Figure 93–14.

There may exist situations when the OLT requires that an ONU go through the discovery sequence again and reregister. Similarly, there may be situations where an ONU needs to inform the OLT of its desire to deregister. The ONU can then reregister by going through the discovery sequence. For the OLT, the REGISTER message may indicate a value, Reregister or Deregister, that if either is specified will force the receiving ONU into reregistering. For the ONU, the REGISTER_REQ message contains the Deregister bit that signifies to the OLT that this ONU should be deregistered.


OLT
ONU
GATE[1]
DA = MAC Control, SA = OLT MAC address
content = Grant + Sync Time + Discovery Information
Grant start
Random delay
Discovery window
REGISTER_REQ[1]
DA = MAC Control, SA = ONU MAC address
content = Pending grants + Discovery Information + Laser On Time + Laser Off Time
REGISTER[1]
DA = ONU MAC address, SA = OLT MAC address
content = LLID + Sync Time + echo of pending grants + echo of Laser On Time + echo of Laser Off Time
GATE[2]
DA = MAC control, SA = OLT MAC address
content = Grant
REGISTER_ACK[2]
DA = MAC Control, SA = ONU MAC address
content = echo of LLID + echo of Sync Time
Discovery handshake completed

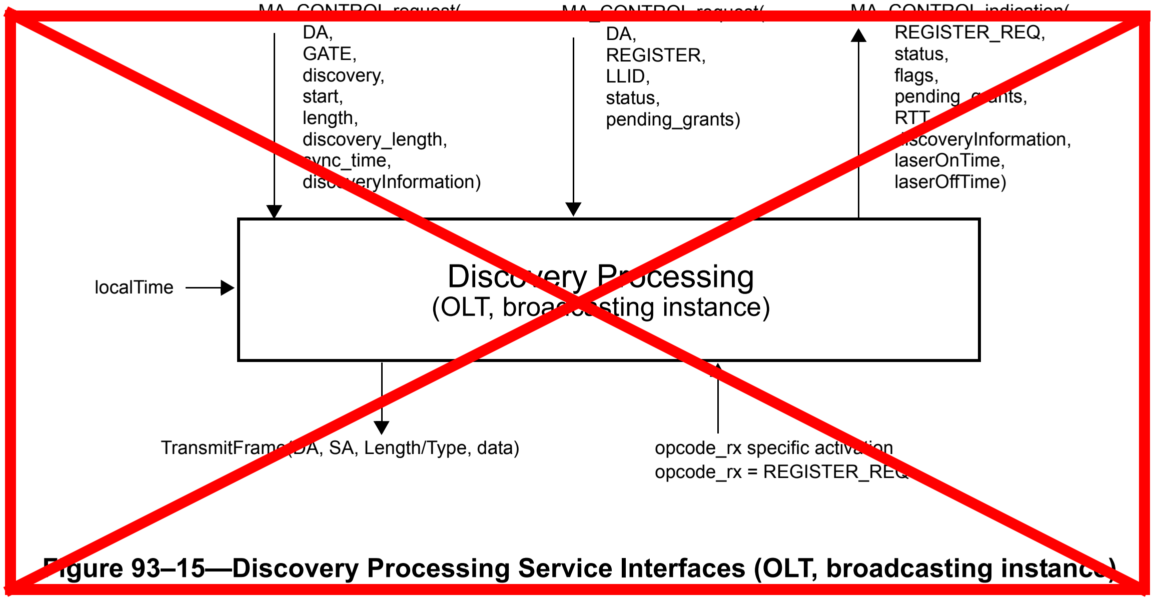
[1] Messages sent on a broadcast channel
[2] Messages sent on unicast channels

Figure 93–14—Discovery Handshake Message Exchange



Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–15—Discovery Processing service interfaces (OLT, broadcasting instance)



~~**Figure 93–15—Discovery Processing Service Interfaces (OLT, broadcasting instance)**~~

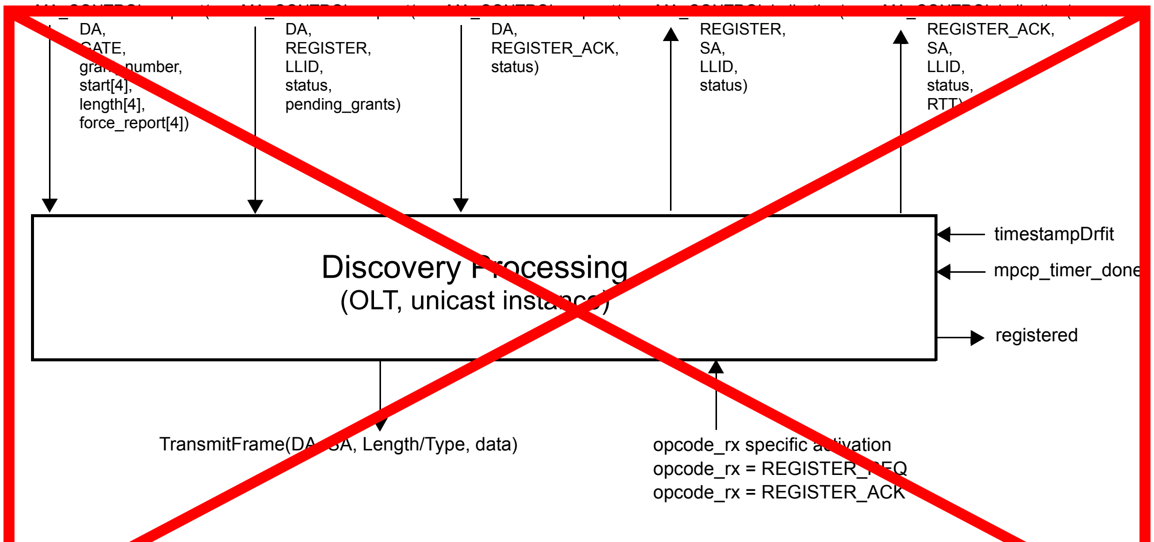


Figure 93–16—Discovery Processing Service Interfaces (OLT, unicasting instance)

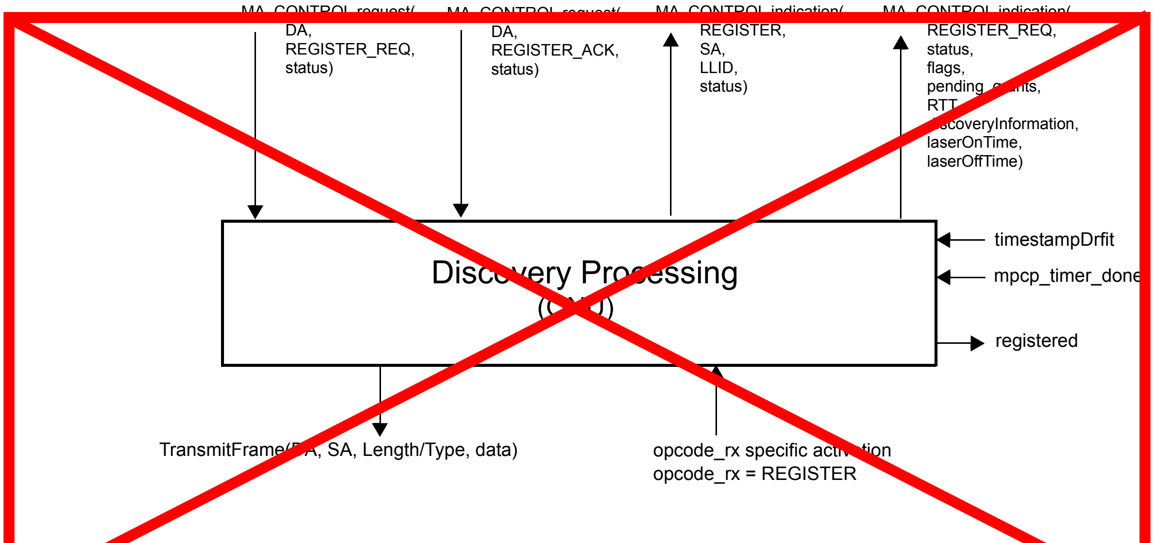
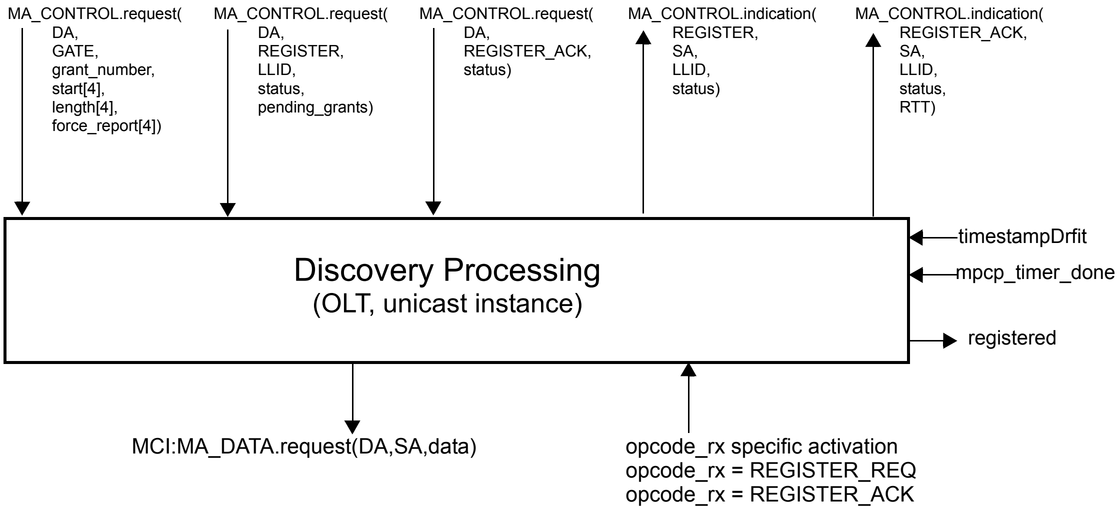
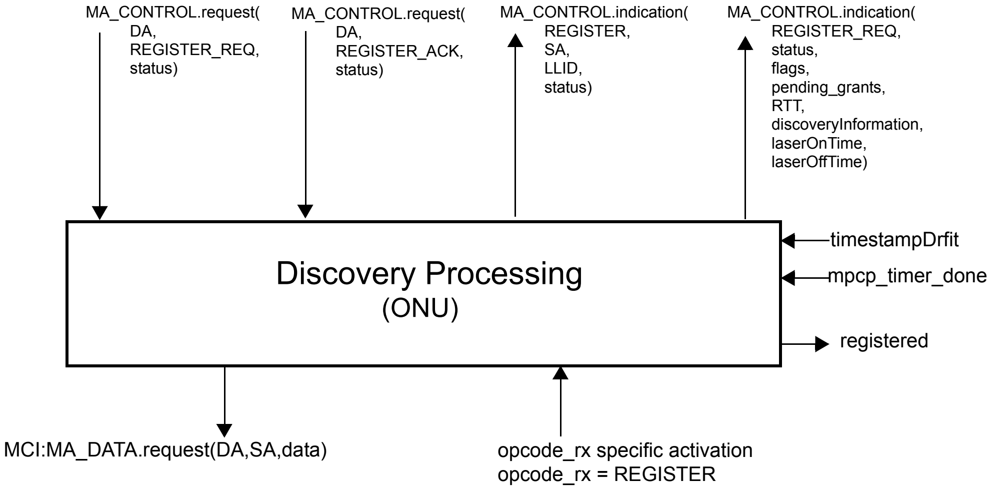


Figure 93–17—Discovery Processing Service Interfaces (ONU)



Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–16—Discovery Processing service interfaces (OLT, unicasting instance)



Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–17—Discovery Processing service interfaces (ONU)

93.3.3.1 Constants

No constants defined.

93.3.3.2 Variables

BEGIN
This variable is defined in Subclause 93.2.2.3.

data_rx

This variable is defined in Subclause 93.2.2.3.

data_tx

This variable is defined in Subclause 93.2.2.3.

grantEndTime

This variable holds the time at which the OLT expects the ONU grant to complete. Failure of a REGISTER_ACK message from an ONU to arrive at the OLT before grantEndTime is a fatal error in the discovery process, and causes registration to fail for the specified ONU, who may then retry to register. The value of grantEndTime is measured in units of time_quantum.

TYPE: 32–bit unsigned

insideDiscoveryWindow

This variable holds the current status of the discovery window. It is set to true when the discovery window opens, and is set to false when the discovery window closes.

TYPE: Boolean

laserOffTime

This variable holds the time required to terminate the laser. It counts in time_quanta units the time period required for turning off the PMD, as specified in Subclause 60.7.13.1.

TYPE: 8 bit unsigned
VALUE: 0x20 (512 ns, default value)

laserOnTime

This variable holds the time required to initiate the PMD. It counts in time_quanta units the time period required for turning on the PMD, as specified in Subclause 60.7.13.1.

TYPE: 8 bit unsigned
VALUE: 0x20 (512 ns, default value)

localTime

This variable is defined in Subclause 93.2.2.2.

opcode_rx

This variable is defined in Subclause 93.2.2.3.

opcode_tx

This variable is defined in Subclause 93.2.2.3.

pendingGrants

This variable holds the maximum number of pending grants that an ONU is able to queue.

TYPE: 16 bit unsigned

registered

This variable holds the current result of the Discovery Process. It is set to true once the discovery process is complete and registration is acknowledged.

TYPE: Boolean

syncTime

This variable holds the time required to stabilize the receiver at the OLT. It counts time_quanta units from the point where transmission output is stable to the point where synchronization has been achieved. The value of syncTime includes gain adjustment interval ($T_{receiver\_settling}$), clock synchronization interval (T_{cdr}), and code–group alignment interval ($T_{code\_group\_align}$), as specified in Subclause 60.7.13.2. The OLT conveys the value of syncTime to ONUs in Discovery GATE and REGISTER messages. During the synchronization time an asymmetric 10/1 Gb/s ONU transmits only IDLE patterns, and a symmetric 10 Gb/s ONU sends a synchronization pattern of 0x55 (transmission bit sequence 1010 ...) followed by a burst delimiter and idle blocks as defined in Subclause @@92.2.3.5@@.

TYPE: 16 bit unsigned

timestampDrift

This variable is defined in Subclause 93.2.2.3.

93.3.3.3 Functions

~~ReceiveFrame(DA, SA, Length/Type, data)~~

~~This function is defined in Subclause 93.2.2.4.~~

~~TransmitFrame(DA, SA, Length/Type, data)~~

~~This function is defined in Subclause 93.2.2.4.~~

None.

93.3.3.4 Timers

discovery_window_size_timer

This timer is used to wait for the event signaling the end of the discovery window.

VALUE: The timer value is set dynamically based on the parameters received in a DISCOVERY GATE message.

mpcp_timer

This timer is used to measure the arrival rate of MPCP frames in the link. Failure to receive frames is considered a fatal fault and leads to deregistration.

93.3.3.5 Messages

MA_DATA.indication(DA, SA, m_sdu, receiveStatus)

The service primitive is defined in 2.3.2.

MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, GATE, discovery, start, length, discovery_length, sync_time, discoveryInformation)

The service primitive used by the MAC Control client at the OLT to initiate the Discovery Process. This primitive takes the following parameters:

DA:	multicast or unicast MAC address.
GATE:	opcode for GATE MPCPDU as defined in Table 31A–1.
discovery:	flag specifying that the given GATE message is to be used for discovery only.
start:	start time of the discovery window.
length:	length of the grant given for discovery.
discovery_length:	length of the discovery window process.
sync_time:	the time interval required to stabilize the receiver at the OLT.
discoveryInformation:	speed(s) the OLT is capable of receiving and speed(s) at which the discovery window will open for.

MA_CONTROL.request(DA, GATE, grant_number, start[4], length[4], force_report[4])

This service primitive is used by the MAC Control client at the OLT to issue the GATE message to an ONU. This primitive takes the following parameters:

DA:	multicast MAC Control address as defined in Annex 31B.
GATE:	opcode for GATE MPCPDU as defined in Table 31A–1.
grant_number:	number of grants issued with this GATE message. The number of grants ranges from 0 to 4.
start[4]:	start times of the individual grants. Only the first grant_number elements of the array are used.

length[4]: lengths of the individual grants. Only the first grant_number elements of the array are used.

force_report[4]: flags indicating whether a REPORT message should be generated in the corresponding grant. Only the first grant_number elements of the array are used.

MA_CONTROL.request(DA, REGISTER_REQ, status)

The service primitive used by a client at the ONU to request the Discovery Process to perform a registration. This primitive takes the following parameters:

DA: multicast MAC Control address as defined in Annex 31B.

REGISTER_REQ: opcode for REGISTER_REQ MPCPDU as defined in Table 31A–1.

status: This parameter takes on the indication supplied by the flags field in the REGISTER_REQ MPCPDU as defined in Table 93–4.

MA_CONTROL.indication(REGISTER_REQ, status, flags, pending_grants, RTT, discoveryInformation, laserOnTime, laserOffTime)

The service primitive issued by the Discovery Process to notify the client and Layer Management that the registration process is in progress. This primitive takes the following parameters:

REGISTER_REQ: opcode for REGISTER_REQ MPCPDU as defined in Table 31A–1.

status: This parameter holds the values incoming or retry. Value incoming is used at the OLT to signal that a REGISTER_REQ message was received successfully. The value retry is used at the ONU to signal to the client that a registration attempt failed and will be repeated.

flags: This parameter holds the contents of the flags field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery Process is in the OLT.

pending_grants: This parameters holds the contents of the pending_grants field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the OLT.

RTT: The measured round trip time to/from the ONU is returned in this parameter. RTT is stated in time_quanta units. This parameter holds a valid value only when the primitive is generated by the Discovery Process in the OLT.

discoveryInformation: This parameter holds the contents of the discovery information field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery ~~process~~ Process in the OLT.

laserOnTime: This parameter holds the contents of the laserOn field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery ~~process~~ Process in the OLT.

laserOffTime: This parameter holds the contents of the laserOff field in the REGISTER_REQ message. This parameter holds a valid value only when the primitive is generated by the Discovery ~~process~~ Process in the OLT.

MA_CONTROL.request(DA, REGISTER, LLID, status, pending_grants)

The service primitive used by the MAC Control client at the OLT to initiate acceptance of an ONU. This primitive takes the following parameters:

DA: Unicast MAC address or multicast MAC Control address as defined in Annex 31B.
REGISTER: opcode for REGISTER MPCPDU as defined in Table 31A–1.
LLID: This parameter holds the logical link identification number assigned by the MAC Control client.
status: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in Table 93–6.
pending_grants: This parameters echoes back the pending_grants field that was previously received in the REGISTER_REQ message.

MA_CONTROL.indication(REGISTER, SA, LLID, status)

This service primitive is issued by the Discovery Process at the OLT or an ONU to notify the MAC Control client and Layer Management of the result of the change in registration status. This primitive takes the following parameters:

REGISTER: opcode for REGISTER MPCPDU as defined in Table 31A–1.
SA This parameter represents is the MAC address of the OLT.
LLID This parameter holds the logical link identification number assigned by the MAC Control client.
status This parameter holds the value of accepted/denied/deregistered/reregistered.

MA_CONTROL.request(DA, REGISTER_ACK, status)

This service primitive is issued by the MAC Control clients at the ONU and the OLT to acknowledge the registration. This primitive takes the following parameters:

DA: multicast MAC Control address as defined in Annex 31B.
REGISTER_ACK: opcode for REGISTER_ACK MPCPDU as defined in Table 31A–1.
status: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in Table 93–7.

MA_CONTROL.indication(REGISTER_ACK, SA, LLID, status, RTT)

This service primitive is issued by the Discovery Process at the OLT to notify the client and Layer Management that the registration process has completed. This primitive takes the following parameters:

REGISTER_ACK: opcode for REGISTER_ACK MPCPDU as defined in Table 31A–1.
SA This parameter represents the MAC address of the reciprocating device (ONU address at the OLT, and OLT address at the ONU).
LLID This parameter holds the logical link identification number assigned by the MAC Control client.
status This parameter holds the value of accepted/denied/reset/deregistered.

RTT
: The measured round trip time to/from the ONU is returned in this parameter. RTT is stated in time_quanta units. This parameter holds a valid value only when the invoking Discovery Process is in the OLT

Opcode–specific function(opcode)
: Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

93.3.3.6 State Diagram

Discovery process in the OLT shall implement the discovery window setup state diagram shown in Figure 93–18, request processing state diagram as shown in Figure 93–19, register processing state diagram as shown in Figure 93–20, and final registration state diagram as shown in Figure 93–21. The discovery process in the ONU shall implement registration state diagram as shown in Figure 93–22.

Instantiation of state diagrams as described in Figure 93–18, Figure 93–19, and Figure 93–20 is performed only at the Multipoint MAC Control instances attached to the appropriate broadcast LLID(s) (0x7FFF and/or 0x7FFE for 1 Gb/s and 10 Gb/s EPON, respectively). Instantiation of state diagrams as described in Figure 93–21 and Figure 93–22 is performed for every Multipoint MAC Control instance, except the instance attached to the broadcast channel. .

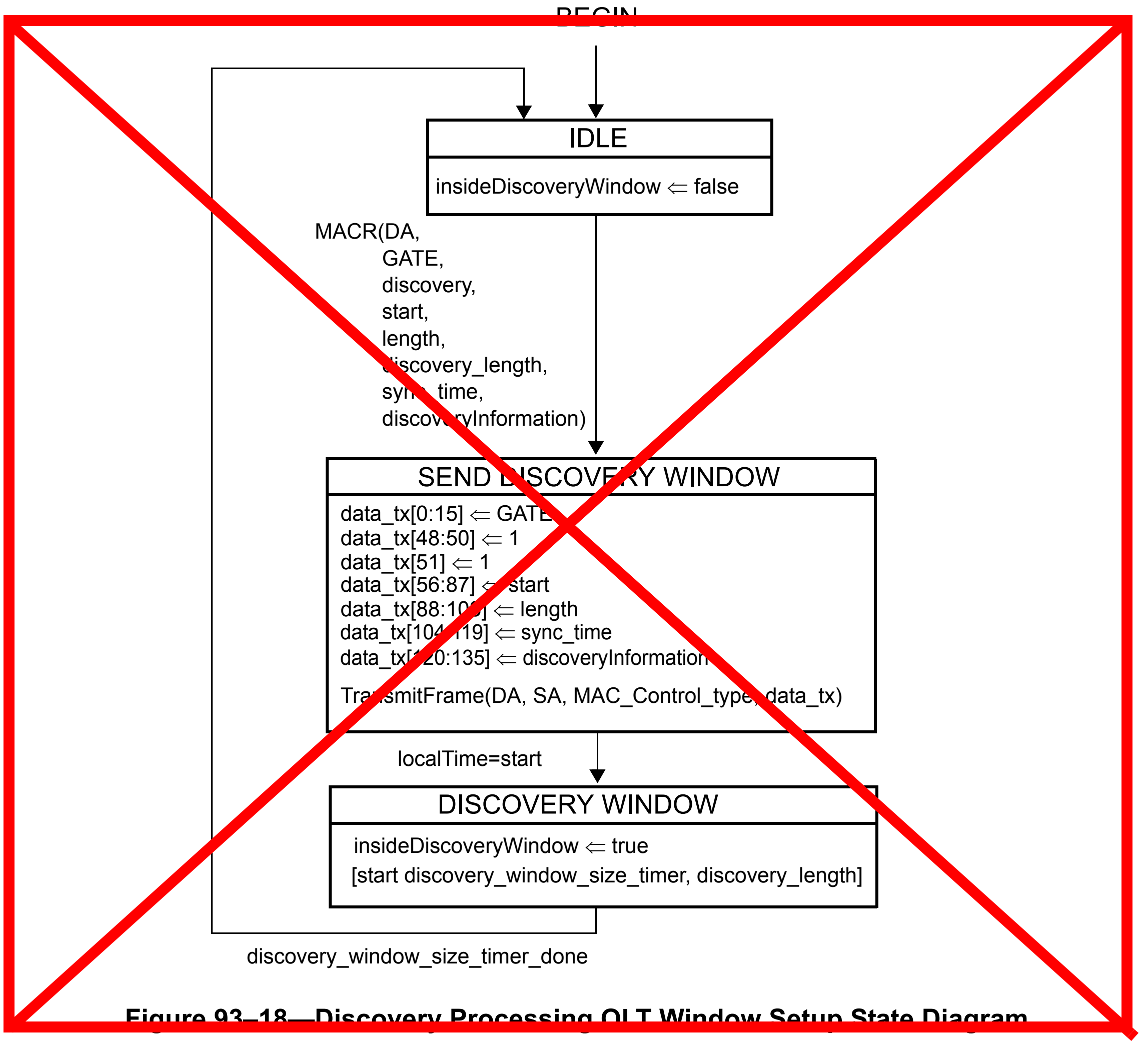

BEGIN
IDLE
insideDiscoveryWindow ⇐ false
MACR(DA,
GATE,
discovery,
start,
length,
discovery_length,
sync_time,
discoveryInformation)
SEND DISCOVERY WINDOW
data_tx[0:15] ⇐ GATE
data_tx[48:50] ⇐ 1
data_tx[51] ⇐ 1
data_tx[56:87] ⇐ start
data_tx[88:103] ⇐ length
data_tx[104:119] ⇐ sync_time
data_tx[120:135] ⇐ discoveryInformation
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
localTime=start
DISCOVERY WINDOW
insideDiscoveryWindow ⇐ true
[start discovery_window_size_timer, discovery_length]
discovery_window_size_timer_done

Figure 93–18—Discovery Processing OLT Window Setup State Diagram

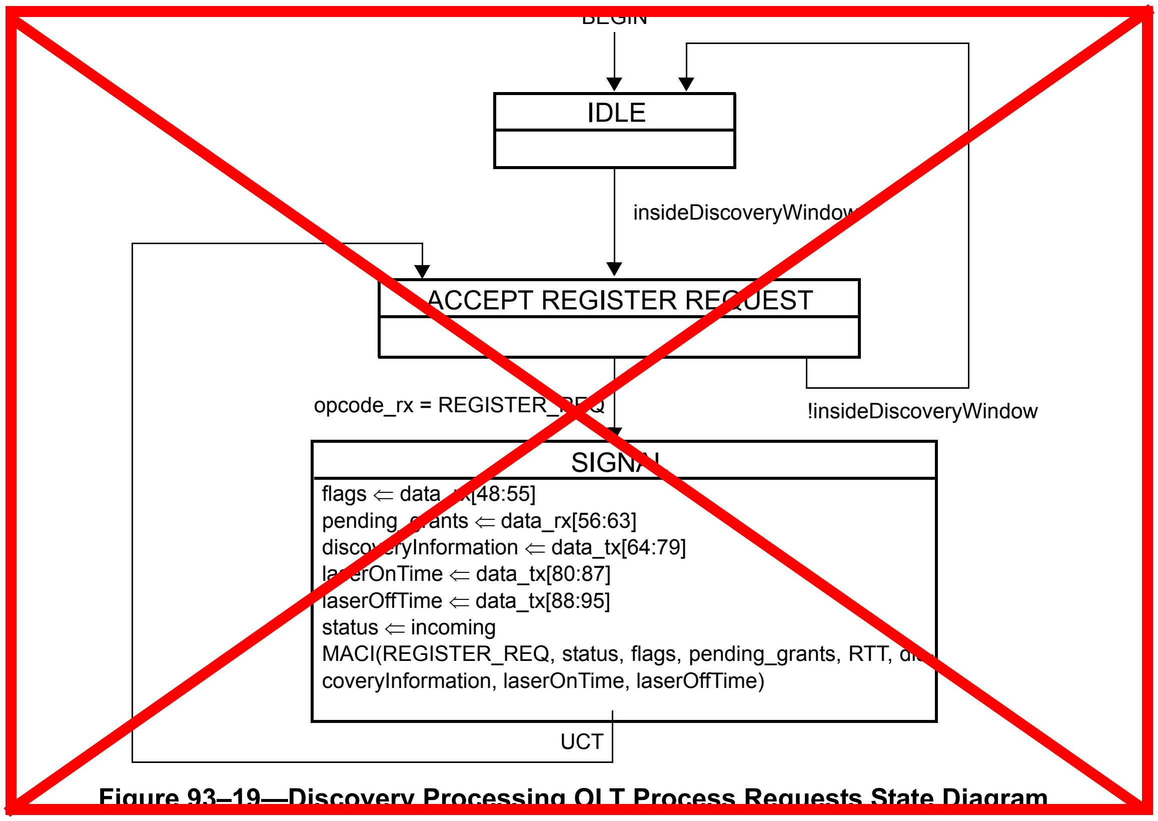

BEGIN
IDLE
insideDiscoveryWindow
ACCEPT REGISTER REQUEST
opcode_rx = REGISTER_REQ
!insideDiscoveryWindow
SIGNAL
flags ⇐ data_rx[48:55]
pending_grants ⇐ data_rx[56:63]
discoveryInformation ⇐ data_tx[64:79]
laserOnTime ⇐ data_tx[80:87]
laserOffTime ⇐ data_tx[88:95]
status ⇐ incoming
MACI(REGISTER_REQ, status, flags, pending_grants, RTT, discoveryInformation, laserOnTime, laserOffTime)
UCT

Figure 93–19—Discovery Processing OLT Process Requests State Diagram

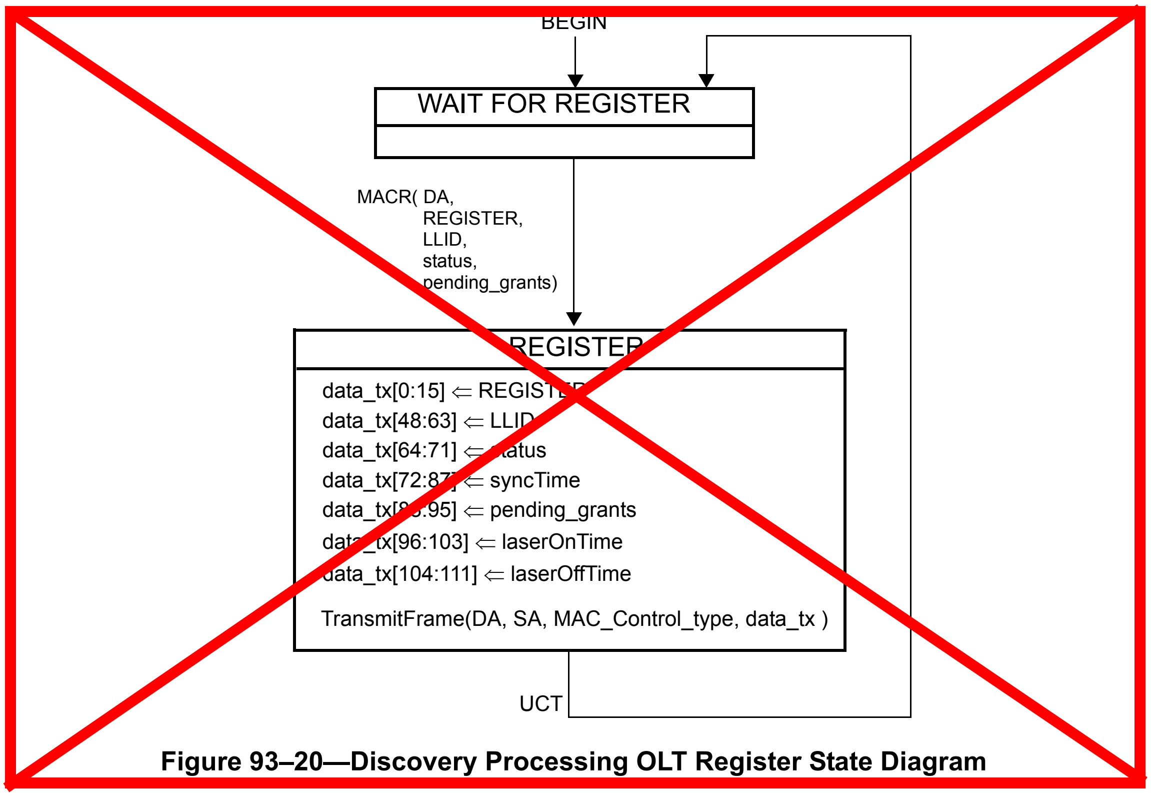
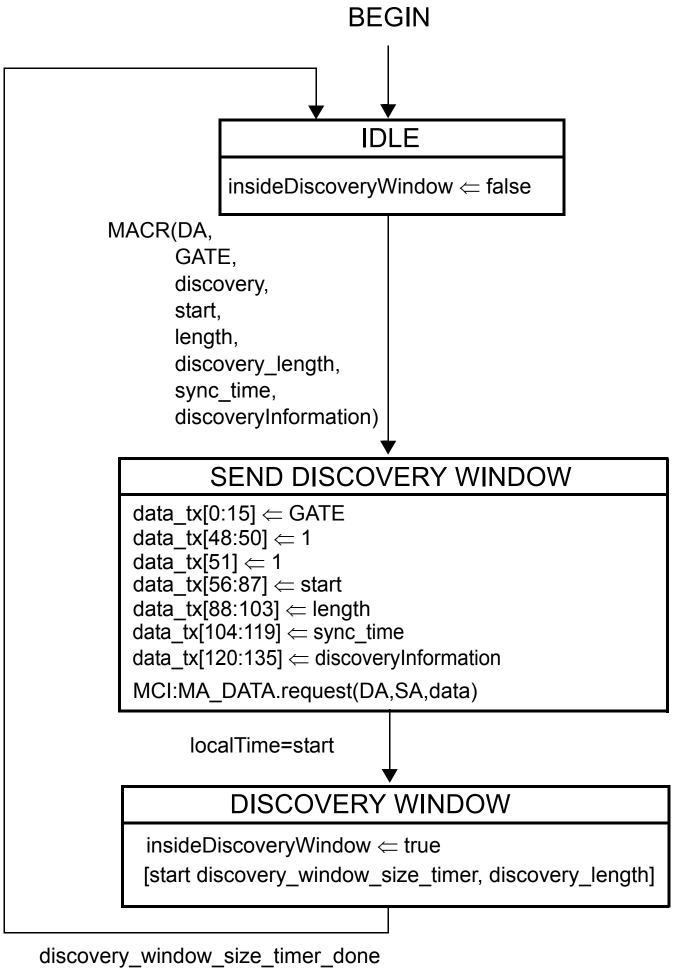

BEGIN
WAIT FOR REGISTER
MACR(DA,
REGISTER,
LLID,
status,
pending_grants)
REGISTER
data_tx[0:15] ⇐ REGISTER
data_tx[48:63] ⇐ LLID
data_tx[64:71] ⇐ status
data_tx[72:87] ⇐ syncTime
data_tx[88:95] ⇐ pending_grants
data_tx[96:103] ⇐ laserOnTime
data_tx[104:111] ⇐ laserOffTime
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
UCT

Figure 93–20—Discovery Processing OLT Register State Diagram



Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–18—Discovery Processing OLT Window Setup state diagram

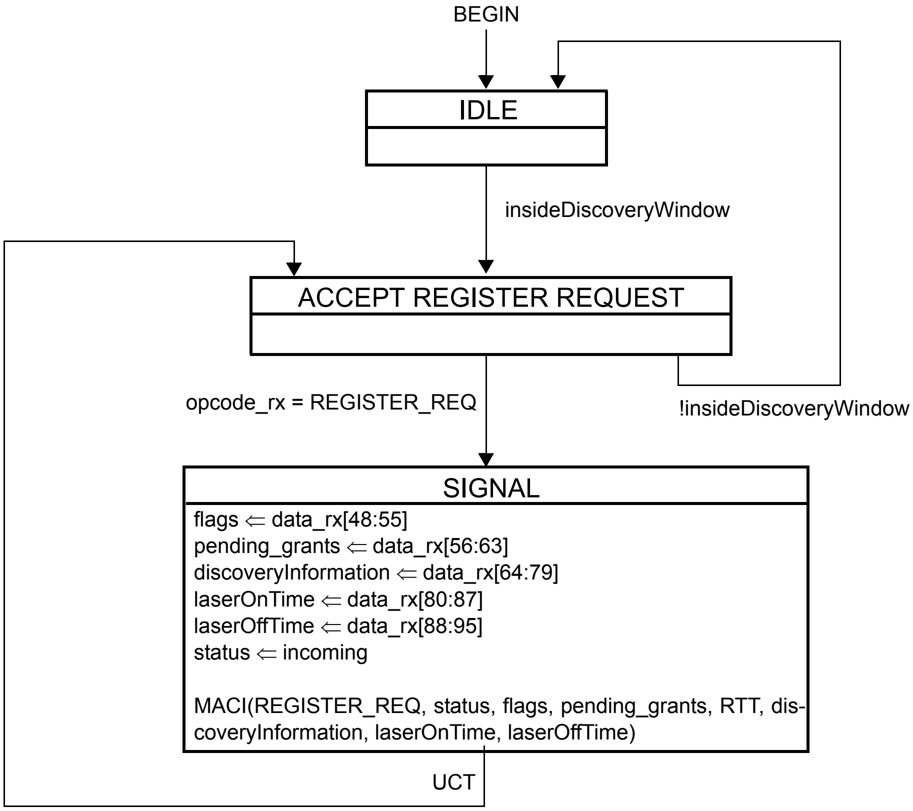

BEGIN
IDLE
insideDiscoveryWindow
ACCEPT REGISTER REQUEST
opcode_rx = REGISTER_REQ
!insideDiscoveryWindow
SIGNAL
flags ⇐ data_rx[48:55]
pending_grants ⇐ data_rx[56:63]
discoveryInformation ⇐ data_rx[64:79]
laserOnTime ⇐ data_rx[80:87]
laserOffTime ⇐ data_rx[88:95]
status ⇐ incoming
MACI(REGISTER_REQ, status, flags, pending_grants, RTT, discoveryInformation, laserOnTime, laserOffTime)
UCT

Figure 93–19—Discovery Processing OLT Process Requests State Diagram

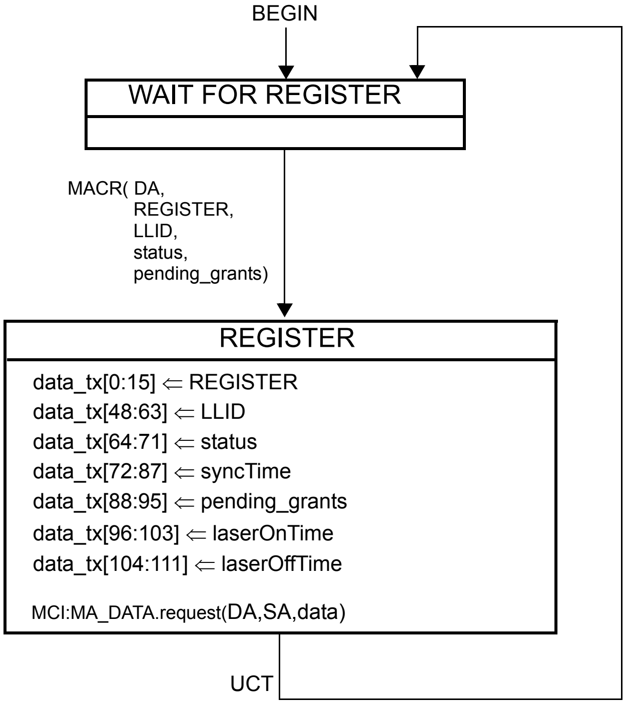
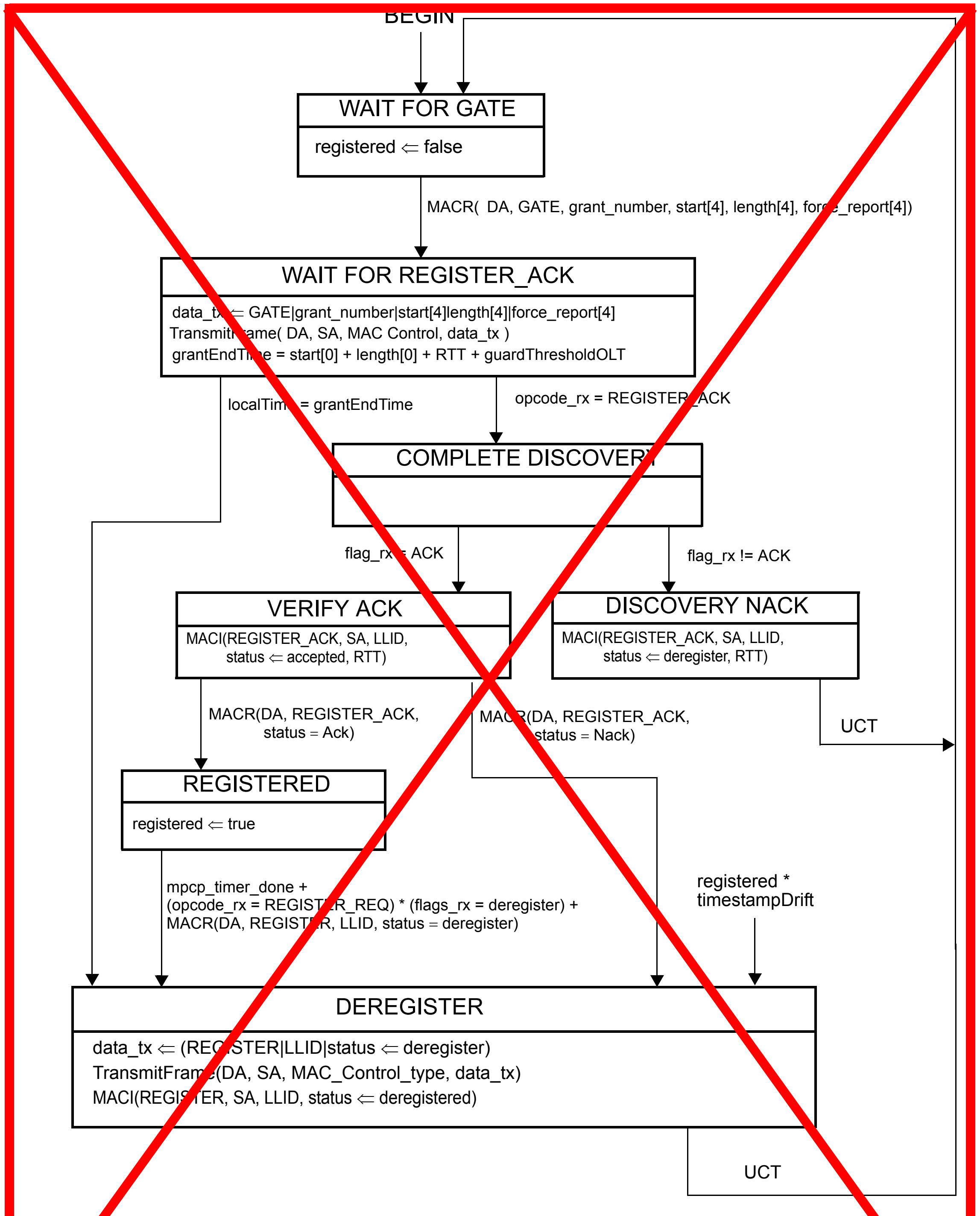
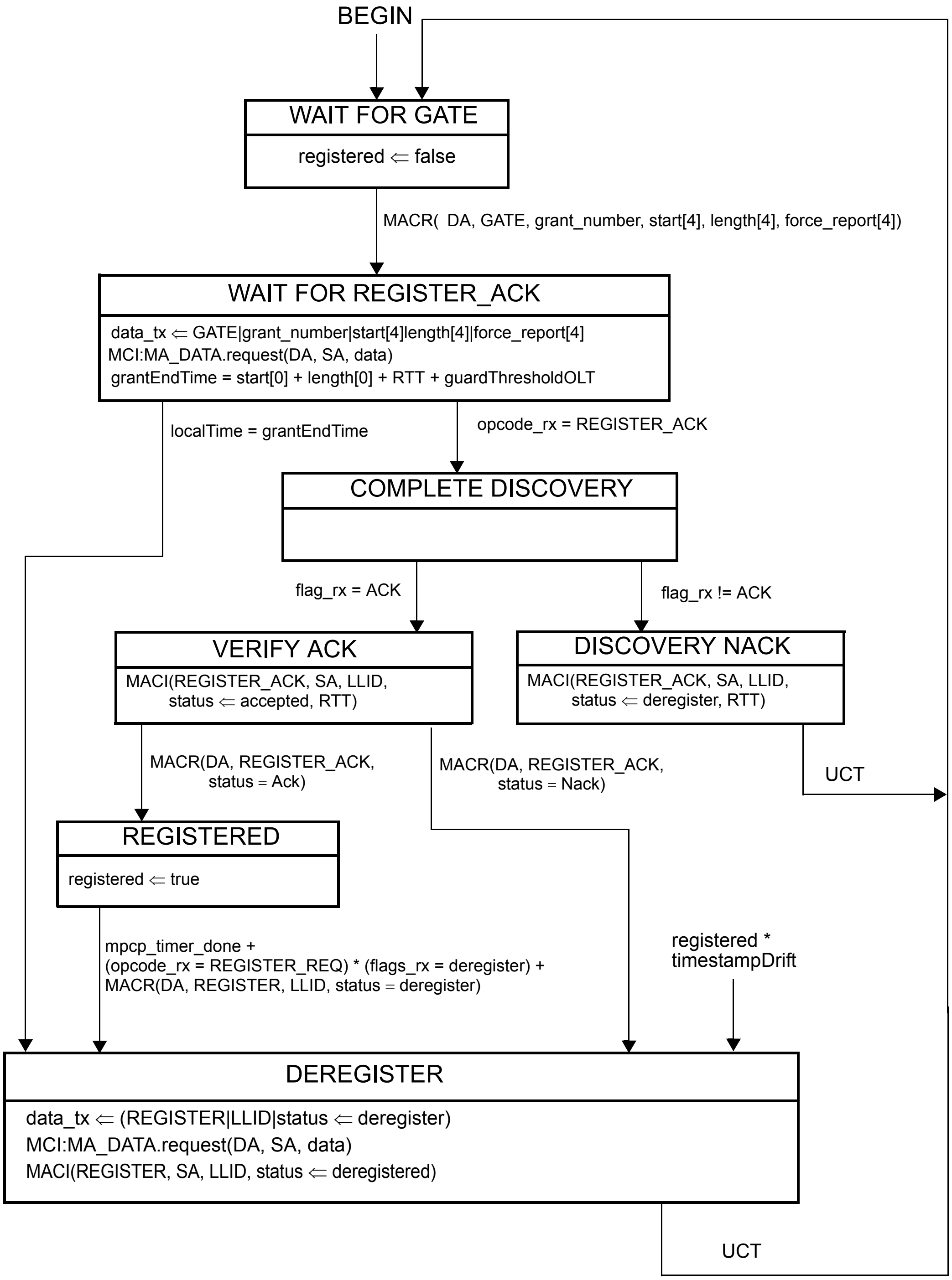

BEGIN
WAIT FOR REGISTER
MACR(DA,
REGISTER,
LLID,
status,
pending_grants)
REGISTER
data_tx[0:15] ⇐ REGISTER
data_tx[48:63] ⇐ LLID
data_tx[64:71] ⇐ status
data_tx[72:87] ⇐ syncTime
data_tx[88:95] ⇐ pending_grants
data_tx[96:103] ⇐ laserOnTime
data_tx[104:111] ⇐ laserOffTime
MCI:MA_DATA.request(DA,SA,data)
UCT

Figure 93–20—Discovery Processing OLT Register State Diagram


BEGIN
WAIT FOR GATE
registered ⇐ false
MACR(DA, GATE, grant_number, start[4], length[4], force_report[4])
WAIT FOR REGISTER_ACK
data_tx ⇐ GATE|grant_number|start[4]|length[4]|force_report[4]
TransmitFrame(DA, SA, MAC Control, data_tx)
grantEndTime = start[0] + length[0] + RTT + guardThresholdOLT
localTime = grantEndTime
opcode_rx = REGISTER_ACK
COMPLETE DISCOVERY
flag_rx = ACK
flag_rx != ACK
VERIFY ACK
MACI(REGISTER_ACK, SA, LLID,
status ⇐ accepted, RTT)
DISCOVERY NACK
MACI(REGISTER_ACK, SA, LLID,
status ⇐ deregister, RTT)
MACR(DA, REGISTER_ACK,
status = Ack)
MACR(DA, REGISTER_ACK,
status = Nack)
UCT
REGISTERED
registered ⇐ true
mpcp_timer_done +
(opcode_rx = REGISTER_REQ) * (flags_rx = deregister) +
MACR(DA, REGISTER, LLID, status = deregister)
registered *
timestampDrift
DEREGISTER
data_tx ⇐ (REGISTER|LLID|status ⇐ deregister)
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
MACI(REGISTER, SA, LLID, status ⇐ deregistered)
UCT

NOTE—The MAC Control Client issues the grant following the REGISTER message, taking the ONU processing delay of REGISTER message into consideration.

Figure 93–21—Discovery Processing OLT Final Registration State Diagram


BEGIN
WAIT FOR GATE
registered ⇐ false
MACR(DA, GATE, grant_number, start[4], length[4], force_report[4])
WAIT FOR REGISTER_ACK
data_tx ⇐ GATE|grant_number|start[4]length[4]|force_report[4]
MCI:MA_DATA.request(DA, SA, data)
grantEndTime = start[0] + length[0] + RTT + guardThresholdOLT
localTime = grantEndTime
opcode_rx = REGISTER_ACK
COMPLETE DISCOVERY
flag_rx = ACK
flag_rx != ACK
VERIFY ACK
MACI(REGISTER_ACK, SA, LLID,
status ⇐ accepted, RTT)
DISCOVERY NACK
MACI(REGISTER_ACK, SA, LLID,
status ⇐ deregister, RTT)
MACR(DA, REGISTER_ACK,
status = Ack)
MACR(DA, REGISTER_ACK,
status = Nack)
UCT
REGISTERED
registered ⇐ true
mpcp_timer_done +
(opcode_rx = REGISTER_REQ) * (flags_rx = deregister) +
MACR(DA, REGISTER, LLID, status = deregister)
registered *
timestampDrift
DEREGISTER
data_tx ⇐ (REGISTER|LLID|status ⇐ deregister)
MCI:MA_DATA.request(DA, SA, data)
MACI(REGISTER, SA, LLID, status ⇐ deregistered)
UCT

NOTE— The MAC Control Client issues the grant following the REGISTER message, taking the ONU processing delay of REGISTER message into consideration.

Figure 93–21—Discovery Processing OLT Final Registration State Diagram

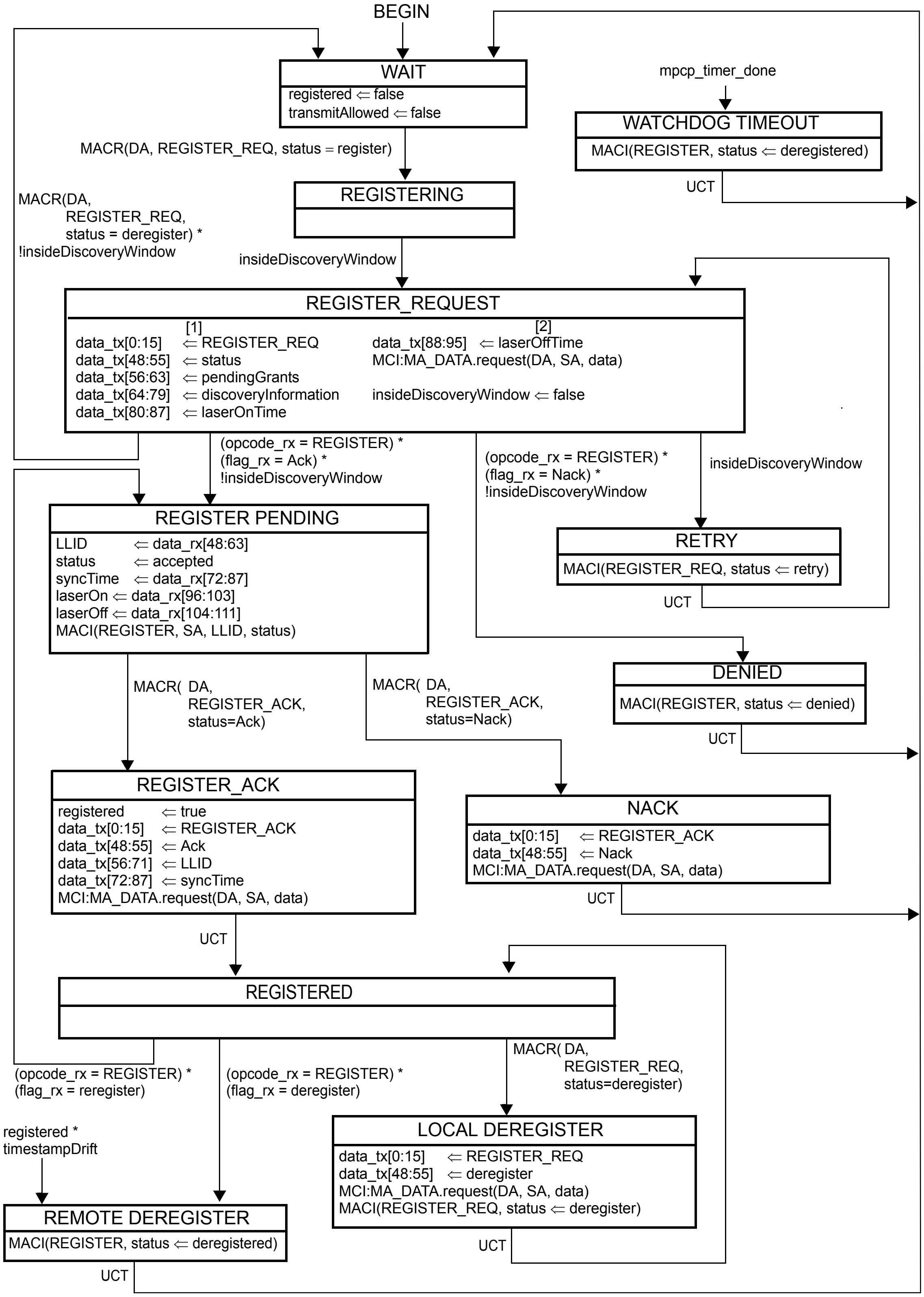

BEGIN
WAIT
registered ⇐ false
transmitAllowed ⇐ false
mpcp_timer_done
WATCHDOG TIMEOUT
MACI(REGISTER, status ⇐ deregistered)
UCT
MACR(DA, REGISTER_REQ, status = register)
REGISTERING
MACR(DA, REGISTER_REQ, status = deregister) * !insideDiscoveryWindow
insideDiscoveryWindow
REGISTER_REQUEST
[1]
data_tx[0:15] ⇐ REGISTER_REQ
data_tx[48:55] ⇐ status
data_tx[56:63] ⇐ pendingGrants
data_tx[64:79] ⇐ discoveryInformation
data_tx[80:87] ⇐ laserOnTime
[2]
data_tx[88:95] ⇐ laserOffTime
MCI:MA_DATA.request(DA, SA, data)
insideDiscoveryWindow ⇐ false
(opcode_rx = REGISTER) * (flag_rx = Ack) * !insideDiscoveryWindow
(opcode_rx = REGISTER) * (flag_rx = Nack) * !insideDiscoveryWindow
insideDiscoveryWindow
REGISTER PENDING
LLID ⇐ data_rx[48:63]
status ⇐ accepted
syncTime ⇐ data_rx[72:87]
laserOn ⇐ data_rx[96:103]
laserOff ⇐ data_rx[104:111]
MACI(REGISTER, SA, LLID, status)
RETRY
MACI(REGISTER_REQ, status ⇐ retry)
UCT
DENIED
MACI(REGISTER, status ⇐ denied)
UCT
MACR(DA, REGISTER_ACK, status=Ack)
MACR(DA, REGISTER_ACK, status=Nack)
REGISTER_ACK
registered ⇐ true
data_tx[0:15] ⇐ REGISTER_ACK
data_tx[48:55] ⇐ Ack
data_tx[56:71] ⇐ LLID
data_tx[72:87] ⇐ syncTime
MCI:MA_DATA.request(DA, SA, data)
NACK
data_tx[0:15] ⇐ REGISTER_ACK
data_tx[48:55] ⇐ Nack
MCI:MA_DATA.request(DA, SA, data)
UCT
UCT
REGISTERED
(opcode_rx = REGISTER) * (flag_rx = reregister)
(opcode_rx = REGISTER) * (flag_rx = deregister)
MACR(DA, REGISTER_REQ, status=deregister)
registered * timestampDrift
LOCAL DEREGISTER
data_tx[0:15] ⇐ REGISTER_REQ
data_tx[48:55] ⇐ deregister
MCI:MA_DATA.request(DA, SA, data)
MACI(REGISTER_REQ, status ⇐ deregister)
REMOTE DEREGISTER
MACI(REGISTER, status ⇐ deregistered)
UCT
UCT

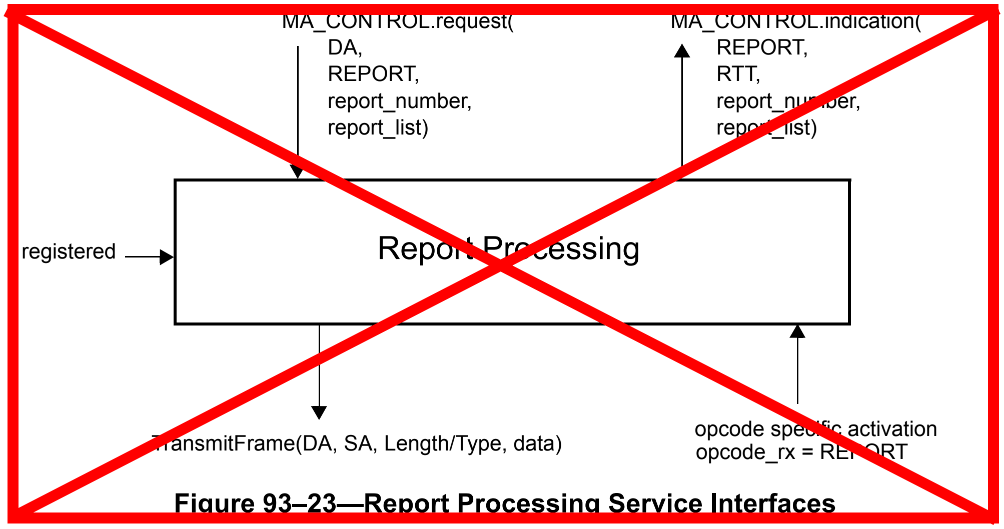
Figure 93–22—Discovery Processing ONU Registration State Diagram

93.3.4 Report Processing

The Report Processing functional block has the responsibility of dealing with queue report generation and termination in the network. Reports are generated by higher layers and passed to the MAC Control sublayer by the MAC Control clients. Status reports are used to signal bandwidth needs as well as for arming the OLT watchdog timer.

Reports shall be generated periodically, even when no request for bandwidth is being made. This keeps a watchdog timer in the OLT from expiring and deregistering the ONU. For proper operation of this mechanism the OLT shall grant the ONU periodically.

The Report Processing functional block, and its MPCP protocol elements are designed for use in conjunction with an IEEE 802.1P capable bridge.



~~Figure 93–23—Report Processing Service Interfaces~~

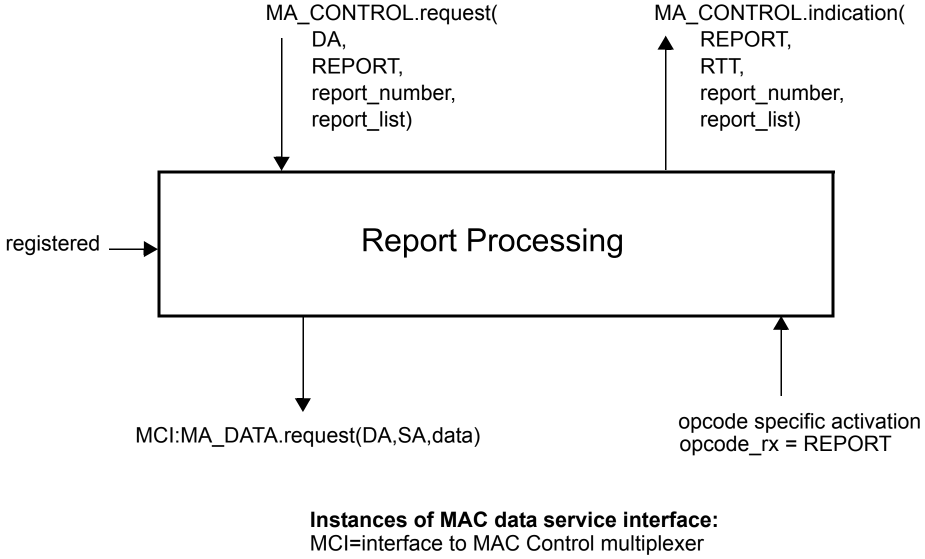


Figure 93–23—Report Processing service interfaces

93.3.4.1 Constants

mpcp_timeout

This constant represents the maximum allowed interval of time between two MPCPDU messages. Failure to receive at least one frame within this interval is considered a fatal fault and leads to deregistration.

TYPE 32–bit unsigned
VALUE 0x03B9ACA0 (1 second)

report_timeout

This constant represents the maximum allowed interval of time between two REPORT messages generated by the ONU.

TYPE 32–bit unsigned
VALUE 0x002FAF08 (50 milliseconds)

93.3.4.2 Variables

BEGIN

This variable is used when initiating operation of the functional block state diagram. It is set to true following initialization and every reset.

TYPE: Boolean

data_rx

This variable is defined in Subclause 93.2.2.3.

data_tx

This variable is defined in Subclause 93.2.2.3.

opcode_rx

This variable is defined in Subclause 93.2.2.3.

opcode_tx

This variable is defined in Subclause 93.2.2.3.

registered

This variable is defined in Subclause 93.3.3.2.

93.3.4.3 Functions

~~TransmitFrame(DA, SA, Length/Type, data)~~

~~This function is defined in Subclause 93.2.2.4.~~

None.

93.3.4.4 Timers

report_periodic_timer

ONUs are required to generate REPORT MPCPDUs with a periodicity of less than report_timeout value. This timer counts down time remaining before a forced generation of a REPORT message in an ONU.

mpcp_timer

This timer is defined in Subclause 93.3.3.4.

93.3.4.5 Messages

MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, REPORT, report_number, report_list)

This service primitive is used by a MAC Control client to request the Report Process at the ONU to transmit a queue status report. This primitive may be called at variable intervals, independently of the granting process, in order to reflect the time varying aspect of the network. This primitive uses the following parameters:

DA: multicast MAC Control address as defined in Annex 31B.

REPORT: opcode for REPORT MPCPDU as defined in Table 31A–1.

report_number: the number of queue status report sets located in report list. The report_number value ranges from 0 to a maximum of 13.

report_list: the list of queue status reports. A queue status report consists of two fields: valid and status. The parameter valid, is a Boolean array with length of 8, ‘0’ or false indicates that the corresponding status field is not present (the length of status field is 0), while ‘1’ or true indicates that the corresponding status field is present (the length of status field is 2 octets). The index of the array is meant to reflect the same numbered priority queue in the 802.1P nomenclature.

The parameter status is an array of 16–bit unsigned integer values. This array consists only of entries whose corresponding bit in field valid is set to true.

MA_CONTROL.indication(REPORT, RTT, report_number, report_list)

The service primitive issued by the Report Process at the OLT to notify the MAC Control client and higher layers the queue status of the MPCP link partner. This primitive may be called multiple times, in order to reflect the time–varying aspect of the network. This primitive uses the following parameters:

REPORT: opcode for REPORT MPCPDU as defined in Table 31A–1.

RTT: this parameter holds an updated round trip time value which is recalculated following each REPORT message reception.

report_number: the number of queue status report sets located in report list. The report_number value ranges from 0 to a maximum of 13.

report_list: the list of queue status reports. A queue status report consists of two fields: valid and status. The parameter valid, is a Boolean array with length of 8, ‘0’ or false indicates that the corresponding status field is not present (the length of status field is 0), while ‘1’ or true indicates that the corresponding status field is present (the length of status field is 2 octets). The index of the array is meant to reflect the same numbered priority queue in the 802.1P nomenclature.

The parameter status is an array of 16–bit unsigned integer values. This array consists only of entries whose corresponding bit in field valid is set to true.

Opcode–specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

93.3.4.6 State Diagram

The report process in the OLT shall implement the report processing state diagram as shown in Figure 93–24. The report process in the ONU shall implement the report processing state diagram as shown in Figure 93–25. Instantiation of state diagrams as described is performed for Multipoint MAC Control instances attached to unicast LLIDs only.

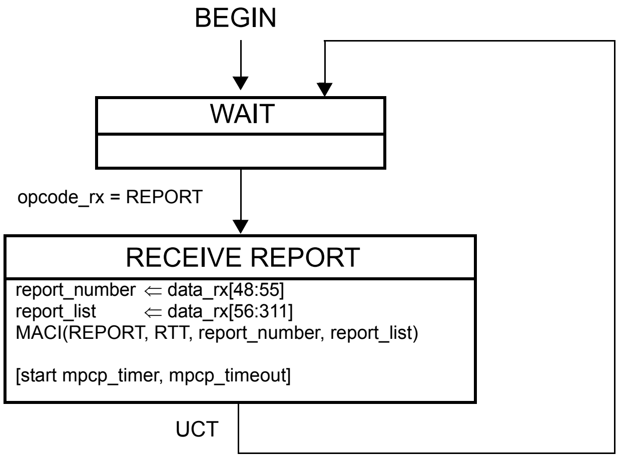

BEGIN
WAIT
opcode_rx = REPORT
RECEIVE REPORT
report_number ⇐ data_rx[48:55]
report_list ⇐ data_rx[56:311]
MACI(REPORT, RTT, report_number, report_list)
[start mpcp_timer, mpcp_timeout]
UCT

Figure 93–24—Report Processing State Diagram at OLT

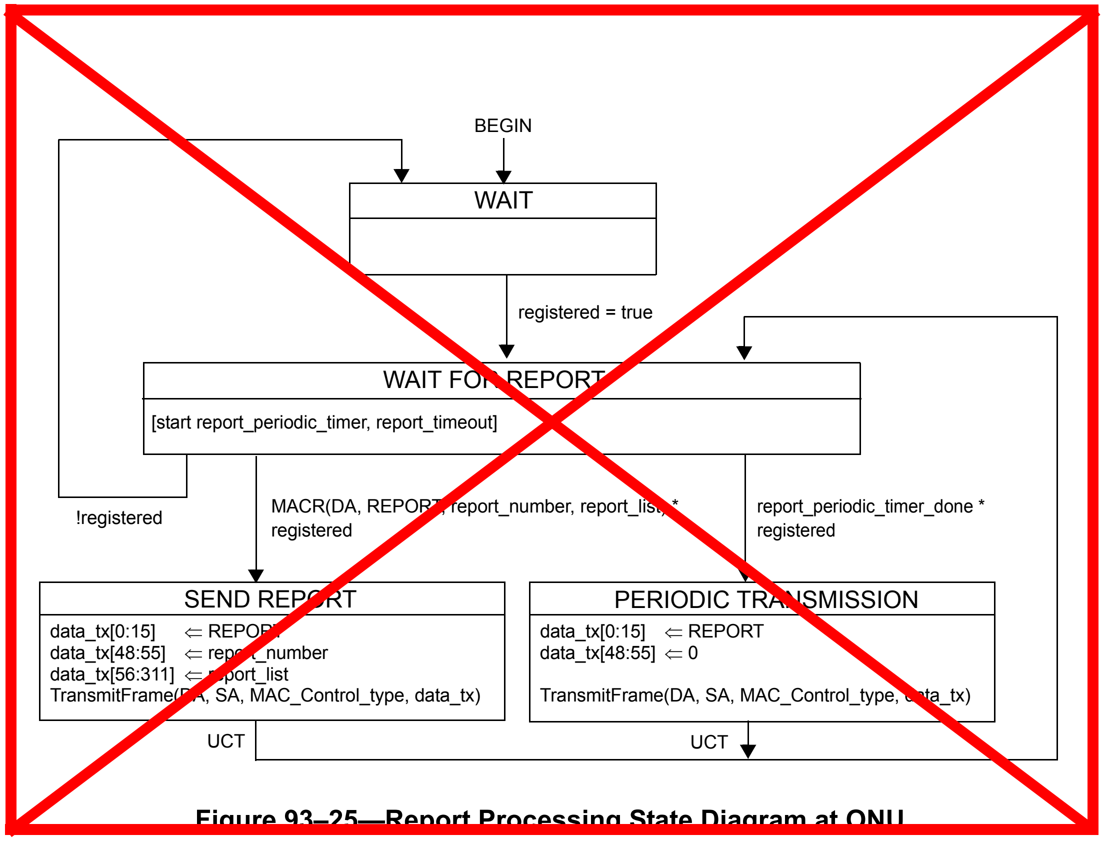

BEGIN
WAIT
registered = true
WAIT FOR REPORT
[start report_periodic_timer, report_timeout]
!registered
MACR(DA, REPORT, report_number, report_list) *
registered
report_periodic_timer_done *
registered
SEND REPORT
data_tx[0:15] ⇐ REPORT
data_tx[48:55] ⇐ report_number
data_tx[56:311] ⇐ report_list
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
PERIODIC TRANSMISSION
data_tx[0:15] ⇐ REPORT
data_tx[48:55] ⇐ 0
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
UCT
UCT

Figure 93–25—Report Processing State Diagram at ONU

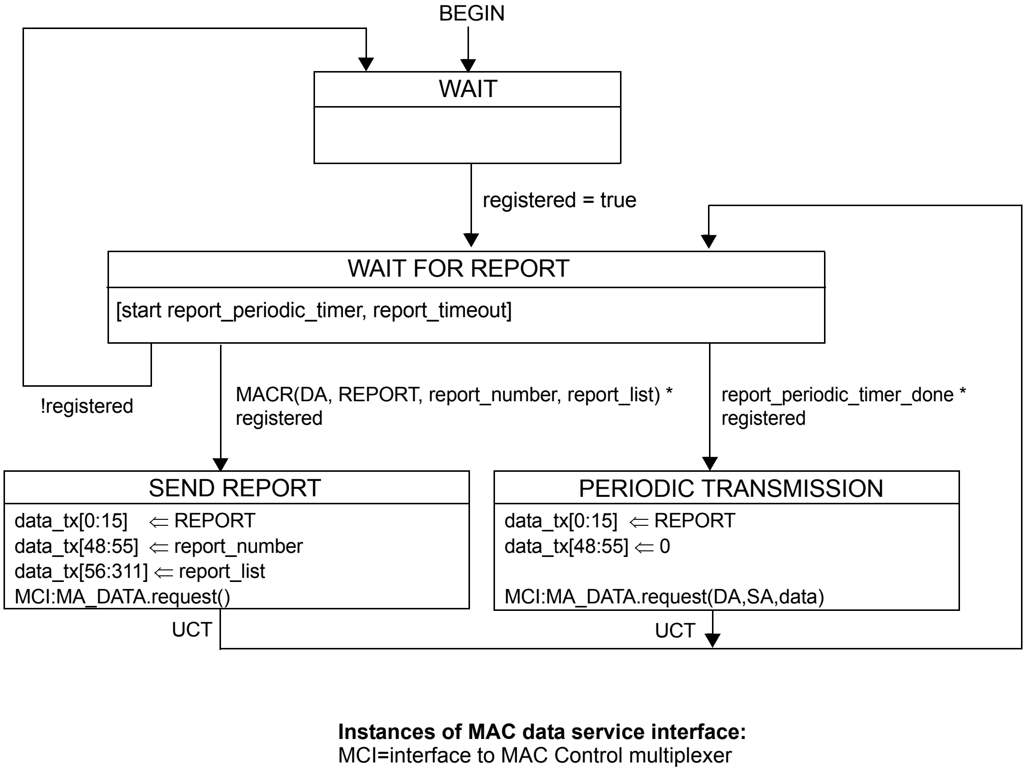

BEGIN
WAIT
registered = true
WAIT FOR REPORT
[start report_periodic_timer, report_timeout]
!registered
MACR(DA, REPORT, report_number, report_list) * registered
report_periodic_timer_done * registered
SEND REPORT
data_tx[0:15] ⇐ REPORT
data_tx[48:55] ⇐ report_number
data_tx[56:311] ⇐ report_list
MCI:MA_DATA.request()
PERIODIC TRANSMISSION
data_tx[0:15] ⇐ REPORT
data_tx[48:55] ⇐ 0
MCI:MA_DATA.request(DA,SA,data)
UCT
UCT
Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–25—Report Processing state diagram at ONU

93.3.5 Gate Processing

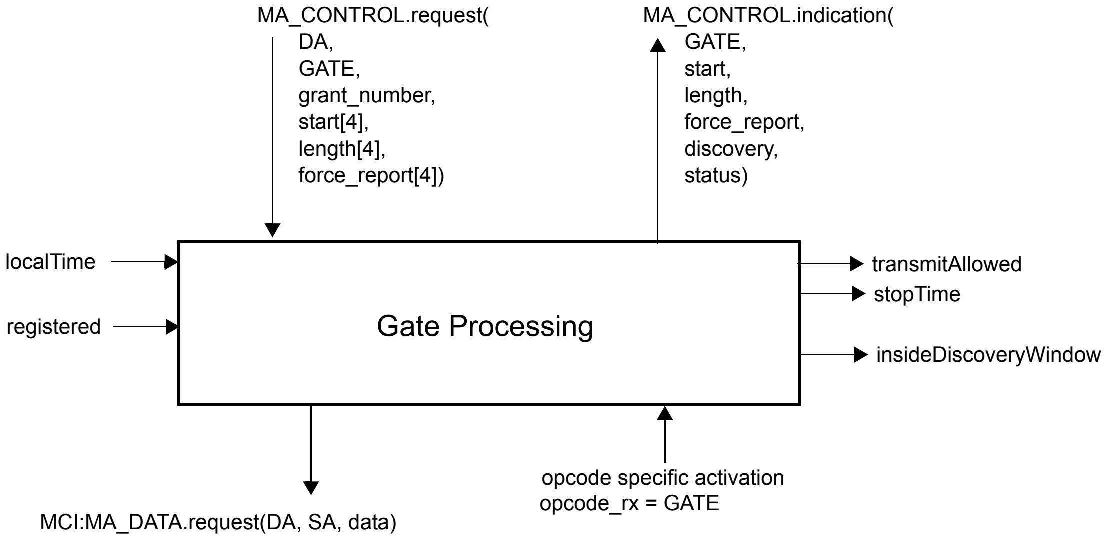
A key concept pervasive in Multipoint MAC Control is the ability to arbitrate a single transmitter out of a plurality of ONUs. The OLT controls an ONU’s transmission by the assigning of grants.

The transmitting window of an ONU is indicated in GATE message where start time and length are specified. An ONU will begin transmission when its localTime counter matches start_time value indicated in the GATE message. An ONU will conclude its transmission with sufficient margin to ensure that the laser is turned off before the grant length interval has elapsed.

Multiple outstanding grants may be issued to each ONU. The OLT shall not issue more than the maximal supported maximal outstanding grants as advertised by the ONU during registration (see pending grants in Subclause 93.3.6.3).

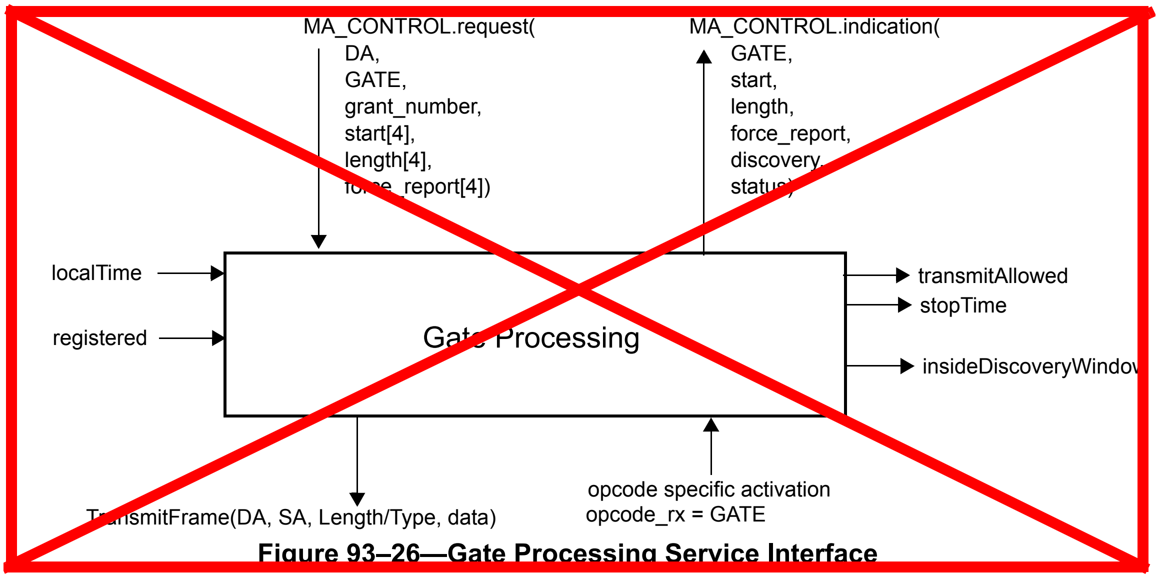
In order to maintain the watchdog timer at the ONU, grants are periodically generated. For this purpose empty GATE messages may be issued periodically.

When registered, the ONU ignores all gate messages where the discovery flag is set.



Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–26—Gate Processing Service Interface



~~**Figure 93–26—Gate Processing Service Interface**~~

93.3.5.1 Constants

discoveryGrantLength

This constant represents the duration of ONU’s transmission during discovery attempt. The value of discoveryGrantLength includes MPCPDU transmission time and tailGuard as defined in Subclause 93.2.2.1. discoveryGrantLength is represented in units of time_quanta.

TYPE:	32 bit unsigned
VALUE:	0x00000005 (672 bits)

gate_timeout

This constant represents the maximum allowed interval of time between two GATE messages generated by the OLT to the same ONU.
TYPE 32–bit unsigned
VALUE 0x002FAF08 (50 milliseconds)

max_future_grant_time

This constant holds the time limiting the future time horizon for a valid incoming grant.
TYPE: 32 bit unsigned
VALUE: 0x03B9ACA0 (1 second)

min_processing_time

This constant is the time required for the ONU processing time.
TYPE: 32 bit unsigned
VALUE: 0x00000400 (16.384 us)

tqSize

This constant is defined in Subcluase 93.2.2.1.

93.3.5.2 Variables

BEGIN

This variable is used when initiating operation of the functional block state diagram. It is set to true following initialization and every reset.
TYPE: Boolean

counter

This variable is used as a loop iterator counting the number of incoming grants in a GATE message.
TYPE: integer

currentGrant

This variable is used for local storage of a pending grant state during processing. It is dynamically set by the Gate Processing functional block and is not exposed.
The state is a structure field composed of multiple subfields.
TYPE: structure {
DA 48 bit unsigned, a.k.a MAC address type
start 32 bit unsigned
length 16 bit unsigned
force_report Boolean
discovery Boolean}

data_rx

This variable is defined in Subclause 93.2.2.3.

data_tx

This variable is defined in Subclause 93.2.2.3.

effectiveLength

This variable is used for temporary storage of a normalized net time value. It holds the net effective length of a grant normalized for elapsed time, and compensated for the periods required to turn the laser on and off, and waiting for receiver lock.
TYPE: 32 bit unsigned

fecEnabled

This variable is defined in Subclause 93.2.2.3.

grantList

This variable is used for storage of the list of pending grants. It is dynamically set by the Gate Processing functional block and is not exposed. Each time a grant is received it is added to the list.

The list elements are structure fields composed of multiple subfields.

The list is indexed by the start subfield in each element for quick searches.

TYPE: list of elements having the structure define in currentGrant

insideDiscoveryWindow

This variable is defined in Subclause 93.3.3.2.

maxDelay

This variable holds the maximum delay that can be applied by an ONU before sending the REGISTER MPCPDU. This delay is calculated such that the ONU would have sufficient time to transmit the REGISTER message and its associated overhead (FEC parity date, end–of–frame sequence, etc.) and terminate the laser before the end of the discovery grant.

TYPE 16 bit unsigned

nextGrant

This variable is used for local storage of a pending grant state during processing. It is dynamically set by the Gate Processing functional block and is not exposed. The content of the variable is the next grant to become active.

TYPE: element having same structure as defined in currentGrant

nextStopTime

This variable holds the value of the localTime counter corresponding to the end of the next grant.

TYPE: 32 bit unsigned

registered

This variable is defined in Subclause 93.3.3.2.

stopTime

This variable is defined in Subclause 93.2.2.3.

syncTime

This variable is defined in Subclause 93.3.3.2.

transmitAllowed

This variable is defined in Subclause 93.2.2.3.

93.3.5.3 Functions

empty(list)

This function is use to check whether the list is empty. When there are no elements queued in the list, the function returns true. Otherwise, a value of false is returned.

confirmDiscovery(data)

This functon is used to check whether the current Discovery Window is open for the given ONU (TRUE) or not (FALSE). For 1 Gb/s ONUs, this function always returns TRUE. For 10 Gb/s ONUs, this function operates as follows: @@TBD@@

InsertInOrder(sorted_list, inserted_element)

This function is used to queue an element inside a sorted list. The queueing order is sorted. In the condition that the list is full the element may be discarded. The length of the list is dynamic and it's maximal size equals the value advertised during registration as maximum number of pending grants.

IsBroadcast(grant)

This function is used to check whether its argument represents a broadcast grant, i.e., grant given to multiple ONUs. This is determined by the destination MAC address of the corresponding GATE message. The function returns the value true when MAC address is a global assigned MAC Control address as defined in Annex 31B, and false otherwise.

PeekHead(sorted_list)

This function is used to check the content of a sorted list. It returns the element at the head of the list without dequeuing the element.

Random(r)

This function is used to compute a random integer number uniformly distributed between 0 and r. The randomly generated number is then returned by the function.

RemoveHead(sorted_list)

This function is used to dequeue an element from the head of a sorted list. The return value of the function is the dequeued element.

TransmitFrame(DA, SA, Length/Type, data)

This function is defined in Subclause 93.2.2.4.

93.3.5.4 Timers

gntStTmr

This timer is used to wait for the event signaling the start of a grant window.

VALUE: The timer value is dynamically set according to the signaled grant start time.

gntWinTmr

This timer is used to wait for the event signaling the end of a grant window.

VALUE: The timer value is dynamically set according to the signaled grant length.

gate_periodic_timer

The OLT is required to generate GATE MPCPDUs with a periodicity of less than gate_timeout value. This timer counts down time remaining before a forced generation of a GATE message in the OLT.

mpcp_timer

This timer is defined in Subclause 93.3.3.4.

rndDlyTmr

This timer is used to measure a random delay inside the discovery window. The purpose of the delay is to apriori reduce the probability of transmission overlap during the registration process, and thus lowering the expectancy of registration time in the PON.

VALUE: A random value less than the net discovery window size less the REGISTER_REQ MPCPDU frame size less the idle period and laser turn on and off delays less the preamble size less the IFG size. The timer value is set dynamically based on the parameters passed from the client.

93.3.5.5 Messages

MA_DATA.request (DA, SA, m_sdu)

The service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, GATE, grant_number, start[4], length[4], force_report[4])

This service primitive is defined in Subclause 93.3.3.5.

MA_CONTROL.indication(GATE, start, length, force_report, discovery, status)

This service primitive issued by the Gate Process at the ONU to notify the MAC Control client and higher layers that a grant is pending. This primitive is invoked multiple times when a single GATE message arrives with multiple grants. It is also generated at the start and end of each grant as it becomes active. This primitive uses the following parameters:

GATE: opcode for GATE MPCPDU as defined in Table 31A–1.

start: start time of the grant. This parameter is not present when the status value is deactive.

length: length of the grant. This parameter is not present when the status value is deactive.

force_report: flags indicating whether a REPORT message should be transmitted in this grant. This parameter is not present when the status value is deactive.

discovery: This parameter holds the value true when the grant is to be used for the discovery process, and false otherwise. This parameter is not present when the status value is deactive.

status: This parameter takes the value *arrive* on grant reception, *active* when a grant becomes active, and *deactive* at the end of a grant.

Opcode–specific function(opcode)

Functions exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

93.3.5.6 State Diagrams

The gating process in the OLT shall implement the gate processing state diagram as shown in Figure 93–26. The gating process in the ONU shall implement the gate processing state diagram as shown in Figure 93–27.

Instantiation of state diagrams as described is performed for all Multipoint MAC Control instances.

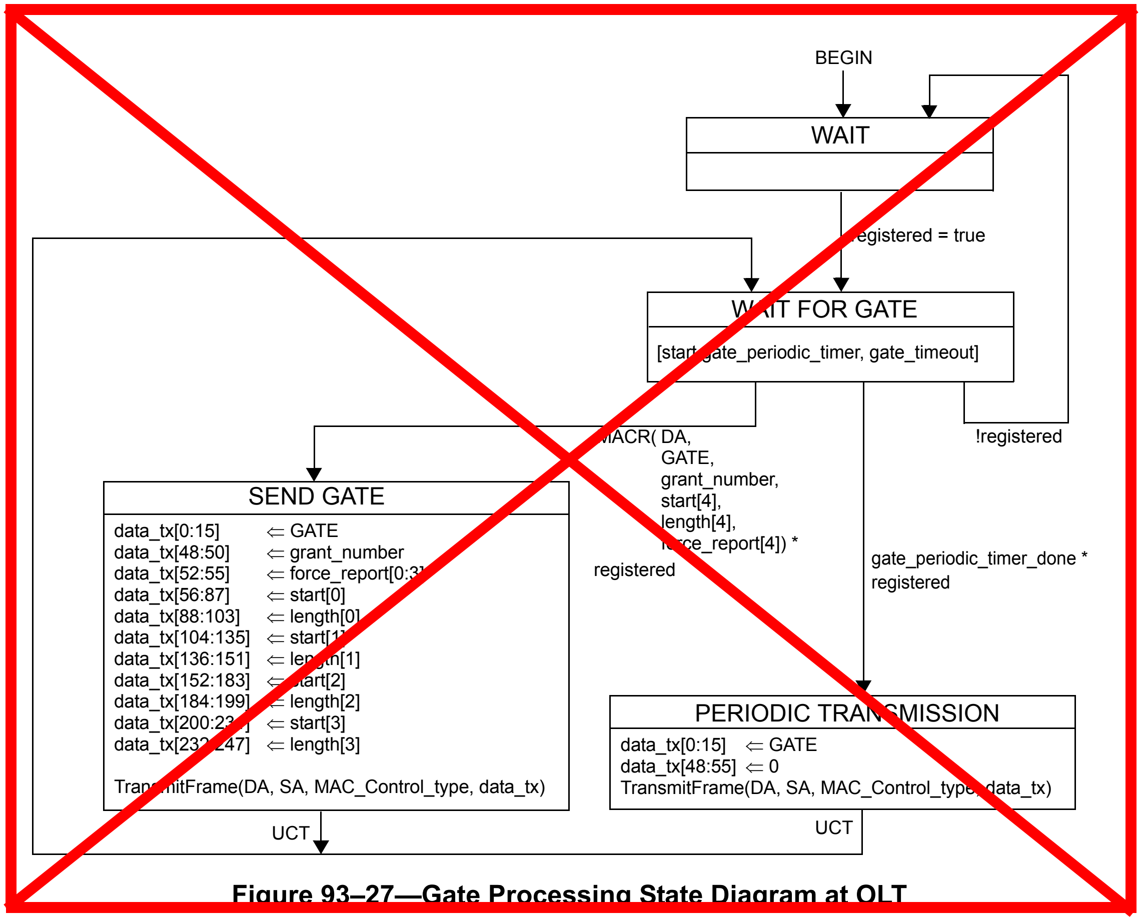
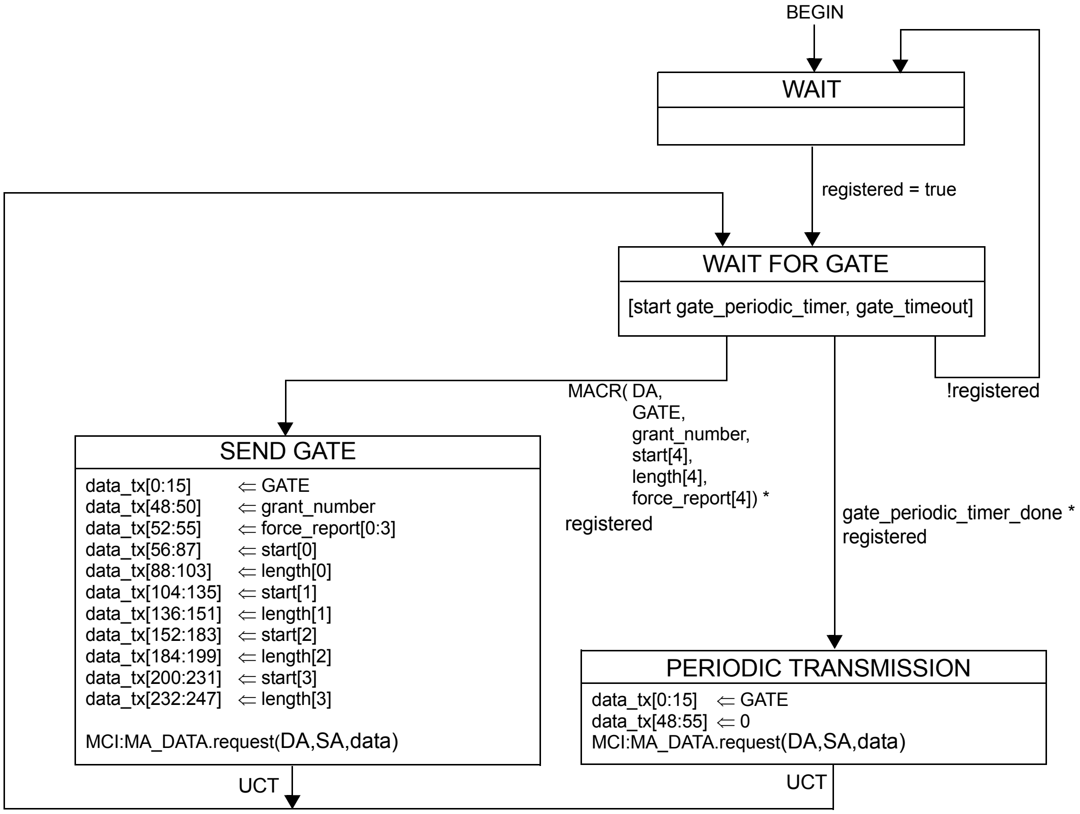

BEGIN
WAIT
registered = true
WAIT FOR GATE
[start gate_periodic_timer, gate_timeout]
!registered
MACR(DA,
GATE,
grant_number,
start[4],
length[4],
force_report[4]) *
registered
gate_periodic_timer_done *
registered
SEND GATE
data_tx[0:15] ⇐ GATE
data_tx[48:50] ⇐ grant_number
data_tx[52:55] ⇐ force_report[0:3]
data_tx[56:87] ⇐ start[0]
data_tx[88:103] ⇐ length[0]
data_tx[104:135] ⇐ start[1]
data_tx[136:151] ⇐ length[1]
data_tx[152:183] ⇐ start[2]
data_tx[184:199] ⇐ length[2]
data_tx[200:231] ⇐ start[3]
data_tx[232:247] ⇐ length[3]
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
UCT
PERIODIC TRANSMISSION
data_tx[0:15] ⇐ GATE
data_tx[48:55] ⇐ 0
TransmitFrame(DA, SA, MAC_Control_type, data_tx)
UCT

Figure 93–27—Gate Processing State Diagram at OLT


BEGIN
WAIT
registered = true
WAIT FOR GATE
[start gate_periodic_timer, gate_timeout]
!registered
MACR(DA,
GATE,
grant_number,
start[4],
length[4],
force_report[4]) *
registered
gate_periodic_timer_done *
registered
SEND GATE
data_tx[0:15] ⇐ GATE
data_tx[48:50] ⇐ grant_number
data_tx[52:55] ⇐ force_report[0:3]
data_tx[56:87] ⇐ start[0]
data_tx[88:103] ⇐ length[0]
data_tx[104:135] ⇐ start[1]
data_tx[136:151] ⇐ length[1]
data_tx[152:183] ⇐ start[2]
data_tx[184:199] ⇐ length[2]
data_tx[200:231] ⇐ start[3]
data_tx[232:247] ⇐ length[3]
MCI:MA_DATA.request(DA,SA,data)
PERIODIC TRANSMISSION
data_tx[0:15] ⇐ GATE
data_tx[48:55] ⇐ 0
MCI:MA_DATA.request(DA,SA,data)
UCT
UCT

Instances of MAC data service interface:
MCI=interface to MAC Control multiplexer

Figure 93–27—Gate Processing state diagram at OLT

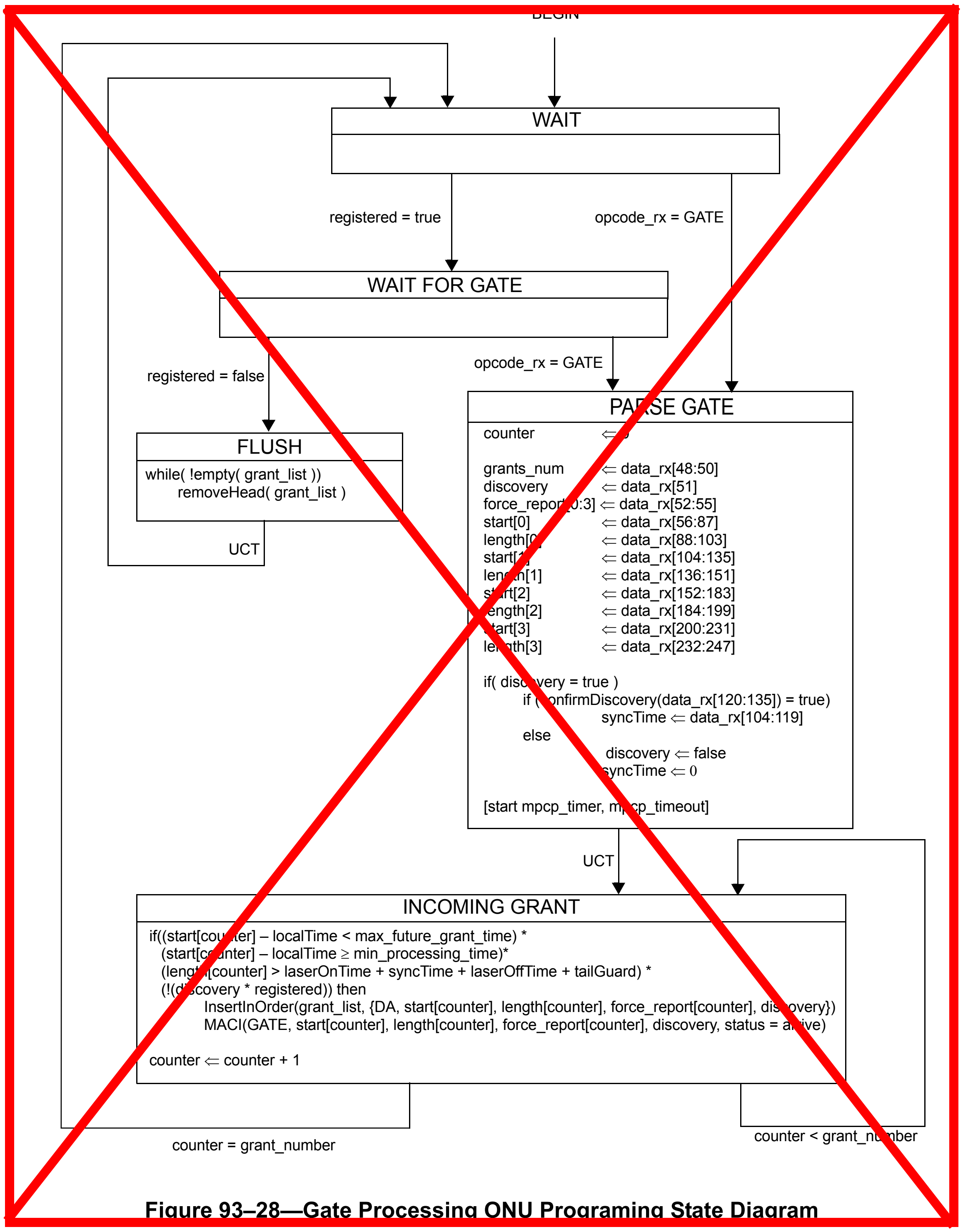

BEGIN
WAIT
registered = true
opcode_rx = GATE
WAIT FOR GATE
registered = false
opcode_rx = GATE
FLUSH
while(!empty(grant_list))
removeHead(grant_list)
UCT
PARSE GATE
counter ⇐ 0
grants_num ⇐ data_rx[48:50]
discovery ⇐ data_rx[51]
force_report[0:3] ⇐ data_rx[52:55]
start[0] ⇐ data_rx[56:87]
length[0] ⇐ data_rx[88:103]
start[1] ⇐ data_rx[104:135]
length[1] ⇐ data_rx[136:151]
start[2] ⇐ data_rx[152:183]
length[2] ⇐ data_rx[184:199]
start[3] ⇐ data_rx[200:231]
length[3] ⇐ data_rx[232:247]
if(discovery = true)
if (confirmDiscovery(data_rx[120:135]) = true)
syncTime ⇐ data_rx[104:119]
else
discovery ⇐ false
syncTime ⇐ 0
[start mpcp_timer, mpcp_timeout]
UCT
INCOMING GRANT
if((start[counter] – localTime < max_future_grant_time) *
(start[counter] – localTime ≥ min_processing_time)*
(length[counter] > laserOnTime + syncTime + laserOffTime + tailGuard) *
(!(discovery * registered)) then
InsertInOrder(grant_list, {DA, start[counter], length[counter], force_report[counter], discovery})
MACI(GATE, start[counter], length[counter], force_report[counter], discovery, status = arrive)
counter ⇐ counter + 1
counter = grant_number
counter < grant_number

Figure 93–28—Gate Processing ONU Programing State Diagram

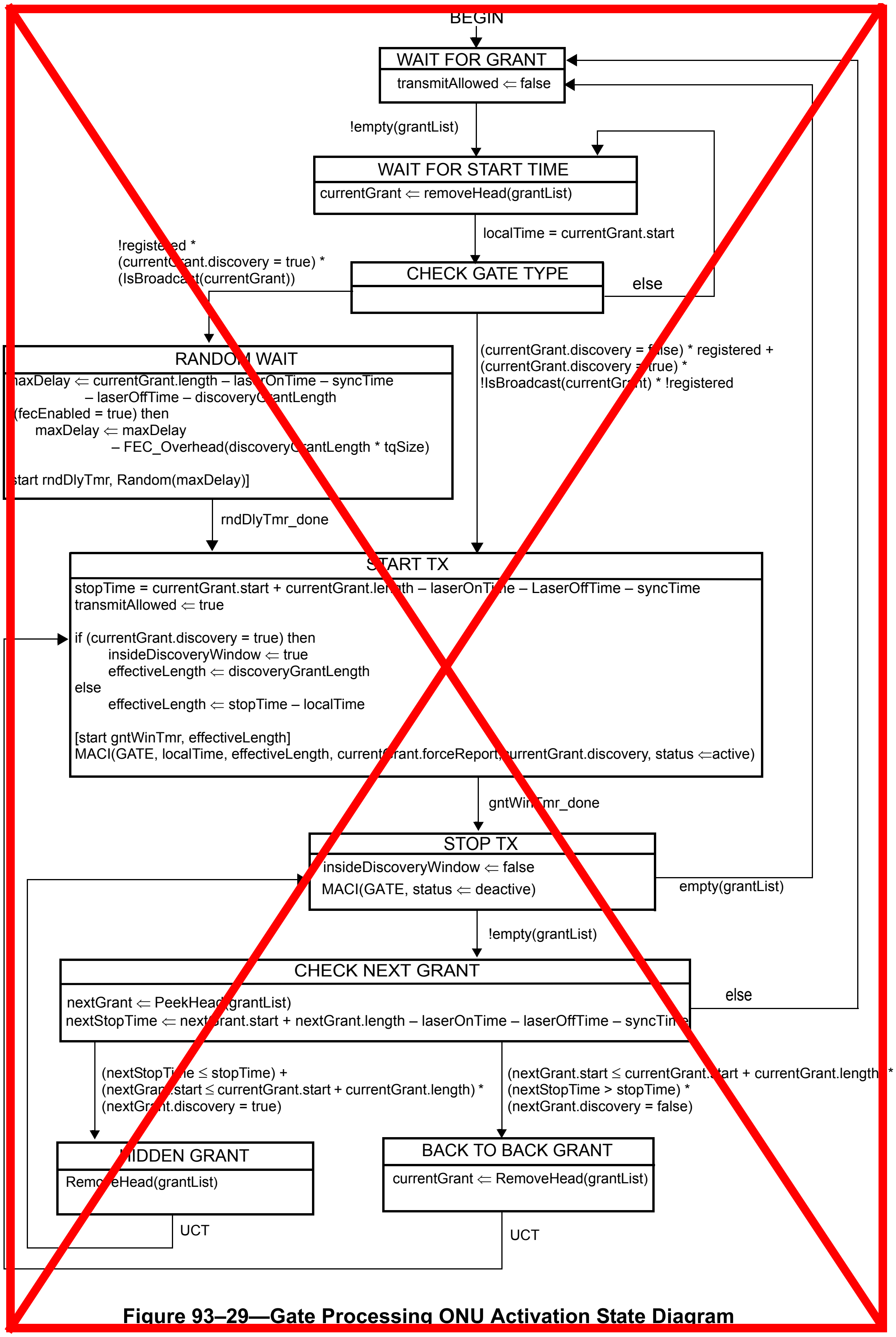
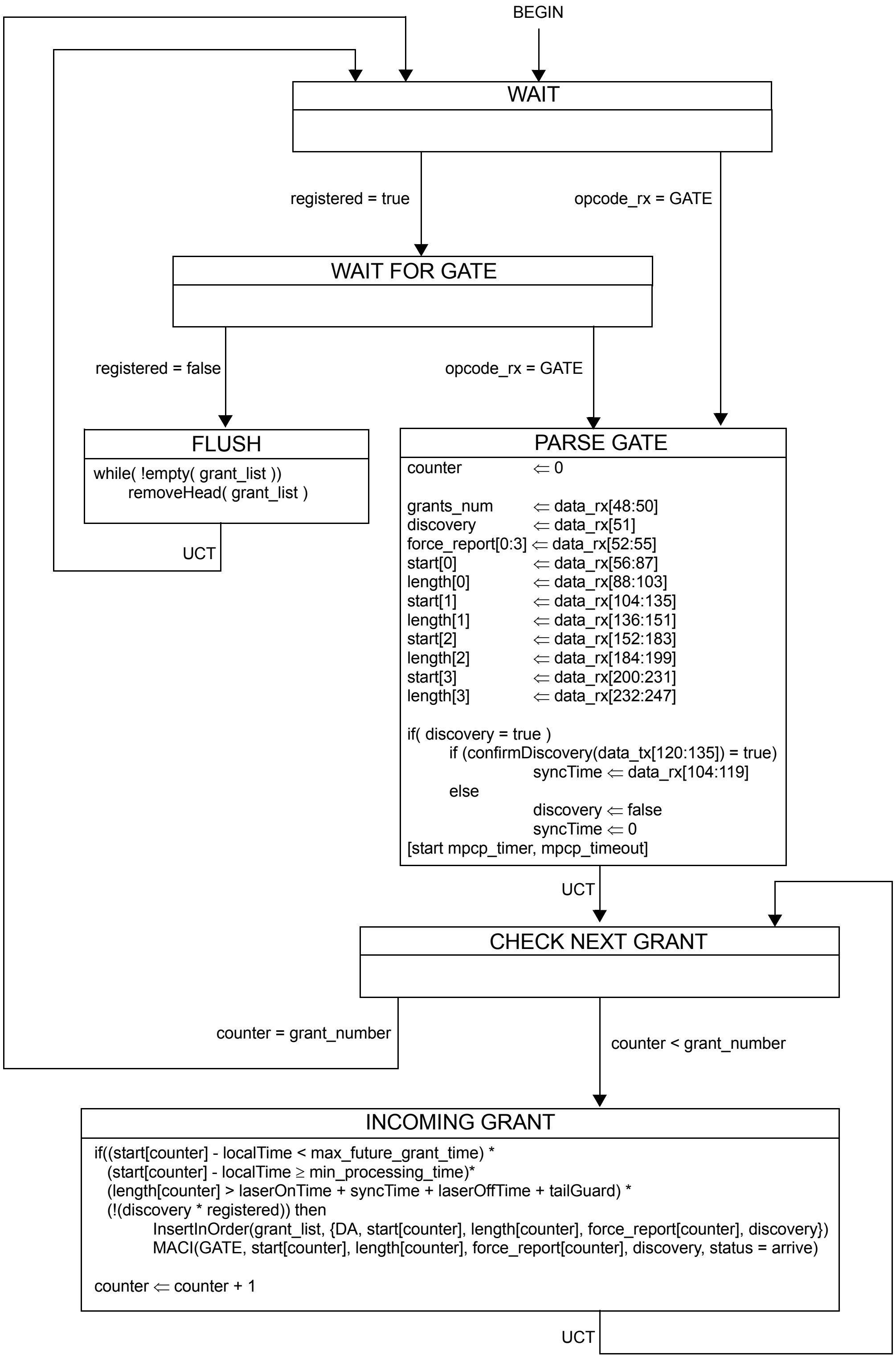

BEGIN
WAIT FOR GRANT
transmitAllowed ⇐ false
!empty(grantList)
WAIT FOR START TIME
currentGrant ⇐ removeHead(grantList)
localTime = currentGrant.start
!registered *
(currentGrant.discovery = true) *
(IsBroadcast(currentGrant))
CHECK GATE TYPE
else
RANDOM WAIT
maxDelay ⇐ currentGrant.length – laserOnTime – syncTime
– laserOffTime – discoveryGrantLength
if (fecEnabled = true) then
maxDelay ⇐ maxDelay
– FEC_Overhead(discoveryGrantLength * tqSize)
[start rndDlyTmr, Random(maxDelay)]
(currentGrant.discovery = false) * registered +
(currentGrant.discovery = true) *
!IsBroadcast(currentGrant) * !registered
rndDlyTmr_done
START TX
stopTime = currentGrant.start + currentGrant.length – laserOnTime – LaserOffTime – syncTime
transmitAllowed ⇐ true
if (currentGrant.discovery = true) then
insideDiscoveryWindow ⇐ true
effectiveLength ⇐ discoveryGrantLength
else
effectiveLength ⇐ stopTime – localTime
[start gntWinTmr, effectiveLength]
MACI(GATE, localTime, effectiveLength, currentGrant.forceReport, currentGrant.discovery, status ⇐active)
gntWinTmr_done
STOP TX
insideDiscoveryWindow ⇐ false
MACI(GATE, status ⇐ deactive)
empty(grantList)
!empty(grantList)
CHECK NEXT GRANT
nextGrant ⇐ PeekHead(grantList)
nextStopTime ⇐ nextGrant.start + nextGrant.length – laserOnTime – laserOffTime – syncTime
else
(nextStopTime ≤ stopTime) +
(nextGrant.start ≤ currentGrant.start + currentGrant.length) *
(nextGrant.discovery = true)
(nextGrant.start ≤ currentGrant.start + currentGrant.length) *
(nextStopTime > stopTime) *
(nextGrant.discovery = false)
HIDDEN GRANT
RemoveHead(grantList)
BACK TO BACK GRANT
currentGrant ⇐ RemoveHead(grantList)
UCT
UCT

Figure 93–29—Gate Processing ONU Activation State Diagram



```
while( !empty( grant_list ))
    removeHead( grant_list )
```

```
counter          ⇐ 0

grants_num       ⇐ data_rx[48:50]
discovery        ⇐ data_rx[51]
force_report[0:3] ⇐ data_rx[52:55]
start[0]         ⇐ data_rx[56:87]
length[0]        ⇐ data_rx[88:103]
start[1]         ⇐ data_rx[104:135]
length[1]        ⇐ data_rx[136:151]
start[2]         ⇐ data_rx[152:183]
length[2]        ⇐ data_rx[184:199]
start[3]         ⇐ data_rx[200:231]
length[3]        ⇐ data_rx[232:247]

if( discovery = true )
        if (confirmDiscovery(data_tx[120:135]) = true)
                syncTime ⇐ data_rx[104:119]
        else
                discovery ⇐ false
                syncTime ⇐ 0
[start mpcp_timer, mpcp_timeout]
```

```
if((start[counter] - localTime < max_future_grant_time) *
   (start[counter] - localTime ≥ min_processing_time)*
   (length[counter] > laserOnTime + syncTime + laserOffTime + tailGuard) *
   (!(discovery * registered)) then
        InsertInOrder(grant_list, {DA, start[counter], length[counter], force_report[counter], discovery})
        MACI(GATE, start[counter], length[counter], force_report[counter], discovery, status = arrive)

counter ⇐ counter + 1
```

Figure 93–28—Gate Processing ONU Programing State Diagram

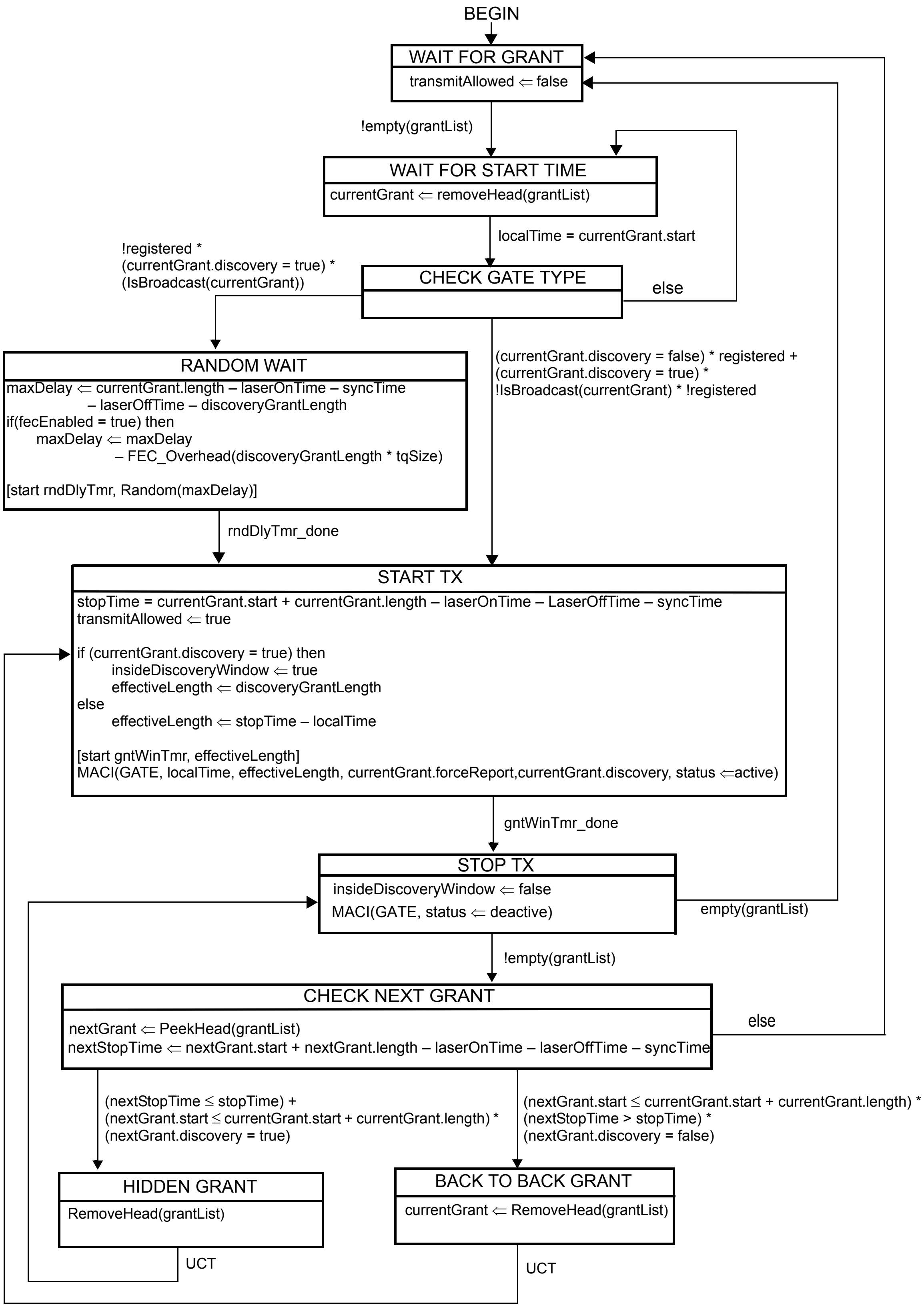

BEGIN
WAIT FOR GRANT
transmitAllowed ⇐ false
!empty(grantList)
WAIT FOR START TIME
currentGrant ⇐ removeHead(grantList)
localTime = currentGrant.start
!registered *
(currentGrant.discovery = true) *
(IsBroadcast(currentGrant))
CHECK GATE TYPE
else
RANDOM WAIT
maxDelay ⇐ currentGrant.length – laserOnTime – syncTime
– laserOffTime – discoveryGrantLength
if(fecEnabled = true) then
maxDelay ⇐ maxDelay
– FEC_Overhead(discoveryGrantLength * tqSize)
[start rndDlyTmr, Random(maxDelay)]
(currentGrant.discovery = false) * registered +
(currentGrant.discovery = true) *
!IsBroadcast(currentGrant) * !registered
rndDlyTmr_done
START TX
stopTime = currentGrant.start + currentGrant.length – laserOnTime – LaserOffTime – syncTime
transmitAllowed ⇐ true
if (currentGrant.discovery = true) then
insideDiscoveryWindow ⇐ true
effectiveLength ⇐ discoveryGrantLength
else
effectiveLength ⇐ stopTime – localTime
[start gntWinTmr, effectiveLength]
MACI(GATE, localTime, effectiveLength, currentGrant.forceReport,currentGrant.discovery, status ⇐active)
gntWinTmr_done
STOP TX
insideDiscoveryWindow ⇐ false
MACI(GATE, status ⇐ deactive)
empty(grantList)
!empty(grantList)
CHECK NEXT GRANT
nextGrant ⇐ PeekHead(grantList)
nextStopTime ⇐ nextGrant.start + nextGrant.length – laserOnTime – laserOffTime – syncTime
else
(nextStopTime ≤ stopTime) +
(nextGrant.start ≤ currentGrant.start + currentGrant.length) *
(nextGrant.discovery = true)
(nextGrant.start ≤ currentGrant.start + currentGrant.length) *
(nextStopTime > stopTime) *
(nextGrant.discovery = false)
HIDDEN GRANT
RemoveHead(grantList)
BACK TO BACK GRANT
currentGrant ⇐ RemoveHead(grantList)
UCT
UCT

Figure 93–29—Gate Processing ONU Activation State Diagram

93.3.6 MPCPDU structure and encoding

MPCP PDU (MPCPDU) are basic IEEE 802.3 frames; they shall not be tagged (see Clause 3). The MPCPDU structure is shown in Figure 93–30, and is further defined in the following definitions:

a) Destination Address (DA). The DA in MPCPDU is the MAC Control Multicast address as specified in the annexes to Clause 31, or the individual MAC address associated with the port to which the MPCPDU is destined.
b) Source Address (SA). The SA in MPCPDU is the individual MAC address associated with the port through which the MPCPDU is transmitted. For MPCPDUs originating at the OLT end, this can be the address any of the individual MACs. These MACs may all share a single unicast address, as explained in Subclause 93.1.2.
c) Length/Type. MPCPDUs are always Type encoded, and carry the MAC_Control_Type field value as specified in Subclause 31.4.1.3.
d) Opcode. The opcode identifies the specific MPCPDU being encapsulated. Values are defined in Table 31A–1.
e) Timestamp. The timestamp field conveys the content of the localTime register at the time of transmission of the MPCPDUs. This field is 32 bits long and counts time in units of time_quanta.
f) Data/Reserved/PAD. These 40 octets are used for the payload of the MPCPDUs. When not used they would be filled with zeros on transmission, and be ignored on reception.
g) FCS. This field is the Frame Check Sequence, typically generated by the underlying MAC.

Based on the MAC instance used to generate the specific MPCPDU, the appropriate LLID shall be generated by the RS.

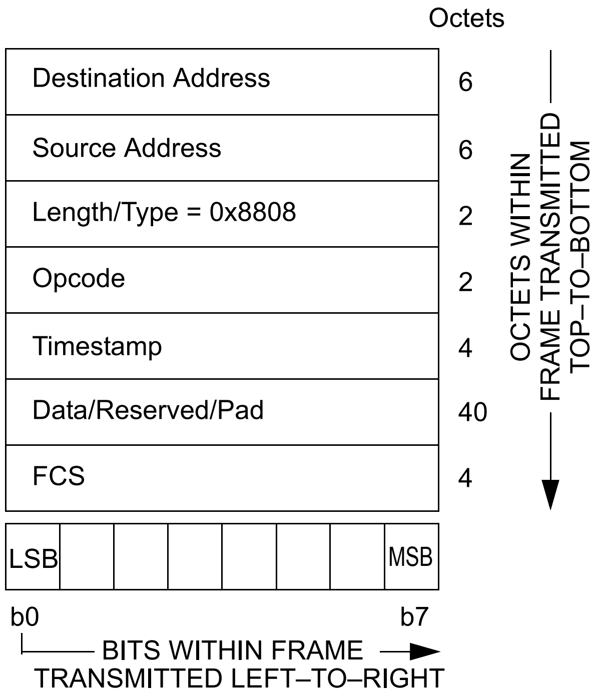


Figure 93–30—Generic MPCPDU

93.3.6.1 GATE description

The purpose of GATE message is to grant transmission windows to ONUs for both discovery messages and normal transmission. Up to four grants can be included in a single GATE message. The number of grants can also be set to zero for using the GATE message as an MPCP keep alive from OLT to the ONU.

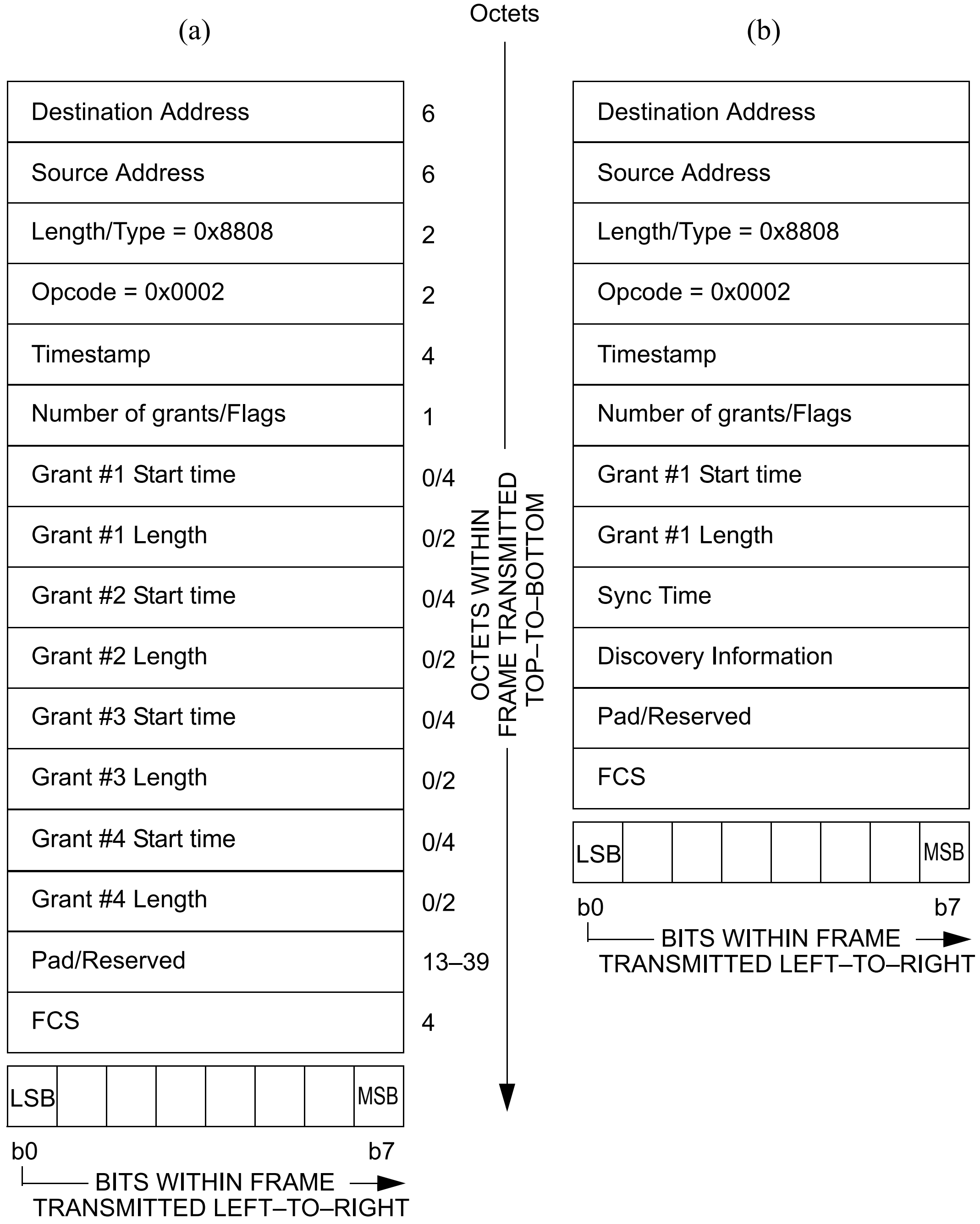


Figure 93–31—GATE MPCPDU: (a) normal GATE MPCPDU, (b) discovery GATE MPCPDU

The GATE MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the GATE MPCPDU is 0x0002.

b) Flags. This is an 8 bit flag register that holds the following flags: The Number of grants field contains the number of grants, composed of valid Length, Start Time pairs in this MPCPDU. This is a number between 0 and 4. Note: when Number of grants is set to 0, sole purpose of message is conveying of timestamp to ONU.
The Discovery flag field indicates that the signaled grants would be used for the discovery process, in which case a single grant shall be issued in the gate message.
The Force Report flag fields ask the ONU to issue a REPORT message related to the corresponding grant number at the corresponding transmission opportunity indicated in this GATE.

c) Grant #n Length. Length of the signaled grant, this is an 16 bit unsigned field. The length is counted in 1 time_quantum increments. There are 4 Grants that are possibly packed into the GATE MPCPDU. The laserOnTime, syncTime, and laserOffTime are included in and thus consume part of Grant #n Length.

d) Grant #n Start Time. Start time of the grant, this is an 32 bit unsigned field. The start time is compared to the local clock, to correlate the start of the grant. Transmitted values shall satisfy the condition Grant #n Start Time < Grant #n+1 Start Time for consecutive grants within the same GATE MPCPDU.

e) Sync Time. This is an unsigned 16 bit value signifying the required synchronization time of the OLT receiver. During the synchronization time the ONU shall send a synchronization pattern of 0x55 (transmission bit sequence 1010 ...) followed by a burst delimiter and idle blocks as defined in @@Subclause 92.2.3.5@@. The value is counted in 1 time_quantum increments. The advertised value includes synchronization requirement on all receiver elements including PMD, PMA and PCS. This field is present only when the gate is a discovery gate, as signaled by the Discovery flag and is not present otherwise.

f) Discovery Information. This is an 16 bit flag register. This field is present only when the gate is a discovery gate, as signaled by the Discovery flag and is not present otherwise. Table 93–1 presents the internal structure of the Discovery Information flag field.

g) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception when constructing a complying MPCP protocol implementation. The size of this field depends on the used Grant #n Length/Start Time entry-pairs as well as the presence of the Sync Time and Discovery Information fields, and varies in length from 13 – 39 accordingly.

The GATE MPCPDU shall be generated by a MAC Control instance mapped to an active ONU, and as such shall be marked with a unicast type of LLID, except when the discovery flag is set where the MAC Control instance is mapped to all ONUs and such frame is marked by the appropriate broadcast LLID (Subclause 93.3.2.3).

Table 93–1—GATE MPCPDU Discovery Information Fields

Bit	Flag Field	Values
0	OLT is 1G upstream capable	0 – OLT supports 1 Gb/s reception 1 – OLT does not support 1 Gb/s reception
1	OLT is 10G upstream capable	0 – OLT does not support 10 Gb/s reception 1 – OLT supports 10 Gb/s reception
2 – 3	reserved	Ignored on reception.
4	OLT is opening 1G discovery window	0 – OLT can receive 1 Gb/s data in this window 1 – OLT cannot receive 1 Gb/s data in this window
5	OLT is opening 10G discovery window	0 – OLT cannot receive 10 Gb/s data in this window 1 – OLT can receive 10 Gb/s data in this window
6 – 15	reserved	Ignored on reception.

Table 93–2—GATE MPCPDU Number of grants/Flags Fields

Bit	Flag Field	Values
0–2	Number of grants	0 – 4
3	Discovery	0 – Normal GATE 1 – Discovery GATE
4	Force Report Grant 1	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 1
5	Force Report Grant 2	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 2
6	Force Report Grant3	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 3
7	Force Report Grant 4	0 – No action required 1 – A REPORT frame should be issued at the corresponding transmission opportunity indicated in Grant 4

93.3.6.2 REPORT description

REPORT messages have several functionalities. Time stamp in each REPORT message is used for round trip (RTT) calculation. In the REPORT messages ONUs indicate the upstream bandwidth needs they request per 802.1Q priority queue. REPORT messages are also used as keep–alives from ONU to OLT. ONUs issue REPORT messages periodically in order to maintain link health at the OLT as defined in Subclause 93.3.4. In addition, the OLT may specifically request a REPORT message.

The REPORT MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the REPORT MPCPDU is 0x0003.
b) Number of Queue Sets. This field specifies the number of requests in the REPORT message. A REPORT frame may hold multiple sets of Report bitmap and Queue #n as specified in the Number of Queue Sets field.
c) Report bitmap. This is an 8 bit flag register that indicates which queues are represented in this REPORT MPCPDU – see Table 93–3

Table 93–3—REPORT MPCPDU Report bitmap fields

Bit	Flag Field	Values
0	Queue 0	0 – queue 0 report is not present; 1 – queue 0 report is present
1	Queue 1	0 – queue 1 report is not present; 1 – queue 1 report is present
2	Queue 2	0 – queue 2 report is not present; 1 – queue 2 report is present
3	Queue 3	0 – queue 3 report is not present; 1 – queue 3 report is present
4	Queue 4	0 – queue 4 report is not present; 1 – queue 4 report is present
5	Queue 5	0 – queue 5 report is not present; 1 – queue 5 report is present
6	Queue 6	0 – queue 6 report is not present; 1 – queue 6 report is present

Table 93–3—REPORT MPCPDU Report bitmap fields

Bit	Flag Field	Values
7	Queue 7	0 – queue 7 report is not present; 1 – queue 7 report is present

d) Queue #n Report. This value represents the length of queue# n at time of REPORT message generation. The reported length shall be adjusted to account for the necessary inter–frame spacing and FEC parity data overhead, if FEC is enabled. The Queue #n Report field is an unsigned 16 bit integer representing transmission request in units of time quanta. This field is present only when the corresponding flag in the Report bitmap is set.

e) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception when constructing a complying MPCP protocol implementation. The size of this field depends on the used Queue Report entries, and accordingly varies in length from 0 to 39.

The REPORT MPCPDU shall be generated by a MAC Control instance mapped to an active ONU, and as such shall be marked with a unicast type of LLID.

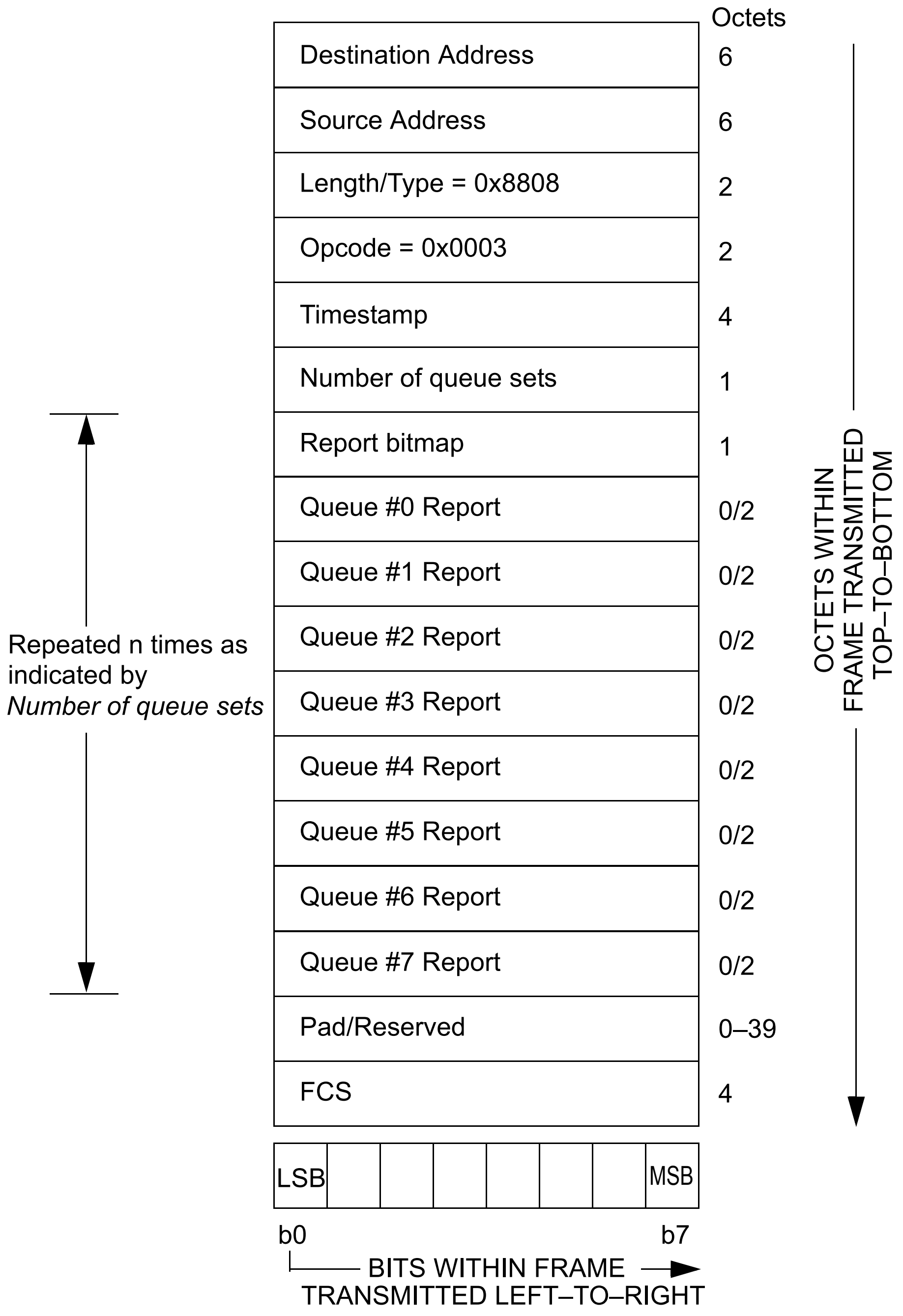


Figure 93–32—REPORT MPCPDU

93.3.6.3 .REGISTER_REQ description

The REGISTER_REQ MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the REGISTER_REQ MPCPDU is 0x0004.
b) Flags. This is an 8 bit flag register that indicates special requirements for the registration.
c) Pending grants. This is an unsigned 8 bit value signifying the maximum number of future grants the ONU is configured to buffer. The OLT should not grant the ONU more than this maximum number of Pending grants vectors comprised of {start, length, force_report, discovery} into the future.
d) Discovery Information. This is an 16 bit flag register. Table 93–5 presents the internal structure of the Discovery Information flag field.
e) Laser On Time. This field is 1 byte long and carries the Laser On Time characteristic for the given ONU transmitter. The value is expressed in the units of time_quanta.
f) Laser Off Time. This field is 1 byte long and carries the Laser Off Time characteristic for the given ONU transmitter. The value is expressed in the units of time_quanta.
g) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception when constructing a complying MPCP protocol implementation..

Table 93–4—REGISTER_REQ MPCPDU Flags fields

Value	Indication	Comment
0	reserved	Ignored on reception.
1	Register	Registration attempt for ONU.
2	reserved	Ignored on reception.
3	Deregister	This is a request to deregister the ONU. Subsequently, the MAC is deallocated and the LLID may be reused.
4–255	reserved	Ignored on reception.

Table 93–5—REGISTER_REQ MPCPDU Discovery Information Fields

Bit	Flag Field	Values
0	ONU is 1G upstream capable	0 – ONU transmitter is capable of 1 Gb/s 1 – ONU transmitter is not capable of 1 Gb/s
1	ONU is 10G upstream capable	0 – ONU transmitter is not capable of 10 Gb/s 1 – ONU transmitter is capable of 10 Gb/s
2 – 3	reserved	Ignored on reception.
4	1G registration attempt	0 – 1 G registration is attempted 1 – 1 G registration is not attempted
5	10 G registration attempt	0 – 10 G registration is not attempted 1 – 10 G registration is attempted
6 – 15	reserved	Ignored on reception.

The REGISTER_REQ MPCPDU shall be generated by a MAC Control instance mapped to an undiscovered ONU, and as such shall be marked with a broadcast type of LLID (Subclause 93.3.2.3).

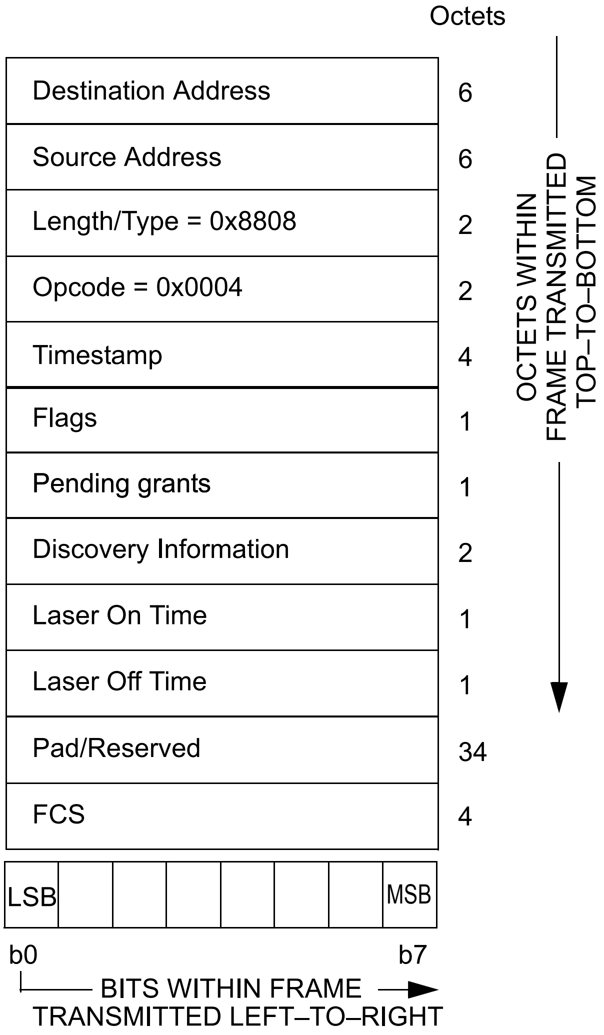


Figure 93–33—REGISTER_REQ MPCPDU

93.3.6.4 REGISTER description

The REGISTER MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) DA. The destination address used shall be an individual MAC address.
b) Opcode. The opcode for the REGISTER MPCPDU is 0x0005.
c) Assigned Port. This field holds a 16 bit unsigned value reflecting the LLID of the port assigned following registration.
d) Flags. this is an 8 bit flag register that indicates special requirements for the registration..

Table 93–6—REGISTER MPCPDU Flags field

Value	Indication	Comment
0	Reserved	Ignored on reception.
1	Reregister	The ONU is explicitly asked to re–register.

Table 93–6—REGISTER MPCPDU Flags field

Value	Indication	Comment
2	Deregister	This is a request to deallocate the port and free the LLID. Subsequently, the MAC is deallocated.
3	Ack	The requested registration is successful.
4	Nack	The requested registration attempt is denied by the higher–layer–entity.
5–255	Reserved	Ignored on reception.

e) Sync Time. This is an unsigned 16 bit value signifying the required synchronization time of the OLT receiver. During the synchronization time the ONU transmits only IDLE code–pairs. The value is counted in 1 time_quantum increments. The advertised value includes synchronization requirement on all receiver elements including PMD, PMA and PCS.

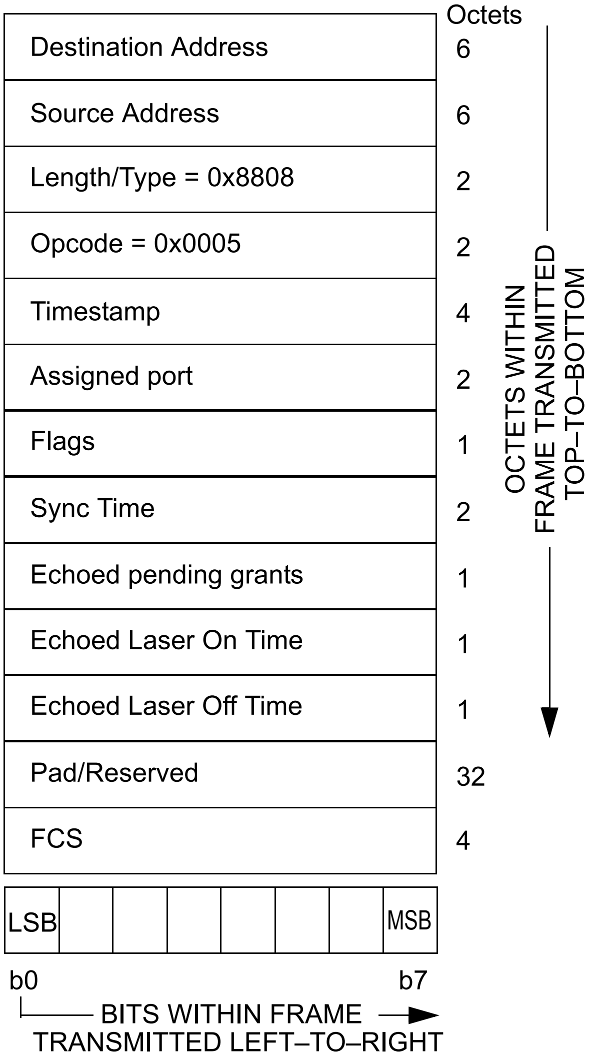


Figure 93–34—REGISTER MPCPDU

f) Echoed pending grants. This is an unsigned 8 bit value signifying the number of future grants the ONU may buffer before activating. The OLT should not grant the ONU more than this number of grants into the future.

g) Echoed Laser On Time. This is an unsigned 8 bit value signifying the Laser On Time for the given ONU transmitter. The value is expressed in the units of time_quanta. The value is delivered to the ONU for confirmation purposes only and its utilization is not prescribed in this specification.

h) Echoed Laser Off Time. This is an unsigned 8 bit value signifying the Laser Off Time for the given ONU transmitter. The value is expressed in the units of time_quanta. The value is delivered to the ONU for confirmation purposes only and its utilization is not prescribed in this specification.

i) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception when constructing a complying MPCP protocol implementation.

The REGISTER MPCPDU shall be generated by a MAC Control instance mapped to all ONUs and such frame is marked by the broadcast LLID (Subclause 93.3.2.3).

93.3.6.5 REGISTER_ACK description

The REGISTER_ACK MPCPDU is an instantiation of the Generic MPCPDU, and is further defined using the following definitions:

a) Opcode. The opcode for the REGISTER_ACK MPCPDU is 0x0006.

b) Flags. this is an 8 bit flag register that indicates special requirements for the registration.Echoed assigned port. This field holds a 16 bit unsigned value reflecting the LLID of the port assigned following registration.

c) Echoed Sync Time. This is an unsigned 16 bit value echoing the required synchronization time of the OLT receiver as previously advertised (Subclause 93.3.6.4).

Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored at reception when constructing a complying MPCP protocol implementation.

Table 93–7—REGISTER_ACK MPCPDU Flags fields

Value	Indication	Comment
0	Nack	The requested registration attempt is denied by the higher–layer–entity.
1	Ack	The registration process is successfully acknowledged.
2–255	Reserved	Ignored on reception.

The REGISTER_ACK MPCPDU shall be generated by a MAC Control instance mapped to an active ONU, and as such shall be marked with a unicast type of LLID.

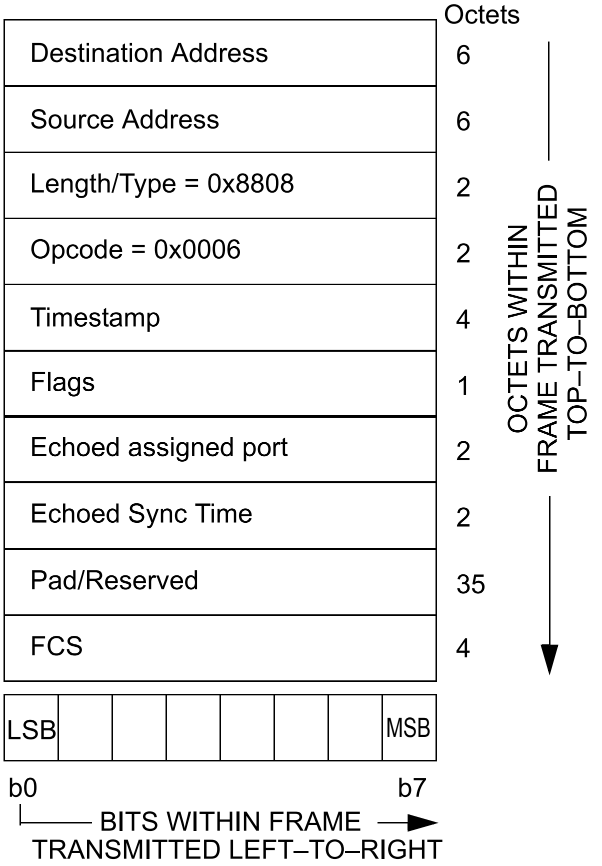


Figure 93–35—REGISTER_ACK MPCPDU

93.4 Discovery process in dual-rate systems (informative)

The enhancements introduced to the Clause 93 discovery process for EPONs facilitate the coexistence of 10G–EPON with legacy EPON.

93.4.1 OLT speed specific discovery

The discovery GATE MPCPDU is defined in Clause 64 for 1 Gb/s operation and in Clause 93 for 10 Gb/s operation. An additional field (Discovery Information field) was added to the 10 Gb/s discovery GATE MPCPDU. This field allows the OLT to relay speed specific information regarding the discovery window to the different ONUs that may co–exist on the same PON. The OLT has the ability to transmit common discovery GATE MPCPDUs on both the 1 Gb/s transmit path and 10 Gb/s transmit path, or it can send completely separate and independent GATE messages on these different paths. For each discovery window, the OLT is capable of opening windows for individual speeds or multiple speeds.

These different combinations allow the OLT Discovery Agent to open a number of discovery windows for all of the different ONU types. Table 93–8 shows the different types of windows that are possible, along with the necessary LLID and discovery information that also needs to be present in the discovery GATE MPCPDUs. For some combinations, it is necessary for the OLT Discovery Agent to open overlapping discovery windows by sending discovery GATE MPCPDUs on both the 1 Gb/s and 10 Gb/s downstream broadcast channels.

Table 93–8—Discovery GATE MPCPDUs for all ONU types.

ONU types on the PON [DS/US transmission speed]	LLID of discovery GATE(s)	Discovery Information			
		Upstream Capable		Discovery Window	
		1G	10G	1G	10G
1/1 Gb/s	0x7FFF	1	0	1	0
10/1 Gb/s	0x7FFE	1	0	1	0
1/1 Gb/s and 10/1 Gb/s	0x7FFF, 0x7FFE[a]	1	0	1	0
10/10 Gb/s	0x7FFE	0	1	0	1
10/1 Gb/s and 10/10 Gb/s	0x7FFE	1	1	1	1
1/1 Gb/s, 10/1 Gb/s, and 10/10 Gb/s	0x7FFF, 0x7FFE[a]	1	1	1	1

[a] Two discovery GATE MPCPDUs are transmitted in the downstream broadcast channel: one with the LLID of 0x7FFF transmitted on the 1 Gb/s downstream broadcast channel and another one the LLID of 0x7FFE transmitted on the 10 Gb/s downstream broadcast channel.

Figure 93–36 shows the three primary combinations of discovery windows and the different types of REGISTER_REQ MPCPDUs that may be received during the window. Figure 93–36(a) shows reception of messages from 1 Gb/s and 10/1 Gb/s ONUs. Figure 93–36(b) shows reception of messages from 10 Gb/s ONUs. Figure 93–36(c) shows reception of messages from all types of ONUs.

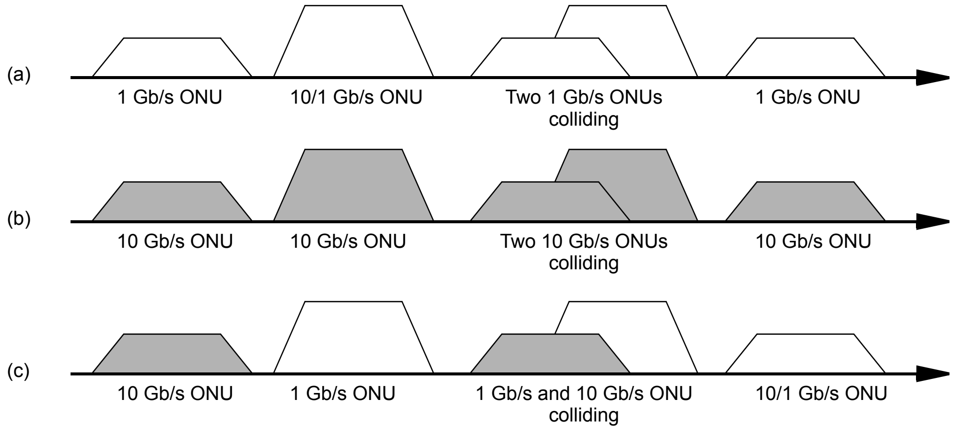


Figure 93–36—Combinations of REGISTER_REQ MPCPDUs during discovery window for 10G–EPON and EPON coexisting on the same PON.

93.4.2 ONU speed specific registration

A legacy 1 Gb/s ONU will only receive discovery GATE messages transmitted by the OLT on the 1 Gb/s broadcast channel. Operation and registration of these ONUs remains the same as previously, since no changes have been made to the existing 1 Gb/s discovery process.

A 10/1 Gb/s ONU is only capable of receiving discovery GATE MPCPDU transmitted by the OLT on the 10 Gb/s broadcast channel. These messages need to be parsed, and if a 1 Gb/s discovery window is opened, the ONU may attempt to register on the EPON.

A dual speed ONU capable of asymetric 10/1 Gb/s operation or symmetric 10/10 Gb/s operation is also only capable of receiving discovery GATE MPCPDU transmitted by the OLT on the 10 Gb/s broadcast channel. These messages need to be parsed, and the ONU makes the registration decision based on the available information. The ONU should attempt to register based during the discovery window announced as supporting the highest speed common to both the OLT and ONU. Table 93–9 shows the action the ONU should take based on the ONU transmit capabilities and the received discovery information.

Table 93–9—ONU action during discovery window

OLT Discovery Information				ONU Tx capability[a]		ONU Action
Upstream Capable		Discovery Window				
1G	10G	1G	10G	1G	10G	
1	0	1	0	1	X	Attempt 1G registration
0	1	0	1	X	1	Attempt 10G registration
1	1	0	1	0	1	Attempt 10G registration
1	1	0	1	1	0	Wait for 1G discovery window
1	1	0	1	1	1	Attempt 10G registration
1	1	1	0	0	1	Wait for 10G discovery window
1	1	1	0	1	0	Attempt 1G registration
1	1	1	0	1	1	Wait for 10G discovery window
1	1	1	1	0	1	Attempt 10G registration
1	1	1	1	1	0	Attempt 1G registration
1	1	1	1	1	1	Attempt 10G registration

[a] "+" in the 1G/10G column of the ONU Tx capability indicates that the given ONU Tx is capable of operation at the given data rate.

93.5 Protocol implementation conformance statement (PICS) proforma for Clause 93, Multipoint MAC Control[a]

93.5.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 93 Multipoint MAC Control, shall complete the following protocol implementation conformance statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

93.5.2 Identification

93.5.2.1 Implementation identification

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

93.5.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.3~~-2005~~-2008, Clause 93, Multipoint MAC Control
Identification of amendments and corrigenda to this PICS proforma that have been completed as part of this PICS	
Have any Exception items been required? No [ ] Yes [ ] (See Clause 21; the answer Yes means that the implementation does not conform to IEEE Std 802.3~~-2005~~-2008.)	
Date of Statement	

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

93.5.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
*OLT	OLT functionality	93.1	Device supports functionality required for OLT	O/1	Yes [ ] No [ ]
*ONU	ONU functionality	93.1	Device supports functionality required for ONU	O/1	Yes [ ] No [ ]

93.5.4 PICS proforma tables for Multipoint MAC Control

93.5.4.1 Compatibility Considerations

Item	Feature	Subclause	Value/Comment	Status	Support
CC1	Delay through MAC and PHY	93.3.2.4	Maximum delay variation of 16 ns (1 time_quantum)	M	Yes [ ]
CC2	OLT grant time delays	93.3.2.4	Not grant nearer than 1024 time_quanta into the future	OLT:M	Yes [ ]
CC3	ONU processing delays	93.3.2.4	Must process all messages in less than 1024 time_quanta	ONU:M	Yes [ ]
CC4	OLT grant issuance	93.3.2.4	Not grant more than one message every 1024 time_quanta	OLT:M	Yes [ ]

93.5.4.2 Multipoint MAC Control

Item	Feature	Subclause	Value/Comment	Status	Support
OM1	OLT localTime	93.2.2.2	Track transmit clock	OLT:M	Yes [ ]
OM2	ONU localTime	93.2.2.2	Track receive clock	ONU:M	Yes [ ]
OM3	Random wait for transmitting REGISTER_REQ messages	93.3.3	Shorter than length of discovery time window	ONU:M	Yes [ ]
OM4	Periodic report generation	93.3.4	Reports are generated periodically	ONU:M	Yes [ ]
OM5	Periodic granting	93.3.4	Grants are issued periodically	OLT:M	Yes [ ]
OM6	Issuing of grants	93.3.5	Not issue more than maximal supported grants	OLT:M	Yes [ ]

93.5.4.3 State ~~Diagrams~~diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	Multipoint Transmission Control	93.2.2.7	Meets the requirements of Figure 93–9	M	Yes [ ]
SM2	OLT Control Parser	93.2.2.7	Meets the requirements of Figure 93–10	M	Yes [ ]
SM3	ONU Control Parser	93.393.2.2.7	Meets the requirements of Figure 93–11	M	Yes [ ]
SM4	OLT Control Multiplexer	93.2.2.793.3	Meets the requirements of Figure 93–12	OLT:M	Yes [ ]
SM5	ONU Control Multiplexer	93.2.2.793.3	Meets the requirements of Figure 93–13	OLT:M	Yes [ ]
SM6	Discovery Processing OLT Window Setup	93.3.3.6	Meets the requirements of Figure 93–18	OLT:M	Yes [ ]
SM7	Discovery Processing OLT Process Requests	93.3.3.6	Meets the requirements of Figure 93–19	OLT:M	Yes [ ]
SM8	Discovery Processing OLT Register	93.3.3.6	Meets the requirements of Figure 93–20	ONU:M	Yes [ ]
SM9	Discovery Processing OLT Final Registration	93.3.3.6	Meets the requirements of Figure 93–21	OLT:M	Yes [ ]
SM10	Discovery Processing ONU Registration	93.3.3.6	Meets the requirements of Figure 93–22	ONU:M	Yes [ ]
SM11	Report Processing at OLT	93.3.4.6	Meets the requirements of Figure 93–24	OLT:M	Yes [ ]
SM12	Report Processing at ONU	93.3.4.6	Meets the requirements of Figure 93–25	ONU:M	Yes [ ]
SM13	Gate Processing at OLT	93.3.5.6	Meets the requirements of Figure 93–27	OLT:M	Yes [ ]
SM14	Gate Processing at ONU	93.3.5.6	Meets the requirements of Figure 93–28	ONU:M	Yes [ ]
SM15	Gate Processing ONU Activation	93.3.5.6	Meets the requirements of Figure 93–29	ONU:M	Yes [ ]

93.5.4.4 MPCP

Item	Feature	Subclause	Value/Comment	Status	Support
MP1	~~VLAN Tags~~MPCPDU structure	93.3.6	~~MPCPDU are not tagged~~ As in Figure 64–30	M	Yes [ ]
MP2	LLID for MPCPDU	93.3.6	RS generates LLID for MPCPDU	M	Yes [ ]
MP3	Grants during discovery	93.3.6.1	Single grant in GATE message during discovery	OLT:M	Yes [ ]
MP4	Grant start time	93.3.6.1	Grants within one GATE MPCPDU are sorted by their Start time values	OLT:M	Yes [ ]
MP5	TX during synchronization	93.3.6.1	Transmit IDLE code groups	ONU:M	Yes [ ]
MP6	GATE generation	93.3.6.1	GATE generated for active ONU except during discovery	OLT:M	Yes [ ]
MP7	GATE LLID	93.3.6.1	Unicast LLID except for discovery	OLT:M	Yes [ ]
MP8	REPORT issuing	93.3.6.2	Issues REPORT periodically	ONU:M	Yes [ ]
MP9	REPORT generation	93.3.6.2	Generated by active ONU	ONU:M	Yes [ ]
MP10	REPORT LLID	93.3.6.2	REPORT has unicast LLID	ONU:M	Yes [ ]
MP11	REGISTER_REQ generation	93.3.6.3	Generated by undiscovered ONU	ONU:M	Yes [ ]
MP12	REGISTER_REQ LLID	93.3.6.3	Use broadcast LLID	ONU:M	Yes [ ]
MP13	REGISTER DA address	93.3.6.4	Use individual MAC address	OLT:M	Yes [ ]
MP14	REGISTER generation	93.3.6.4	Generated for all ONUs	OLT:M	Yes [ ]
MP15	REGISTER_ACK generation	93.3.6.5	Generated by active ONU	ONU:M	Yes [ ]
MP16	REGISTER_ACK LLID	93.3.6.5	Use unicast LLID	ONU:M	Yes [ ]