102. ~~-~~Multipoint MAC Control for ~~10G–EPON~~EPoC

102.1 Overview

This clause deals with the mechanism and control protocols required in order to reconcile the ~~10 Gb/s P2MP topology~~ EPoC (EPON Protocol over Coax) into the Ethernet framework. The P2MP medium is a ~~passive optical~~ coax cable distribution network (~~PON), an optical network with no~~ CCDN) in which active and passive elements are present in the signal's path 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~~ EPON protocol, such a network is referred to as ~~Ethernet passive optical~~ EPON Protocol over coax network (~~EPON~~EPoC).

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

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

A simplified P2MP topology example is depicted in ~~Figure 4–1~~Figure 102–1. ~~Clause~~ Clause 67 provides additional examples of P2MP topologies.

In case of FDD mode, downstream and upstream directions are separated in frequency. In case of TDD mode, downstream and upstream directions are separated in time, with a portion of time allocated for downstream operations (DS Transmission Window) and another portion of time allocated for upstream operations (US Transmission Window). To facilitate the transitions from one direction to the other, guard intervals are typically inserted in between. An example of TDD timeline, as seen from the CLT side, is shown in Figure 102–2.

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

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~~ a coax cable multipoint network by defining a Multipoint MAC Control sublayer as an extension of the MPCP defined in Clause 77 and of the MAC Control sublayer defined in ~~Clause 31~~Clause 31, and supporting current and future operations as defined in ~~Clause 31and~~ Clause 31and annexes.

Each ~~PON~~ CCDN consists of a node located at the root of the tree assuming the role of ~~OLT~~CLT, and multiple nodes located at the tree leaves assuming roles of ~~ONUs~~CNUs. The network operates by allowing ~~only~~ a ~~single ONU~~ subset of CNUs multiplexed in frequency to transmit in the upstream direction at a time. The

MPCP located at the ~~OLT~~ CLT is responsible for timing the different transmissions. Reporting of congestion by the different ~~ONUs~~ CNUs may assist in optimally allocating the bandwidth across the ~~PON~~CCDN.

Automatic discovery of end stations is performed, culminating in registration through binding of an ~~ONU~~ CNU to an ~~OLT~~ CLT port by allocation of a Logical Link ID (see LLID in ~~76.2.6.1.3.2~~Y.2.6.1.3.2), and dynamic binding to a MAC connected to the ~~OLT.~~ CLT

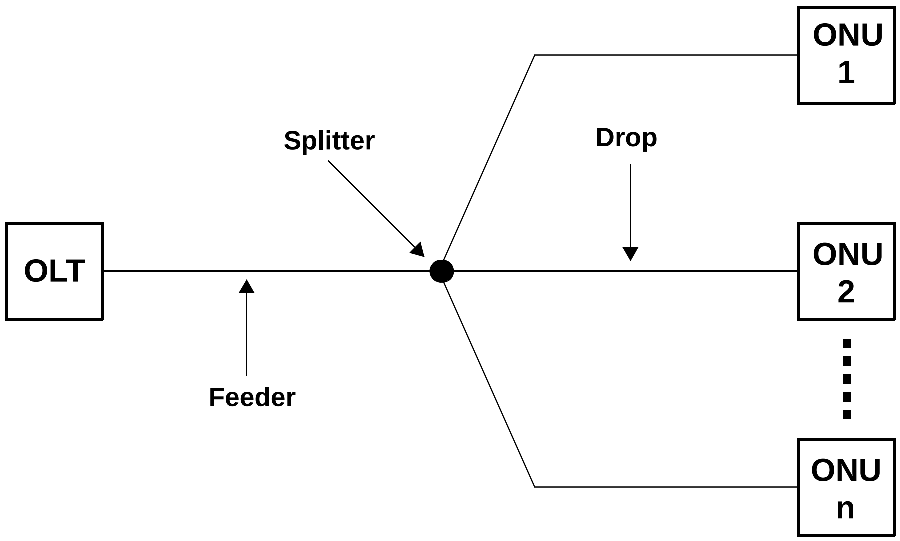


Figure 4–1—PON topology example

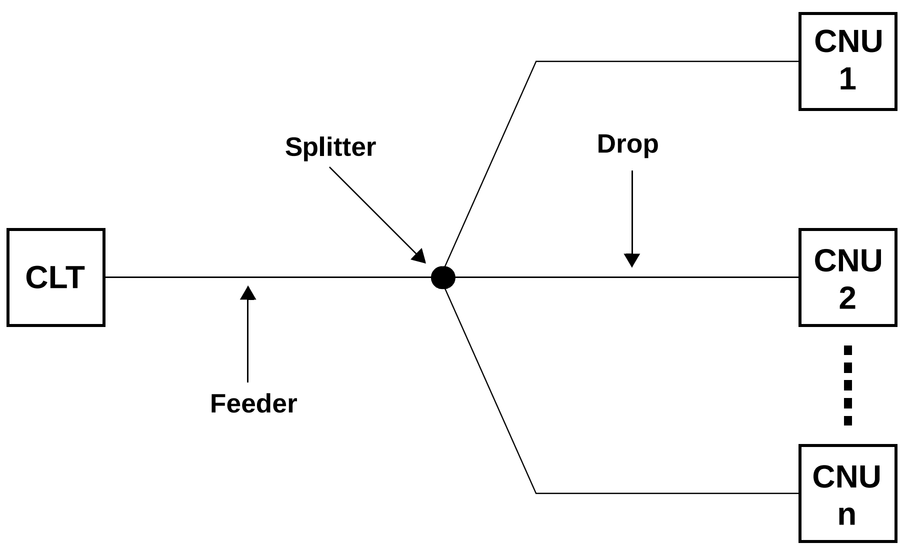


Figure 102–1—CDN topology example

The Multipoint MAC Control functionality shall be implemented for subscriber access devices containing point-to-multipoint Physical Layer devices defined in ~~Clause 75~~Clause X.

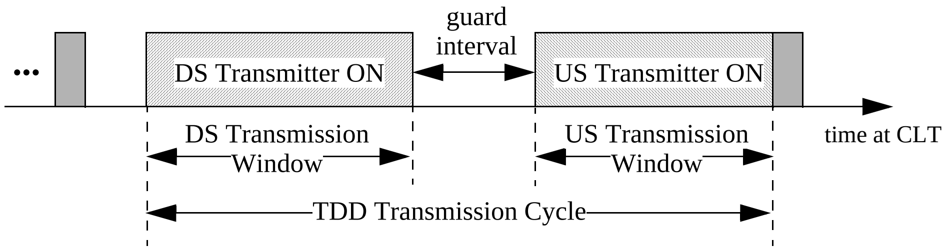


Figure 102–2—TDD timeline example, seen at the CLT side

102.1.1 Goals and objectives

The goals and objectives of this clause are the definition of a point-to-multipoint Ethernet network utilizing ~~an optical medium~~a coax cable medium in FDD or TDD mode.

Specific objectives met include the following:

a) Support of point-to-point Emulation (P2PE) as ~~specified~~specified
b) Support multiple LLIDs and MAC Clients at the ~~OLT~~CLT
c) Support a single LLID per ~~ONU~~CNU
d) Support a mechanism for single copy broadcast
e) Flexible architecture allowing dynamic allocation of ~~bandwidth~~bandwidth
f) Use of 32 bit timestamp for timing distribution
g) MAC Control based architecture
h) Ranging of discovered devices for improved network ~~performance~~performance
i) Continuous ranging for compensating round trip time variation

102.1.2 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 4–2~~Figure 102–3 ~~and Figure 4–3 depict~~ 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.

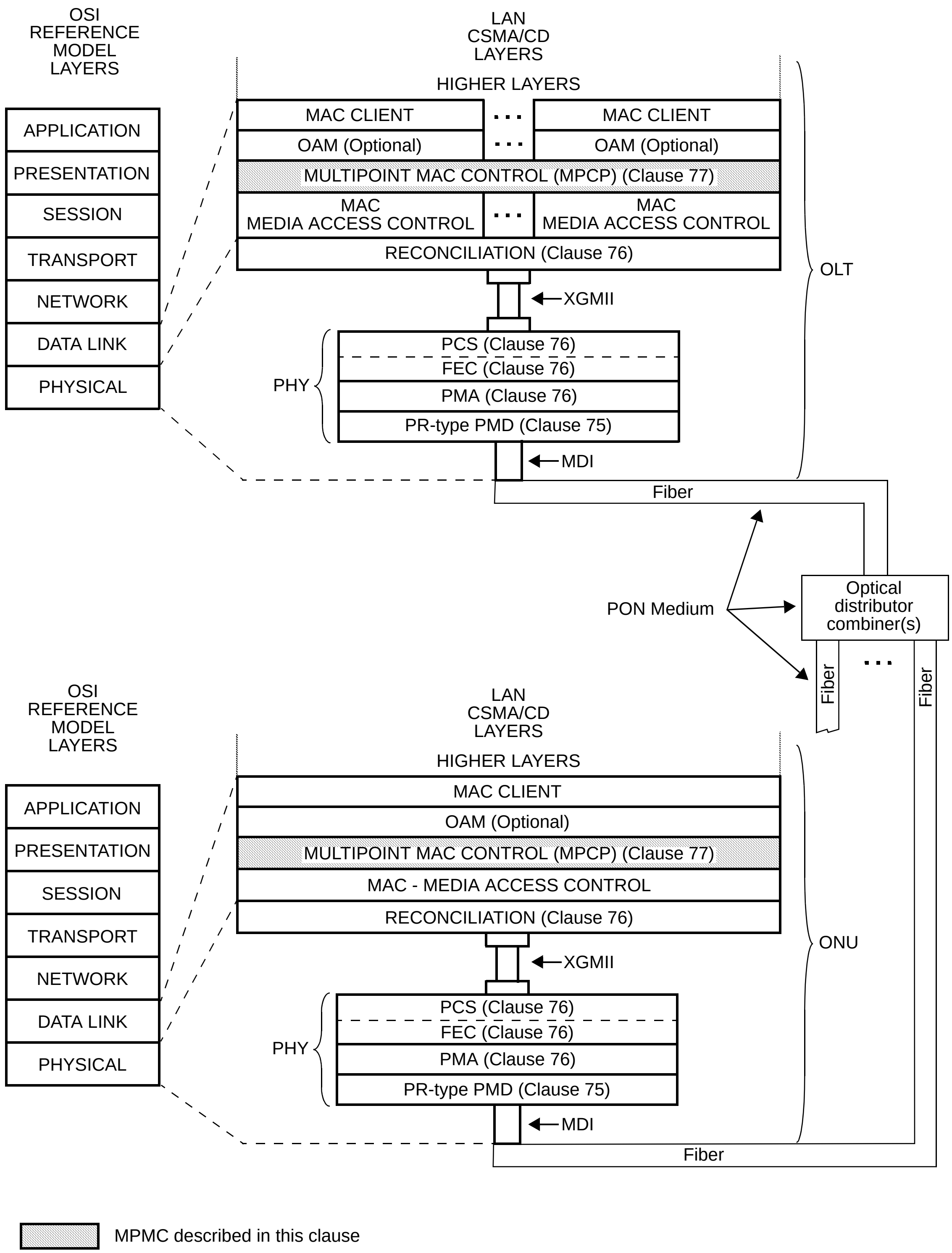
Multipoint MAC Control is defined using the mechanisms and precedents of the MAC Control sublayer. The MAC Control sublayer has extensive functionality designed to manage the real-time control and manipulation of MAC sublayer operation. This clause specifies the extension of the MAC Control mechanism 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 4–2 and Figure 4–3~~Figure 102–3, 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~~ CLT and the ~~ONU~~CNU. An additional MAC is instantiated to communicate to all ~~10G–EPON ONUs~~ EPoC CNUs 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 ~~10G–EPON ONUs~~EPoC CNUs. This MAC instance is referred to as Single Copy Broadcast (SCB).

The ~~ONU~~ CNU 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~~ CNU (see ~~76.2.6.1.3~~Y.2.6.1.3).

Although ~~Figure 4–2 and Figure 4–3~~Figure 102–3 and supporting text describe multiple MACs within the ~~OLT~~CLT, a single unicast MAC address may be used by the ~~OLT~~CLT. Within the ~~EPON~~ EPoC Network, MACs are uniquely identified by their LLIDs, which are dynamically assigned by the registration process~~.~~ .



XGMII= 10 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 4–2—Relationship of Multipoint MAC Control and the OSI protocol stack for 10/10G–EPON (10 Gb/s downstream and 10 Gb/s upstream)

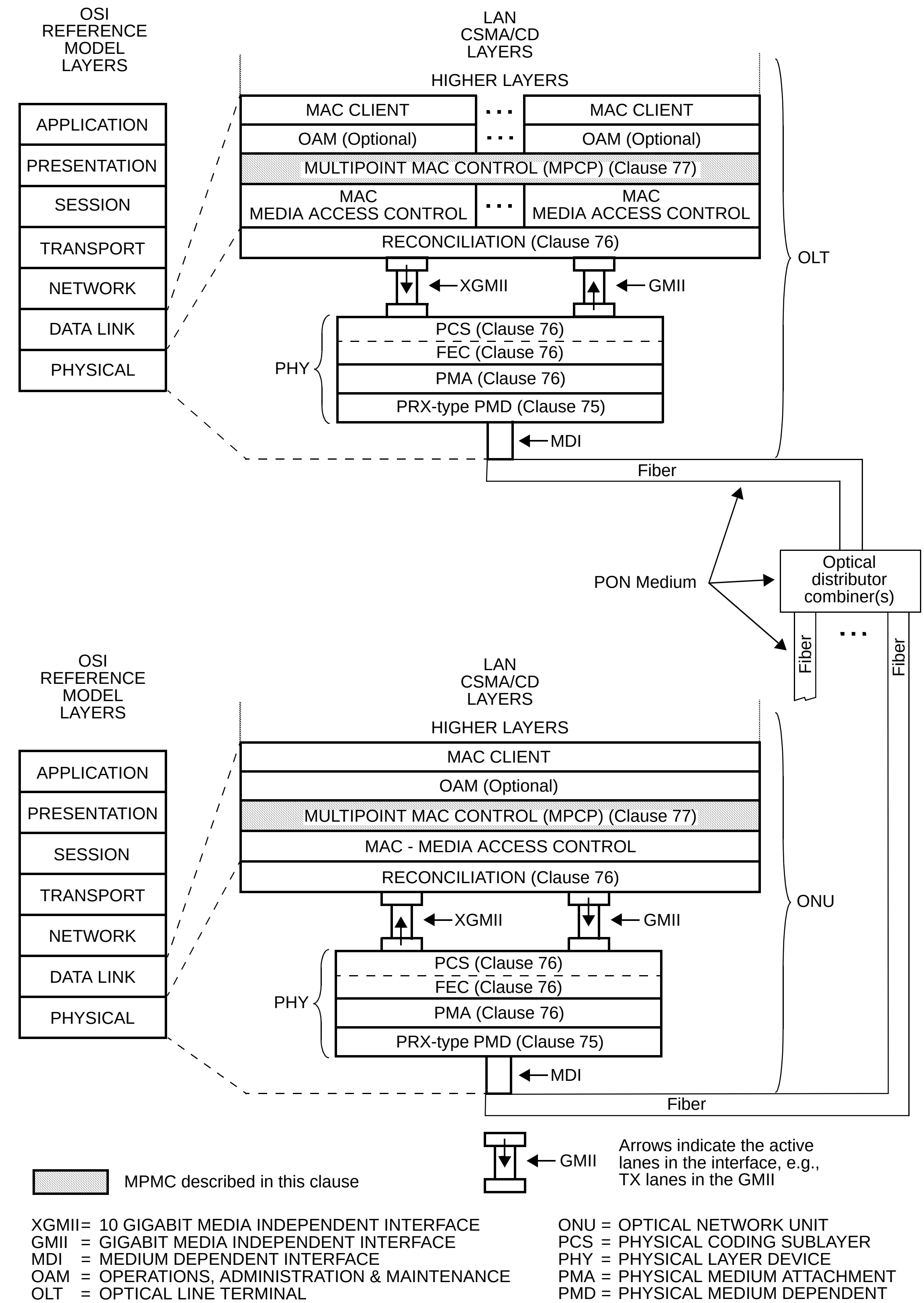


Figure 4–3—Relationship of Multipoint MAC Control and the OSI protocol stack for 10/1G–EPON (10 Gb/s downstream and 1 Gb/s upstream)

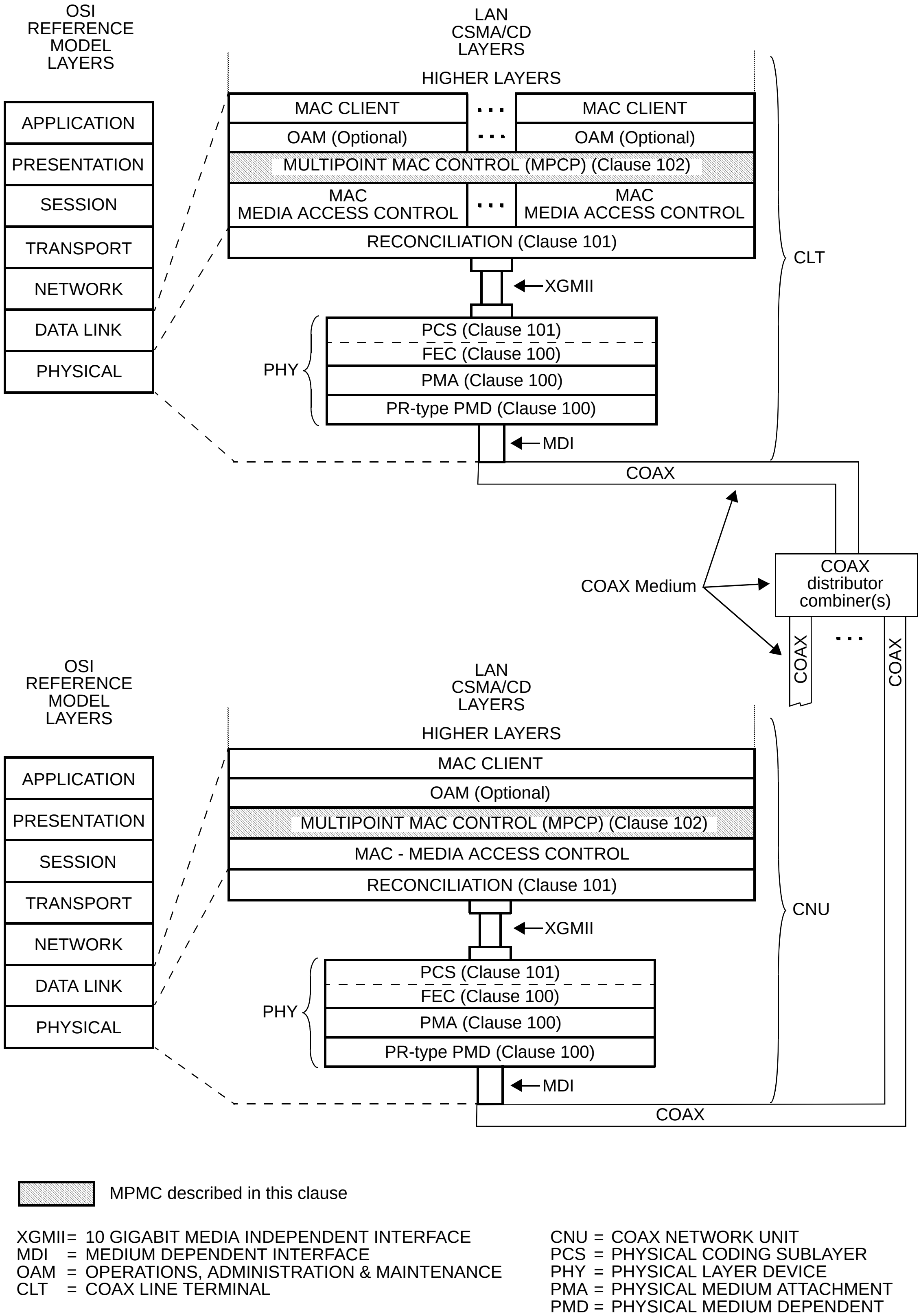


Figure 102–3—Relationship of Multipoint MAC Control and the OSI protocol stack for EPoC

102.1.3 Functional block diagram

~~Figure 4–4~~Figure 102–4 provides a functional block diagram of the Multipoint MAC Control architecture~~.~~.

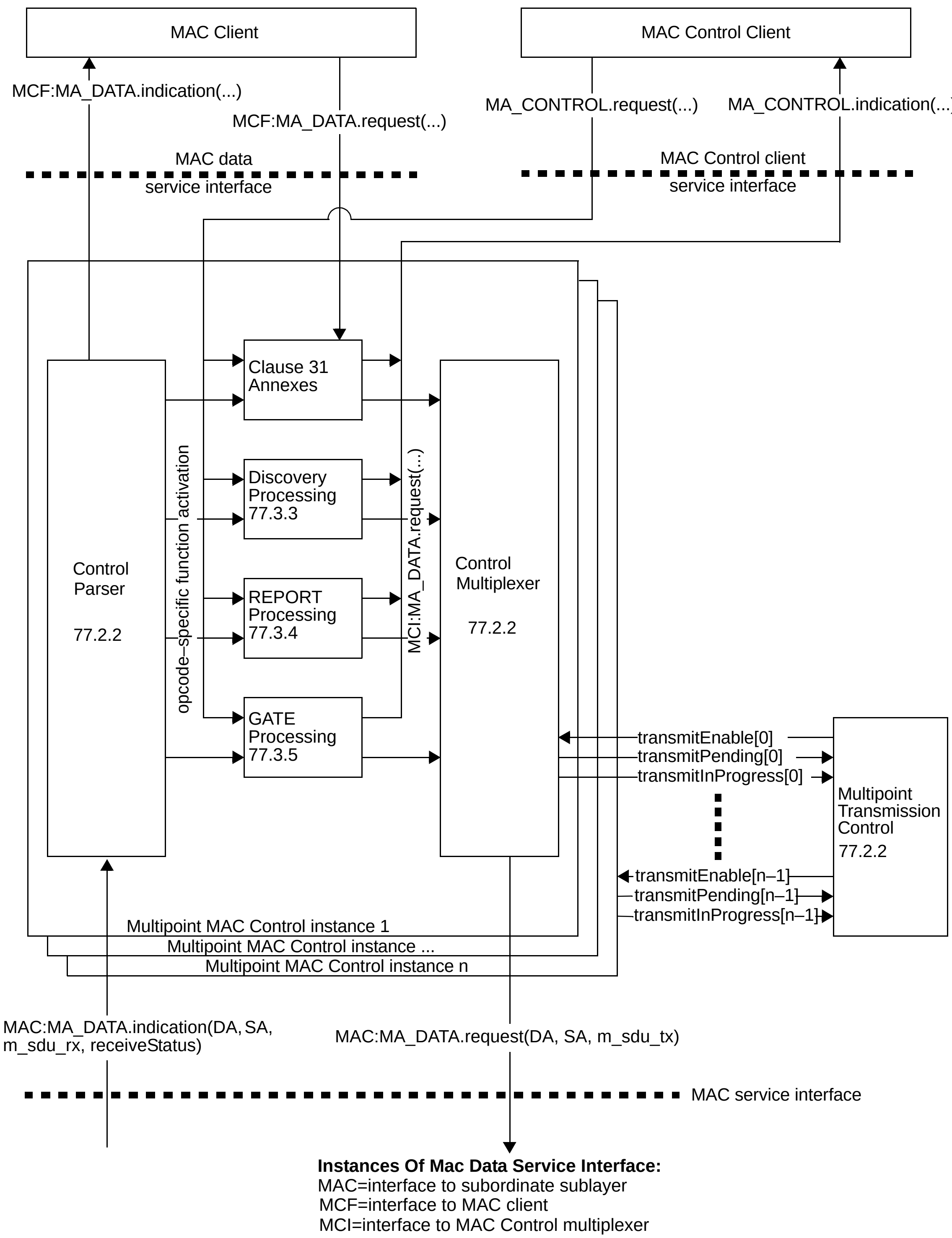


Figure 4–4—Multipoint MAC Control functional block diagram

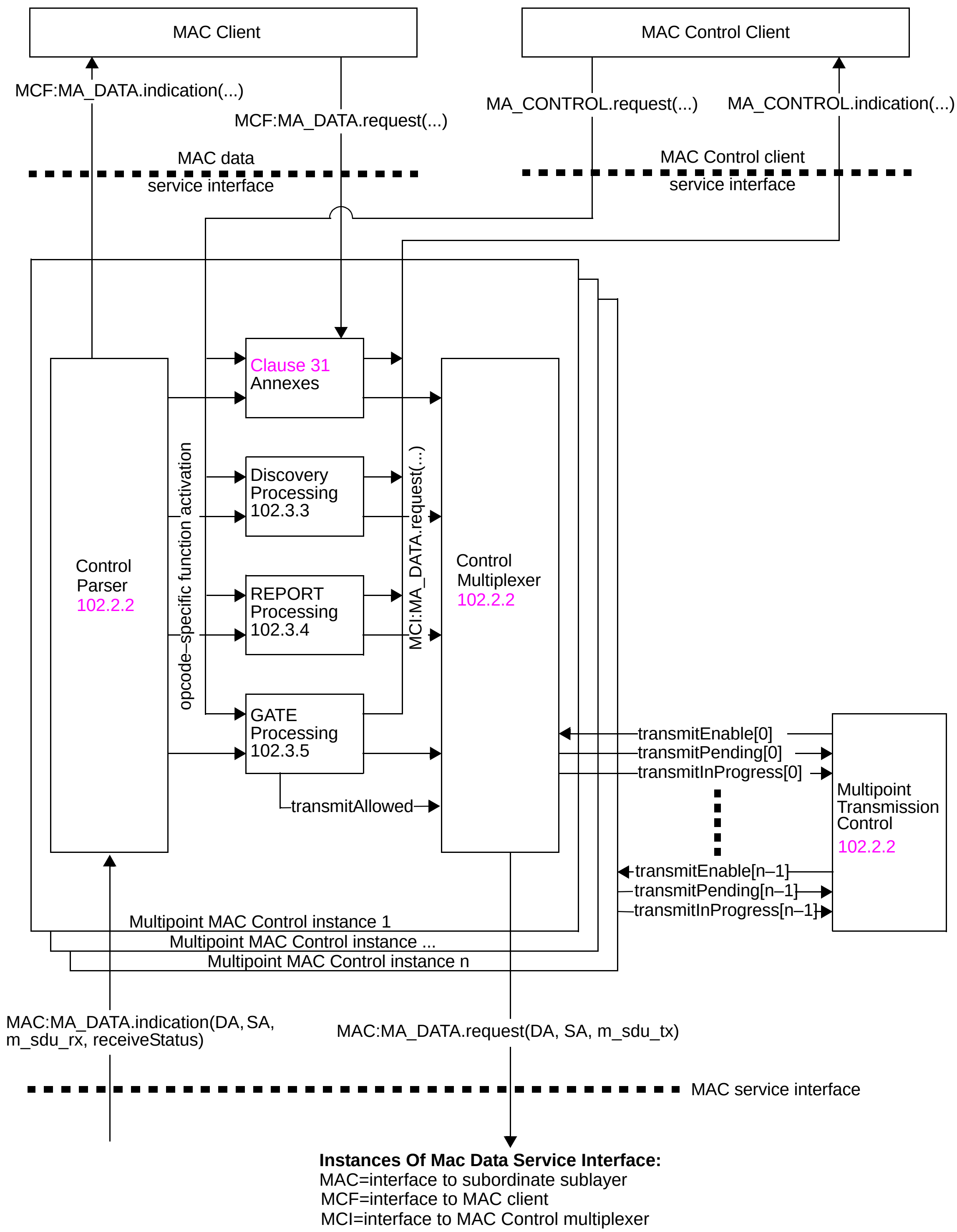


Figure 102–4—Multipoint MAC Control functional block diagram

102.1.4 Service interfaces

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

102.1.5 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 ~~21.5~~21.5. State diagram timers follow the conventions of ~~14.2.3.2~~ 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 ~~3.1~~ 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–~~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*)~~

a > b is equivalent to !(a < b or a = b)

a ≥ b is equivalent to !(a < b)

a ≤ b is equivalent to !(a > b)

102.2 Multipoint MAC Control operation

As depicted in ~~Figure 4–4~~Figure 102–4, 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~~ 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~~ 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.

102.2.1 Principles of Multipoint MAC Control

As depicted in ~~Figure 4–4~~Figure 102–4, 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~~ CLT to communicate with each ~~ONU~~CNU. The individual MAC instances utilize the point-to-point emulation service between the ~~OLT~~ CLT and the ~~ONU~~ CNU as defined in ~~76.2~~Y.2.

At the ~~ONU~~CNU, a single MAC instance is used to communicate with a MAC instance at the ~~OLT~~CLT. 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 MAC:MA_DATA.indication primitive in MAC j leads to the assertion of the MCF:MA_DATA.indication primitive to Client j. Conversely, the assertion of the request service interface in Client i leads to the assertion of the 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 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 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 ~~3.4~~3.4, are not passed to the Multipoint MAC Control sublayer in response to a 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 results in generation of the MAC:MA_DATA.indication primitive on that MAC's interface. Only one receive MAC interface is 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~~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.

102.2.1.1 Ranging and timing process

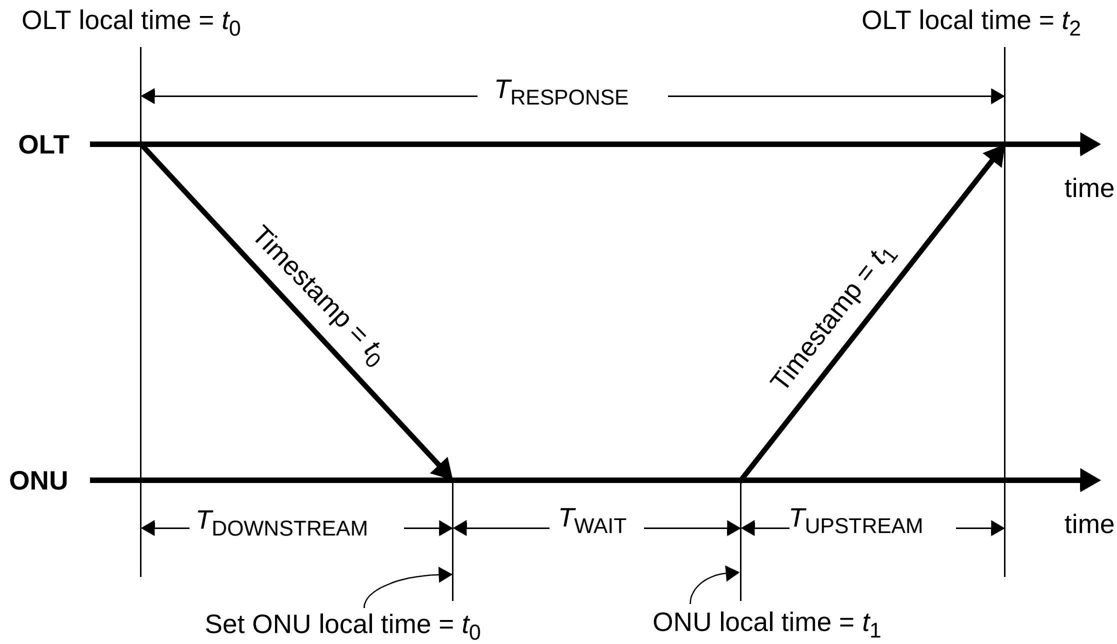
Both the ~~OLT~~ CLT and the ~~ONU~~ CNU have 32 bit counters that increment every 16 ns. These counters provide a local time stamp. When either device transmits an MPCPDU, it maps its counter value into the timestamp field. The time of transmission of the first octet of the MPCPDU frame from the MAC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

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

When the ~~OLT~~ CLT receives MPCPDUs, it uses the received timestamp value to calculate or verify a round trip time between the ~~OLT~~ CLT and the ~~ONU~~CNU. The Round Trip Time (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~~CLT's and ~~ONU~~CNU's clocks exceeds some predefined threshold. This condition can be independently detected by the ~~OLT~~ CLT or an ~~ONU~~CNU. The ~~OLT~~ CLT detects this condition when an absolute difference between new and old RTT values measured for a given ~~ONU~~ CNU exceeds the value of ~~guardThresholdOLT~~ guardThresholdCLT (see ~~102.2.2.1~~102.2.2.1), as shown in ~~Figure 4–11~~Figure 102–11. An ~~ONU~~ CNU detects the timestamp drift error condition when absolute difference between a timestamp received in an MPCPDU and the localTime counter exceeds ~~guardThresholdONU~~ guardThresholdCNU (see ~~102.2.2.1~~102.2.2.1), as is shown in ~~Figure 4–12.~~

Figure 102–12.



$T_{DOWNSTREAM}$ = downstream propagation delay

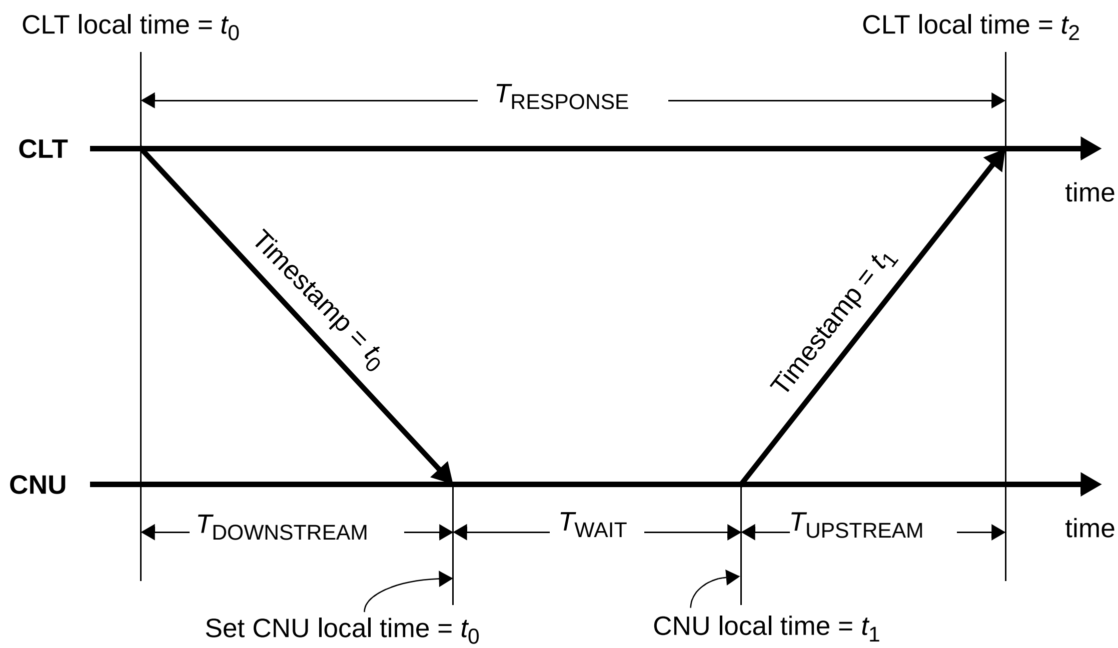
$T_{UPSTREAM}$ = upstream propagation delay

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 4–5—Round trip time calculation



$T_{DOWNSTREAM}$ = downstream propagation delay

$T_{UPSTREAM}$ = upstream propagation delay

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

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

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

Figure 102–5—Round trip time calculation

102.2.2 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.

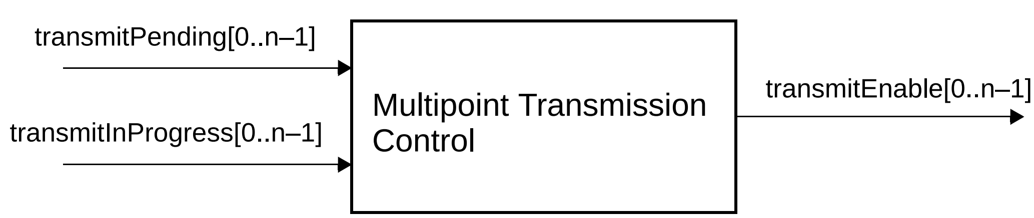


Figure 4–6—Multipoint Transmission Control service interfaces

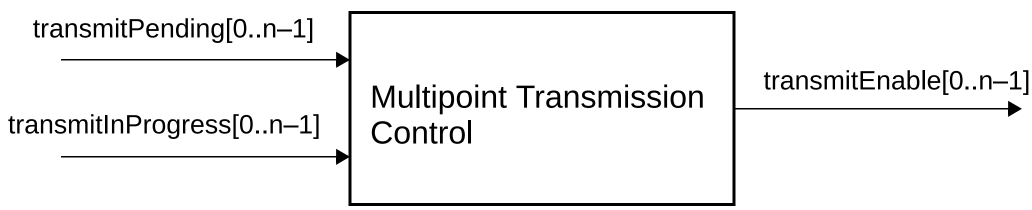


Figure 102–6—Multipoint Transmission Control service interfaces

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 4–4~~Figure 102–4).

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~~ 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 MAC:MA_DATA.request service primitive is generated for transmission. At the ~~OLT~~CLT, 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~~CNU, the Gate Processing functional block interfaces for upstream transmission administration~~.~~.

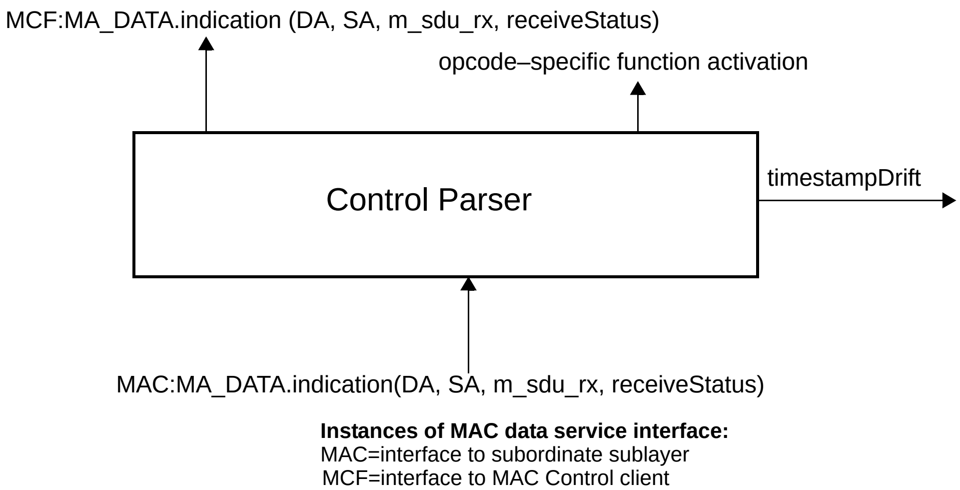
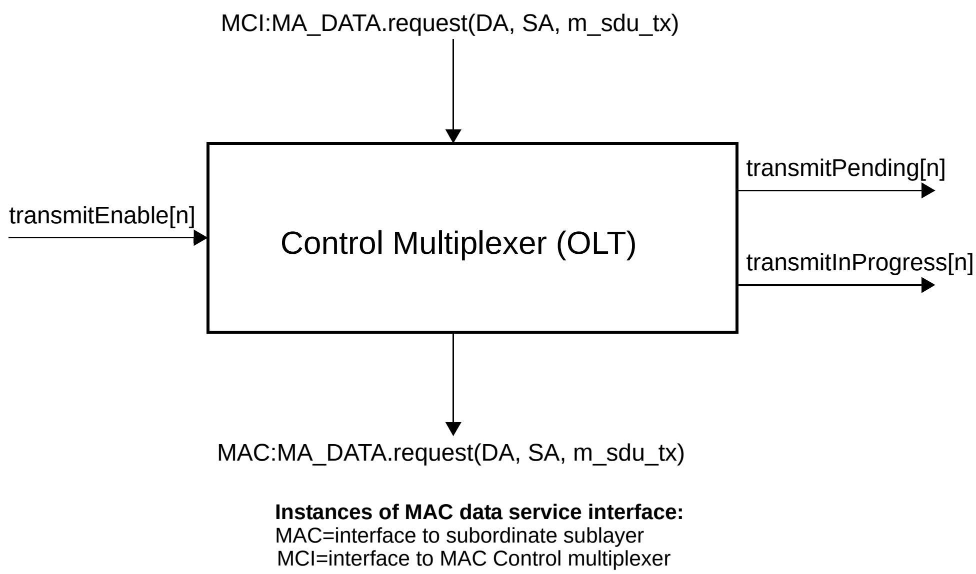
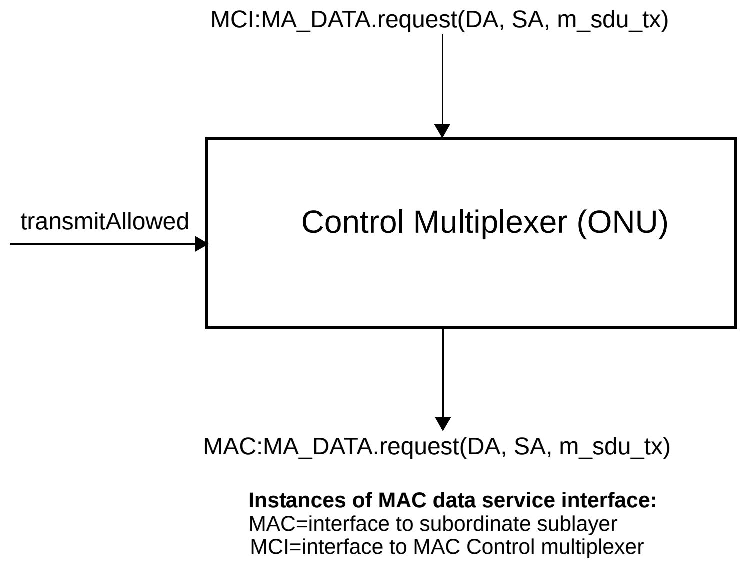


Figure 4–7—Control Parser service interfaces



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

Figure 4–8—OLT Control Multiplexer service interfaces



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

Figure 4–9—ONU Control Multiplexer service interfaces

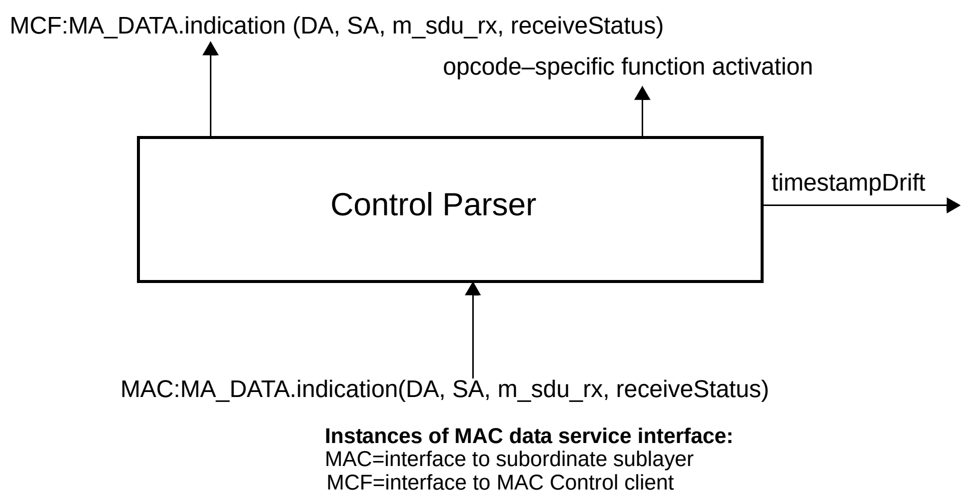
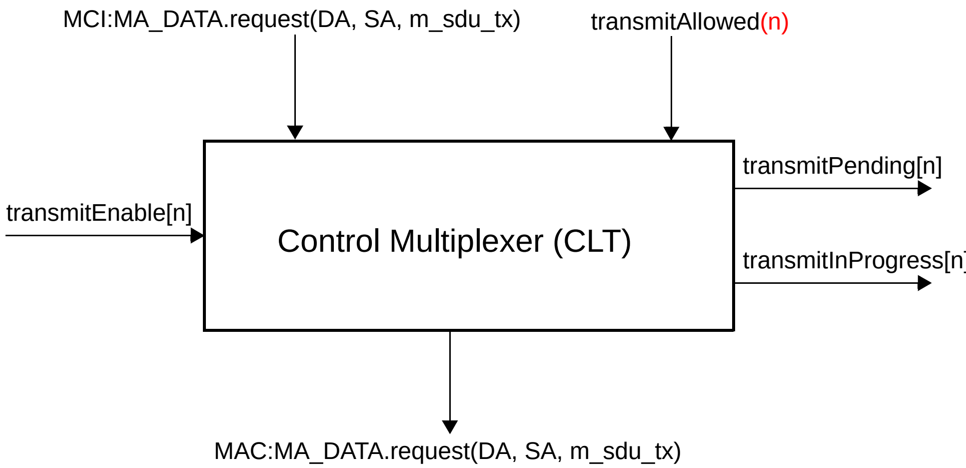


Figure 102–7—Control Parser service interfaces

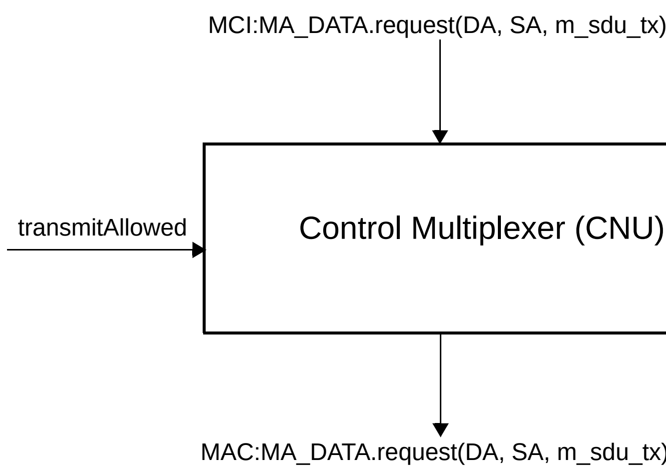


Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCI=interface to MAC Control multiplexer

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

Figure 102–8—CLT Control Multiplexer service interfaces

Editorial Note:In Figure 102-8 the baseline material did not include the “(n)” for “transmitAllowed”, the editor will add a comment to formalize this change.



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 102–9—CNU Control Multiplexer service interfaces

102.2.2.1 Constants

FEC_CODEWORD_SIZE
TYPE: integer
~~FEC_CODEWORD_SIZE~~
~~TYPE: integer~~
This constant represents the size of FEC codeword in octets (FEC_PAYLOAD_SIZE + FEC_PARITY_SIZE).
~~Value: 248~~
Value: (TBD)

FEC_PARITY_SIZE
TYPE: integer
~~FEC_PARITY_SIZE~~
~~TYPE: integer~~
This constant represents the size of FEC codeword parity field in octets~~.~~
~~Value: 32~~.
Value: (TBD)

FEC_PAYLOAD_SIZE
TYPE: integer
~~FEC_PAYLOAD_SIZE~~
~~TYPE: integer~~
This constant represents the size of FEC codeword payload in octets~~.~~
~~VALUE: 216~~.
VALUE: (TBD)

guardThresholdCLT
TYPE: integer
~~guardThresholdOLT~~
~~TYPE: integer~~
This constant holds the maximum amount of drift allowed for a timestamp received at the ~~OLT~~CLT. This value is measured in units of time_quantum.
~~VALUE: 12~~
VALUE: 12

guardThresholdCNU
TYPE: integer
~~guardThresholdONU~~
~~TYPE: integer~~
This constant holds the maximum amount of drift allowed for a timestamp received at the ~~ONU~~CNU. This value is measured in units of time_quantum.
~~VALUE: 8~~
VALUE: 8

MAC_Control_type
TYPE: integer
~~MAC_Control_type~~
~~TYPE: integer~~
The value of the Length/Type field as defined in ~~31.4.1.3~~31.4.1.
~~VALUE: 0x8808~~3.
VALUE: 0x8808

tailGuard

TYPE: integer

~~tailGuard~~

~~TYPE: integer~~

This constant holds the value used to reserve space at the end of the upstream transmission at the ~~ONU~~ CNU in addition to the 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 interpacket gap. The sizes of the above listed MAC overhead items are described in ~~3.1.1~~3.1.1. The size of the minimum IPG is described in ~~4A.4.2~~4A.4.

~~VALUE: 38~~2.

VALUE: 38

~~time_quantum~~

~~This variable is defined in 64.2.2.1.~~

time_quantum

This variable is defined in 64.2.2.1.

tqSize

TYPE: integer

~~tqSize~~

~~TYPE: integer~~

This constant represents time_quantum in octet transmission times.

~~VALUE: 20~~

VALUE: 20

EDITORS NOTE: the list of constants will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/ decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

102.2.2.2 Counters

localTime

TYPE: 32 bit unsigned

~~localTime~~

~~TYPE: 32 bit unsigned~~

This variable holds the value of the local timer used to control MPCP operation. This variable is advanced by a timer at 62.5 MHz, and counts in time_quanta. At the ~~OLT~~ CLT the counter shall track the transmit clock, while at the ~~ONU~~ CNU the counter shall track the receive clock. For accuracy of receive clock see ~~76.4.1.2~~Y.4.1.2. It is reloaded with the received timestamp value (from the ~~OLT~~CLT) by the Control Parser (see ~~Figure 4–12~~Figure 102–12). Changing the value of this variable while running using Layer Management is highly undesirable and is unspecified.

102.2.2.3 Variables

BEGIN

TYPE: Boolean

~~BEGIN~~

~~TYPE: Boolean~~

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

fecOffset

TYPE: 32 bit unsigned

fecOffset

TYPE: 32 bit unsigned

A variable that advances by 1 PHY_DATA_SIZE after every 8 ×(PHY_DATA_SIZE + PHY_OVERHEAD_SIZE) bit timestimes (see EPoC de-rating equation 102-1). After reaching the value of FEC_CODEWORD_SIZE, this variable is reset to zero. In the OLTCLT, this variable is initialized to 0 at system initialization. In the ONUCNU, this variable is assigned in the GATE Processing ONU CNU Activation state diagram (see Figure 4–14Figure 102–14).

NOTE–Notation fecOffset[1:0] refers to two least significant bits of this variable.

data_rx

TYPE: bit array

data_rx

TYPE: bit array

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.

data_tx

TYPE: bit array

data_tx

TYPE: bit array

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.

grantStart

TYPE: Boolean

grantStart

TYPE: Boolean

This variable indicates beginning of a grant transmission. It is set to true in the GATE Processing ONU Activation state diagram (see Figure 4–30Figure 102–29 and Figure 102–31) when a new grant activates. It is reset to false after the transmission of the first frame in the grant (see Figure 4–14Figure 102–14). This variable is defined in ONU CNU and, for TDD mode only, also in the CLT.

newRTT

TYPE: 16 bit unsigned

newRTT

TYPE: 16 bit unsigned

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

m_sdu_rx

TYPE: bit array

m_sdu_rx

TYPE: bit array

Equal to the concatenation of the Length/Type and data_rx variables.

m_sdu_tx

TYPE: bit array

m_sdu_tx

TYPE: bit array

Equal to the concatenation of the Length/Type and data_tx variables.

m_sdu_ctl

TYPE: bit array

~~m_sdu_ctl~~

~~TYPE: bit array~~

Equal to the concatenation of the MAC_Control_type and data_tx variables.

OctetsRemaining

TYPE: 32 bit unsigned

~~OctetsRemaining~~

~~TYPE: 32 bit unsigned~~

This variable is an alias for the expression (((stopTime – localTime) × tqSize) – tqOffset). It denotes the number of octets that can be transmitted between the current time and the end of the grant.

OctetsRequired

TYPE: 16 bit unsigned

~~OctetsRequired~~

~~TYPE: 16 bit unsigned~~

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~~ the transmission window. The value of OctetsRequired includes packet transmission time, tailGuard defined in ~~102.2.2.1~~102.2.2.1, and FEC parity data overhead. This variable is measured in units of octets.

opcode_rx

TYPE: 16 bit unsigned

~~opcode_rx~~

~~TYPE: 16 bit unsigned~~

This variable holds an opcode of the last received MPCPDU.

opcode_tx

~~opcode_tx~~

TYPE: 16 bit unsigned

~~This variable holds an opcode of an outgoing MPCPDU.~~

~~opcode_tx~~

~~TYPE: 16 bit unsigned~~

This variable holds an opcode of an outgoing MPCPDU.

packet_initiate_delay

TYPE: 16 bit unsigned

~~packet_initiate_delay~~

~~TYPE: 16 bit unsigned~~

This variable is used to set the ~~time−out~~ time?ut interval for packet_initiate_timer defined in ~~102.2.2.5~~102.2.2.5. The packet_initiate_delay value is represented in units of octets.

RTT

TYPE: 16 bit unsigned

~~RTT~~

~~TYPE: 16 bit unsigned~~

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

stopTime

TYPE: 32 bit unsigned

~~stopTime~~

~~TYPE: 32 bit unsigned~~

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 ~~102.3.5.~~102.3.5

timestamp
TYPE: 32 bit unsigned
~~timestamp~~
~~TYPE: 32 bit unsigned~~
This variable holds the value of timestamp of the last received MPCPDU frame.

timestampDrift
TYPE: Boolean
~~timestampDrift~~
~~TYPE: Boolean~~
This variable is used to indicate whether an error is signaled as a result of uncorrectable timestamp drift.

tqOffset
TYPE: 8 bit unsigned
~~tqOffset~~
~~TYPE: 8 bit unsigned~~
This variable denotes the offset (in octet times) of the current actual time from the localTime variable (which maintains the current time in units of time_quanta).

transmitAllowed
TYPE: Boolean
~~transmitAllowed~~
~~TYPE: Boolean~~
This variable is used to control PDU transmission at the ~~ONU~~CNU and at the CLT. 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.

transmitEnable
TYPE: Boolean array
~~transmitEnable~~
~~TYPE: Boolean array~~
This array contains one element per each Multipoint MAC Control instance. Elements of this array are used to control the transmit path in the Multipoint MAC Control instance at the ~~OLT~~CLT. Setting an element to TRUE indicates that the selected instance is permitted to transmit a frame. Setting it to FALSE inhibits the transmission of frames in the selected instance. Only one element of transmitEnable should be set to TRUE at a time.

transmitInProgress
TYPE: Boolean array
~~transmitInProgress~~
~~TYPE: Boolean array~~
This array contains one element per each Multipoint MAC Control instance. The element *j* of this array set to on indicates that the Multipoint MAC Control instance *j* is in the process of transmitting a frame.

transmitPending
TYPE: Boolean array
~~transmitPending~~
~~TYPE: Boolean array~~

This array contains one element per each Multipoint MAC Control instance. The element *j* of this array set to on indicates that the Multipoint MAC Control instance *j* is ready to transmit a frame.

PHY_DATA_SIZE
TYPE: integer
The number of octets constituting the denominator in the EPoC de-rating Equation (102–1). To normalize the effective data rate, the MPCP control multiplexer waits PHY_OVERHEAD_SIZE octets per every PHY_DATA_SIZE octets transmitted.
Value: {TBD}

$$\beta = \frac{XGMII\_Rate}{PCS\_Rate} = \frac{PHY\_DATA\_SIZE + PHY\_OVERHEAD}{PHY\_DATA\_SIZE} \quad (102–1)$$

PHY_OVERHEAD_SIZE
TYPE: integer
The number of octets constituting (together with PHY_DATA_SIZE) the numerator in the EPoC de-rating Equation (102–1). To normalize the effective data rate, the MPCP control multiplexer waits PHY_OVERHEAD_SIZE octets per every PHY_DATA_SIZE octets transmitted.
Value: {TBD}

EDITORS NOTE: The list of variables will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/ decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled.

102.2.2.4 Functions

abs(n)
~~abs(n)~~
This function returns the absolute value of the parameter n.

Opcode–specific function(opcode)
~~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.

CheckGrantSize(length)
~~CheckGrantSize(length)~~
This function calculates the future time at which the transmission of the current frame (including the FEC parity overhead) is completed~~-~~.

$$\text{CheckGrantSize(length)} = \left\lceil \frac{\text{fecOffset} + \text{length}}{\text{FEC\_PAYLOAD\_SIZE}} \right\rceil \times \text{FEC\_CODEWORD\_SIZE} - \text{fecOffset}$$

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

$$eckGrantSize\langle length\rangle = \left\lceil \frac{fecOffset + length}{FECPAYLOADSIZE} \right\rceil \times FEC\ CODEWORD\ SIZE - fecOffse \quad (102–2)$$

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

PMD_Overhead(length)

~~FEC_Overhead(length)~~

This function calculates the additional amount of time (in octet times) that the MPCP control multiplexer waits following transmission of a frame of size ‘length’ by the MAC. The additional time is added to allow the insertion of parity data into the frame by the PHY ~~layer. As described in 76.3.2.4, FEC encoder adds 32 parity octets for each block of 216~~ layer and to adjust the data rate to the effective data ~~or control octets~~rate supported by the PCS and PMD. ~~FEC_Overhead~~PMD_Overhead(~~)~~).-returns the number of octets that the PHY inserts during transmission of a particular packet and its subsequent IPG. Parameter ‘length’ represents the size of an entire frame including preamble, SFD, DA, SA, Length/Type, FCS, and IPG. The following formula is used to calculate the overhead~~:~~

$$\mathrm{FEC\_Overhead(length)} = 12 + \mathrm{FEC\_PARITY\_SIZE} \times \left\lfloor \frac{\mathrm{fecOffset} + \mathrm{length}}{\mathrm{FEC\_PAYLOAD\_SIZE}} \right\rfloor$$

NOTE—The notation $\lfloor x \rfloor$ represents a *floor* function, which returns the value of its argument *x* rounded down to the nearest integer.

:

$$PMD\_Overhead((length), \beta) = \quad (102\text{–}3)$$

$$12 + \left\lceil (\beta - 1) \times length + \beta \times \left(FEC\_PARITY\_SIZE \times \left\lfloor \frac{fecOffset + length}{FEC\_PAYLOAD\_SIZE} \right\rfloor\right) \right\rceil$$

NOTE—The notation $\lfloor x \rfloor$ represents a floor function, which returns the value of its argument x rounded down to the nearest integer.

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

select()

~~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()

~~SelectFrame()~~

This function enables the interface, which has a pending frame. If multiple interfaces have frames waiting at the same time, only one interface is 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)

~~sizeof(sdu)~~

This function returns the size of the sdu in octets.

transmissionPending()

~~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:

$$transmissionPending(\ ) = transmitPending(0) + transmitPending(1) + \ldots + transmitPending(n - 1) \quad (102\text{–}4)$$

EDITORS NOTE: the text below represents the above equation in the same format as in the baseline document. This format may not be compatible with IEEE Style

transmissionPending() =
transmitPending[0] +
transmitPending[1] ~~+~~ +
... ~~+~~ +
transmitPending[n~~–~~-1]
where n is the total number of Multipoint MAC Control instances.

END OF EDITORS NOTE

EDITORS NOTE: the list of function will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/ decisions/decisions.html) once EPoC-specific FEC and PMD overhead details are settled. In particular, further checks are needed for the function CheckGrantSize(), in relation to data rata adaption changes.

102.2.2.5 Timers

packet_initiate_timer
~~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, this timer increases interframe spacing just enough to accommodate the extra parity data to be added by the FEC encoder.

102.2.2.6 Messages

MAC:MA_DATA.indication(DA, SA, m_sdu, receiveStatus~~)~~
~~The service primitive is defined in 31.3.~~)
The service primitive is defined in 31.3.

MCF:MA_DATA.indication(DA, SA, m_sdu, receiveStatus~~)~~
~~The service primitive is defined in 31.3.~~)
The service primitive is defined in 31.3.

MAC:MA_DATA.request (DA, SA, m_sdu)
~~MAC:MA_DATA.request (DA, SA, m_sdu)~~

The service primitive is defined in ~~31.3~~31.3. The action invoked by this service primitive is not considered to end until the transmission of the frame by the MAC has concluded. The ability of the MAC control layer to determine this is implementation dependent.

MCF:MA_DATA.request (DA, SA, m_sdu)
~~The service primitive is defined in 31.3.~~
The service primitive is defined in 31.3.

102.2.2.7 State diagrams

The Multipoint transmission control function in the ~~OLT~~ CLT shall implement state diagram shown in ~~Figure 4–10~~Figure 102–10. Control parser function in the ~~OLT~~ CLT shall implement state diagram shown in ~~Figure 4–11~~Figure 102–11. Control parser function in the ~~ONU~~ CNU shall implement state diagram shown in ~~Figure 4–12~~Figure 102–12. Control multiplexer function in the ~~OLT~~ CLT shall implement state diagram shown in ~~Figure 4–13~~Figure 102–13. Control multiplexer function in the ~~ONU~~ CNU shall implement state diagram shown in ~~Figure 4–14~~Figure 102–14.

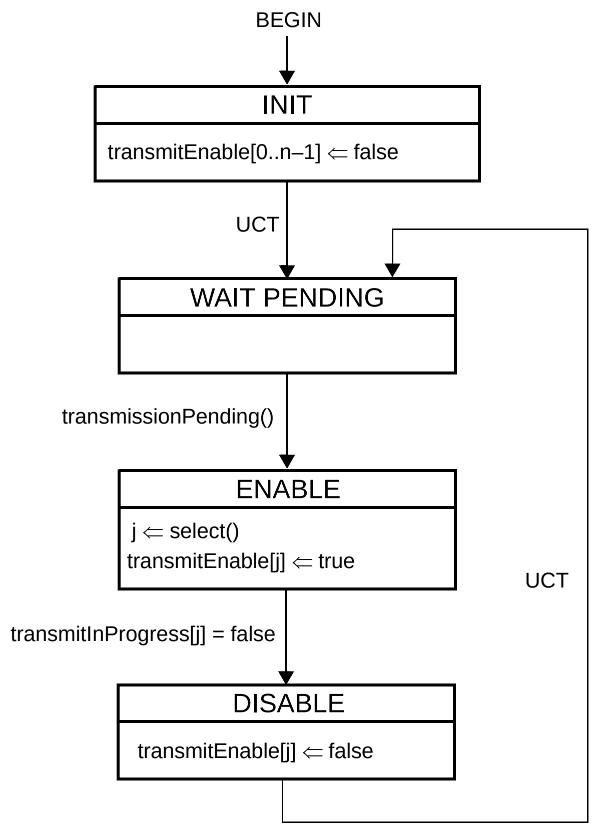
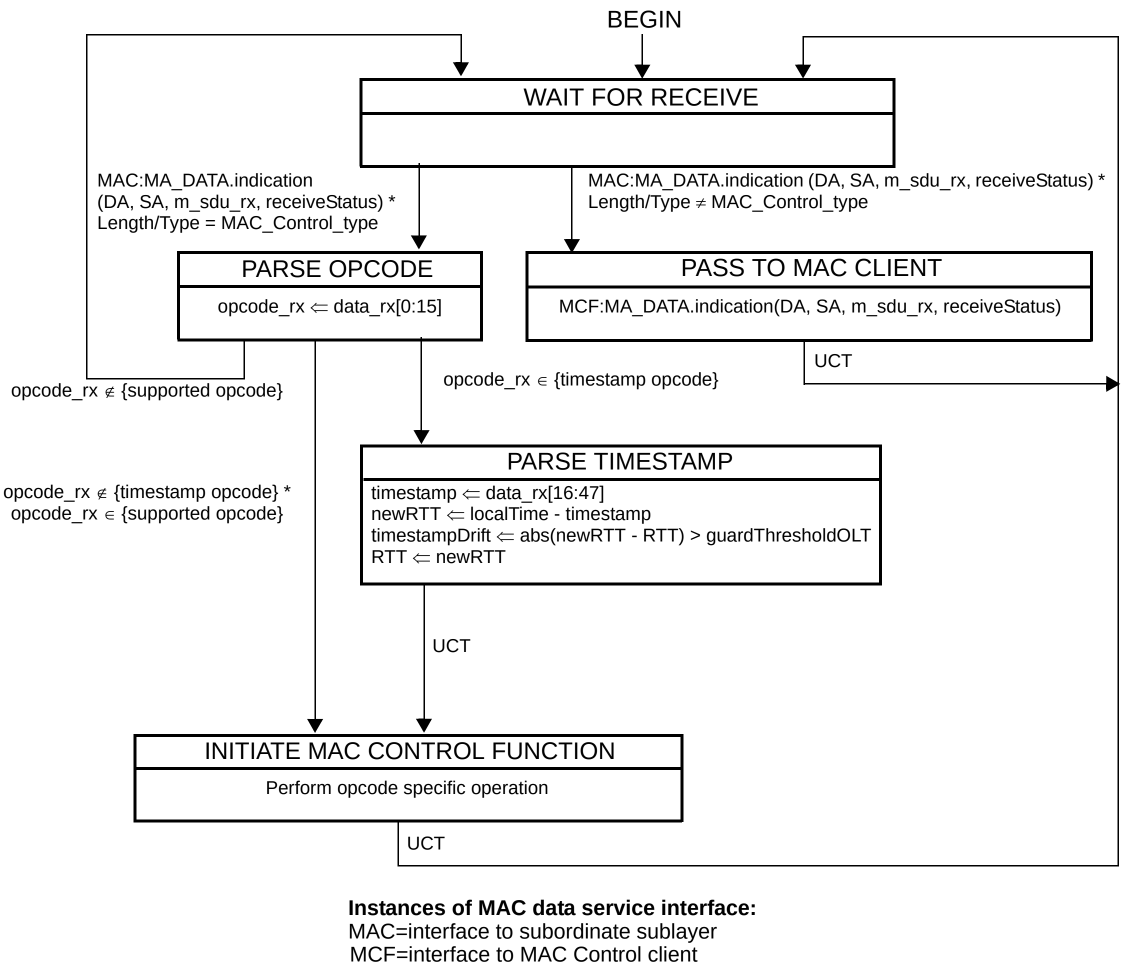


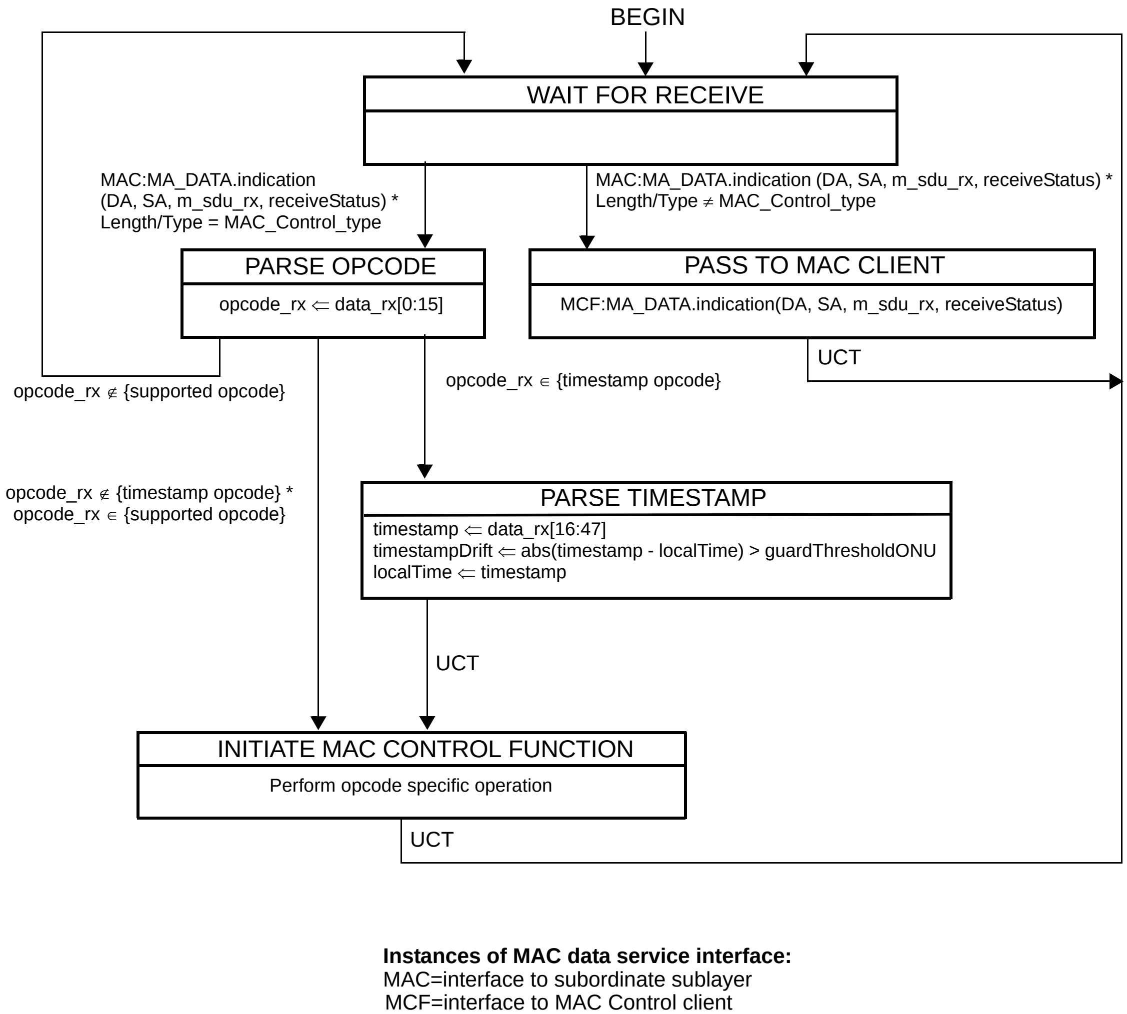
Figure 4–10—OLT Multipoint Transmission Control state diagram



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 4–11—OLT Control Parser state diagram



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 4–12—ONU Control Parser state diagram

:

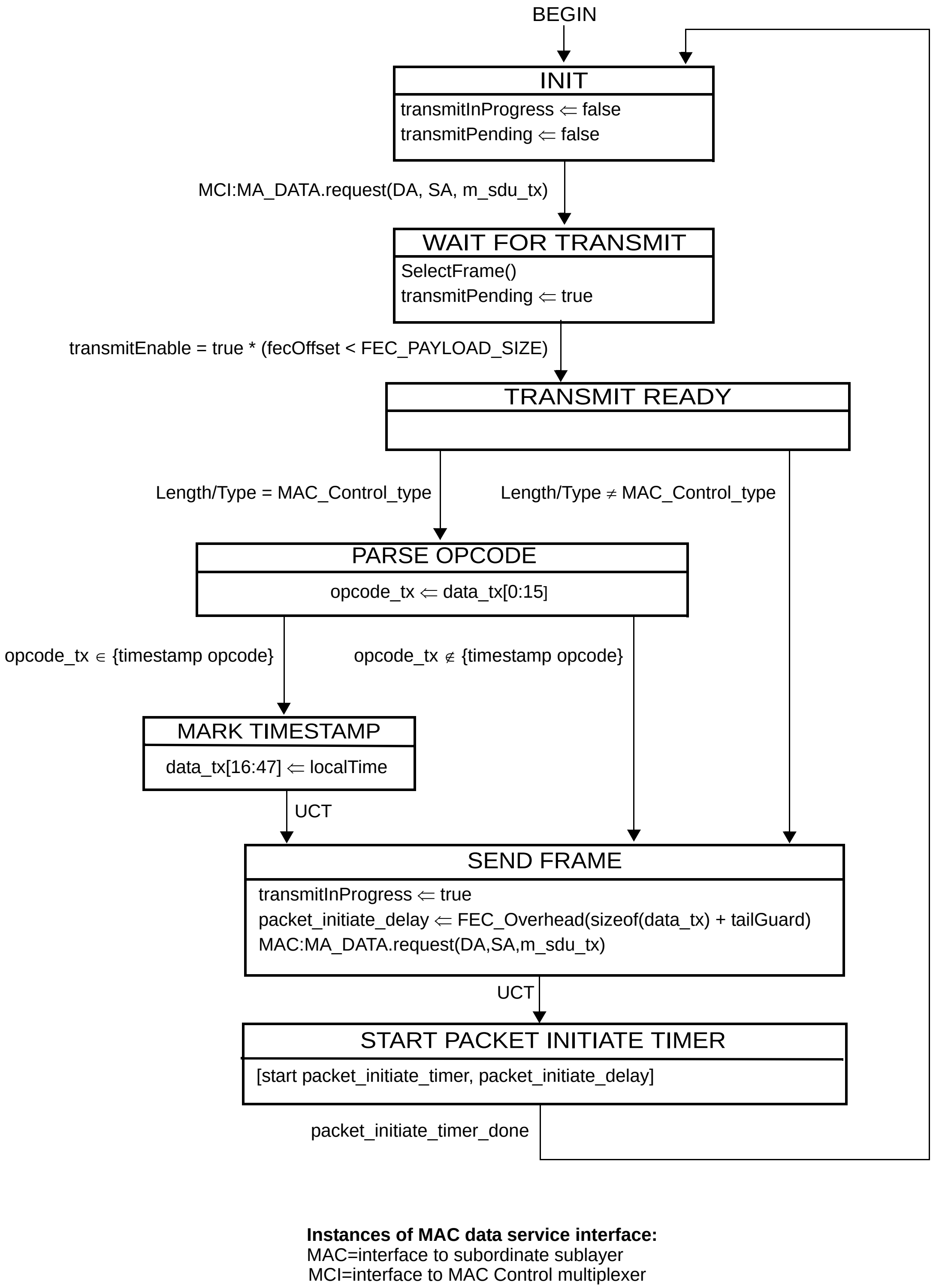


Figure 4–13—OLT Control Multiplexer state diagram

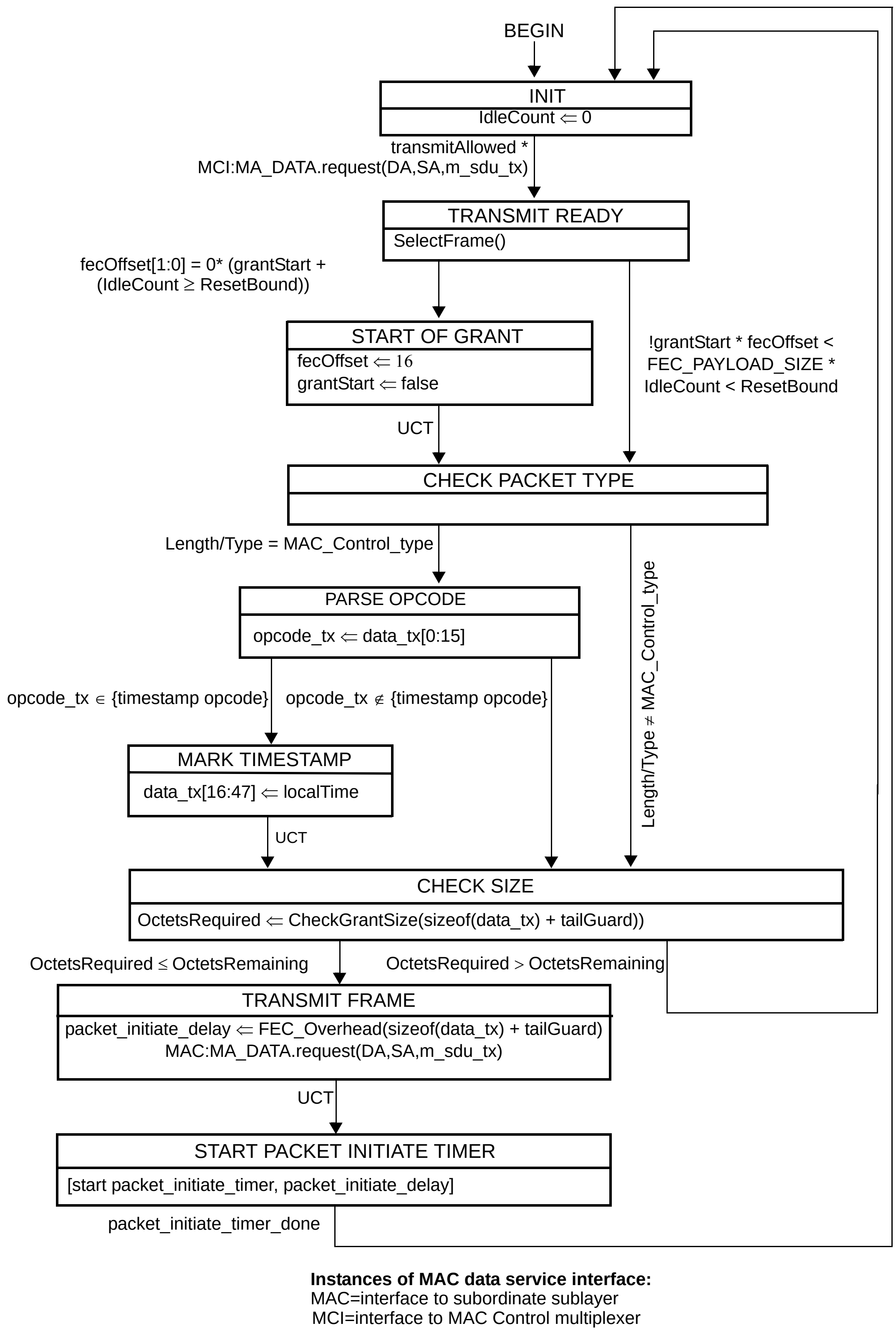


Figure 4–14—ONU Control Multiplexer state diagram

EDITORS NOTE: Figures 102–13 and 102–14 will be updated per technical decision #44 (http://www.ieee802.org/3/bn/public/decisions/decisions.html) once EPoC-specific FEC and PMD over-head details are settled.

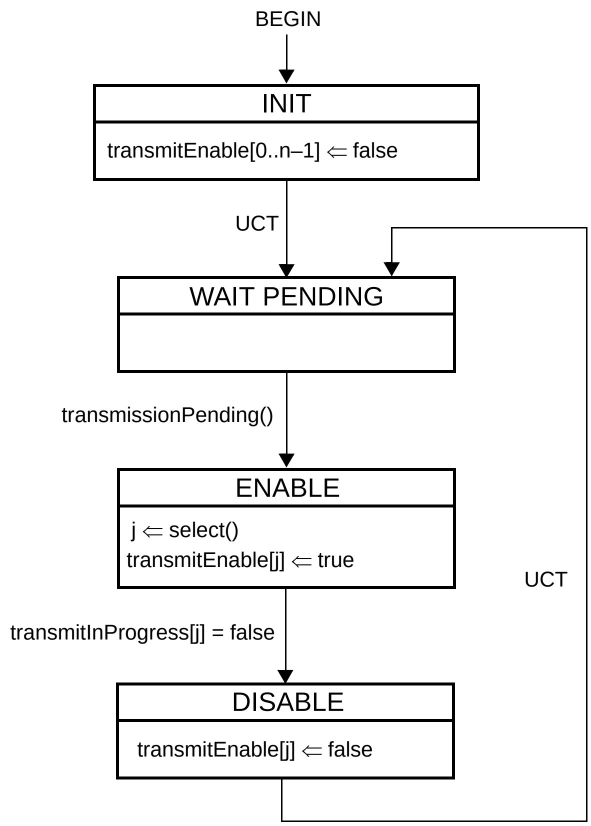
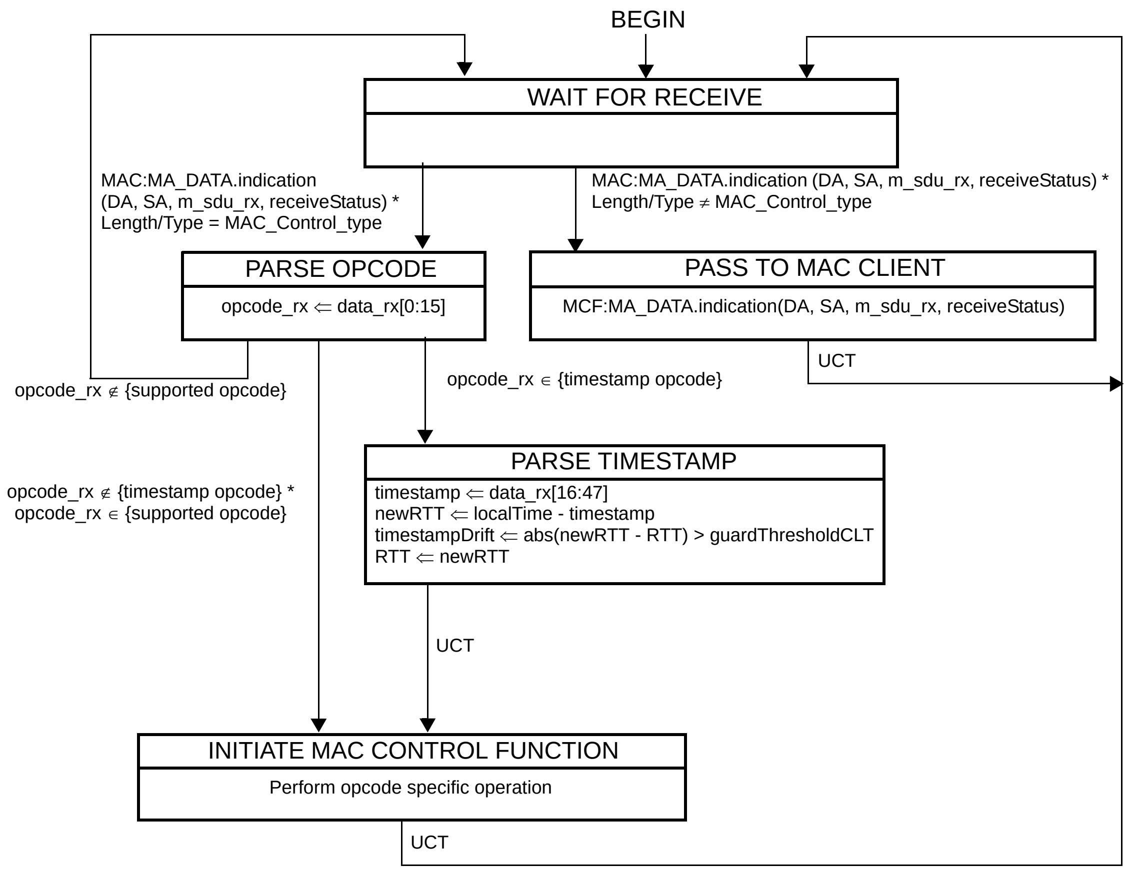


Figure 102–10—CLT Multipoint Transmission Control state diagram

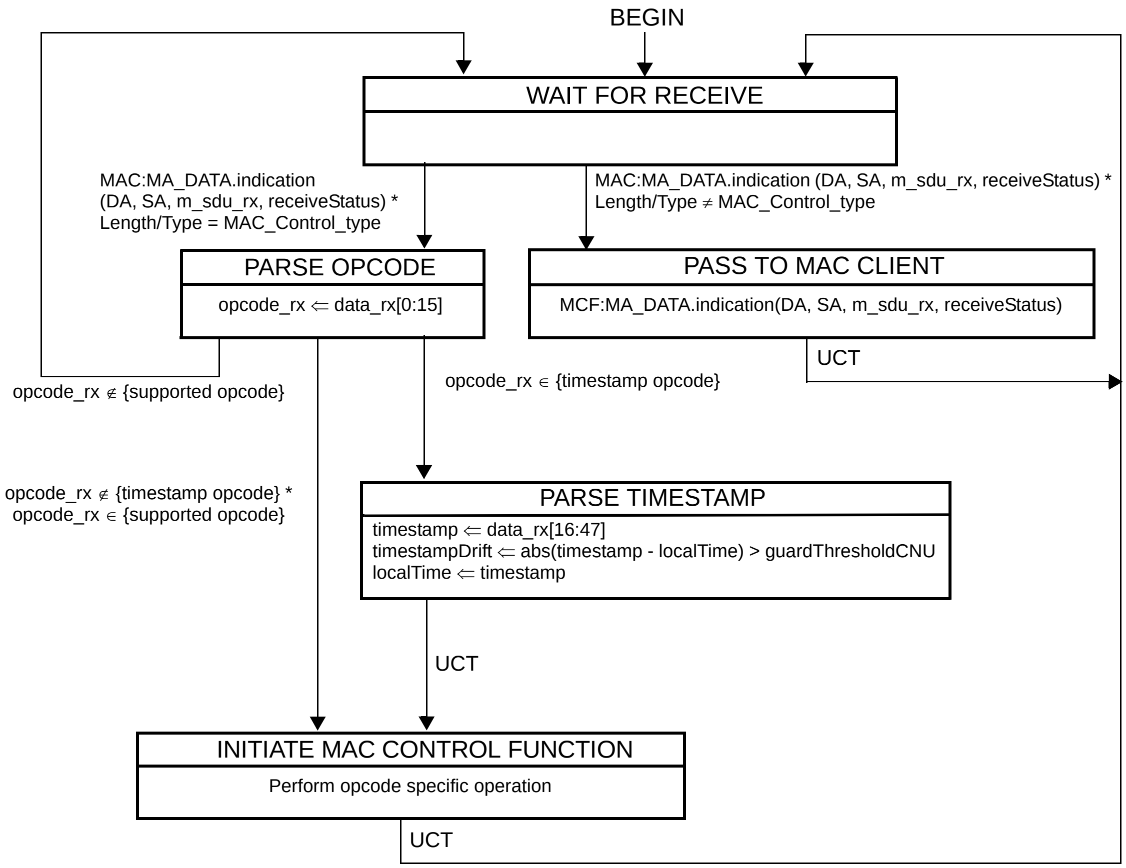


Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

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 102–11—CLT Control Parser state diagram



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 102–12—CNU Control Parser state diagram

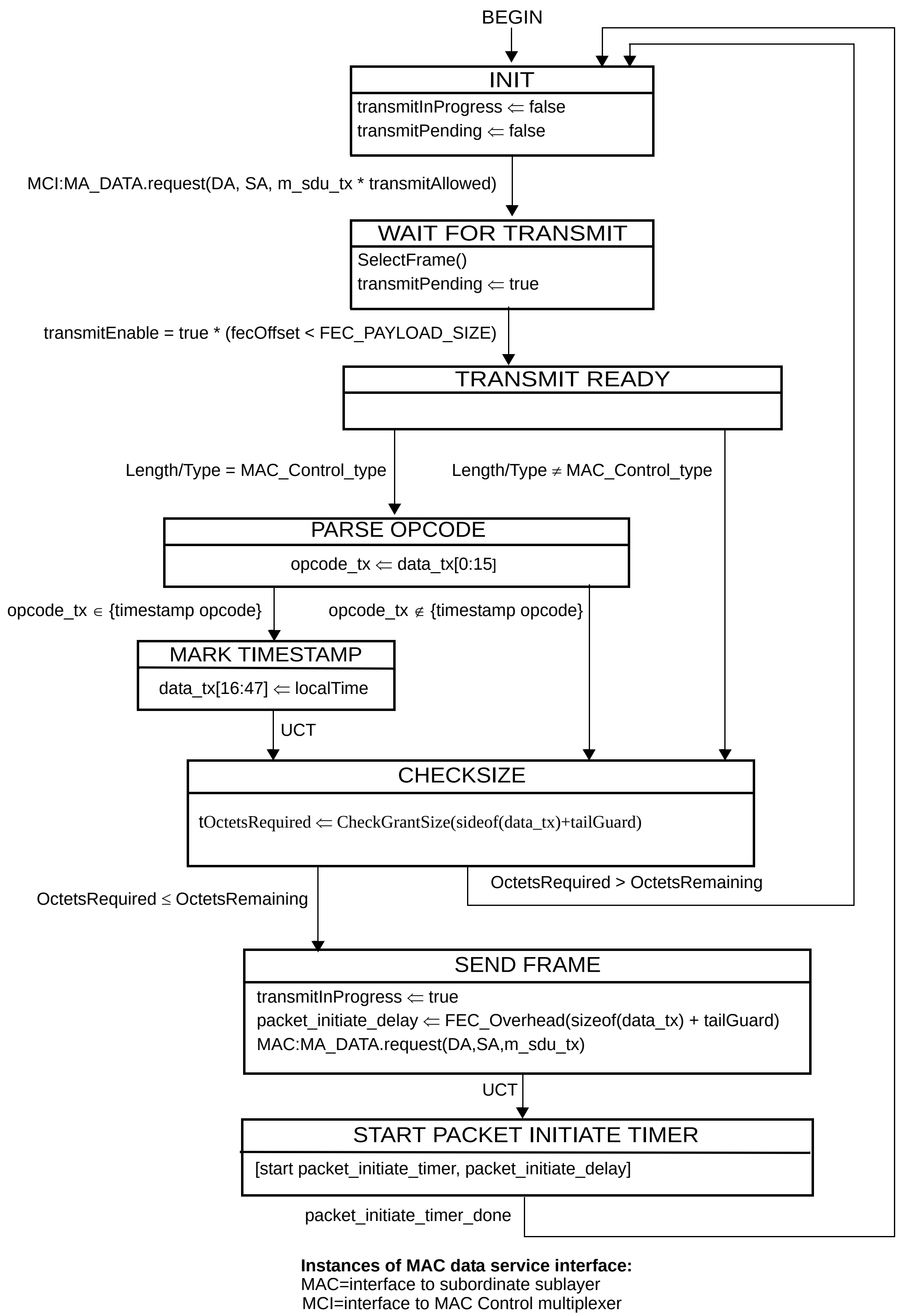


Figure 102–13—CLT Control Multiplexer state diagram

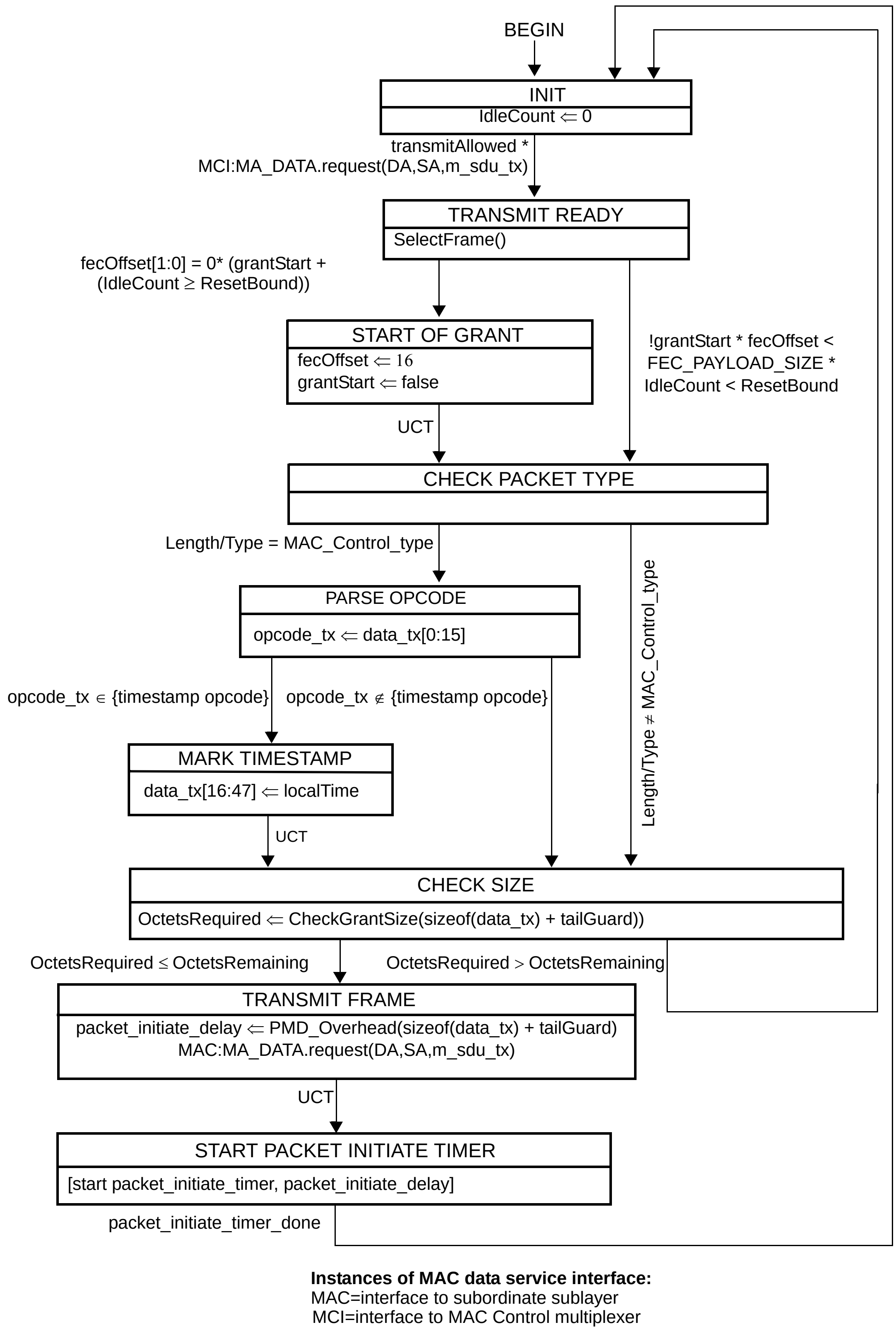


Figure 102–14—CNU Control Multiplexer state diagram

102.3 Multipoint Control Protocol (MPCP)

As depicted in ~~Figure 4–4~~Figure 102–4, the Multipoint MAC Control functional block comprises the following functions:

a) Discovery Processing. This block manages the discovery process, through which an ~~ONU~~ CNU 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~~ CNU to the ~~OLT~~CLT.

c) Gate Processing. This block manages the generation and collection of gate messages, through which multiplexing of multiple transmitters ~~is~~ and TDD mode control are achieved.

As depicted in ~~Figure 4–4~~Figure 102–4, 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.

102.3.1 Principles of Multipoint Control Protocol

Multipoint MAC Control enables a MAC Client to participate in a point-to-multipoint ~~optical~~ coax cable 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.

    c1) TDD mode operations is achieved by allowing downstream transmissions only during the DS Transmission window.

d) Multiple MACs operate on a shared medium by allowing only a single MAC to transmit upstream across the network at any given time ~~across the network using a time-division multiple access (TDMA) method~~and frequency.

e) Such gating of upstream transmission and of TDD downstream transmission ~~is~~ are 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 asymmetric, with the ~~OLT~~ CLT assuming the role of master, and the ~~ONU~~ CNU assuming the role of slave.

102.3.2 Compatibility considerations

102.3.2.1 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 ~~Annex 31B~~

Annex 31B supplement the constraints found at ~~102.3.2.4~~102.3.2.4.

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

102.3.2.2 Optional Shared LAN emulation

By combining P2PE, suitable filtering rules at the ~~ONU~~CNU, and suitable filtering and forwarding rules at the ~~OLT~~CLT, 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 76.2.6.1.3.2.~~

Specific behaviour of the filtering layer at the RS is specified in Y.2.6.1.3.2.

102.3.2.3 Multicast and single copy broadcast support

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

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

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

Configuration of SCB channels as well as filtering and marking of frames for support of SCB is defined in ~~76.2.6.1.3.2~~ Y.2.6.1.3.2 for ~~10G-EPON~~ EPOC compliant Reconciliation Sublayers~~.~~.

102.3.2.4 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 MAC.

The ~~OLT~~ CLT shall not grant less than 1024 time_quanta into the future, in order to allow the ~~ONU~~ CNU processing time when it receives a gate message. The ~~ONU~~ CNU shall process all messages in less than this period. The ~~OLT~~ CLT shall not issue more than one message every 1024 time_quanta to a single ~~ONU~~CNU. The unit of time_quantum is defined in ~~102.2.2.1~~102.2.2.1.

102.3.3 Discovery processing

Discovery is the process whereby newly connected or off-line ~~ONUs~~ CNUs are provided access to the ~~PON~~EPoC. The process is driven by the ~~OLT~~CLT, which periodically makes available Discovery Windows during which off-line ~~ONUs~~ CNUs are given the opportunity to make themselves known to the ~~OLT~~CLT. The periodicity of these windows is unspecified and left up to the implementor. The ~~OLT~~ CLT 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 ~~102.3.6.1~~102.3.6.1. With the appropriate settings of individual flags contained in this ~~16~~ 16 bit wide field, the ~~OLT~~ CLT notifies all the ~~ONUs~~ CNUs about its upstream and downstream channel transmission capabilities. Note that the ~~OLT~~ CLT may simultaneously support more than one data rate in the given transmission direction.

Off-line ~~ONUs~~CNUs, upon receiving a Discovery GATE MPCPDU, wait for the period to begin and then transmit a REGISTER_REQ MPCPDU to the ~~OLT~~CLT. Discovery windows are unique in that they are the only times when multiple ~~ONUs~~ CNUs can access the ~~PON~~ CCDN simultaneously, and transmission overlap can occur. In order to reduce transmission overlaps, a contention algorithm is used by all ~~ONUs~~CNUs. Measures are taken to reduce the probability for overlaps by artificially simulating a random distribution of distances from the ~~OLT~~CLT. Each ~~ONU~~ CNU waits a random amount of time before transmitting the REGISTER_REQ MPCPDU that is shorter than the length of the discovery window. It should be noted that multiple valid REGISTER_REQ MPCPDUs can be received by the ~~OLT~~ CLT during a single discovery window. Included in the REGISTER_REQ MPCPDU is the ~~ONU~~CNU's MAC address and number of maximum pending grants. Additionally, a registering ~~ONU~~ CNU notifies the ~~OLT~~ CLT of its transmission capabilities in the upstream and downstream channels by setting appropriately the flags in the Discovery Information field, as specified in ~~102.3.6.3~~102.3.6.3.

Note that even though a compliant ~~ONU~~ CNU 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 is 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~~ CNU notifies the ~~OLT~~ CLT of the ~~laser~~ RF on/off times, by setting appropriate values in the ~~Laser~~ RF On Time and ~~Laser~~ RF Off Time fields, where both values are expressed in the units of time_quanta.

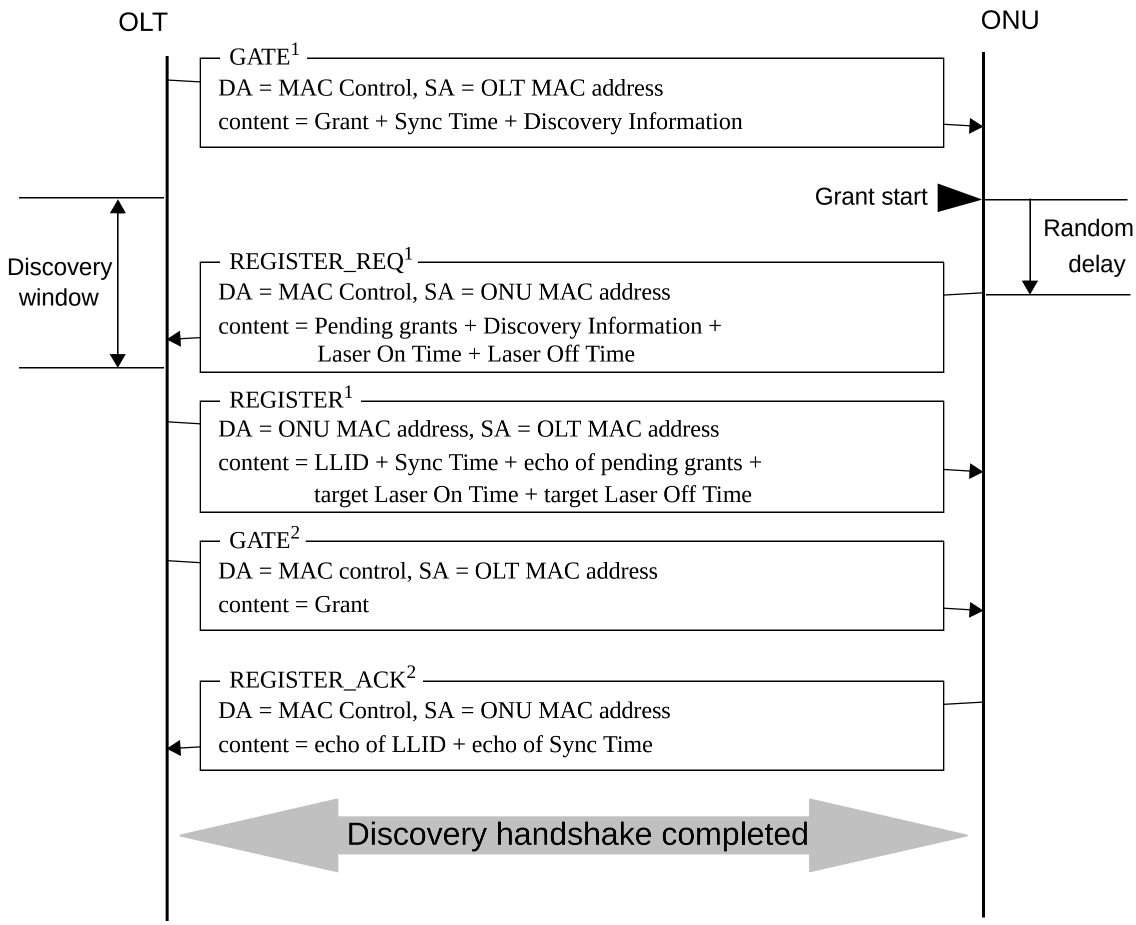
Upon receipt of a valid REGISTER_REQ MPCPDU, the ~~OLT~~ CLT registers the ~~ONU~~CNU, allocating and assigning a new port identity (LLID), and bonding a corresponding MAC to the LLID.

The next step in the process is for the ~~OLT~~ CLT to transmit a REGISTER MPCPDU to the newly discovered ~~ONU~~CNU, which contains the ~~ONU~~CNU's LLID, and the ~~OLT~~CLT's required synchronization time. Moreover, the ~~OLT~~ CLT echoes the maximum number of pending grants. The ~~OLT~~ CLT also sends the target value of ~~laser~~ RF on time and ~~laser~~ RF off time, which may be different than ~~laser~~ RF on time and ~~laser~~ RF off time delivered by the ~~ONU~~ CNU in the REGISTER_REQ MPCPDU.

The ~~OLT~~ CLT now has enough information to schedule the ~~ONU~~ CNU for access to the ~~PON~~ CCDN and transmits a standard GATE message allowing the ~~ONU~~ CNU to transmit a REGISTER_ACK. Upon receipt of the REGISTER_ACK, the discovery process for that ~~ONU~~ CNU is complete, the ~~ONU~~ CNU 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~~CNU. The discovery message exchange is illustrated in ~~Figure 4–15~~Figure 102–15.

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

-



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

Figure 4–15—Discovery handshake message exchange

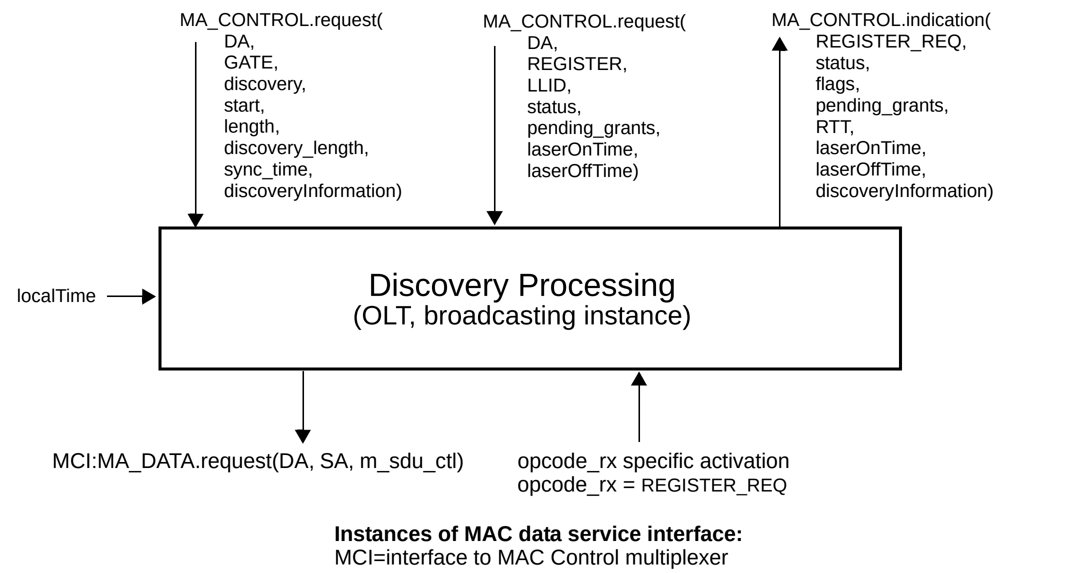


Figure 4–16—Discovery Processing service interfaces (OLT, broadcasting instance)

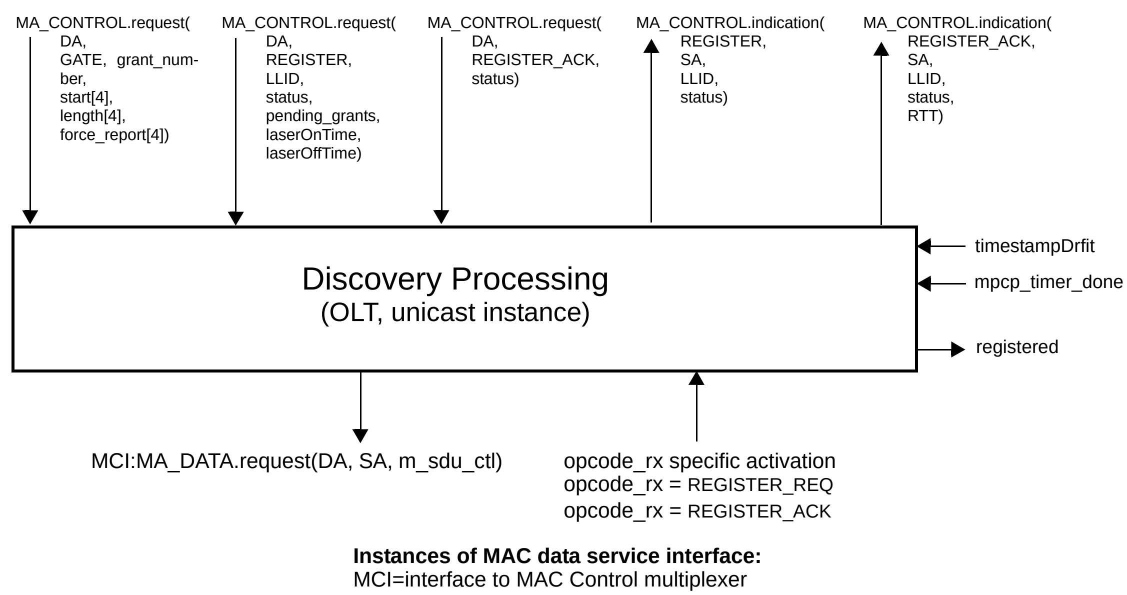


Figure 4–17—Discovery Processing service interfaces (OLT, unicasting instance)

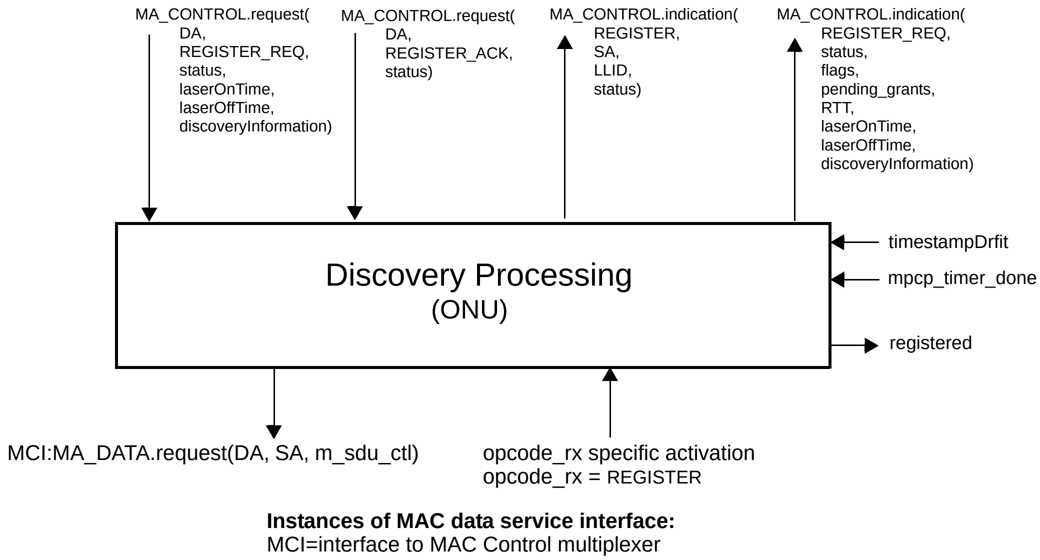
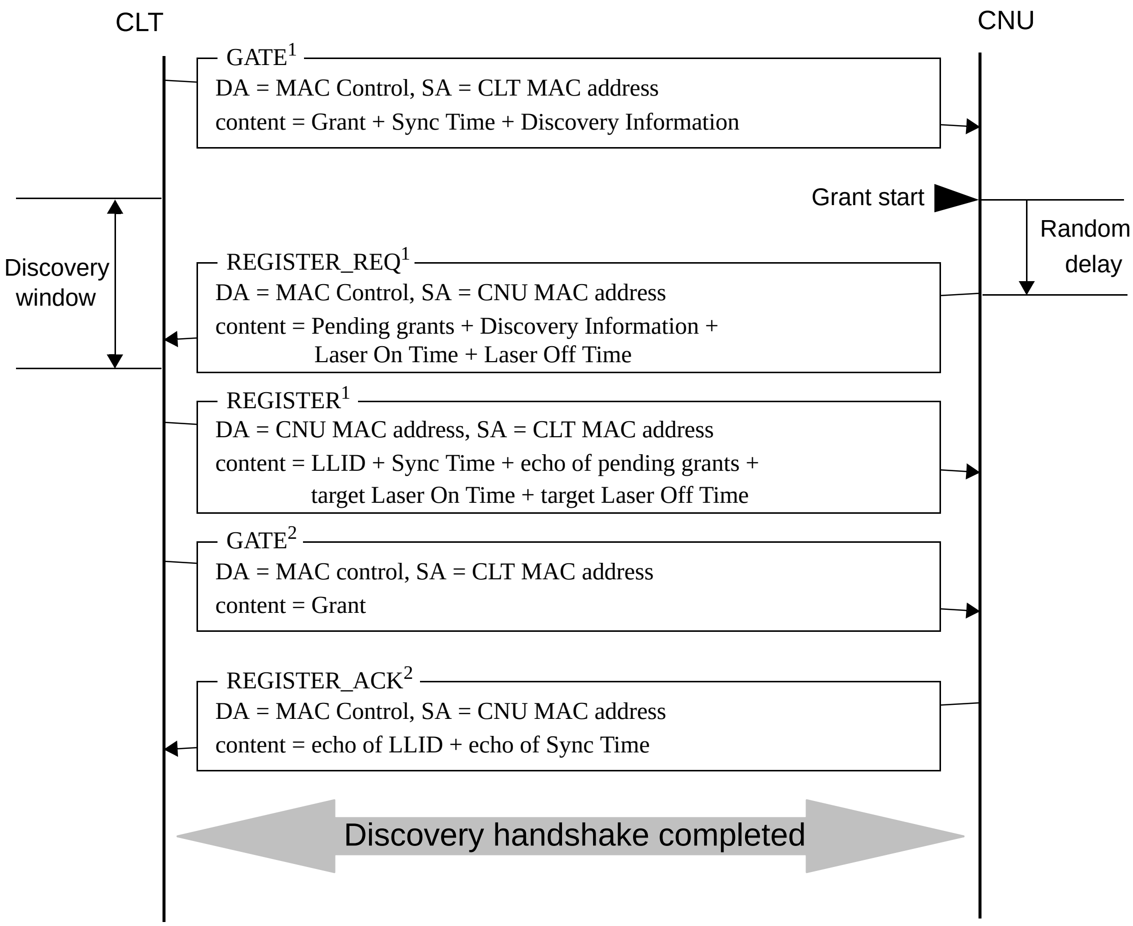


Figure 4–18—Discovery Processing service interfaces (ONU)



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

Figure 102–15—Discovery handshake message exchange

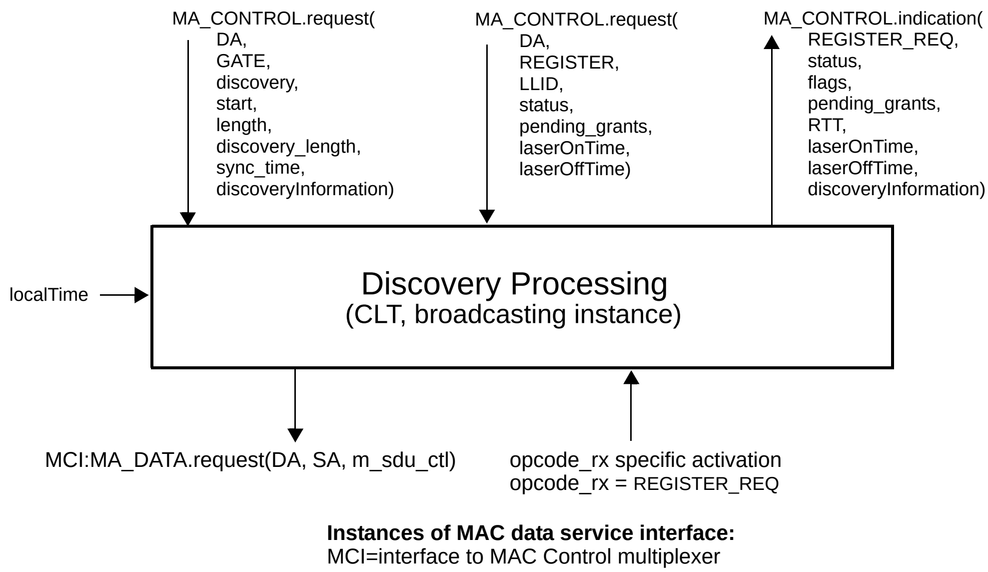


Figure 102–16—Discovery Processing service interfaces (CLT, unicasting instance)

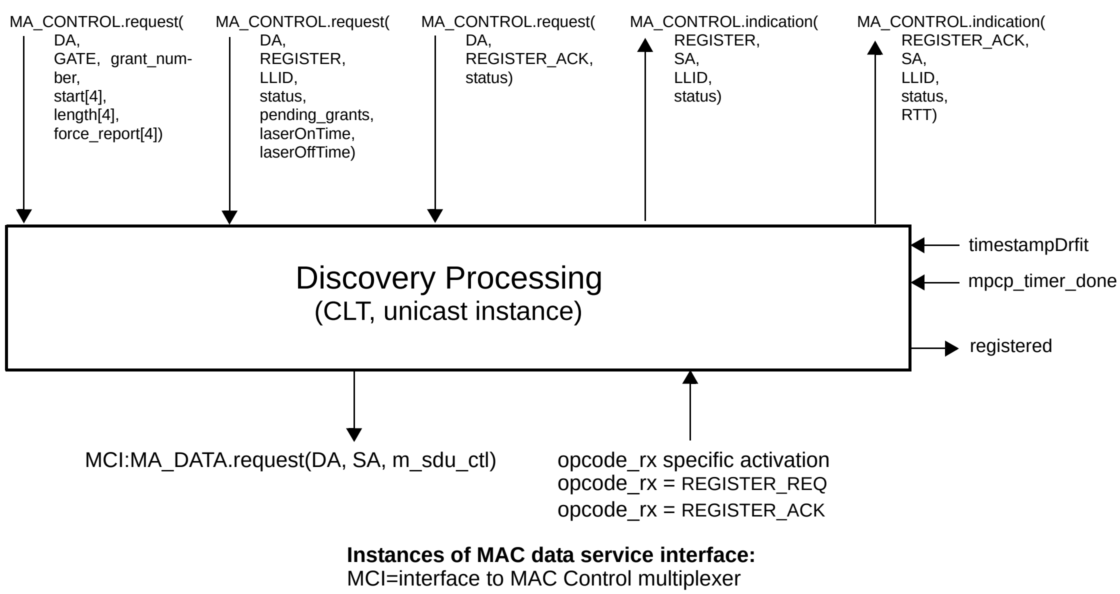


Figure 102–17—Discovery Processing service interfaces (CNU)

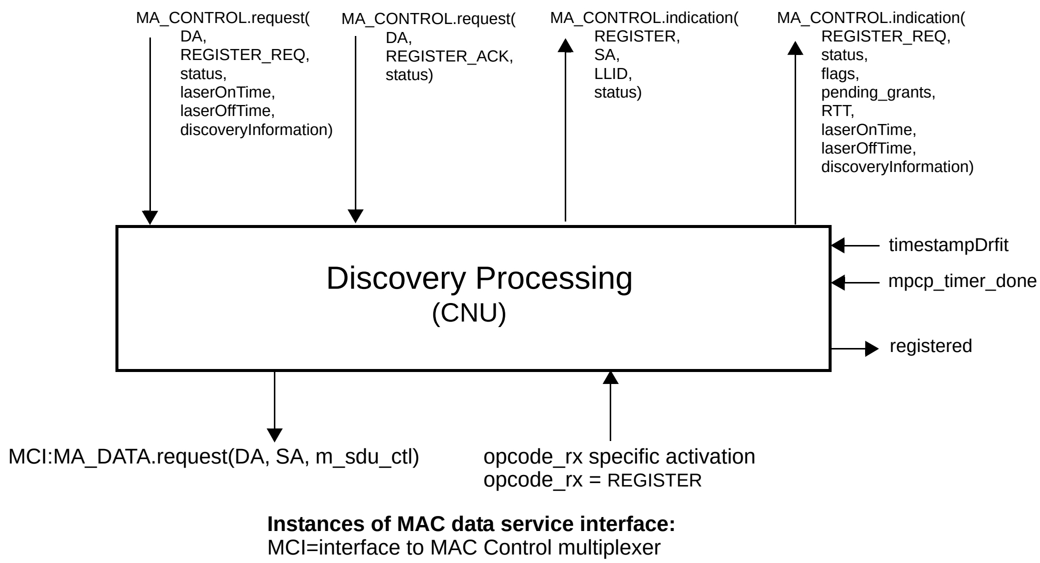


Figure 102–18—Discovery Processing service interfaces (CNU)

102.3.3.1 Constants

rfOffTimeCapability
TYPE: 8 bit unsigned
~~laserOffTimeCapability~~
~~TYPE: 8 bit unsigned~~
This constant represents the time required to terminate the ~~laser~~RF, in units of time_quantum. While the default value corresponds to a maximum allowed Toff (as specified in ~~Table 75–8~~Table 102–8 and ~~Table 75–9~~Table 102–9), implementations may set it to the actual value time period required for turning off the PMD, as specified in ~~75.7.14~~X.7.14.
~~VALUE: 0x20 (512 ns, default value)~~
VALUE: 0x20 (512 ns, default value)
EDITORS NOTE: the cross reference to X.7.14 cannot be resolved as the paragraph does not exist.

rfOnTimeCapability
TYPE: 8 bit unsigned
~~laserOnTimeCapability~~
~~TYPE: 8 bit unsigned~~
This constant represents the time required to initialize the ~~laser~~RF, in units of time_quantum. While the default value corresponds to a maximum allowed Ton (as specified in ~~Table 75–8~~Table 102–8 and ~~Table 75–9~~Table 102–9), implementations may set it to the actual value time period required for turning on the PMD, as specified in ~~75.7.14~~X.7.14.
~~VALUE: 0x20 (512 ns, default value)~~
VALUE: 0x20 (512 ns, default value)
EDITORS NOTE: the cross reference to X.7.14 cannot be resolved as the paragraph does not exist.

102.3.3.2 Variables

BEGIN

~~BEGIN~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

data_rx

~~data_rx~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

data_tx

~~data_tx~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

grantEndTime

TYPE: 32 bit unsigned

~~grantEndTime~~

~~TYPE: 32 bit unsigned~~

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

insideDiscoveryWindow

TYPE: Boolean

~~insideDiscoveryWindow~~

~~TYPE: Boolean~~

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.

rfOffTime

TYPE: 8 bit unsigned

~~laserOffTime~~

~~TYPE: 8 bit unsigned~~

This variable holds the time required to terminate the ~~laser~~RF. It counts in time_quanta units the time period required for turning off the PMD, as specified by the value of Toff in ~~75.7.14~~X.7.14.

~~VALUE: laserOffTimeCapability (default value)~~

VALUE: rfOffTimeCapability (default value)

EDITORS NOTE: the cross reference to X.7.14 cannot be resolved as the paragraph does not exist.

rfOnTime

TYPE: 8 bit unsigned

~~laserOnTime~~

~~TYPE: 8 bit unsigned~~

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 by the value of Ton in ~~75.7.14~~X.7.14.

~~VALUE: laserOnTimeCapability (default value)~~

VALUE: rfOnTimeCapability (default value)

EDITORS NOTE: the cross reference to X.7.14 cannot be resolved as the paragraph does not exist.

localTime

~~localTime~~

This variable is defined in ~~102.2.2.2~~102.2.2.2.

m_sdu_ctl
~~m_sdu_ctl~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

opcode_rx
~~opcode_rx~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

pendingGrants
TYPE: 16 bit unsigned
~~pendingGrants~~
~~TYPE: 16 bit unsigned~~
This variable holds the maximum number of pending grants that an ~~ONU~~ CNU is able to queue.

registered
TYPE: Boolean
~~registered~~
~~TYPE: Boolean~~
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.

syncTime
TYPE: 16 bit unsigned
~~syncTime~~
~~TYPE: 16 bit unsigned~~
This variable holds the time required to stabilize the receiver at the ~~OLT~~CLT. 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 (Treceiver_settling), clock synchronization interval (Tcdr), and ~~code-group~~ code?roup alignment interval (Tcode_group_align), as specified in ~~75.7.14~~X.7.14. The ~~OLT~~ CLT conveys the value of syncTime to ~~ONUs~~ CNUs in Discovery GATE and REGISTER messages. During the synchronization time a ~~10/1G-EPON ONU transmits only IDLE patterns, and a 10/10G-EPON ONU~~ CNU sends synchronization pattern (SP, see ~~76.3.2.5.2~~Y.3.2.5.2) followed by burst delimiter pattern (BURST_DELIMITER, see ~~76.3.2.5.2~~Y.3.2.5.2).

EDITORS NOTE: the cross reference to X.7.14 cannot be resolved as the paragraph does not exist.

timestampDrift
~~timestampDrift~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

102.3.3.3 Functions

None.

102.3.3.4 Timers

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

102.3.3.5 Messages

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

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

This service primitive is defined in 2.3.2.

MA_DATA.request (DA, SA, m_sdu)

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

This service primitive is defined in 2.3.2.

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

~~The~~ This service primitive is used by the MAC Control client at the ~~OLT~~ CLT 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.~~
GATE: Opcode for GATE MPCPDU as defined in Table 31A?. 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~~sync_time:The time interval required to stabilize the receiver at the CLT.
discoveryInformation: This parameter represents the Discovery Information field in GATE MPCPDU as specified in ~~102.3.6.1~~102.3.6.1, defining the speed(s) the ~~OLT~~ CLT is capable of receiving and speed(s) at which the discovery window is opened for.

MA_CONTROL.request(DA, GATE, grant_number, start[4], length[4], force_report[4])
~~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~~ CLT to issue the GATE message to an ~~ONU~~CNU and to issue local grants for downstream transmission in TDD mode. This primitive takes the following parameters:

DA~~:~~: Multicast MAC Control address as defined in ~~Annex 31B.~~Annex 31B.
GATE: Opcode for GATE MPCPDU as defined in ~~Table 31A–1.~~Table 31A?.
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,rfOnTime,rfOffTime, discoveryInformation)
~~MA_CONTROL.request(DA, REGISTER_REQ, status, laserOnTime, laserOffTime, discoveryInformation)~~

~~The~~ This service primitive is used by a client at the ~~ONU~~ CNU 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.~~Annex 31B.

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

status: This parameter takes on the indication supplied by the flags field in the REGISTER_REQ MPCPDU as defined in ~~Table 4–5~~Table 102–5.

~~laserOnTime~~rfOnTime:This parameter holds the ~~laserOnTime~~ rfOnTime value, expressed in units of time_quanta, as reported by MAC client and specified in ~~102.3.6.3~~102.3.6.3.

~~laserOffTime~~rfOffTime:This parameter holds the ~~laserOffTime~~ rfOffTime value, expressed in units of time_quanta, as reported by MAC client and specified in ~~102.3.6.3~~102.3.6.3.

discoveryInformation~~:~~ :This parameter represents the Discovery Information field, as specified in ~~102.3.6.3~~102.3.6.3, defining the speed(s) the ~~ONU~~ CNU is capable of transmitting and speed(s) at which the registration attempt is made.

MA_CONTROL.indication(REGISTER_REQ, status, flags, pending_grants, RTT, ~~laserOnTime~~rfOnTime, ~~laserOffTime~~rfOffTime, discoveryInformation)
~~The service primitive is 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:~~
~~MA_CONTROL.indication(REGISTER_REQ, status, flags, pending_grants, RTT, laserOnTime, laserOffTime, discoveryInformation)~~

The service primitive is 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~~Table 31A?.

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

pending_grants~~:~~ : This parameter 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~~CLT.

RTT~~:~~ : The measured round trip time to/from the ~~ONU~~ CNU 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~~CLT.

~~laserOnTime:~~ rfOnTime:This parameter holds the contents of the ~~laserOnTime~~ rfOnTime 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~~CLT.

~~laserOffTime:~~ rfOffTime: This parameter holds the contents of the ~~laserOffTime~~ rfOffTime 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~~CLT.

discoveryInformation~~:~~: This parameter holds the contents of the Discovery Information field in the REGISTER_REQ MPCPDU. This parameter holds a valid value only when the primitive is generated by the Discovery process in the ~~OLT~~CLT.

MA_CONTROL.request(DA, REGISTER, LLID, status, pending_grants, rfOnTime, rfOffTime)
~~MA_CONTROL.request(DA, REGISTER, LLID, status, pending_grants, laserOnTime, laserOffTime)~~

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

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

REGISTER~~:~~: Opcode for REGISTER MPCPDU as defined in ~~Table 31A–1.~~Table 31A?.

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 4–7.~~Table 102–7.

pending_grants~~:~~: This parameters echoes back the pending_grants field that was previously received in the REGISTER_REQ message.

~~laserOnTime:~~ rfOnTime: This parameter carries the target value of ~~Laser~~ RF On Time for the given ~~ONU~~ CNU transmitter. This value may be different than the ~~laserOnTime~~ rfOnTime value carried in the REGISTER_REQ MPCPDU received from the corresponding ~~ONU~~ CNU MAC during Discovery stage.

laserOffTime~~:~~: This parameter carries the target value of ~~Laser~~ RF Off Time for the given ~~ONU~~ CNU transmitter. This value may be different than the ~~laserOffTime~~ rfOffTime value carried in the REGISTER_REQ MPCPDU received from the corresponding ~~ONU~~ CNU MAC during Discovery stage.

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

This service primitive is issued by the Discovery Process at the ~~OLT~~ CLT or an ~~ONU~~ CNU 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.~~Table 31A?.

SA~~:~~: This parameter represents the MAC address of the ~~OLT.~~ CLT.

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)
~~MA_CONTROL.request(DA, REGISTER_ACK, status)~~

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

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

REGISTER_ACK~~:~~: Opcode for REGISTER_ACK MPCPDU as defined in ~~Table 31A–1~~Table 31A?.

status~~:~~: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in ~~Table 4–8~~Table 102–8.

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

This service primitive is issued by the Discovery Process at the ~~OLT~~ CLT 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~~Table 31A?.

SA~~:~~: This parameter represents the MAC address of the reciprocating device (~~ONU~~ CNU address at the ~~OLT~~CLT, and ~~OLT~~ CLT address at the ~~ONU~~CNU).

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~~ CNU is returned in this parameter. RTT is stated in time_quanta units. This parameter holds a valid value only when the invoking Discovery Process in the ~~OLT~~CLT.

Opcode-specific function(opcode)
~~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.

102.3.3.6 State Diagrams

The Discovery Process in the ~~OLT~~ CLT shall implement the discovery window setup state diagram shown in ~~Figure 4–19~~Table 102–19, request processing state diagram as shown in ~~Figure 4–20~~Table 102–20, register processing state diagram as shown in ~~Figure 4–21~~Table 102–21, and final registration state diagram as shown in ~~Figure 4–22~~Table 102–22. The discovery process in the ~~ONU~~ CNU shall implement the registration state diagram as shown in ~~Figure 4–23~~Table 102–23.

Instantiation of state diagrams as described in ~~Figure 4–19~~Table 102–19, ~~Figure 4–20~~Table 102–20, and ~~Figure 4–21~~Table 102–21 is performed only at the Multipoint MAC Control instances attached to the broadcast LLID (0x7FFE). Instantiation of state diagrams as described in ~~Figure 4–22~~Table 102–22 and ~~Figure 4–23~~Table 102–23 is performed for every Multipoint MAC Control instance, except the instance attached to the broadcast channel.

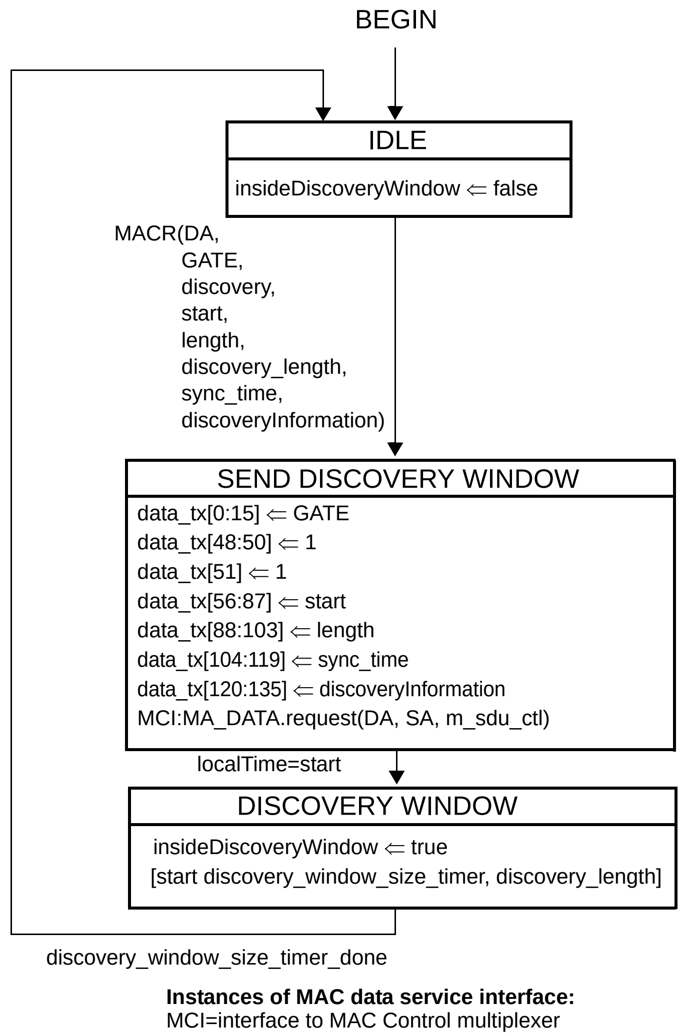


Figure 4–19—Discovery Processing OLT Window Setup state diagram

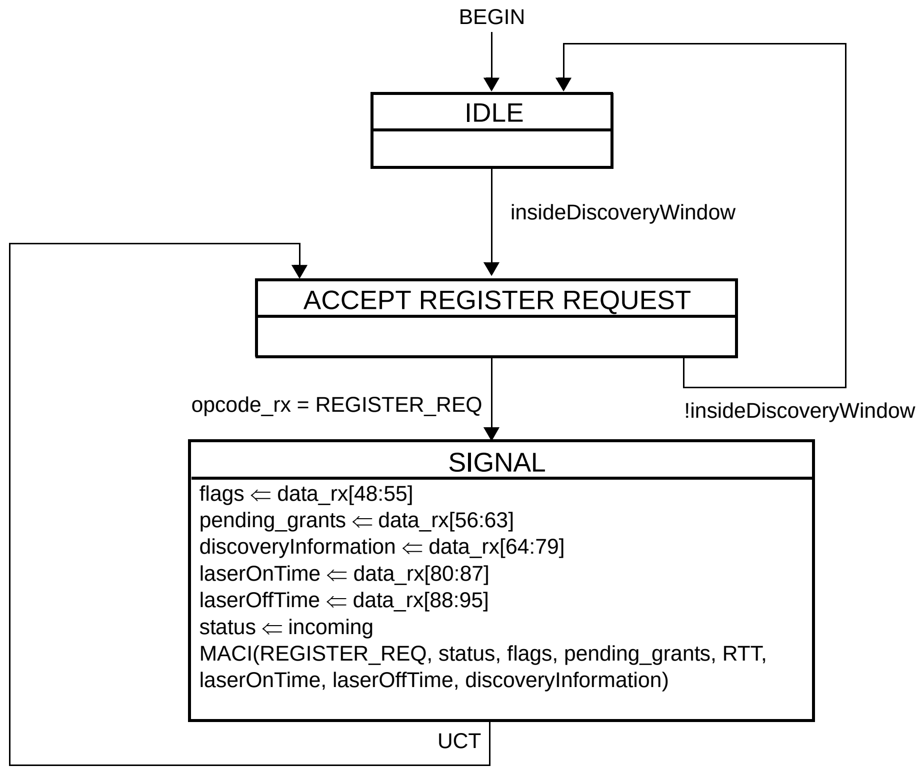
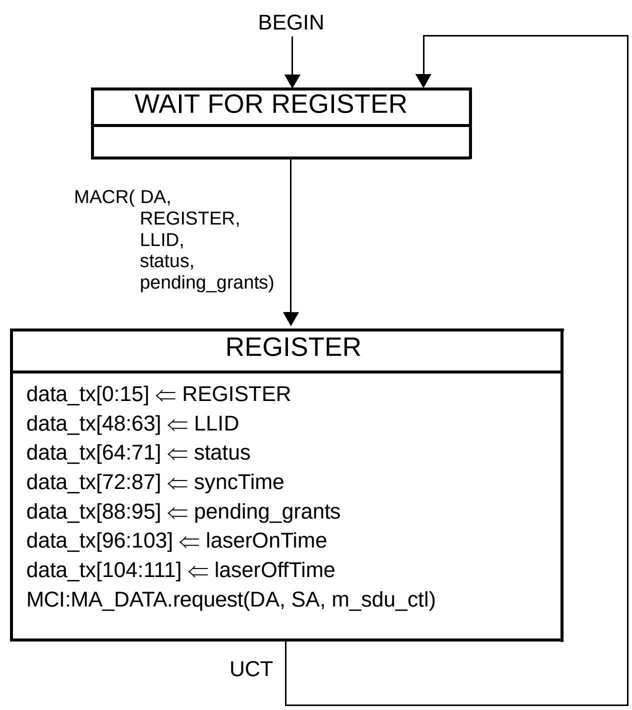
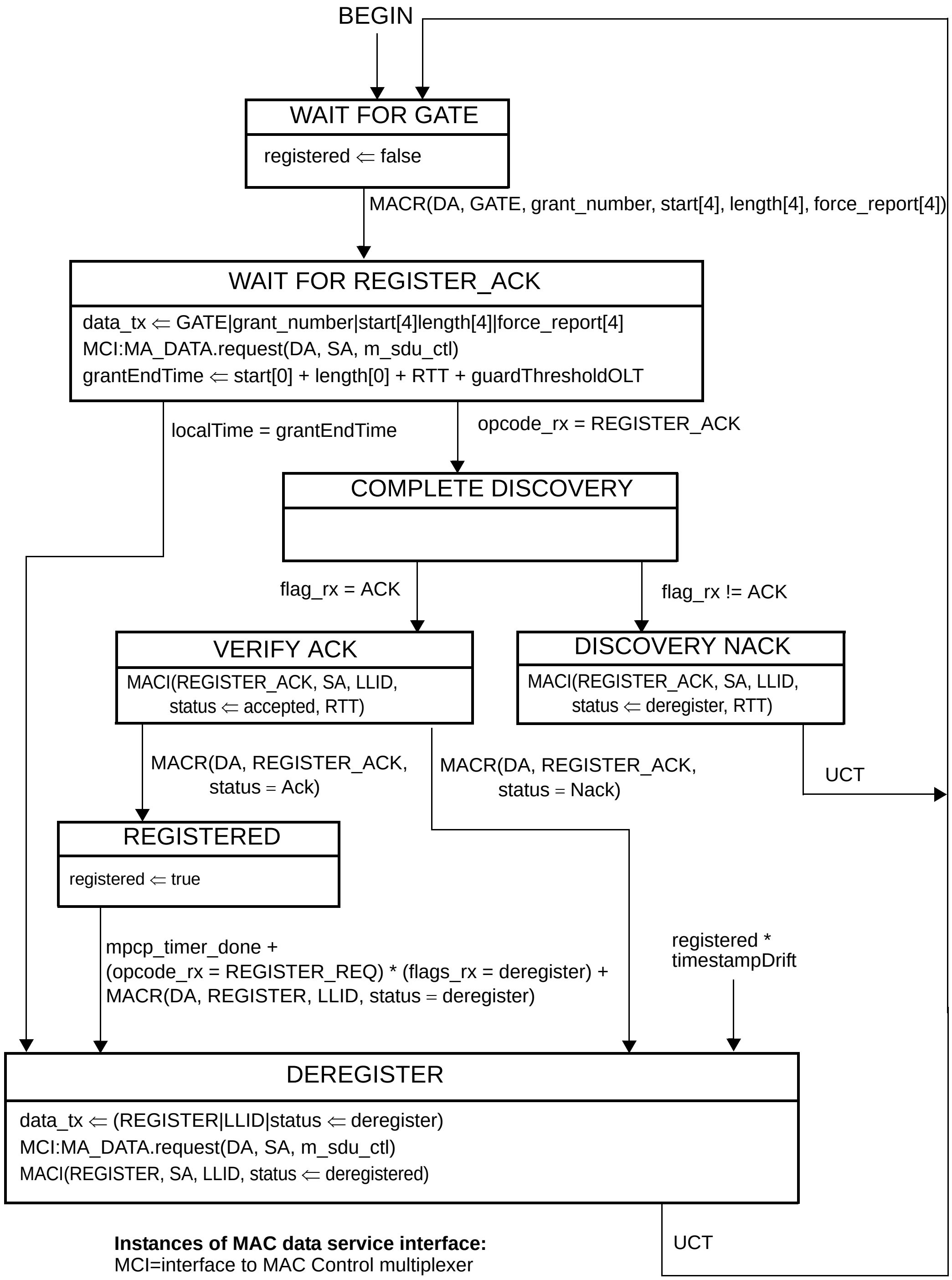


Figure 4–20—Discovery Processing OLT Process Requests state diagram



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

Figure 4–21—Discovery Processing OLT Register state diagram



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

Figure 4–22—Discovery Processing OLT Final Registration state diagram

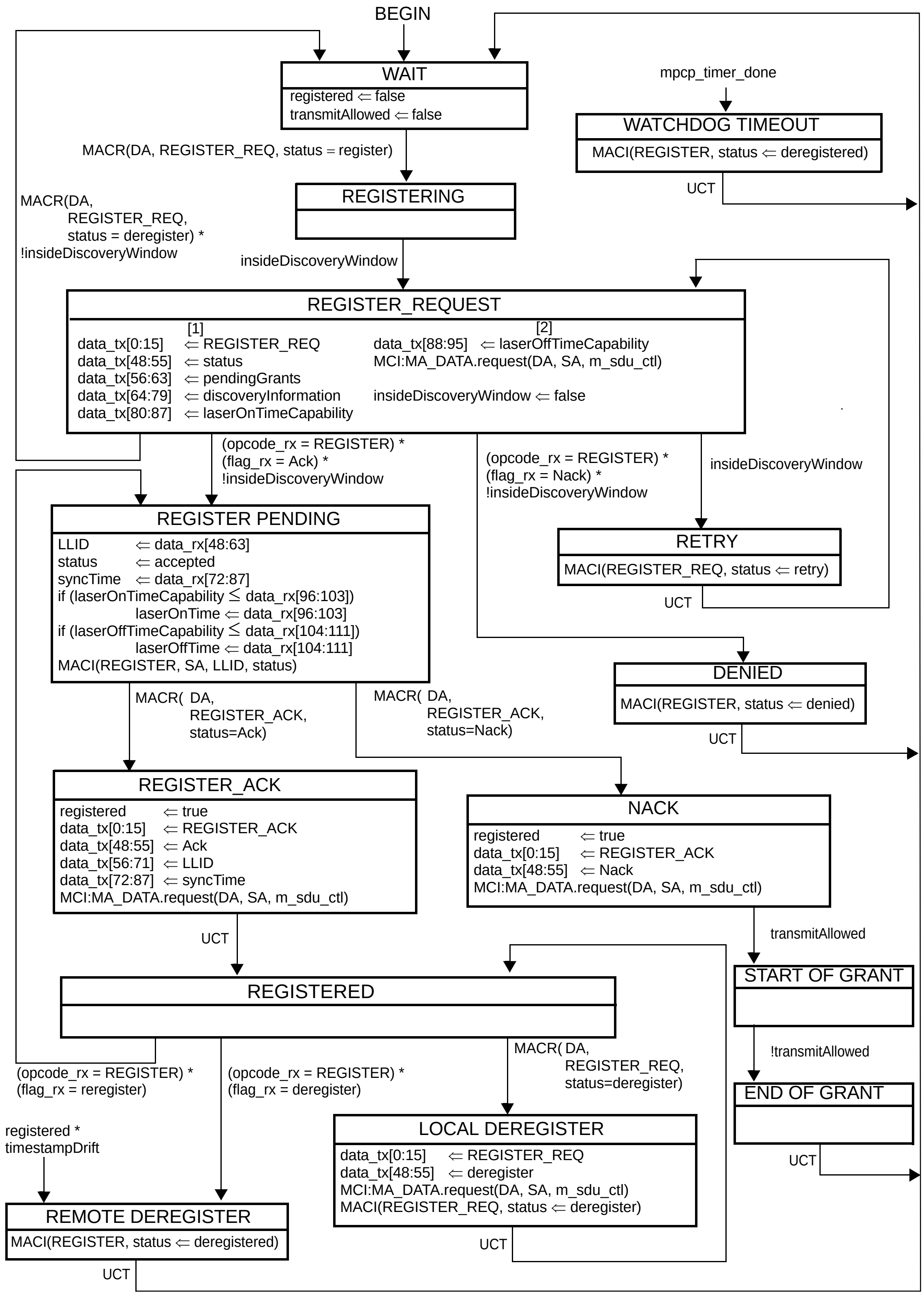


Figure 4–23—Discovery Processing ONU Registration state diagram

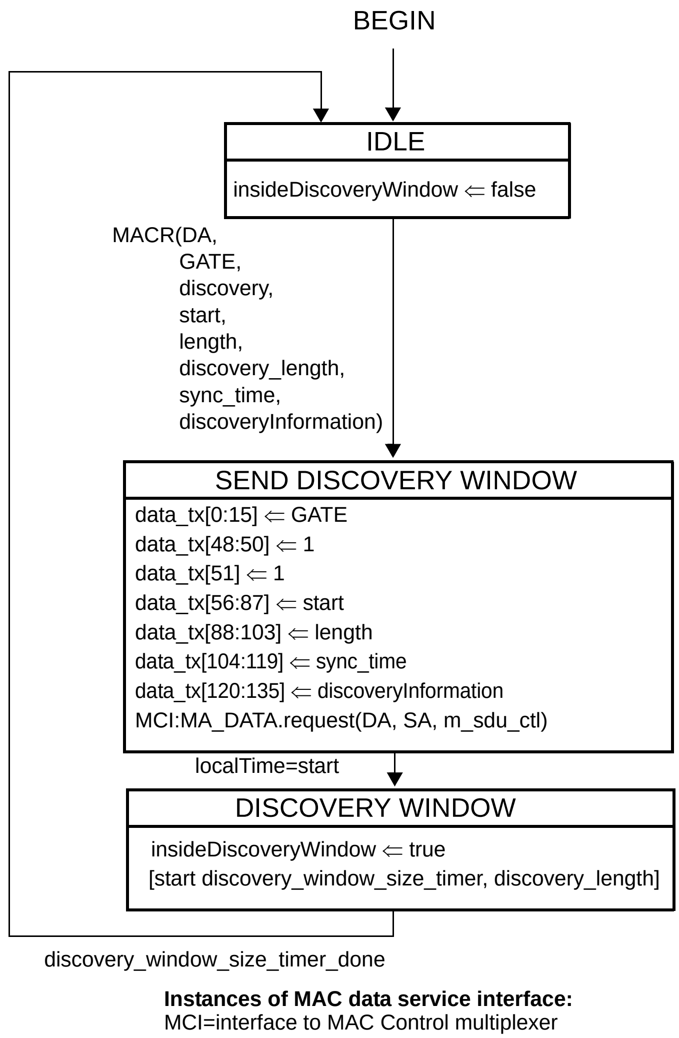


Figure 102–19—Discovery Processing CLT Window Setup state diagram

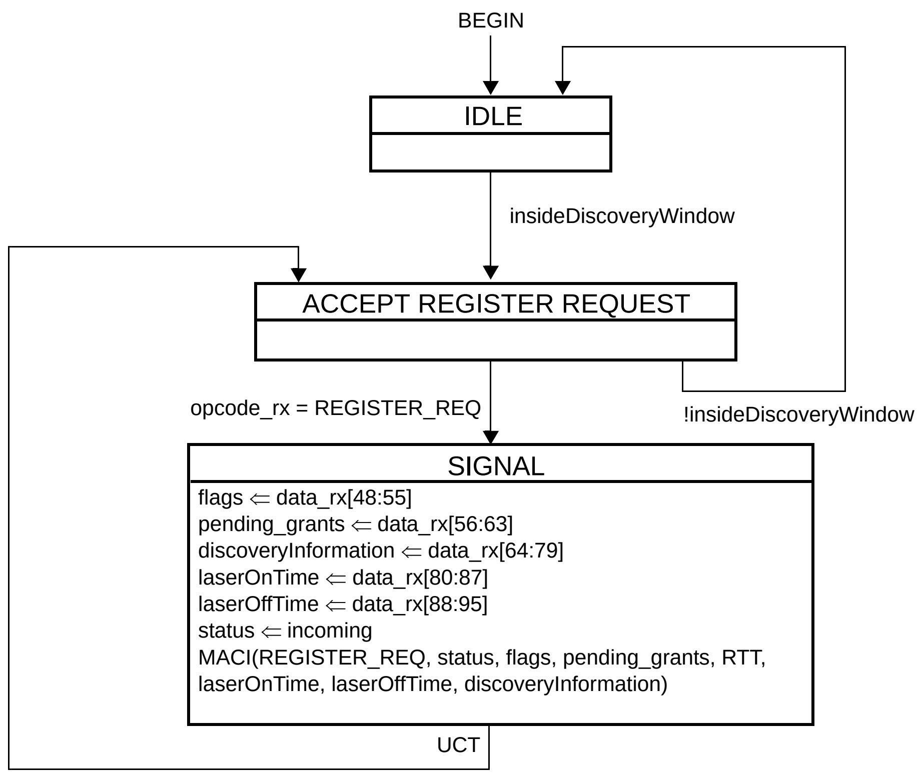
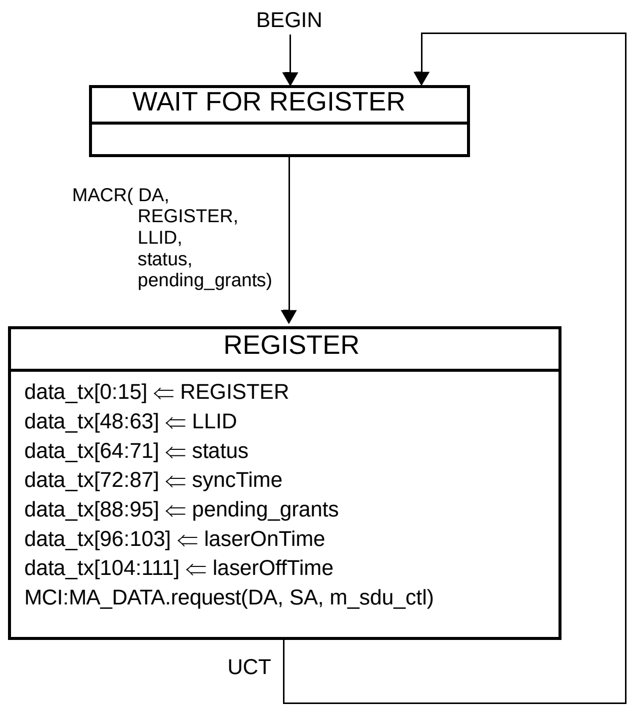
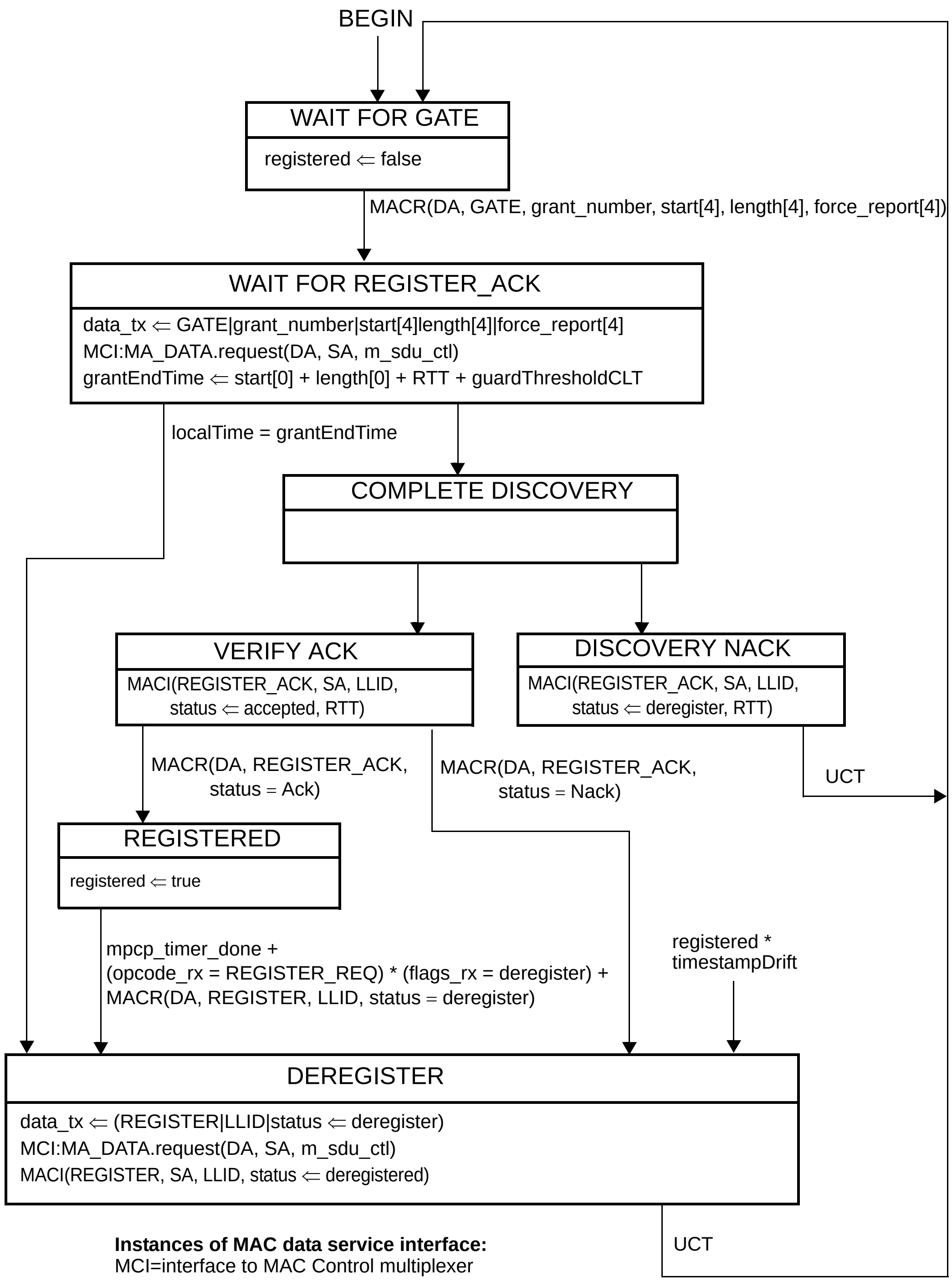


Figure 102–20—Discovery Processing CLT Process Requests state diagram



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

Figure 102–21—Discovery Processing CLT Register state diagram



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

Figure 102–22—Discovery Processing CLT Final Registration state diagram

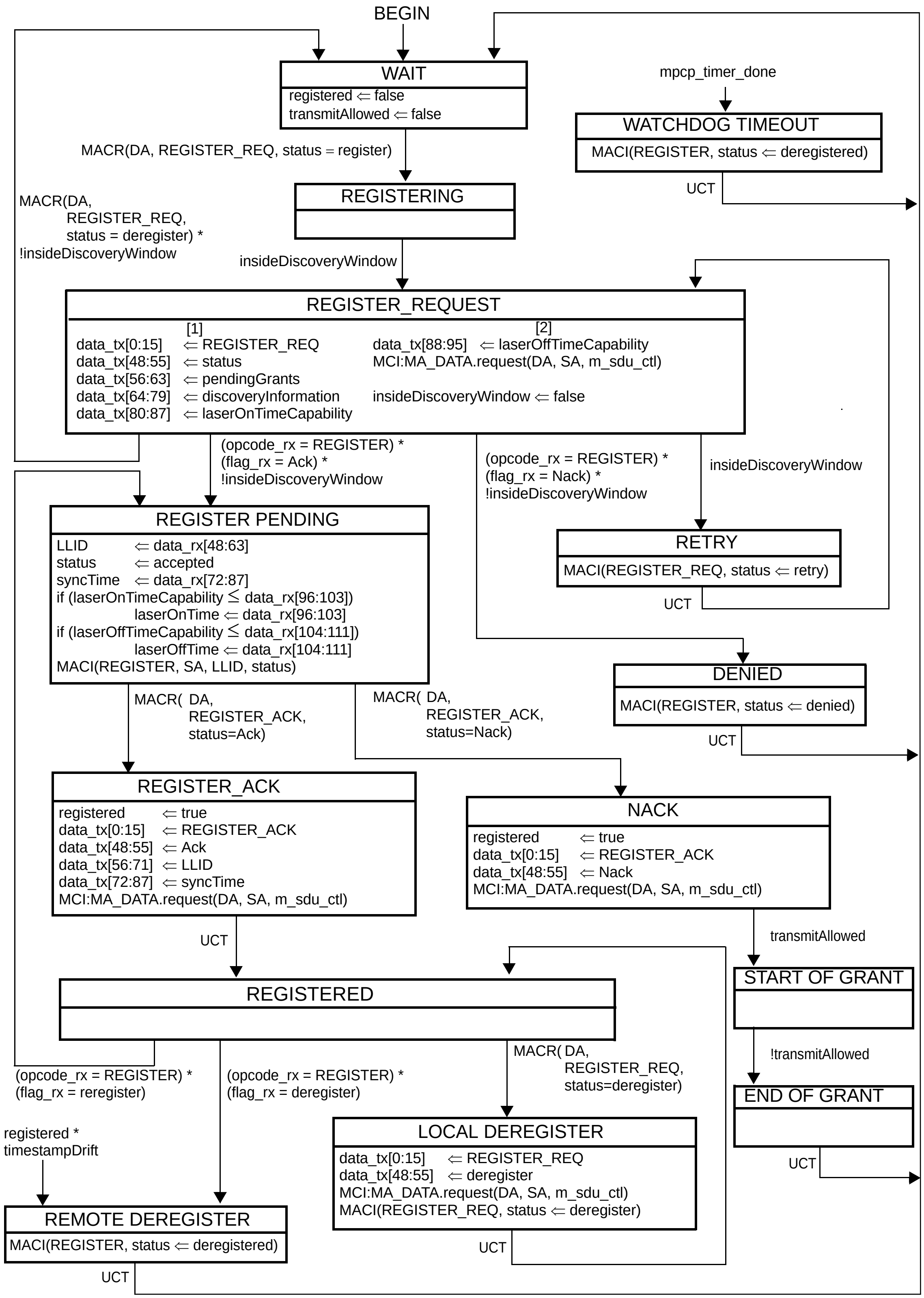


Figure 102–23—Discovery Processing CNU Registration state diagram

102.3.4 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~~ CLT watchdog timer.

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

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

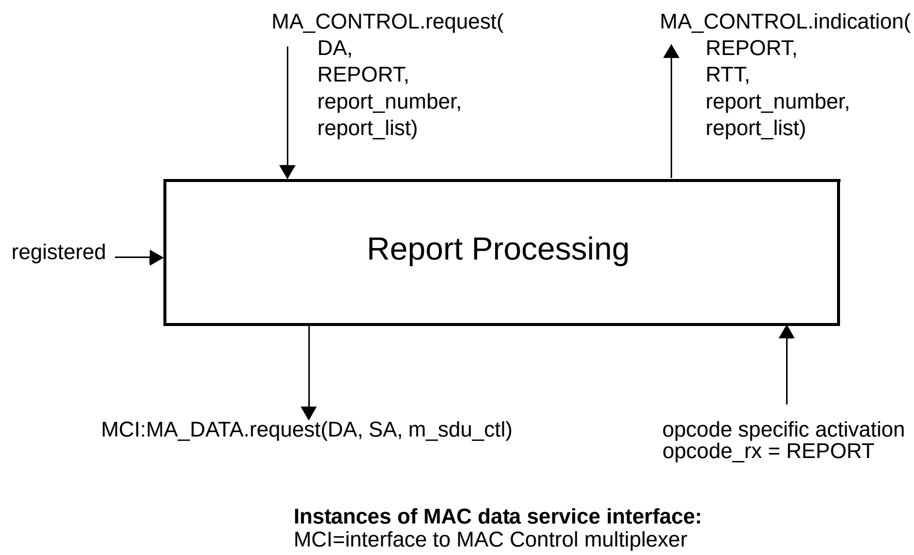
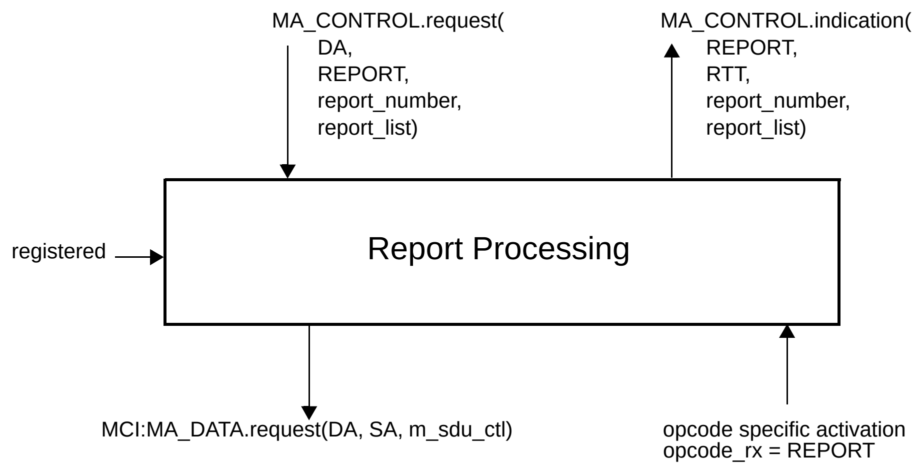


Figure 4–24—Report Processing service interfaces



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

Figure 102–24—Report Processing service interfaces

102.3.4.1 Constants

None.

102.3.4.2 Variables

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

data_rx
~~data_rx~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

data_tx
~~data_tx~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

m_sdu_ctl
~~m_sdu_ctl~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

mpcp_timeout
TYPE: 32 bit unsigned
~~mpcp_timeout~~
~~TYPE: 32 bit unsigned~~
This variable represents the maximum allowed interval of time between two MPCPDU mes-

sages. Failure to receive at least one frame within this interval is considered a fatal fault and leads to deregistration. This variable is expressed in units of time_quanta.
~~VALUE: 0x03B9ACA0 (1 s, default value)~~
VALUE: 0x03B9ACA0 (1 s, default value)

opcode_rx
~~opcode_rx~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

registered
~~registered~~
This variable is defined in ~~102.3.3.2~~102.3.3.2.

report_timeout
TYPE: 32 bit unsigned
~~report_timeout~~
~~TYPE: 32 bit unsigned~~
This variable represents the maximum allowed interval of time between two REPORT messages generated by the ~~ONU~~CNU, expressed in units of time_quanta.
~~VALUE: 0x002FAF08 (50 ms, default value)~~.
VALUE: 0x002FAF08 (50 ms, default value)

102.3.4.3 Functions

None.

102.3.4.4 Timers

report_periodic_timer
~~report_periodic_timer~~
~~ONUs~~ CNUs 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~~CNU.

mpcp_timer
~~mpcp_timer~~
This timer is defined in ~~102.3.3.4~~102.3.3.4.

102.3.4.5 Messages

MA_DATA.request (DA, SA, m_sdu)
~~The service primitive is defined in 2.3.2.~~
The service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, REPORT, report_number, report_list)
~~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~~ CNU 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.~~Annex 31B.
REPORT~~:~~:	Opcode for REPORT MPCPDU as defined in ~~Table 31A–1.~~Table 31A?.

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 of length of 8. The index of an element of this array reflects the numbered priority queue in the IEEE 802.1P nomenclature. An element with the value of '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 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)
~~MA_CONTROL.indication(REPORT, RTT, report_number, report_list)~~

The service primitive is issued by the Report Process at the ~~OLT~~ CLT 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~~ time?arying aspect of the network. This primitive uses the following parameters:

REPORT~~:~~ : Opcode for REPORT MPCPDU as defined in ~~Table 31A–1.~~Table 31A?.

RTT~~:~~ : This parameter holds an updated round trip time value that 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 of length of 8. The index of an element of this array reflects the numbered priority queue in the IEEE 802.1P nomenclature. An element with the value of '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 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)
~~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.

102.3.4.6 State diagrams

The report process in the ~~OLT~~ CLT shall implement the report processing state diagram as shown in ~~Figure 4–25~~Figure 102–25. The report process in the ~~ONU~~ CNU shall implement the report processing state diagram as shown in ~~Figure 4–26~~Figure 102–26. Instantiation of state diagrams as described is performed for Multipoint MAC Control instances attached to unicast LLIDs only.

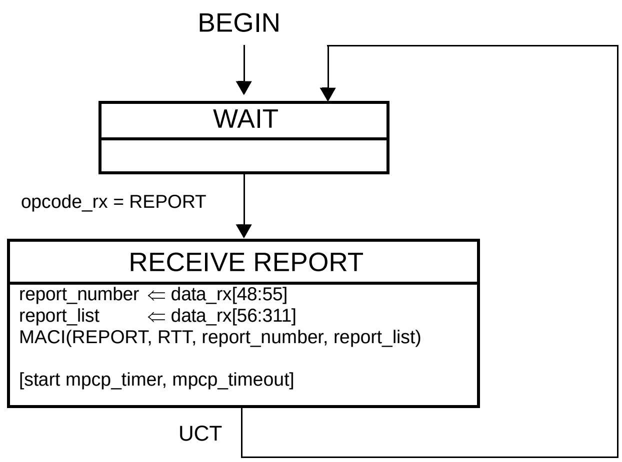


Figure 4–25—Report Processing state diagram at OLT

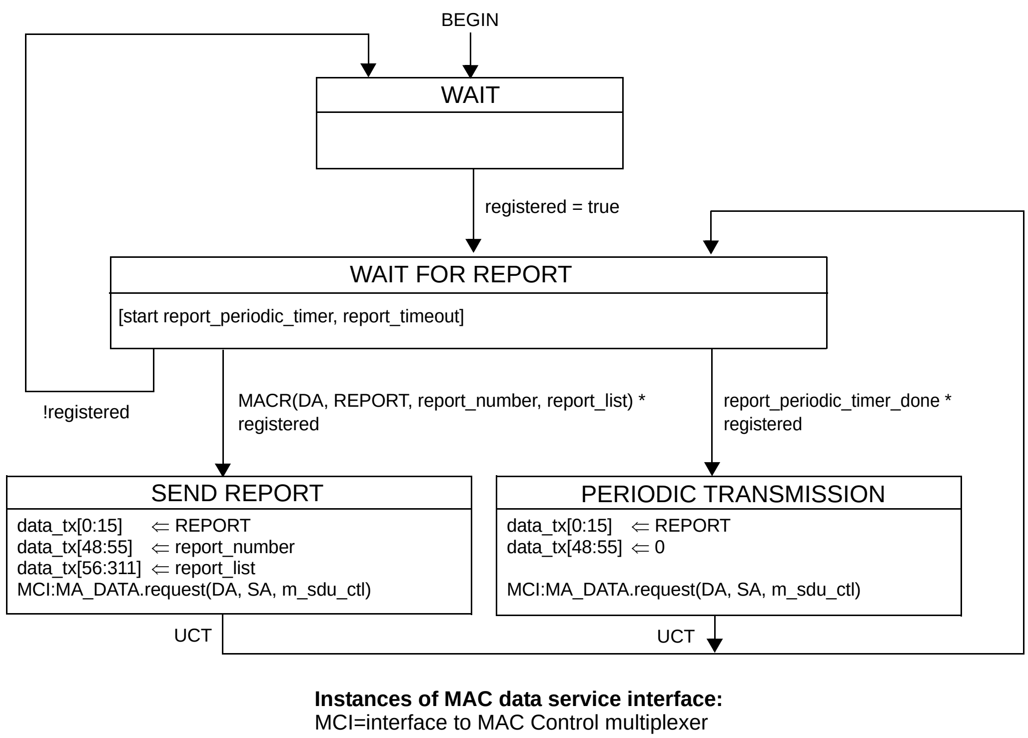


Figure 4–26—Report Processing state diagram at ONU

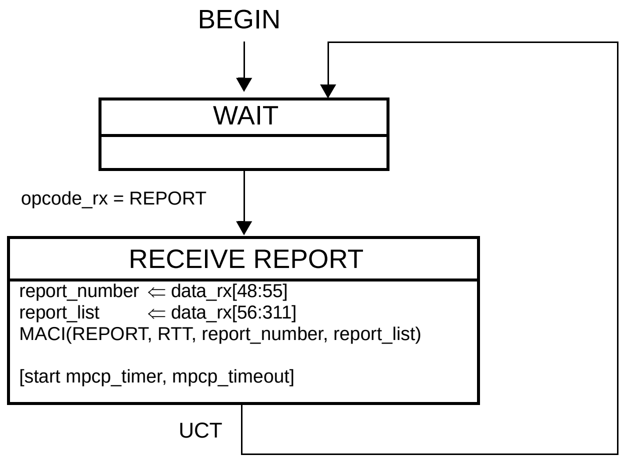


Figure 102–25—Report Processing state diagram at CLT

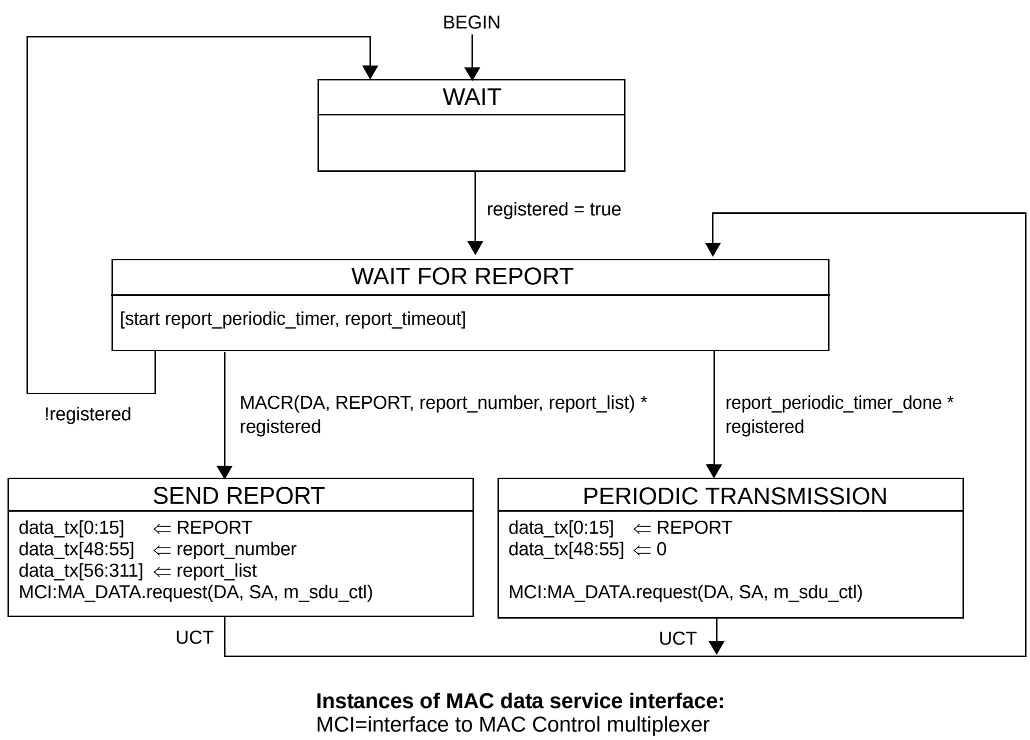


Figure 102–26—Report Processing state diagram at CNU

102.3.5 Gate Processing

A key concept pervasive in Multipoint MAC Control is the ability to arbitrate a single transmitter out of a plurality of ~~ONUs~~CNUs. The ~~OLT~~ CLT controls an ~~ONU~~CNU's transmission by the assigning of grants. In addition for TDD mode, the CLT controls the TDD downstream transmission by the assigning of local grants.

The transmitting window of ~~an ONU~~ a CNU is indicated in the GATE message where start time and length are ~~specified~~specified and the DA field differs from the local address of the CLT. ~~An ONU~~ A CNU begins transmission when its localTime counter matches the start_time value indicated in the GATE message. ~~An ONU~~ A CNU concludes its transmission with sufficient margin to ensure that the ~~laser~~ RF transmitter is turned off before the grant length interval has elapsed.

Multiple outstanding grants may be issued to each ~~ONU~~CNU. The ~~OLT~~ CLT shall not issue more than the maximum supported maximum outstanding grants as advertised by the ~~ONU~~ CNU during registration (see pending grants in ~~102.3.6.3~~102.3.6.3).

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

When registered, the ~~ONU~~ CNU ignores all gate messages where the Discovery flag is set.

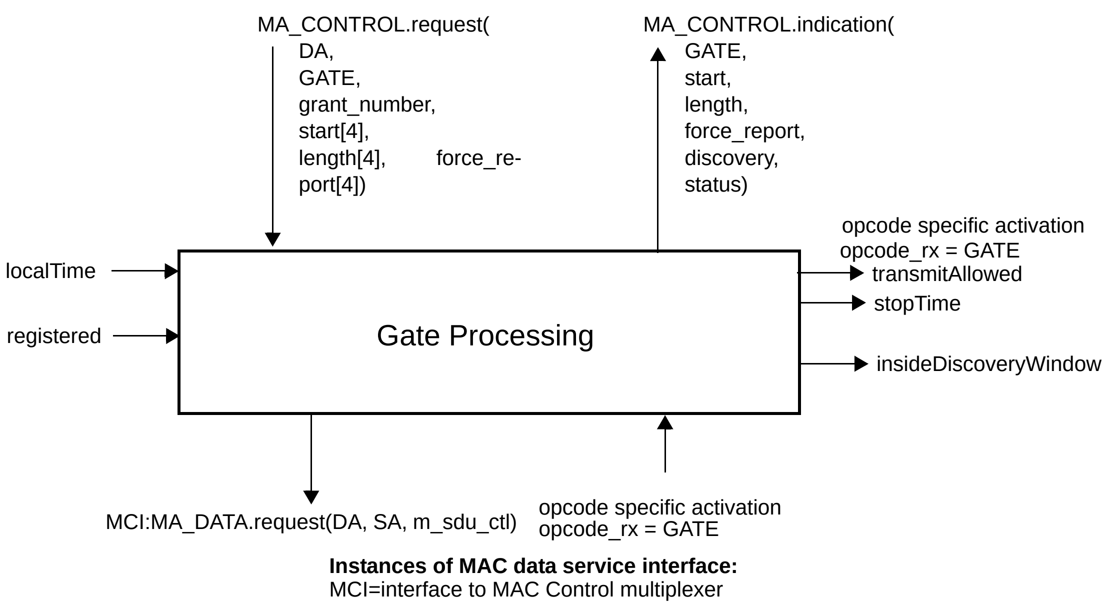


Figure 4–27—Gate Processing service interface

The DS transmission window for TDD mode is indicated in the MAC Request for local grant at CLT side where start time and length are specified and the local grant flag is set. In TDD mode, the CLT begins transmission when its localTime counter matches the start_time value indicated in the local grant. The CLT concludes its transmission with sufficient margin to ensure that the RF transmitter is turned off before the local grant length interval has elapsed.

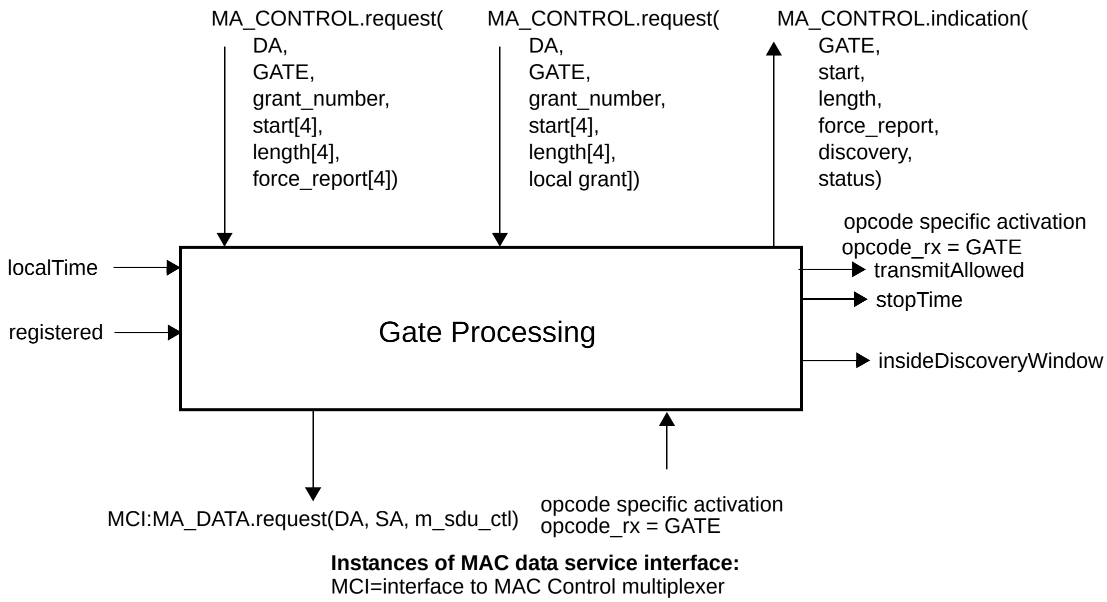


Figure 102–27—Gate Processing service interface

102.3.5.1 Constants

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

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

minGrantLength
TYPE: 32 bit unsigned
~~minGrantLength~~
~~TYPE: 32 bit unsigned~~
This constant represents the minimum data portion of a grant. The minGrantLength is equal to one FEC codeword (see FEC_CODEWORD_SIZE in ~~102.2.2.1~~102.2.2.1), less the initial 16 idle octets, expressed in units of time_quanta. The minimum grant length accepted by an ~~ONU~~ CNU is equal to minGrantLength + BurstOverhead (see ~~102.3.5.2~~102.3.5.2).
~~VALUE: 12~~
VALUE: 12

tqSize

~~tqSize~~

This constant is defined in ~~102.2.2.1~~102.2.2.1.

102.3.5.2 Variables

BEGIN

TYPE: Boolean

~~BEGIN~~

~~TYPE: Boolean~~

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

BurstOverhead

TYPE: integer

~~BurstOverhead~~

~~TYPE: integer~~

This variable represents the burst overhead and equals the sum of ~~laserOnTime~~rfOnTime, ~~laserOffTime~~rfOffTime, syncTime and an additional two time_quanta to account for END_BURST_DELIMITER and two leading IDLE vectors of the payload. This variable is expressed in units of time_quanta.

counter

TYPE: integer

~~counter~~

~~TYPE: integer~~

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

currentGrant

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

}

~~currentGrant~~

~~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~~

~~}~~

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~~.

The state is a structure field composed of multiple subfields.

data_rx
~~data_rx~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

data_tx
~~data_tx~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

effectiveLength
TYPE: 32 bit unsigned
~~effectiveLength~~
~~TYPE: 32 bit unsigned~~
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~~ RF on and off, and waiting for receiver lock.

gate_timeout
TYPE: 32 bit unsigned
~~gate_timeout~~
~~TYPE: 32 bit unsigned~~
This variable represents the maximum allowed interval of time between two GATE messages generated by the ~~OLT~~ CLT to the same ~~ONU~~CNU, expressed in units of time_quanta~~.~~
~~VALUE: 0x002FAF08 (50 ms, default value)~~.
VALUE: 0x002FAF08 (50 ms, default value)

grantList
TYPE: list of elements having the structure define in currentGrant
~~grantList~~
~~TYPE: list of elements having the structure define in currentGrant~~
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.

grantStart
~~grantStart~~
This variable is defined in ~~102.2.2.3~~102.2.2.3.

insideDiscoveryWindow
~~insideDiscoveryWindow~~
This variable is defined in ~~102.3.3.2~~102.3.3.2.

localGrantList
TYPE: list of elements having the structure define in currentGrant for the local grant
This variable is used for storage of the list of pending local grants used in TDD mode. It is dynamically set by the Gate Processing functional block and is not exposed. Each time a local 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.

maxDelay
TYPE: 16 bit unsigned

~~maxDelay~~

~~TYPE: 16 bit unsigned~~

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

m_sdu_ctl

~~m_sdu_ctl~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

nextGrant

TYPE: element having same structure as defined in currentGrant

~~nextGrant~~

~~TYPE: element having same structure as defined in 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 content of the variable is the next grant to become active.

nextStopTime

TYPE: 32 bit unsigned

~~nextStopTime~~

~~TYPE: 32 bit unsigned~~

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

opcode_rx

~~opcode_rx~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

registered

~~registered~~

This variable is defined in ~~102.3.3.2~~102.3.3.2.

stopTime

~~stopTime~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

syncTime

~~syncTime~~

This variable is defined in ~~102.3.3.2~~102.3.3.2.

transmitAllowed

~~transmitAllowed~~

This variable is defined in ~~102.2.2.3~~102.2.2.3.

102.3.5.3 Functions

empty(list)

~~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)
~~confirmDiscovery(data)~~

This function is used to check whether the current Discovery Window is open for the given ~~ONU~~ CNU (TRUE) or not (FALSE). This function returns values as shown in ~~Table 4–1.~~ Table 102–1.

Table 4–1—Operation of the confirmDiscovery(data) function

OLT Discovery Information: Discovery Window		ONU Tx capability		confirmDiscovery(data) returns
1G	10G	1G	10G	
X	1	0	1	TRUE
1	X	1	0	TRUE
0	1	1	0	FALSE
1	0	0	1	FALSE
0	0	X	X	FALSE[a]

[a]These particular values for the Discovery Window fields should not be normally generated by the OLT.

Table 102–1—Operation of the confirmDiscovery(data) function

CLT Discovery Information: Discovery Window		NU Tx capability		confirmDiscovery(data) returns
1G	10G	1G	10G	
X	1	0	1	TRUE
1	X	1	0	TRUE
1	1	1	0	FALSE
1	0	0	1	FALSE
0	0	X	X	FALSE[a]

[a]These particular values for the Discovery Window fields should not be normally generated by the CLT

InsertInOrder(sorted_list, inserted_element)
~~InsertInOrder(sorted_list, inserted_element)~~

This function is used to queue an element inside a sorted list. The queuing 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 maximum size equals the value advertised during registration as maximum number of pending grants.

IsBroadcast(grant)
~~IsBroadcast(grant)~~

This function is used to check whether its argument represents a broadcast grant, i.e., grant given to multiple ~~ONUs~~CNUs. 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~~Annex 31B, and false otherwise.

PeekHead(sorted_list)
~~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)
~~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)
~~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.

102.3.5.4 Timers

gntWinTmr
~~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
~~gate_periodic_timer~~
The ~~OLT~~ CLT 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~~CLT.

mpcp_timer
~~mpcp_timer~~
This timer is defined in ~~102.3.3.4.~~ 102.3.3.4.

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

102.3.5.5 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])
~~MA_DATA.request (DA, SA, m_sdu)~~

~~The~~ This service primitive is defined in ~~2.3.2~~102.3.3.5.

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

~~This service primitive is defined in 102.3.3.5.~~

This service primitive is used by the MAC Control client at the CLT to issue the GATE message to an CNU and to issue local grants for downstream transmission in TDD mode. This primitive takes the following parameters:

DA:	Multicast MAC Control address as defined in Annex 31B.
GATE:	Opcode for GATEMPCPDU 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.
local_grant:	Flag specifying that the given GATE message is to be used for local grant only

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

This service primitive issued by the Gate Process at the ~~ONU~~ CNU 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~~Table 31A?.
start~~:~~:	start time of the grant. This parameter is not present when the parameter status value is equal to deactive.
length~~:~~:	Length of the grant. This parameter is not present when the parameter status value is equal to deactive.
force_report~~:~~:	Flags indicating whether a REPORT message should be transmitted in this grant. This parameter is not present when the parameter status value is equal to 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 parameter status value is equal to 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.

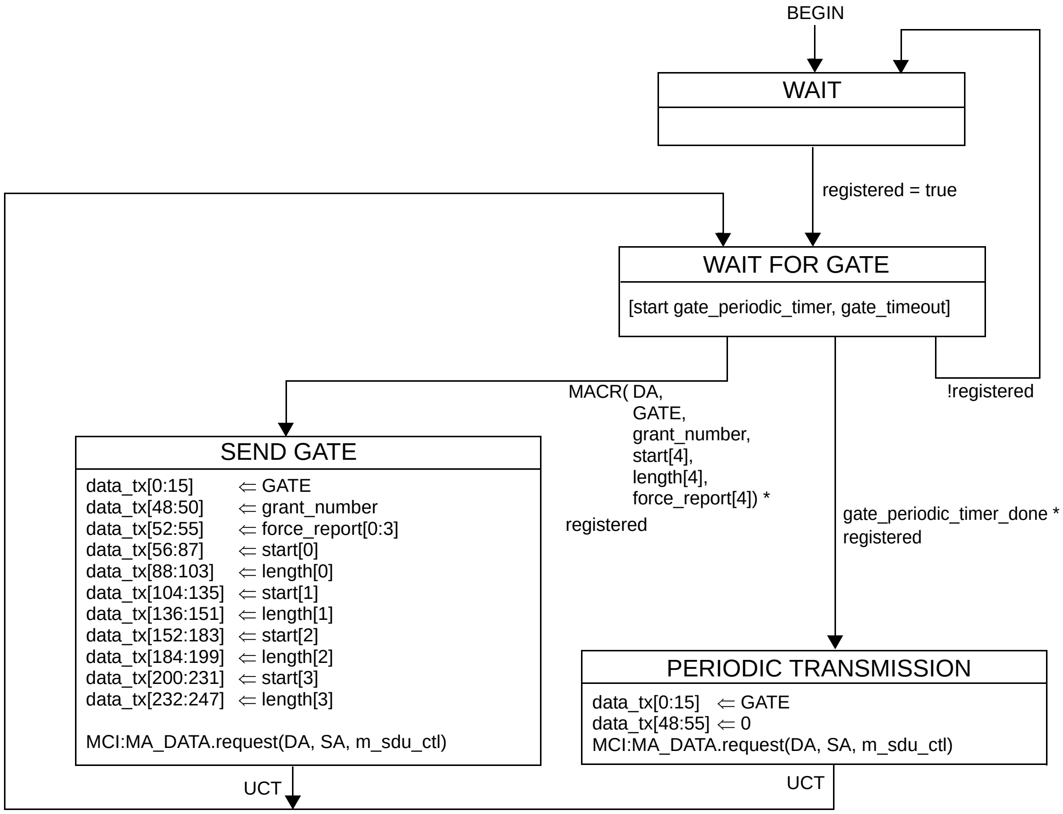
Opcode-specific function(opcode)
~~Opcode-specific function(opcode)~~

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

102.3.5.6 State diagrams

The gating process in the ~~OLT~~ CLT shall implement the Gate processing state diagram as shown in ~~Figure 4–28~~Figure 102–28 and, for TDD mode only, in Figure 102–29. The gating process in the ~~ONU~~ CNU shall implement the Gate processing state diagram as shown in ~~Figure 4–29~~Figure 102–30 and ~~Figure 4–30~~Figure 102–31. Instantiation of state diagrams as described is performed for all Multipoint

MAC Control instances.



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

Figure 4–28—Gate Processing state diagram at OLT

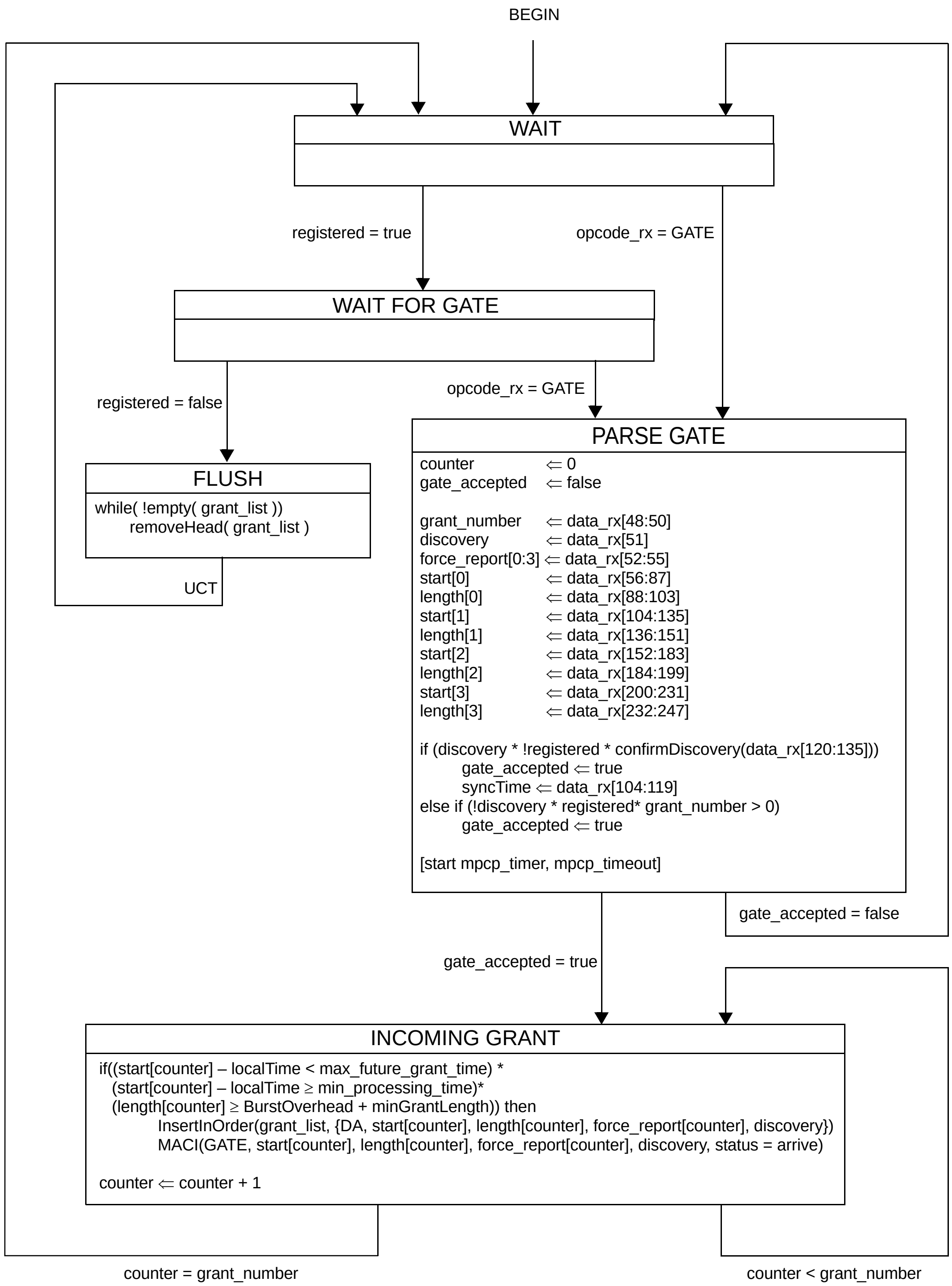


Figure 4–29—Gate Processing ONU Programing state diagram

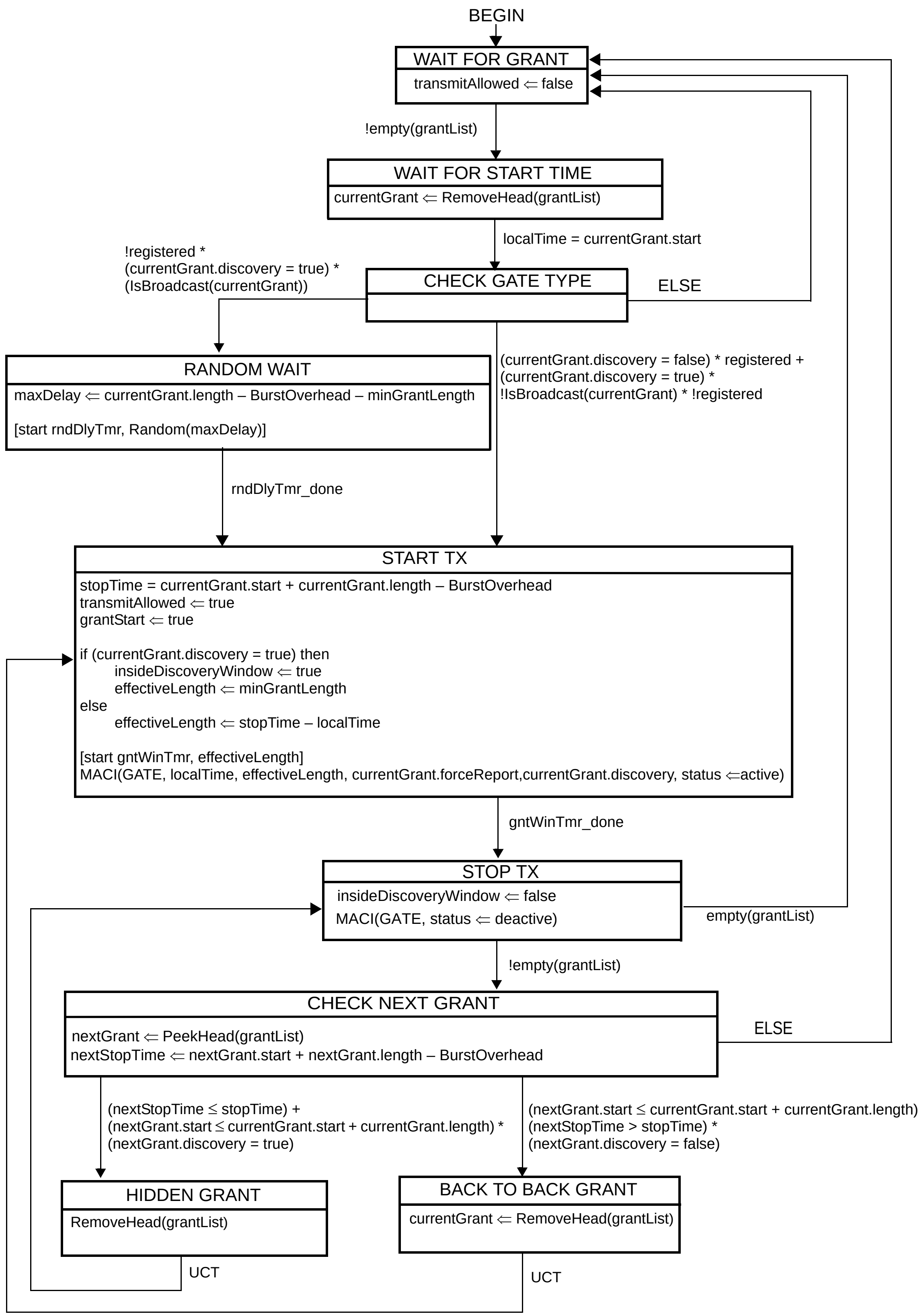
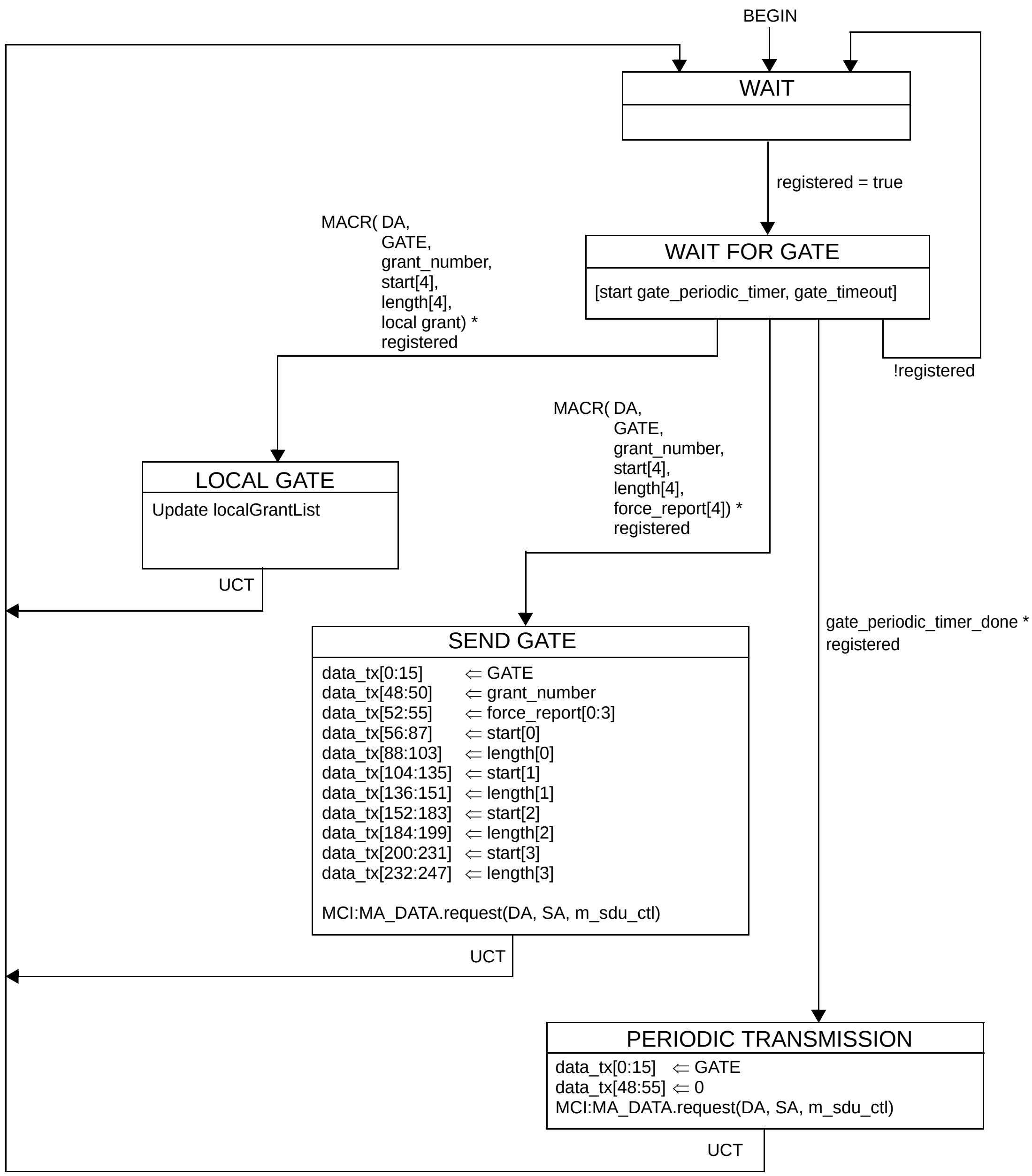


Figure 4–30—Gate Processing ONU Activation state diagram



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

Figure 102–28—Gate Processing state diagram at CLT

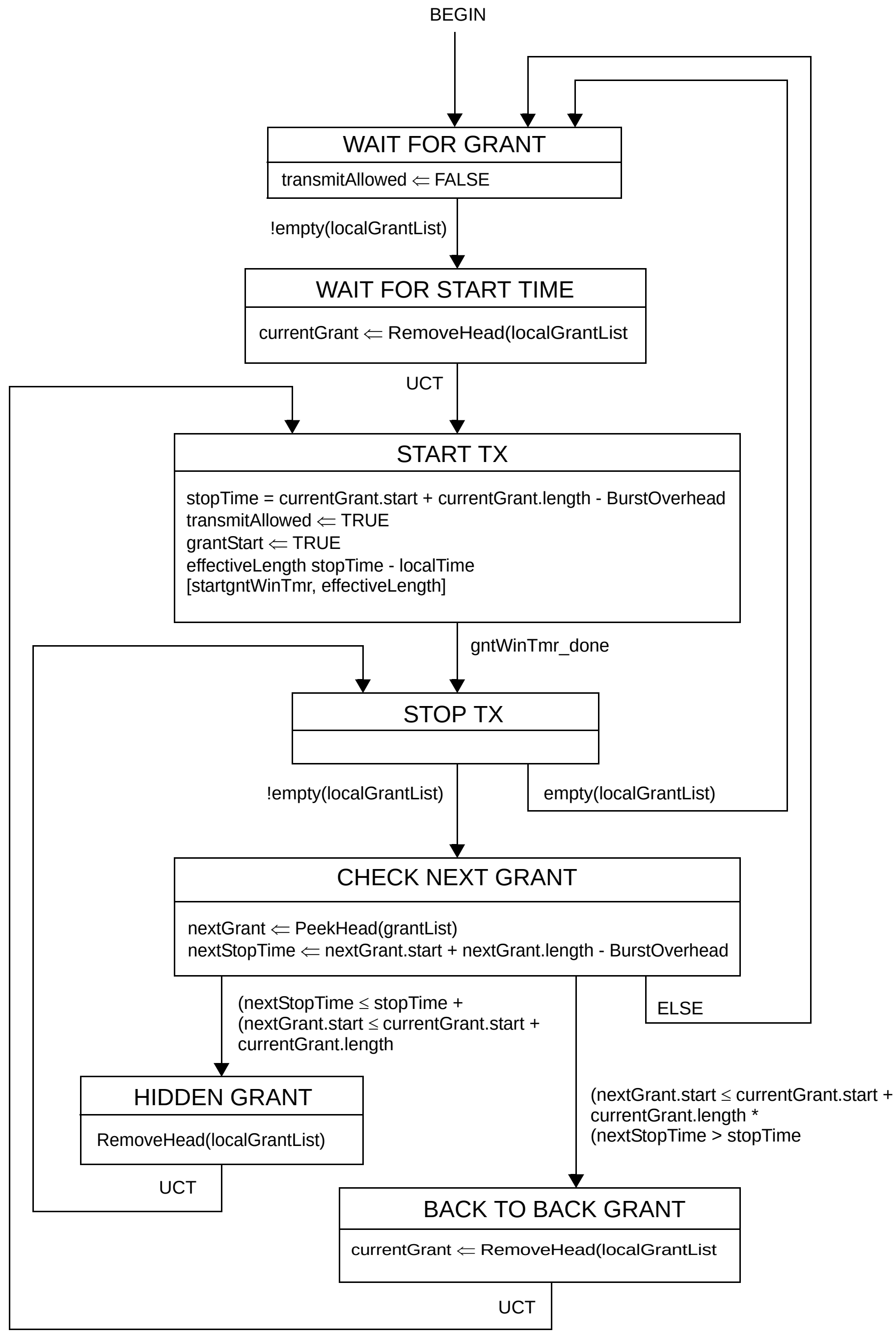


Figure 102–29—Gate Processing CLT Activation state diagram (TDD mode only)

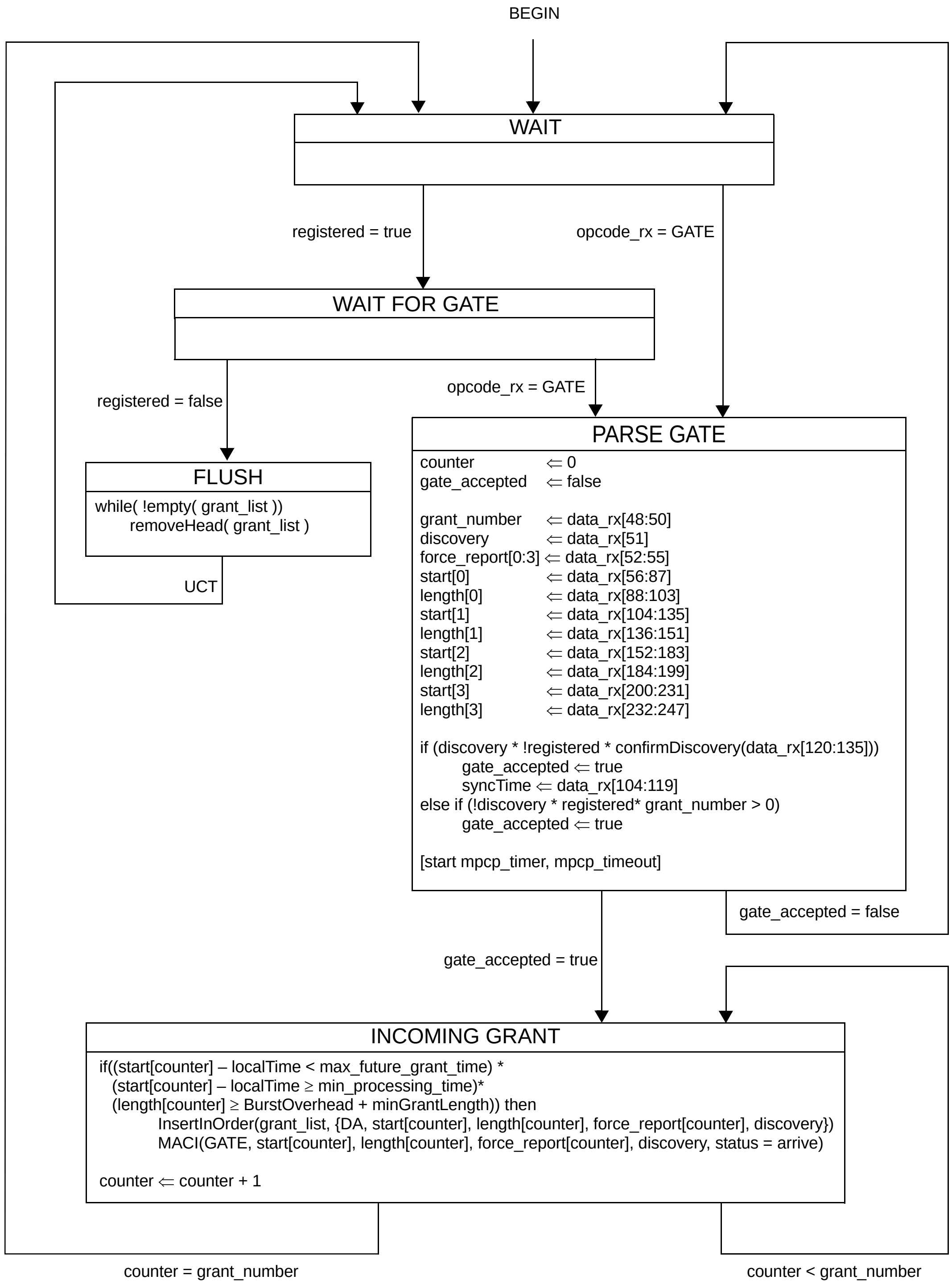


Figure 102–30—Gate Processing CNU Programing state diagram

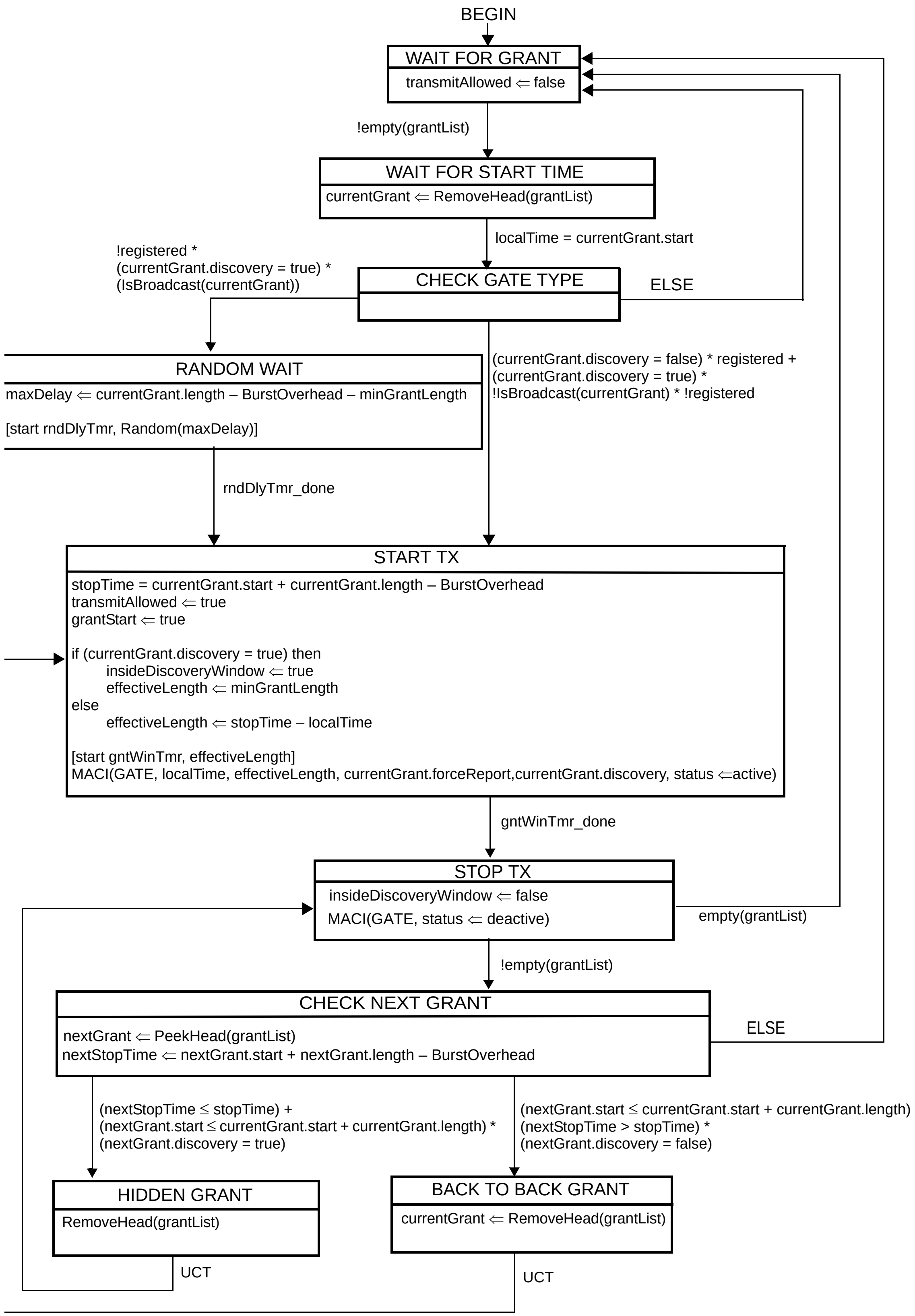


Figure 102–31—Gate Processing CNU Activation state diagram

102.3.6 MPCPDU structure and encoding

The MPCPDU structure shall be as shown in ~~Figure 4–31~~Figure 102–32, 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~~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~~ CLT end, this can be the address any of the individual MACs. These MACs may all share a single unicast address, as explained in ~~102.1.2~~102.1.2.

c) Length/Type. The Length/Type in MPCPDUs carries the MAC_Control_Type field value as specified in ~~31.4.1.3~~31.4.1.3.

d) Opcode. The opcode identifies the specific MPCPDU being encapsulated. Values are defined in ~~Table 31A–1~~Table 31A?.

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.

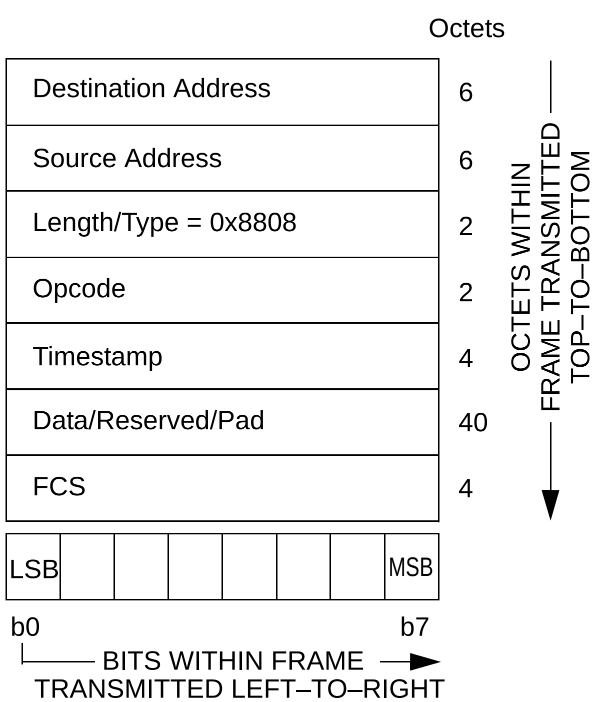


Figure 4–31—Generic MPCPDU

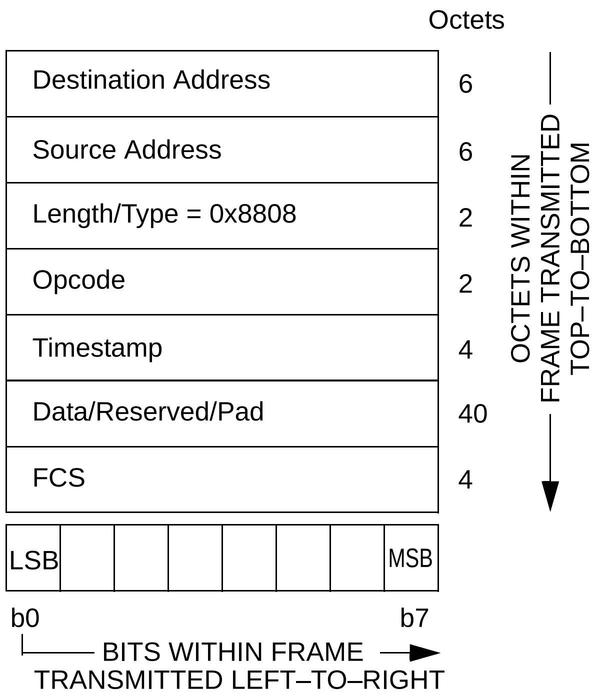


Figure 102–32—Generic MPCPDU

102.3.6.1 GATE description

The purpose of GATE message is to grant transmission windows to ~~ONUs~~ CNUs 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~~ CLT to the ~~ONU.~~ CNU.

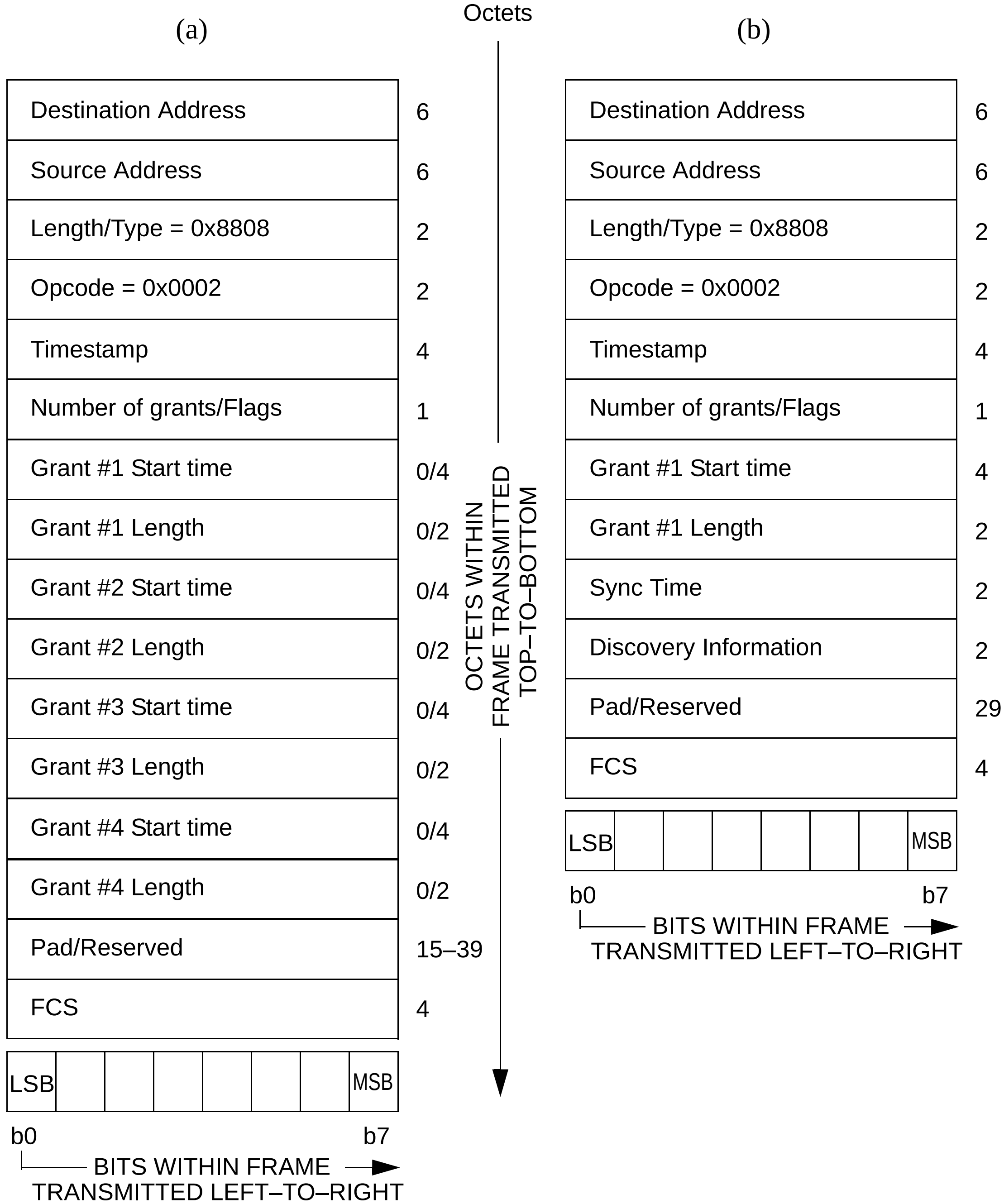


Figure 4–32—GATE MPCPDU: (a) normal GATE MPCPDU, (b) discovery GATE MPCPDU

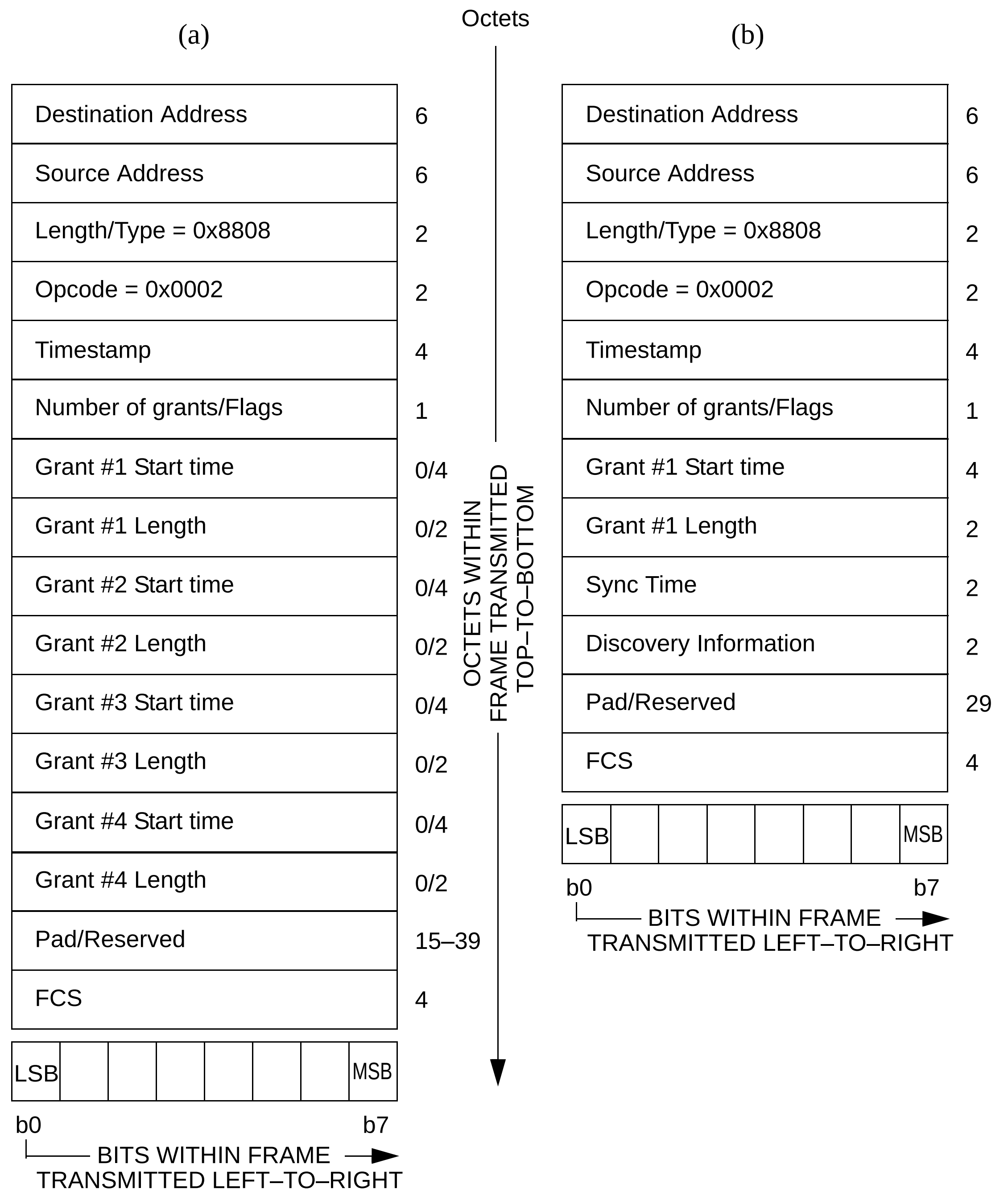


Figure 102–33—GATE MPCPDU: (a) normal GATE MPCPDU, (b) discovery GATE MPCPDU

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: As presented in ~~Table 4–2~~Table 102–2, 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~~CNU.

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 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~~ CNU to issue a REPORT message related to the corresponding grant number at the corresponding transmission opportunity indicated in this GATE~~.~~.

Table 4–2—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

Table 102–2—GATE MPCPDU Number of grants/flags fields

Bit	Flag field	Values
0 – 2	Number of grants	0 – 4

Table 102–2—GATE MPCPDU Number of grants/flags fields

Bit	Flag field	Values
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 Grant 3	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

c) Grant #n Start Time. This 32 bit unsigned field represents the start time of the grant. 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.

d) Grant #n Length. This 16 bit unsigned field represents the length of the grant. The length is counted in 1 time_quantum increments. There are 4 Grants that are possibly packed into the GATE MPCPDU. The ~~laserOnTime~~rfOnTime, syncTime, ~~laserOffTime~~rfOffTime, two initial Idle blocks, FEC parity overhead, and burst terminator sequence (composed of three END_BURST_DELIMITER blocks) are included in and thus consume part of the Grant #n length.

e) Sync Time. This is an unsigned 16 bit value signifying the required synchronization time of the ~~OLT~~ CLT receiver. The ~~ONU~~ CNU calculates the effective grant length by subtracting the syncTime, ~~laserOnTime~~rfOnTime, ~~laserOffTime~~ rfOffTime and END_BURST_DELIMITER from the grant length it received from the ~~OLT~~CLT. 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 a 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 4–3~~Table 102–3 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. 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 15–39 accordingly.

The GATE MPCPDU shall be generated by a MAC Control instance mapped to an active ~~ONU~~CNU, and as such shall be marked with a unicast type of LLID, except when the MPCPDU is a discovery GATE, as indicated by the Discovery flag being set to true. For the discovery procedure, a MAC Control instance is mapped to all ~~ONUs~~CNUs, and therefore, the discovery GATE MPCPDU is marked with the appropriate broadcast LLID (see ~~102.3.2.3~~102.3.2.3)~~.~~.

Table 4–3—GATE MPCPDU discovery information fields

Bit	Flag field	Values
0	OLT is 1G upstream capable	0 – OLT does not support 1 Gb/s reception 1 – OLT supports 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 cannot receive 1 Gb/s data in this window 1 – OLT can 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 102–3—GATE MPCPDU discovery information fields

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

102.3.6.2 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~~ CNUs indicate the upstream bandwidth needs they request per IEEE 802.1Q priority queue. REPORT messages are also used as keep–alives from ~~ONU~~ CNU to ~~OLT~~CLT. ~~ONUs~~ CNUs issue REPORT messages periodically in order to maintain link health at the ~~OLT~~ CLT as defined in ~~102.3.4~~102.3.4. In addition, the ~~OLT~~ CLT 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~~ 8 bit flag register that indicates which queues are represented in this REPORT MPCPDU~~—~~, see ~~Table 4–4~~Table 102–4.

Table 4–4—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
7	Queue 7	0 – queue 7 report is not present; 1 – queue 7 report is present

Table 102–4—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 2report is present
3	Queue 3	0 – queue 3 report is not present; 1 – queue 3report 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
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 and rounded up to the nearest time_quantum to account for the necessary inter–frame spacing and preamble. FEC parity overhead is not included in the reported length. The Queue #n Report field is an unsigned ~~16~~ 16 bit integer representing the 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. 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~~CNU, and as such shall be marked with a unicast type of LLID.

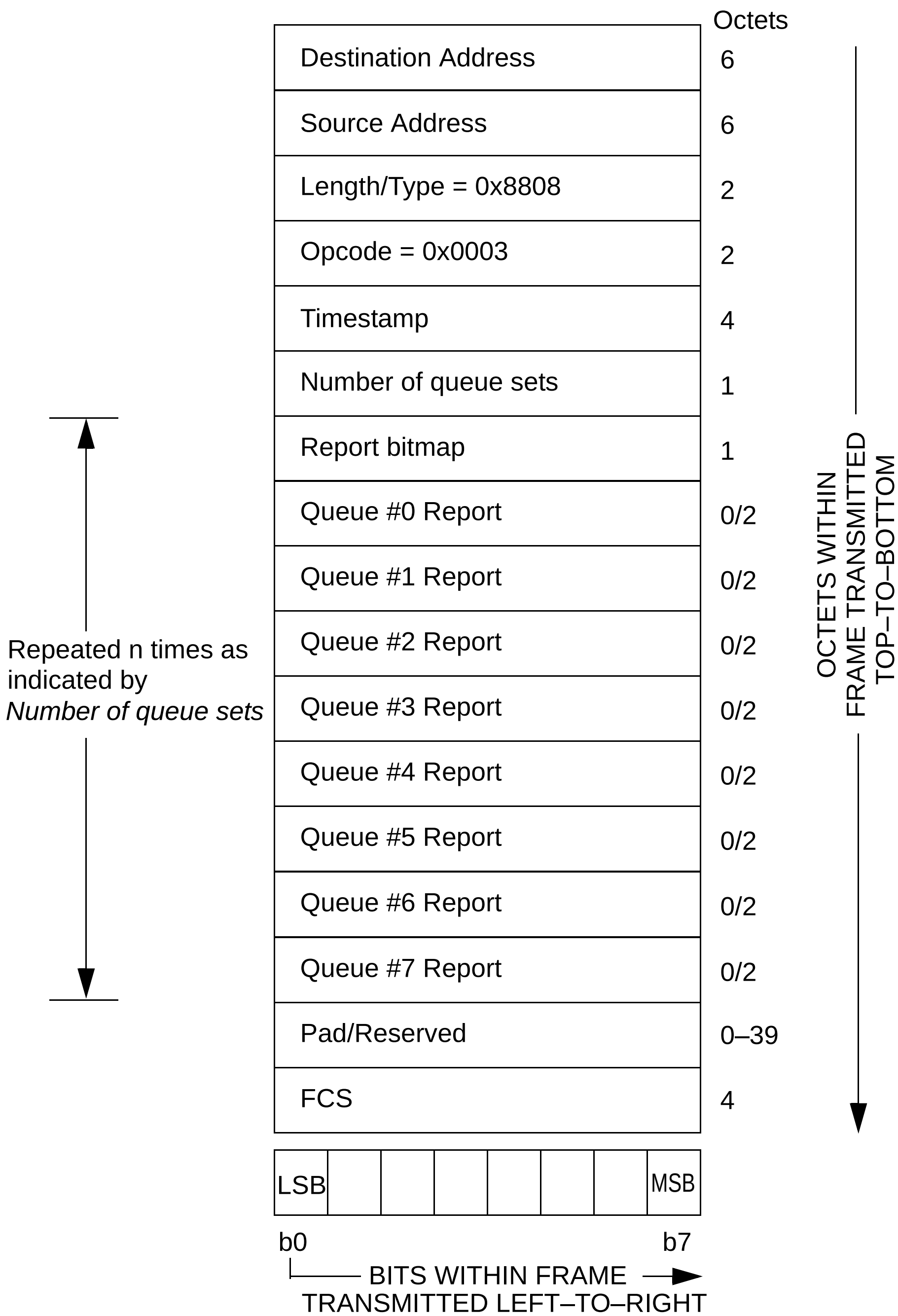


Figure 4–33—REPORT MPCPDU

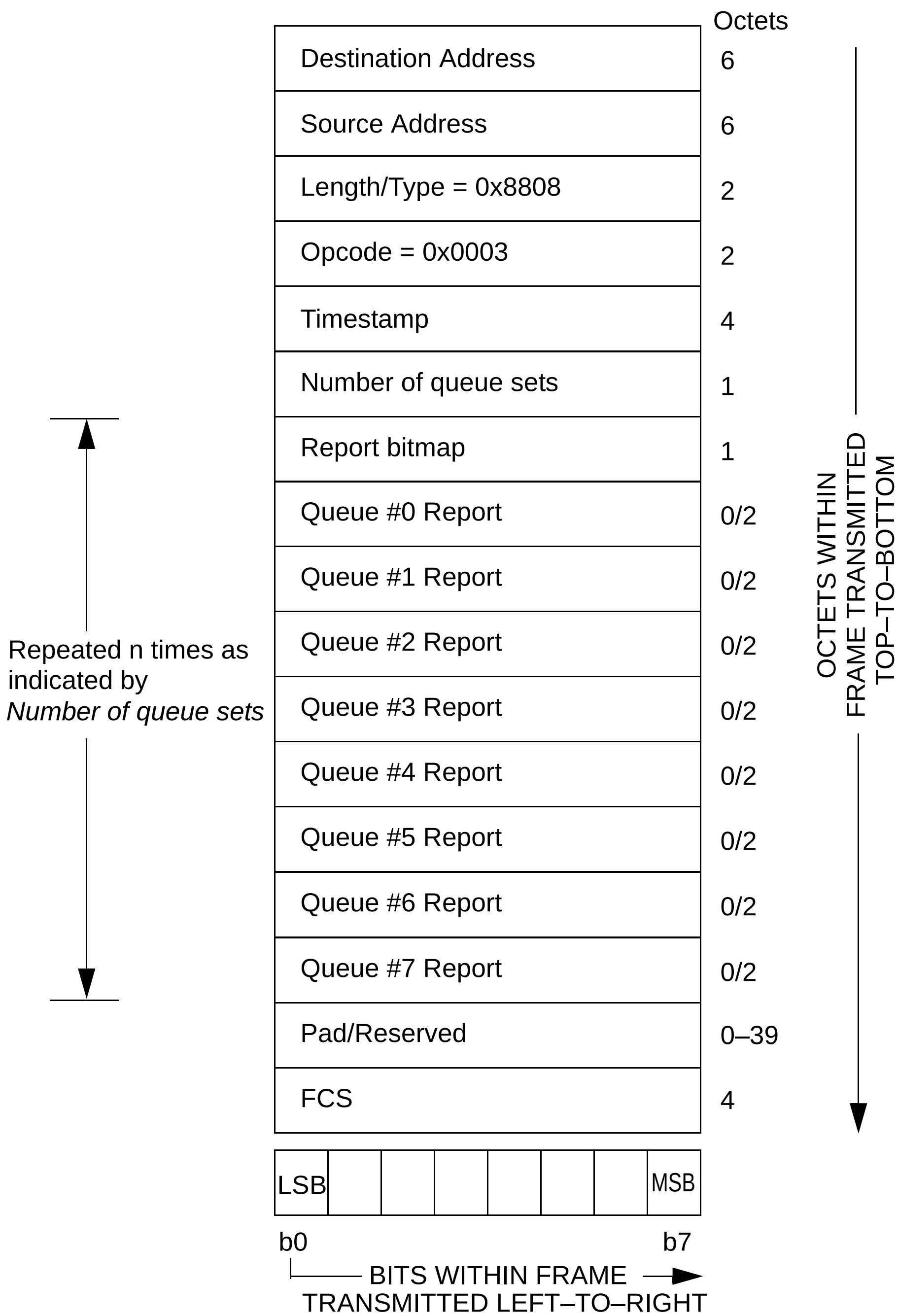


Figure 102–34—REPORT MPCPDU

102.3.6.3 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~~ 8 bit flag register that indicates special requirements for the registration, as presented in ~~Table 4–5~~Table 102–5.

Table 4–5—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 102–5—REGISTER_REQ MPCPDU Flags fields

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

c) Pending grants. This is an unsigned ~~8~~ 8 bit value signifying the maximum number of future grants the ~~ONU~~ CNU is configured to buffer. The ~~OLT~~ CLT should not grant the ~~ONU~~ CNU more than this maximum number of Pending grants vectors comprised of {start, length, force_report, discovery} into the future.

d) Discovery Information. This is a 16 bit flag register. ~~Table 4–6~~Table 102–6 presents the structure of the Discovery Information flag.

Table 4–6—REGISTER_REQ MPCPDU Discovery Information Fields

Bit	Flag field	Values
0	ONU is 1G upstream capable	0 – ONU transmitter is not capable of 1 Gb/s 1 – ONU transmitter is 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 Gb/s registration is not attempted 1 – 1 Gb/s registration is attempted
5	10G registration attempt	0 – 10 Gb/s registration is not attempted 1 – 10 Gb/s registration is attempted

Table 4–6—REGISTER_REQ MPCPDU Discovery Information Fields

Bit	Flag field	Values
6–15	Reserved	Ignored on reception

Table 102–6—REGISTER_REQ MPCPDU Discovery Information Fields

Bit	Flag field	Values
0	CNU is 1G upstream capable	0 – CNU transmitter is not capable of 1 Gb/s 1 – CNU transmitter is capable of 1 Gb/s
1	CNU is 10G upstream capable	0 – CNU transmitter is not capable of 10 Gb/s 1 – CNU transmitter is capable of 10 Gb/s
2 – 3	Reserved	Ignored on reception.
4	1G registration attempt	0 – 1 Gb/s registration is not attempted 1 – 1 Gb/s registration is attempted
5	10G registration attempt	0 – 10 Gb/s registration is not attempted 1 – 10 Gb/s registration is attempted
6 – 255	Reserved	Ignored on reception.

EDITORS NOTE: should below be rfOnTime and rfOffTime? The text seems to use “RF On Time” and RF Off Time” along with “rfOnTime”, “RFOn Time”, “rfOffTime”, and “RFOff Time”. This may be confusing to the reader, recommend using variable simplifying this.

e) ~~Laser On~~ RFOn Time. This field is 1 octet long and carries the ~~Laser~~ RF On Time characteristic for the given ~~ONU~~ CNU transmitter. The value is expressed in the units of time_quanta.

f) ~~Laser Off~~ RFOff Time. This field is 1 octet long and carries the ~~Laser~~ RF Off Time characteristic for the given ~~ONU~~ CNU 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~~.~~.

The REGISTER_REQ MPCPDU shall be generated by a MAC Control instance mapped to an undiscovered ~~ONU~~CNU, and as such shall be marked with a broadcast type of LLID (~~102.3.2.3~~102.3.2.3).

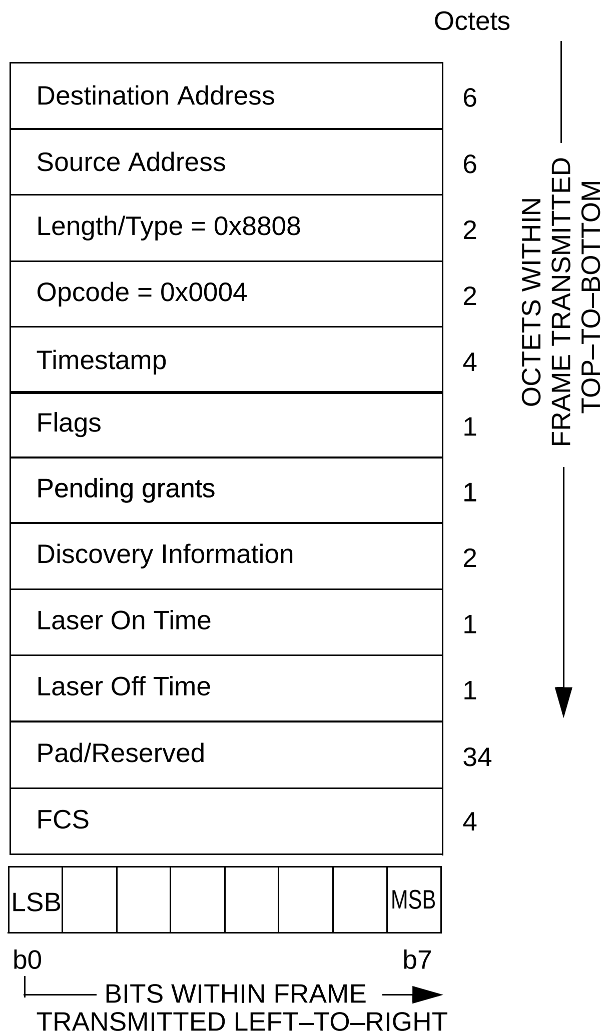


Figure 4–34—REGISTER_REQ MPCPDU

EDITORS NOTE: show Laser On Time and Laser Off Time be replace with RF On time and RF Off Time (or some variant thereof) in the figure below? What about 102-36?

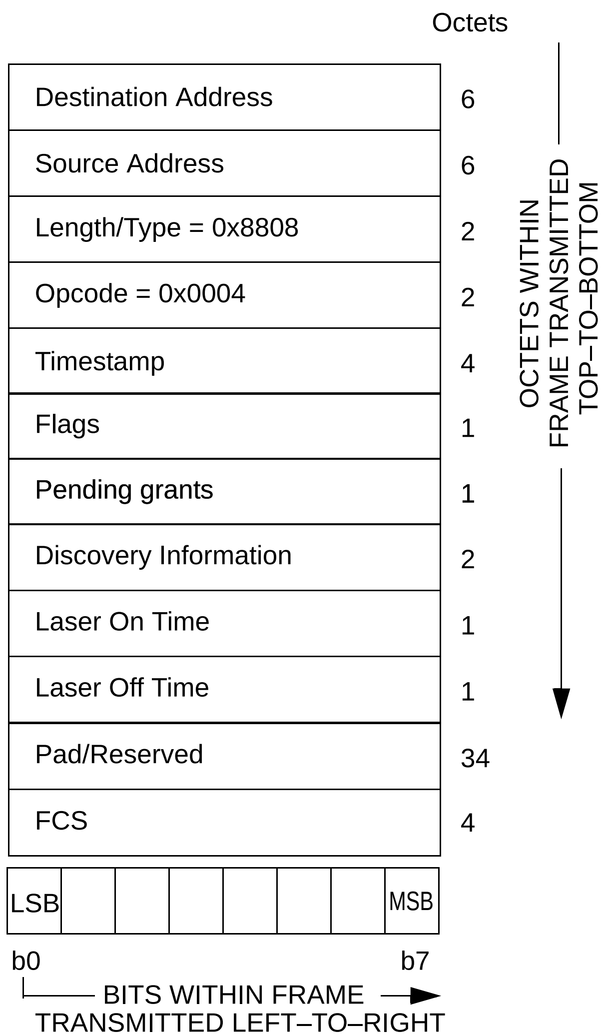


Figure 102–35—REGISTER_REQ MPCPDU

102.3.6.4 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-~~16 bit unsigned value reflecting the LLID of the port assigned following registration.

d) Flags. this is an ~~8-~~8 bit flag register that indicates special requirements for the registration, as presented in ~~Table 4–7.~~Table 102–7.

e) Sync Time. This is an unsigned 16 bit value signifying the required synchronization time of the ~~OLT~~ CLT receiver. The ~~ONU~~ CNU calculates the effective grant length by subtracting the syncTime, ~~laserOnTime~~rfOnTime, ~~laserOffTime~~rfOffTime, and END_BURST_DELIMITER from the grant length it received from the ~~OLT~~CLT. The value is counted in 1 time_quantum increments. The

Table 4–7—REGISTER MPCPDU Flags field

Value	Indication	Comment
0	Reserved	Ignored on reception.
1	Reregister	The ONU is explicitly asked to re-register.
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 MAC Control Client.
5–255	Reserved	Ignored on reception.

advertised value includes synchronization requirement on all receiver elements including PMD, PMA, and PCS.

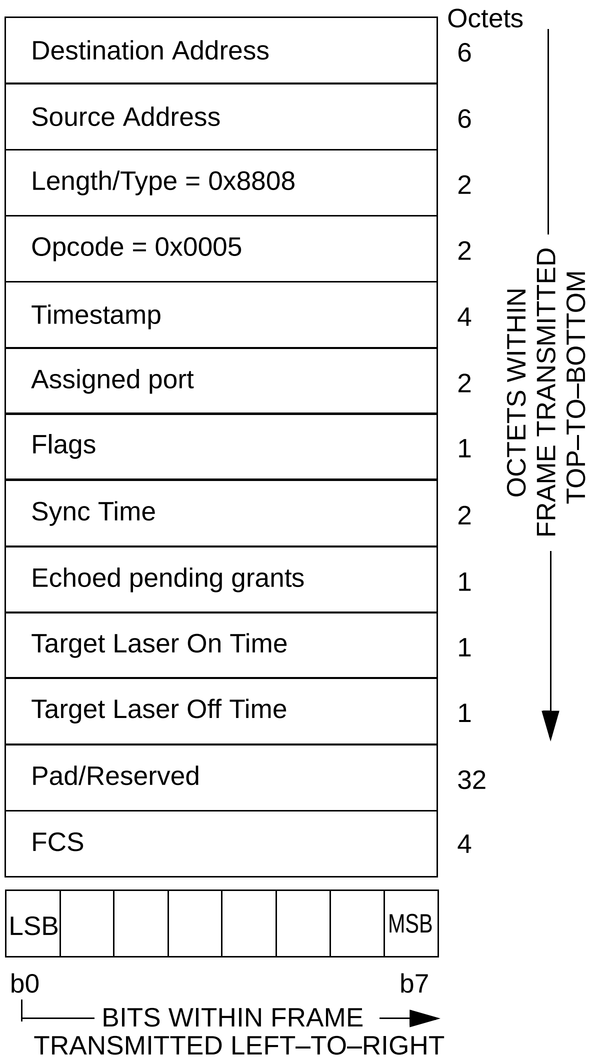


Figure 4–35—REGISTER MPCPDU

Table 102–7—REGISTER MPCPDU Flags field

Value	Indication	Comment
0	Reserved	Ignored on reception.

Table 102–7—REGISTER MPCPDU Flags field

Value	Indication	Comment
1	Reregister	The CNU is explicitly asked to re-register.
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	he requested registration attempt is denied by the MAC Control Client.
5 – 255	Reserved	Ignored on reception.

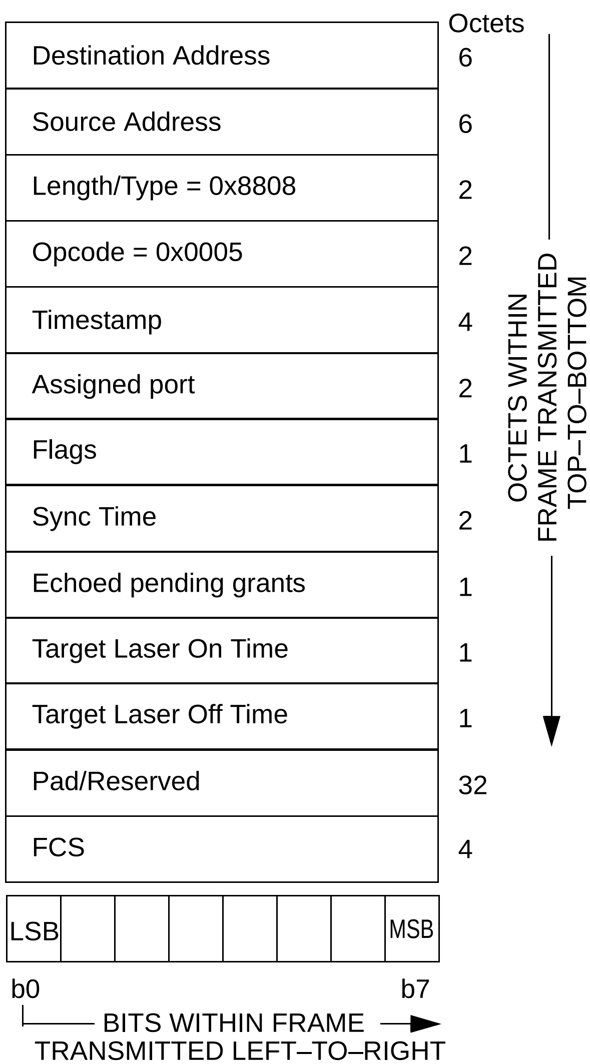


Figure 102–36—REGISTER MPCPDU

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

g) Target ~~Laser On~~ RFn Time. This is an unsigned ~~8~~ 8 bit value, expressed in the units of time_quanta, signifying the ~~Laser~~ RF On Time for the given ~~ONU~~ CNU transmitter. This value may be different from ~~Laser~~ RF On Time delivered by the ~~ONU~~ CNU in the REGISTER_REQ MPCPDU during the Discovery process. The ~~ONU~~ CNU updates the local ~~laserOnTime~~ rfOnTime variable per state diagram in ~~Figure 4–23~~Figure 102–23.

h) Target ~~Laser~~ RF Off Time. This is an unsigned ~~8~~ 8 bit value, expressed in the units of time_quanta, signifying the ~~Laser~~ RF Off Time for the given ~~ONU~~ CNU transmitter. This value may be different from ~~Laser~~ RF Off Time delivered by the ~~ONU~~ CNU in the REGISTER_REQ MPCPDU during the Discovery process. The ~~ONU~~ CNU updates the local ~~laserOffTime~~ rfOffTime variable per state diagram in ~~Figure 4–23~~Figure 102–23.

i) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored on reception.

The REGISTER MPCPDU shall be generated by a MAC Control instance mapped to all ~~ONUs~~ CNUs and such frame is marked by the broadcast LLID (~~102.3.2.3~~102.3.2.3).

102.3.6.5 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~~ 8 bit flag register that indicates special requirements for the registration, as presented in ~~Table 4–8~~Table 102–8.

c) Echoed assigned port. This field holds a ~~16~~ 16 bit unsigned value reflecting the LLID for the port assigned following registration.

d) Echoed Sync Time. This is an unsigned ~~16~~ 16 bit value echoing the required synchronization time of the ~~OLT~~ CLT receiver as previously advertised (~~102.3.6.4~~102.3.6.4).

e) Pad/Reserved. This is an empty field that is transmitted as zeros, and ignored at reception.

Table 4–8—REGISTER_ACK MPCPDU Flags fields

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

Table 102–8—REGISTER_ACK MPCPDU Flags fields

Value	Indication	Comment
0	Nack	The requested registration attempt is denied by the MAC Control Client.
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~~CNU, and as such shall be marked with a unicast type of LLID.

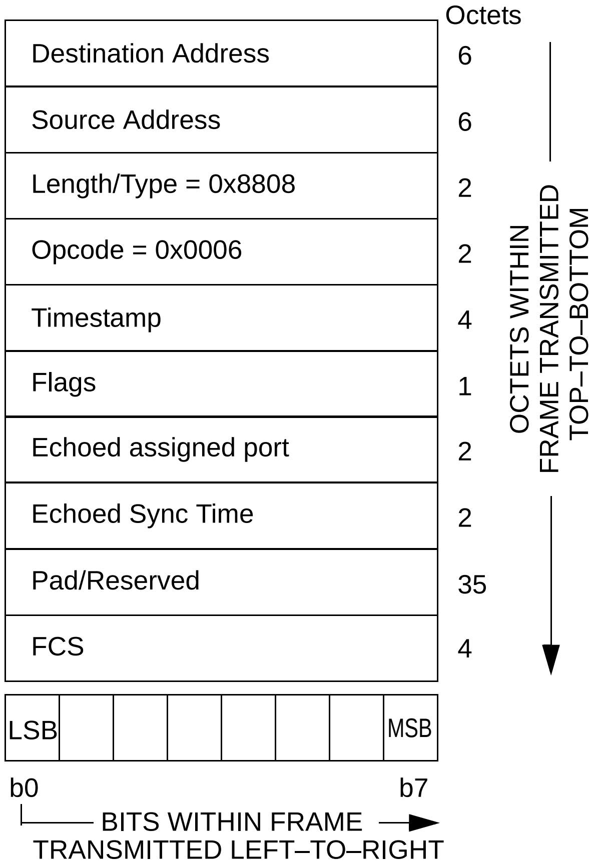


Figure 4–36—REGISTER_ACK MPCPDU

102.4 Discovery Process in dual-rate systems

The enhancements introduced to the ~~Clause 102~~Clause 102 discovery process for EPONs facilitate the coexistence of 10G–~~EPON~~ PON with 1G–~~EPON~~PON.

102.4.1 ~~OLT~~ CLT speed-specific discovery

The discovery GATE MPCPDU is defined in ~~Clause 64~~ Clause 64 for ~~1~~ 1 Gb/s operation and in ~~Clause 102~~ Clause 77 for ~~10~~ 10 Gb/s operation. An additional field (Discovery Information field) was added to the ~~10~~ 10 Gb/s discovery GATE MPCPDU. This field allows the ~~OLT~~ CLT to relay speed-specific information regarding the discovery window to the different ~~ONUs~~ CNUs that may coexist in the same PON. The ~~OLT~~ CLT has the ability to transmit common discovery GATE MPCPDUs on both the ~~1~~ 1 Gb/s transmit path and ~~10~~ 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~~ CLT is capable of opening windows for individual speeds or multiple speeds.

EDITORS NOTE: the above para referenced Clause Z rather than Clause 77 for 10G-EPON.

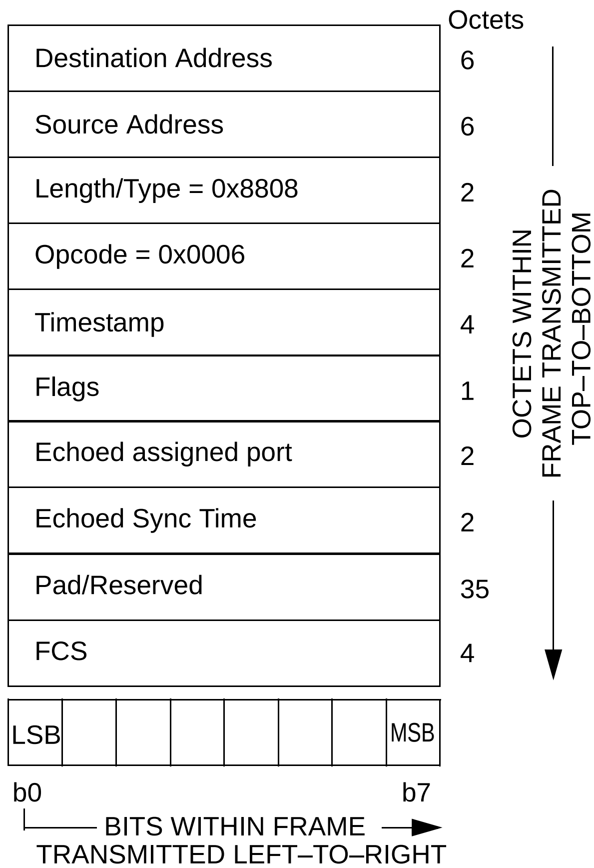


Figure 102–37—REGISTER_ACK MPCPDU

These different combinations allow the ~~OLT~~ CLT MAC Control Client to open a number of discovery windows for all of the different ~~ONU~~ CNU types. ~~Table 4–9~~ Table 102–9shows 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 may be desirable for the ~~OLT~~ CLT MAC Control Client to open overlapping discovery windows. It may do so by sending one discovery GATE MPCPDU on the ~~1~~ 1 Gb/s downstream channel and a similar discovery GATE MPCPDU on the ~~10~~ 10 Gb/s downstream channel; both discovery GATE MPCPDUs having the same Start Time value.

~~Figure 4–37~~Figure 102–38 shows the three primary combinations of discovery windows and the different types of REGISTER_REQ MPCPDUs that may be received during the window. ~~Figure 4–37~~Figure 102–38(a) shows reception of messages from ~~1~~ 1 Gb/s and 10/~~1~~ 1 Gb/s ~~ONUs~~CNUs. ~~Figure 4–37~~Figure 102–38(b) shows reception of messages from ~~10~~ 10 Gb/s ~~ONUs~~CNUs. ~~Figure 4–37~~Figure 102–38(c) shows reception of messages from all types of ~~ONUs~~CNUs.

102.4.2 ~~ONU~~ CNU speed-specific registration

A 1G–~~EPON ONU~~ PON CNU receives only discovery GATE messages transmitted by the ~~OLT~~ CLT in the ~~1~~ 1 Gb/s broadcast channel. Operation and registration of these ~~ONUs~~ CNUs is specified in ~~Clause 64~~Clause 64.

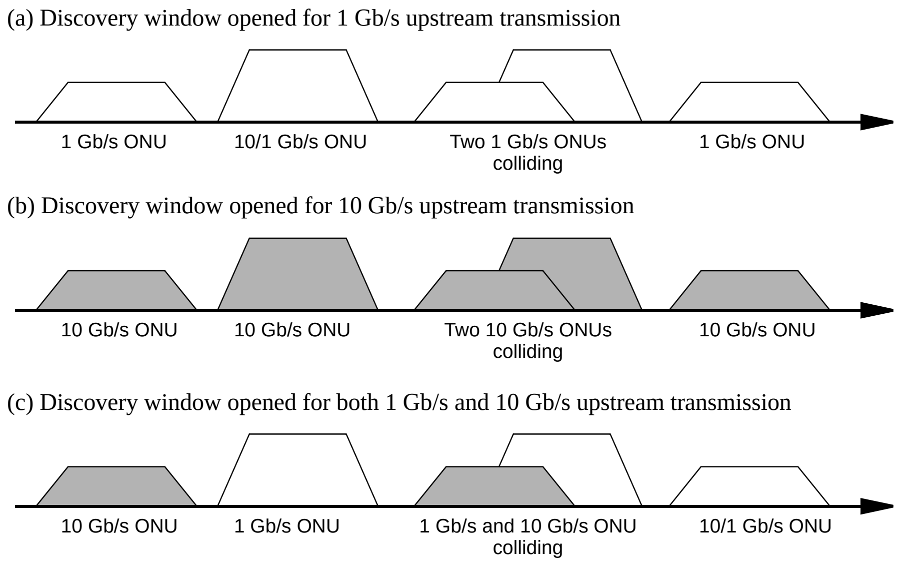


Figure 4–37—Combinations of REGISTER_REQ MPCPDUs during discovery window for 10G–EPON and 1G–EPON coexisting in the same PON

Table 4–9—Discovery GATE MPCPDUs for all ONU types

ONU types targeted by discovery GATE MPCPDU	LLID of discovery GATE(s)	Discovery information			
		Upstream capable		Discovery window	
		1G	10G	1G	10G
1G–EPON	0x7FFF	No Discovery Information field present			
10/1G–EPON	0x7FFE	1	0	1	0
1G–EPON and 10/1G–EPON	0x7FFF[a]	No Discovery Information field present			
	0x7FFE[a]	1	0	1	0
10/10G–EPON	0x7FFE	0	1	0	1
10/1G–EPON and 10/10G–EPON	0x7FFE	1	1	1	1
1G–EPON, 10/1G–EPON, and 10/10G–EPON	0x7FFF[a]	No Discovery Information field present			
	0x7FFE[a]	1	1	1	1

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

A 10/1G–~~EPON ONU~~ PON CNU is only capable of receiving discovery GATE MPCPDU transmitted by the ~~OLT~~ CLT in the ~~10~~ 10 Gb/s broadcast channel. These messages are parsed, and if a ~~1~~ 1 Gb/s discovery window is opened, the ~~ONU~~ CNU may attempt to register in the EPON.

l

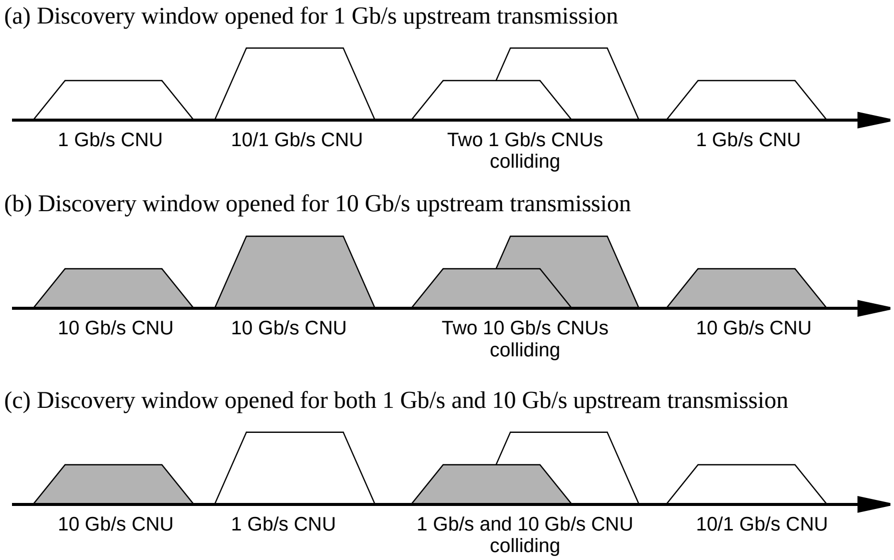


Figure 102–38—Combinations of REGISTER_REQ MPCPDUs during discovery window for 10G–PON and 1G–PON coexisting in the same PON

EDITORS NOTE: will Figure 102-38 need to be updated for EPoC? Baseline material indicate replacing OLT/ONU with CLT/CNU however this appears to be in error

Table 102–9—Discovery GATE MPCPDUs for all CNU types

CNU types targeted by discovery GATE MPCPDU	LLID of discovery GATE(s)	Discovery information			
		Upstream capable		Discovery window	
		1G	10G	1G	10G
1G–PON	0x7FFF	No Discovery Information field present			
10/1G–PON	0x7FFE	1	0	1	0
1G–PON and 10/1G–PON	0x7FFF[a]	No Discovery Information field present			
	0x7FFE[a]	1	0	1	0
10/10G–PON	0x7FFE	0	1	0	1
0/1G–PON and 10/10G–PON	10x7FFE	1	1	1	1
1G–PON, 10/1G–PON, and 10/10G–PON	0x7FFFa	No Discovery Information field present			
	0x7FFEa	1	1	1	1

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

A 10/10G–~~EPON ONU~~ PON CNU is only capable of receiving discovery GATE MPCPDU transmitted by the ~~OLT~~ CLT in the ~~10~~ 10 Gb/s broadcast channel. These messages are parsed, and if a ~~10~~ 10 Gb/s discovery window is opened, the ~~ONU~~ CNU may attempt to register in the EPON.

A dual speed ~~ONU~~ CNU capable of 10/1G–~~EPON~~ PON operation or 10/10G–~~EPON~~ PON operation is also only capable of receiving discovery GATE MPCPDU transmitted by the ~~OLT~~ CLT in the ~~10~~ 10 Gb/s broadcast channel. These messages need to be parsed, and the ~~ONU~~ CNU makes the registration decision based on the available information. The ~~ONU~~ CNU should attempt to register during the discovery window announced as supporting the highest speed common to both the ~~OLT~~ CLT and ~~ONU~~CNU. ~~Table 4–10~~Table 102–10 shows the action the ~~ONU~~ CNU should take based on the ~~ONU~~ CNU transmit capabilities and the received discovery information.

Table 4–10—ONU action during discovery window

OLT Discovery information				ONU Tx capability		ONU action
Upstream capable		Discovery window				
1G	10G	1G	10G	1G	10G	
1	0	1	0	1	X	Attempt 1G registration
1	X	1	X	1	0	Attempt 1G registration
X	1	X	1	X	1	Attempt 10G registration
1	1	0	1	1	0	Wait for 1G discovery window
1	1	1	0	X	1	Wait for 10G discovery window

Table 102–10—CNU action during discovery window

CLT Discovery information				CNU Tx capability		CNU action
Upstream capable		Discovery window				
1G	10G	1G	10G	1G	10G	
1	0	1	0	1	X	Attempt 1G registration
1	X	1	0	1	0	Attempt 1G registration
X	1	X	1	X	1	Attempt 10G registration
1	1	0	1	1	0	Wait for 1G discovery window
1	1	1	0	X	1	Wait for 10G discovery window

The ~~ONU~~ CNU generates the REGISTER_REQ MPCPDU with the same LLID as the discovery GATE MPCPDU it responds to, i.e., 1G–~~EPON ONU~~ PON CNU (per ~~Clause 64~~Clause 64) use LLID 0x7FFF, while the 10G–~~EPON ONUs~~ PON CNUs use LLID 0x7FFE.

102.5 Protocol implementation conformance statement (PICS) proforma for ~~Clause 102~~Clause 102, Multipoint MAC ~~Control~~Control for EPoC[1]

102.5.1 Introduction

The supplier of a protocol implementation that is claimed to conform to ~~Clause 102 Multipoint MAC Control~~Clause 102, clause title, 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~~Clause 21.

102.5.2 Identification

102.5.2.1 Implementation identification

~~Supplier~~Supplier[1]	
Contact point for enquiries about the ~~PICS~~PICS[1]	
Implementation Name(s) and Version(s)[1,3]	
Other information necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)[2]	
NOTE 1—Required for all implementations. NOTE ~~1~~2—~~Only the first three items are required for all implementations; other information may~~ May be completed as appropriate in meeting the requirements for the identification. NOTE ~~2~~3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).	

EDITORS NOTE: Implementation ID table is different in 2012 version.

102.5.2.2 ~~-~~Protocol summary

Identification of protocol standard	IEEE Std 802.3-2012, Clause 102, 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 802.3-2012.)	
Date of Statement	

EDITORS NOTE:Protocol summary table in 2012 version omits blank row.

[1] *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~~.~~.

Identification of protocol standard	IEEE Std 802.3xx-201x, Clause 102, clause title
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 802.3xx-201x.)	

Date of Statement	

102.5.3 Major capabilities/~~options~~ options

Item	Feature	Subclause	Value/Comment	Status	Support
*~~OLT~~ CLT	~~OLT~~ CLT functionality	~~102.1~~102.1	Device supports functionality required for ~~OLT~~CLT	O/1	Yes [ ] No [ ]
*~~ONU~~ CNU	~~ONU~~ CNU functionality	~~102.1~~102.1	Device supports functionality required for ~~ONU~~CNU	O/1	Yes [ ] ~~No [ ]~~

EDITORS NOTE: the remainder of this clause has been copied from Cl 77 2012 edition with OLT/ONU replaced with CLT/CNU and cross-references updated.

4.5.4 PICS proforma tables for Multipoint MAC Control

4.5.4.1 Compatibility considerations

Item	Feature	Subclause	Value/Comment	Status	Support
CC1	Delay through MAC	~~102.3.2.4~~102.3.2.4	Maximum delay variation of 1 time_quantum	M	Yes [ ]
CC2	~~OLT~~ CLT grant time delays	~~102.3.2.4~~102.3.2.4	Not grant nearer than 1024 time_quanta into the future	~~OLT~~ CLT:M	Yes [ ]
CC3	~~ONU~~ CNU processing delays	~~102.3.2.4~~102.3.2.4	Process all messages in less than 1024 time_quanta	~~ONU~~CNU:M	Yes [ ]
CC4	~~OLT~~ CLT grant issuance	~~102.3.2.4~~102.3.2.4	Not grant more than one message every 1024 time_quanta to a single ~~ONU~~CNU	~~OLT~~ CLT:M	Yes [ ]

4.5.4.2 Multipoint MAC Control

Item	Feature	Subclause	Value/Comment	Status	Support
OM1	~~OLT~~ CLT localTime	~~102.2.2.2~~102.2.2.2	Track transmit clock	~~OLT~~-CLT:M	Yes [ ]
OM2	~~ONU~~ CNU localTime	~~102.2.2.2~~102.2.2.2	Track receive clock	~~ONU~~CNU:M	Yes [ ]
OM3	Random wait for transmitting REGISTER_REQ messages	~~102.3.3~~102.3.3	Shorter than length of discovery window	~~ONU~~CNU:M	Yes [ ]
OM4	Periodic report generation	~~102.3.4~~102.3.3.3	Reports are generated periodically	~~ONU~~CNU:M	Yes [ ]
OM5	Periodic granting	~~102.3.4~~102.3.3.4	Grants are issued periodically	~~OLT~~-CLT:M	Yes [ ]
OM6	Issuing of grants	~~102.3.5~~102.3.3.5	Not issue more than maximum supported grants	~~OLT~~-CLT:M	Yes [ ]

4.5.4.3 State diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	Multipoint Transmission Control	102.2.2.7	Meets the requirements of Figure 4–10	M	Yes [ ]
SM2	~~OLT~~ CLT Control Parser	102.2.2.7	Meets the requirements of Figure 4–11	M	Yes [ ]
SM3	~~ONU~~ CNU Control Parser	102.2.2.7	Meets the requirements of Figure 4–14	M	Yes [ ]
SM4	~~OLT~~ CLT Control Multiplexer	102.2.2.7	Meets the requirements of Figure 4–15	~~OLT~~-CLT:M	Yes [ ]
SM5	~~ONU~~ CNU Control Multiplexer	102.2.2.7	Meets the requirements of Figure 4–16	~~OLT~~-CLT:M	Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
SM6	Discovery Processing ~~OLT~~ CLT Window Setup	102.3.3.6	Meets the requirements of Figure 4–19	~~OLT~~CLT:M	Yes [ ]
SM7	Discovery Processing ~~OLT~~ CLT Process Requests	102.3.3.6	Meets the requirements of Figure 4–20	~~OLT~~CLT:M	Yes [ ]
SM8	Discovery Processing ~~OLT~~ CLT Register	102.3.3.6	Meets the requirements of Figure 4–21	~~ONU~~CNU:M	Yes [ ]
SM9	Discovery Processing ~~OLT~~ CLT Final Registration	102.3.3.6	Meets the requirements of Figure 4–22	~~OLT~~CLT:M	Yes [ ]
SM10	Discovery Processing ~~ONU~~ CNU Registration	102.3.3.6	Meets the requirements of Figure 4–23	~~ONU~~CNU:M	Yes [ ]
SM11	Report Processing at ~~OLT~~CLT	102.3.4.6	Meets the requirements of Figure 4–25	~~OLT~~CLT:M	Yes [ ]
SM12	Report Processing at ~~ONU~~CNU	102.3.4.6	Meets the requirements of ~~Figure 4–26~~	~~ONU~~CNU:M	Yes [ ]
SM13	Gate Processing at ~~OLT~~CLT	102.3.5.6	Meets the requirements of ~~Figure 4–28~~	~~OLT~~CLT:M	Yes [ ]
SM14	Gate Processing at ~~ONU~~CNU	102.3.5.6	Meets the requirements of Figure 4–29	~~ONU~~CNU:M	Yes [ ]
SM15	Gate Processing ~~ONU~~ CNU Activation	102.3.5.6	Meets the requirements of Figure 4–30	~~ONU~~CNU:M	Yes [ ]

4.5.4.4 MPCP

Item	Feature	Subclause	Value/Comment	Status	Support
MP1	MPCPDU structure	102.3.6	As in Figure 4–31	M	Yes [ ]
MP2	LLID for MPCPDU	102.3.6	RS generates LLID for MPCPDU	M	Yes [ ]
MP3	Grants during discovery	102.3.6.1	Single grant in GATE message during discovery	~~OLT~~CLT:M	Yes [ ]
MP4	Grant start time	102.3.6.1	Grants within one GATE MPCPDU are sorted by their Start time values	~~OLT~~CLT:M	Yes [ ]
MP5	GATE generation	102.3.6.1	GATE generated for active ~~ONU~~ CNU except during discovery	~~OLT~~CLT:M	Yes [ ]
MP6	GATE LLID	102.3.6.1	Unicast LLID except for discovery	~~OLT~~CLT:M	Yes [ ]
MP7	REPORT issuing	102.3.6.2	Issues REPORT periodically	~~ONU~~CNU:M	Yes [ ]
MP8	REPORT generation	102.3.6.2	Generated by active ~~ONU~~CNU	~~ONU~~CNU:M	Yes [ ]
MP9	REPORT LLID	102.3.6.2	REPORT has unicast LLID	~~ONU~~CNU:M	Yes [ ]
MP10	REGISTER_REQ generation	102.3.6.3	Generated by undiscovered ~~ONU~~CNU	~~ONU~~CNU:M	Yes [ ]
MP11	REGISTER_REQ LLID	102.3.6.3	Use broadcast LLID	~~ONU~~CNU:M	Yes [ ]
MP12	REGISTER DA address	102.3.6.4	Use individual MAC address	~~OLT~~CLT:M	Yes [ ]
MP13	REGISTER generation	102.3.6.4	Generated for all ~~ONUs~~CNUs	~~OLT~~CLT:M	Yes [ ]
MP14	REGISTER_ACK generation	102.3.6.5	Generated by active ~~ONU~~CNU	~~ONU~~CNU:M	Yes [ ]
MP15	REGISTER_ACK LLID	102.3.6.5	Use unicast LLID	~~ONU~~CNU:M	Yes [ ]