103. Multipoint MAC Control for EPoC

103.1 Overview

This clause deals with the mechanism and control protocols required in order to reconcile the EPoC (EPON protocol over coax) into the Ethernet framework. The P2MP medium is a coax cable distribution network (CCDN) in which passive and usually active elements are present in the signal's path from source to destination. When combined with the EPON protocol, such a network is referred to as EPON protocol over coax network (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 cable line terminal (CLT) and the DTEs connected at the branches of the tree are called cable network units (CNU). The CLT typically resides at the service provider’s facility, while the CNUs are located at the subscriber premises.

The EPoC topology is similar to the P2MP topology of EPON with the optical line terminal being replaced by a CLT, the optical network units replaced by CNU and operating over a coaxial network rather than an optical network.

A simplified P2MP topology example is depicted in Figure 103–1.

Topics dealt with in this clause include allocation of upstream transmission resources to different CNUs, discovery and registration of CNUs into the network, and reporting of congestion to higher layers to allow for dynamic bandwidth allocation schemes and statistical multiplexing across the 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 a coax cable distribution network by defining a Multipoint MAC Control sublayer as an extension of the MAC Control sublayer defined in Clause 31, and supporting current and future operations as defined in Clause 31 and annexes.

Each CCDN consists of a node located at the root of the tree assuming the role of CLT, and multiple nodes located at the tree leaves assuming roles of CNUs. The network operates by allowing CNUs multiplexed in frequency and in time to share the upstream medium. The MPCP located at the CLT is responsible for timing the different transmissions. Reporting of congestion by the different CNUs may assist in optimally allocating the bandwidth across the CCDN.

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

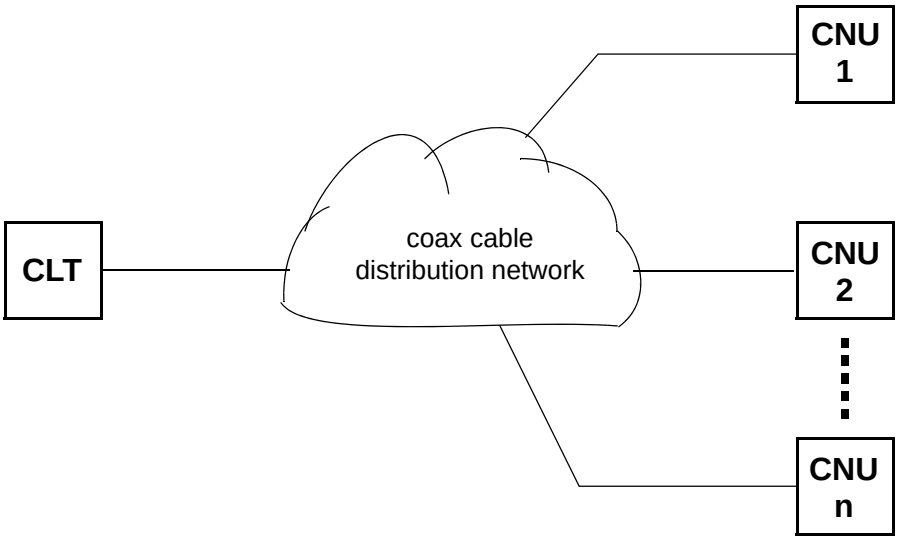


Figure 103–1—Coax cable distribution network topology example

The Multipoint MAC Control functionality shall be implemented for subscriber access devices containing point-to-multipoint Physical Layer devices defined in Clause 100, Clause 101, and Clause 102.

The EPoC Multipoint MAC Control shares much in common with prior versions of the Multipoint MAC Control protocol defined in Clause 64 and Clause 77. There are a number of variables, constants and functions that are complementary to those defined for EPON Multipoint MAC Control but that are unique to EPoC. These are listed in Table 103–1.

Table 103–1—Complementary constants, variables and functions between EPON and EPoC

Clause 77	Clause 103
guardThresholdOLT	*guardThresholdCLT*
guardThresholdONU	*guardThresholdCNU*
laserOffTime	*rfOffTime*
laserOnTime	*rfOnTime*
tqSize	*qSizeC*
fecOffset	*fecOffsetC*
OctetsRemaining	*OctetsRemainingC*
ResetBound	*ResetBoundC*

Table 103–1—Complementary constants, variables and functions between EPON and EPoC

Clause 77	Clause 103
CheckGrantSize	*CheckGrantSizeC*
packet_initiate_timer	*packet_initiate_timerC*
effectiveLength	*effectiveLengthC*
minGrantLength	*minGrantLengthC*
rndDlyTmr	*rndDlyTmrC*

103.1.1 Position of Multipoint MAC Control within the IEEE 802.3 hierarchy

The Multipoint MAC Control in this clause defines the MAC control operation for point-to-multipoint networks over coaxial cable distribution networks. Figure 103–2 depicts the architectural positioning of the Multipoint MAC Control sublayer with respect to the MAC and the MAC Control client. The Multipoint MAC Control sublayer takes the place of the MAC Control sublayer to extend it to support multiple clients and additional MAC control functionality.

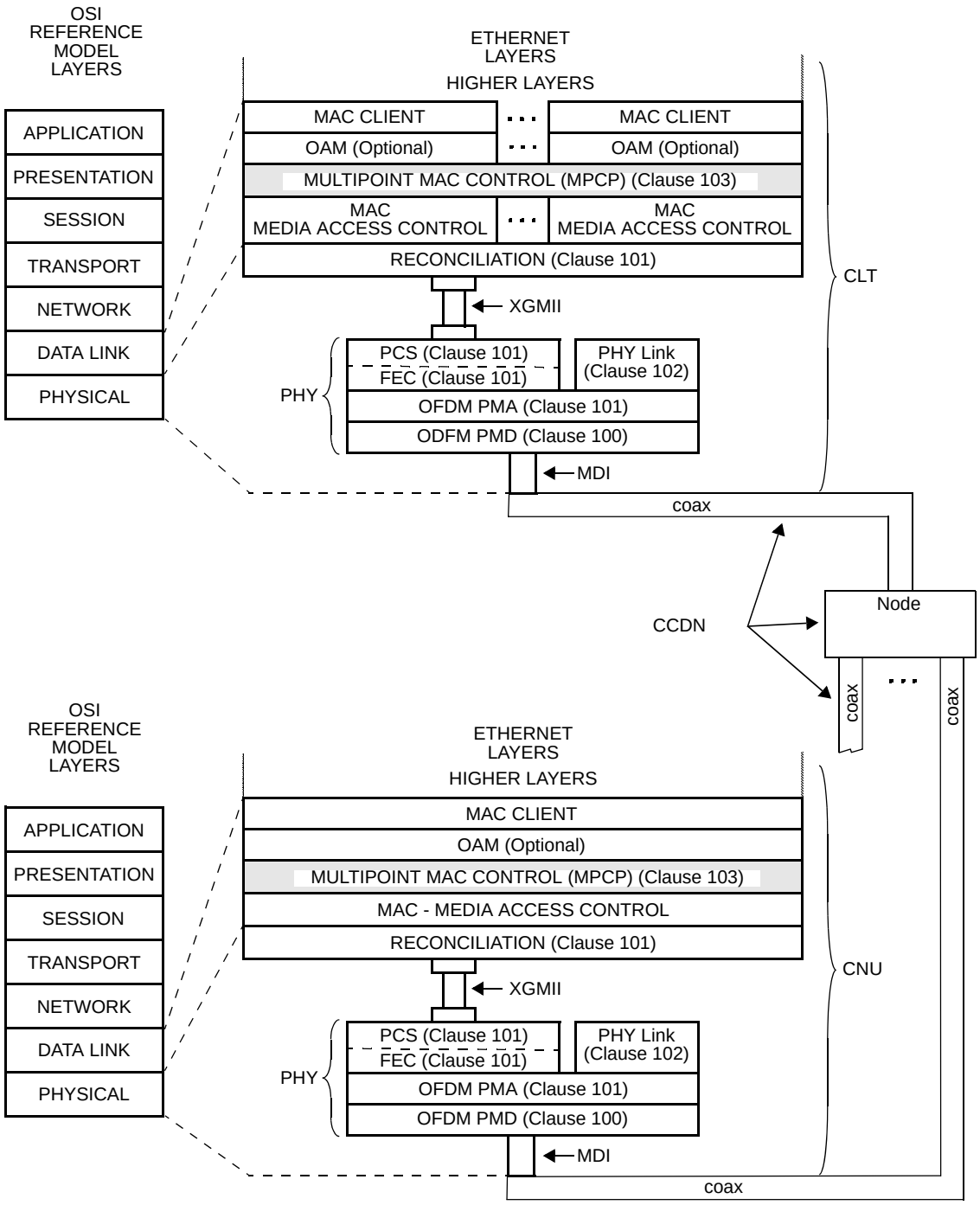
Multipoint MAC Control is defined using the mechanisms and precedents of the MAC Control sublayer. The MAC Control sublayer has extensive functionality designed to manage the real-time control and manipulation of MAC sublayer operation. This clause specifies the extension of the MAC Control mechanism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

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

As depicted in Figure 103–2, the layered system instantiates multiple MAC entities, using a single Physical Layer. The individual MAC instances offer a point-to-point emulation service between the CLT and the CNU. An additional MAC is instantiated to communicate to all 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 EPoC CNUs. This MAC instance is referred to as Single Copy Broadcast (SCB).

The 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 CNU.

Although Figure 103–2 and supporting text describe multiple MACs within the CLT, a single unicast MAC address may be used by the CLT. Within the 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
CCDN= COAX CABLE DISTRIBUTION NETWORK
CLT = COAXIAL LINE TERMINAL
CNU = COAXIAL NETWORK UNIT
PCS = PHYSICAL CODING SUBLAYER
PHY = PHYSICAL LAYER DEVICE
PMA = PHYSICAL MEDIUM ATTACHMENT
PMD = PHYSICAL MEDIUM DEPENDENT

Figure 103–2—Relationship of Multipoint MAC Control and the OSI protocol stack for EPoC

103.1.2 Functional block diagram

Figure 103–3 provides a functional block diagram of the Multipoint MAC Control architecture.

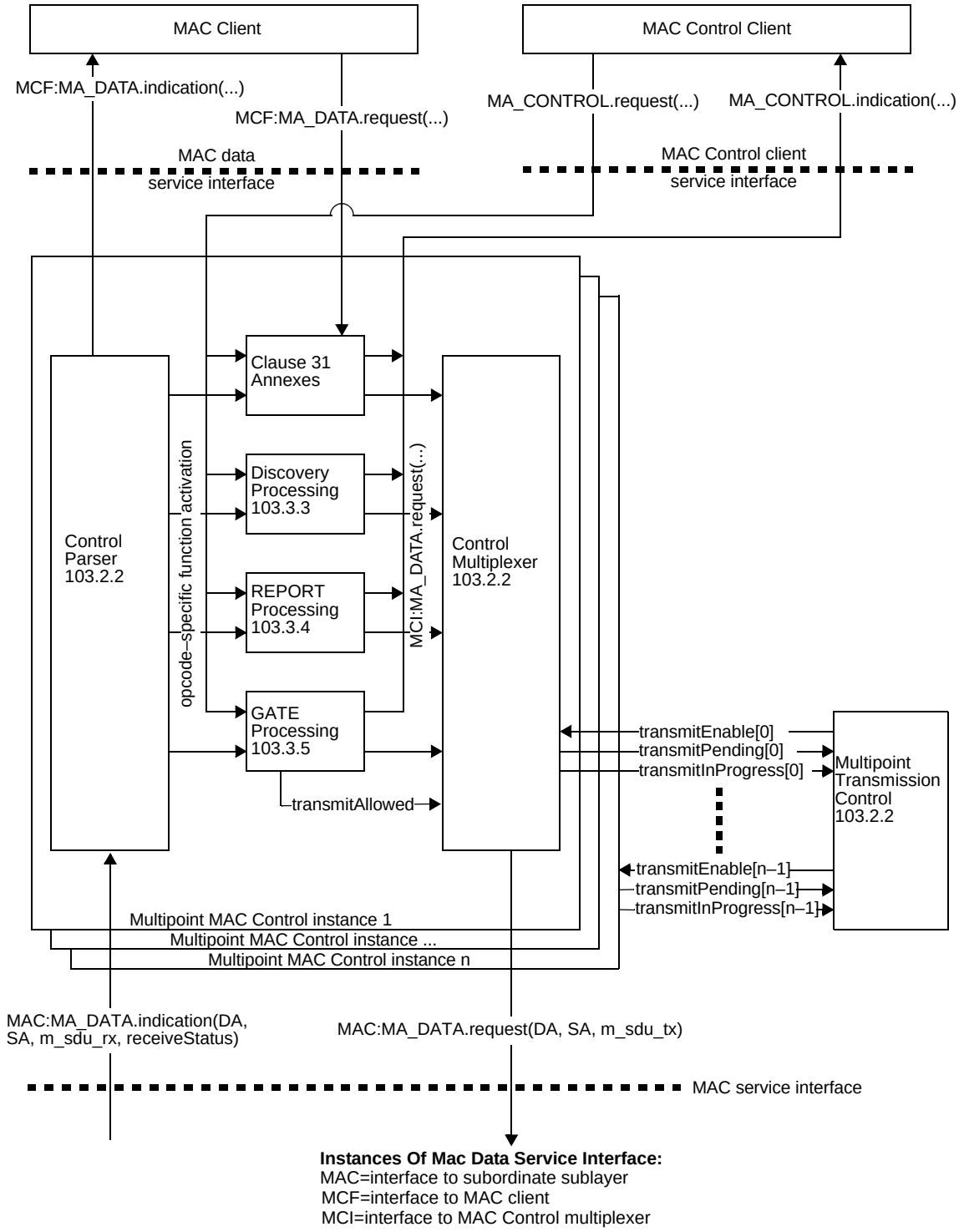


Figure 103–3—Multipoint MAC Control functional block diagram

103.1.3 Service interfaces

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

103.1.4 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. State diagram timers follow the conventions of 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 for bit ordering. When referring to an octet vector, 0 is used to mark the first received octet and so on (for example m_sdu[0..1]).

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

103.1.5 Other conventions

For equations used in this clause the symbol $\lceil x \rceil$ represents a ceiling function that rounds up its argument x to the next highest integer; in pseudo code listing the term ceiling() is used for this function. The symbol $\lfloor x \rfloor$ represents a floor function that rounds down its argument x to the next lowest integer.

103.2 Multipoint MAC Control operation

As depicted in Figure 103–3, the Multipoint MAC Control functional block contains functions very similar to those found in Clause 77. In EPoC the CLT replaces the OLT and the CNU replaces the ONU. Significant differences are noted in the following sections.

103.2.1 Principles of Multipoint MAC Control

The principles of Multipoint MAC Control are the same as those described in 77.2.1 for EPON.

103.2.1.1 Ranging and timing process

The ranging and timing processes for EPoC are the same as those described in 77.2.1.1 for EPON.

103.2.2 Multipoint transmission control, Control Parser, and Control Multiplexer

The purpose and high level functionality of multipoint transmission control is similar to those described in 77.2.2 for EPON. Detailed differences are noted in the definitions below and in Figure 103–3 through Figure 103–13.

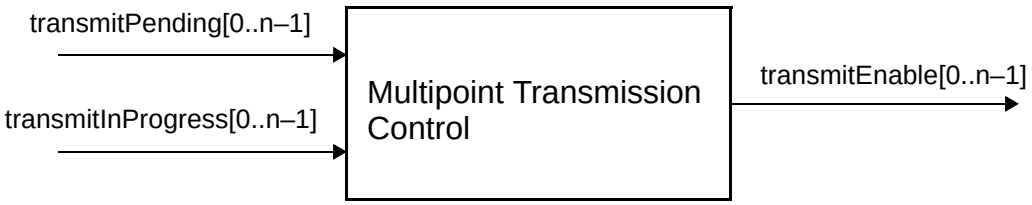


Figure 103–4—Multipoint Transmission Control service interfaces

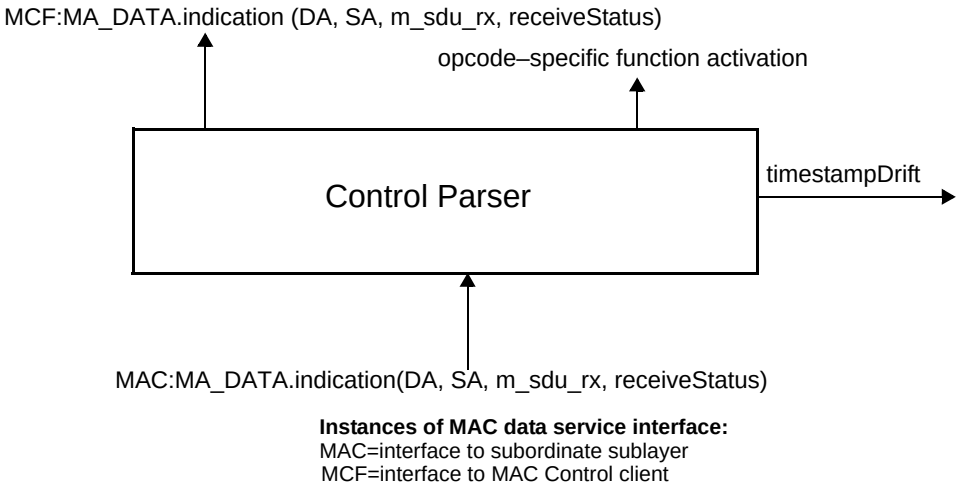
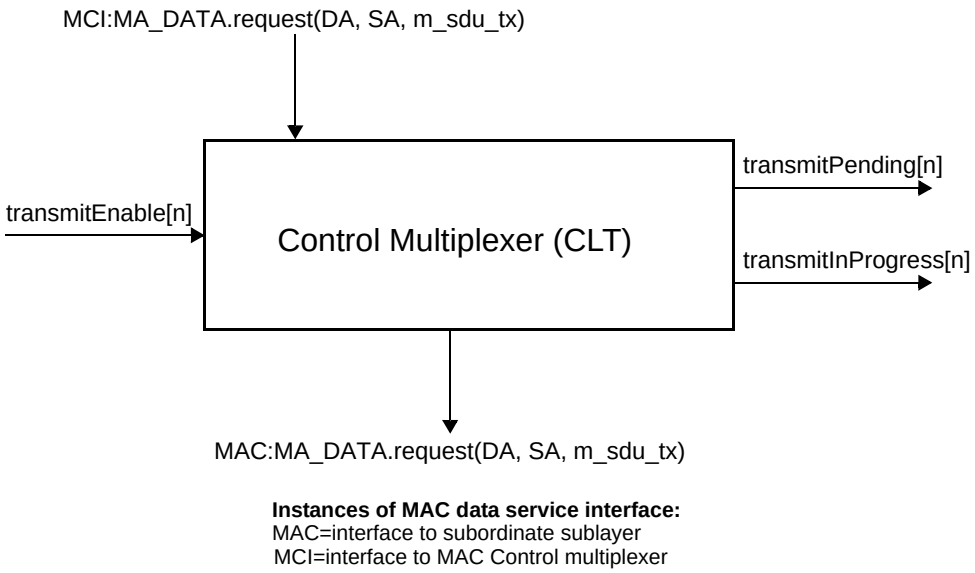
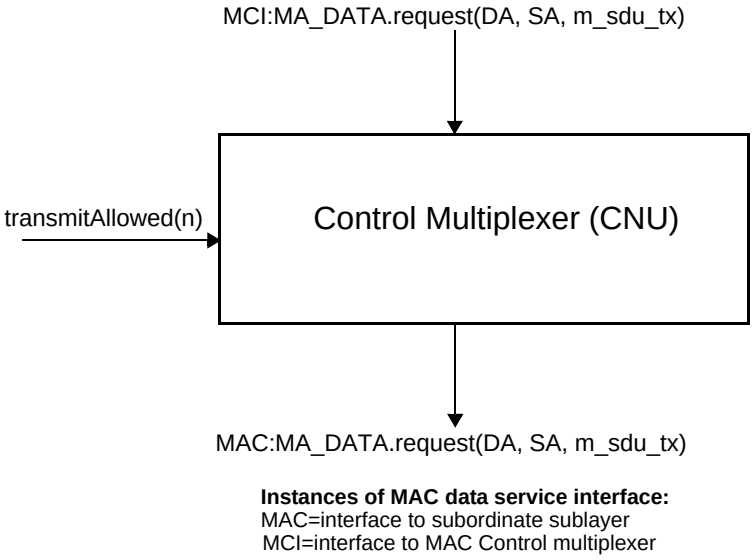


Figure 103–5—Control Parser service interfaces



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

Figure 103–6—CLT Control Multiplexer service interfaces



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

Figure 103–7—CNU Control Multiplexer service interfaces

103.2.2.1 Constants

DS_FEC_CW_Sz
TYPE: unsigned integer
This constant represents the integer number of octets in the FEC codeword and is equal to the sum of DS_FEC_Pld_Sz and DS_FEC_Prty_Sz..
VALUE: 1987

DS_FEC_CW_Sz_FRAC
TYPE: real number
This constant represents the exact size of the FEC codeword expressed in units of octets.
VALUE: 1760+2944/13 or 1760+(1840×64/65/8)

DS_FEC_Prty_Sz
TYPE: unsigned integer
This constant represents the size of FEC codeword parity field in units of octets.
VALUE: 227

DS_FEC_Pld_Sz
TYPE: unsigned integer
This constant represents the size of FEC codeword payload in units of octets.
VALUE: 1760 (220 block of 64 bits as seen from the MAC Table 101–2)

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

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

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

tailGuard
TYPE: unsigned integer
This constant holds the value used to reserve space at the end of the upstream transmission at the 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. The size of the minimum IPG is described in 4A.4.2.
VALUE: 38

time_quantum
This constant is defined in 64.2.2.1 and is 16 ns.

XGMII_Rate
TYPE: unsigned integer
See 101.3.2.1.1.

103.2.2.2 Counters

localTime
: TYPE: 32-bit unsigned integer
This counter is defined in 77.2.2.2 and holds the value of the local timer used to control MPCP operation.

103.2.2.3 Variables

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
: TYPE: bit array
This variable is defined in 77.2.2.3 and represents a 0-based bit array corresponding to the payload of a received MPCPDU.

data_tx
: TYPE: bit array
This variable is defined in 77.2.2.3 and represents a 0-based bit array corresponding to the payload of an MPCPDU being transmitted.

fecOffsetC
: TYPE: 32-bit unsigned integer
A variable that advances by one after every octet time. In the CLT, after reaching the value of *DS_FEC_CW_Sz*, this variable is held for a period of time for PMD derating and then reset to zero as illustrated in Figure 103–8. In the CNU, this variable is assigned in Figure 103–13.

fecPldSz
: TYPE: integer
fecPldSz is an alias for DS_FEC_Pld_Sz

grantStart
: TYPE: Boolean
This variable is defined in 77.2.2.3 and indicates beginning of a grant transmission.

IdleGapCount
: TYPE: 32-bit unsigned integer
This variable defined in 77.2.2.3 represents length of gap between subsequent frames, expressed in the unit of octet time and advances by 1 after every 8-bit times.

initial_derating_delay
: TYPE: 24 bit unsigned integer
This variable is used to set the time-out interval for *derating_timer* defined in 103.2.2.5 expressed in units of octets.

newRTT
: TYPE: 16-bit unsigned integer
This variable is defined in 77.2.2.3 and temporarily holds a newly-measured Round Trip Time.

m_sdu_rx
: TYPE: bit array
This varible is defined in 77.2.2.3 and is equal to the concatenation of the Length/Type and *data_rx* variables.

m_sdu_tx
: TYPE: bit array

This variable is defined in 77.2.2.3 and is equal to the concatenation of the Length/Type and *data_tx* variables.

m_sdu_ctl

TYPE: bit array
This varible is defined in 77.2.2.3 and is equal to the concatenation of the *MAC_Control_type* and *data_tx* variables.

Octet_CLK

TYPE: Boolean
This clear on read Boolean value is TRUE for every octet time period, i.e. the amount of time used to transmit one octet in 10Gb/s MAC data rate.

OctetsRemainingC

TYPE: 32-bit unsigned integer
This variable is an alias for the expression (((*stopTime* – *localTime*) × *tqSizeC*/128) – *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 integer
This variable is defined in 77.2.2.3 and represents a total transmission time of next packet and is used to check whether the next packet fits in the remainder of the transmission window. The value of *OctetsRequired* includes packet transmission time, *tailGuard* defined in 103.2.2.1, and FEC parity data overhead expressed in units of octets.

opcode_rx

TYPE: 16-bit unsigned integer
This variable is defined in 77.2.2.3 and holds an opcode of the last received MPCPDU.

opcode_tx

TYPE: 16-bit unsigned integer
This variable is defined in 77.2.2.3 and holds an opcode of an outgoing MPCPDU.

packet_initiate_delay

TYPE: 18-bit unsigned integer
This variable is defined in 77.2.2.3 and used to set the time-out interval for *packet_initiate_timerC* defined in 103.2.2.5 expressed in units of octets.

PCS_Rate

See 101.3.2.1.2 and Equation (101–2).

ResetBoundC

TYPE: 32-bit unsigned integer
This variable represents the value of *DelayBound* (see 101.3.2.1.2) expressed in units of octet time (i.e., *ResetBoundC* = 8 × *DelayBound*).

RTT

TYPE: 16-bit unsigned integer
This variable is defined in 77.2.2.3 and holds the measured Round Trip Time to the CNU expressed in units of time_quanta.

stopTime

TYPE: 32-bit unsigned integer
This variable is defined in 77.2.2.3 and holds the value of the *localTime* counter corresponding to the end of the nearest grant.

timestamp

TYPE: 32-bit unsigned integer

This variable is defined in 77.2.2.3 and holds the value of *timestamp* of the last received MPCPDU frame.

timestampDrift
TYPE: Boolean
This variable is defined in 77.2.2.3 and used to indicate whether an error is signaled as a result of uncorrectable *timestamp* drift.

tqOffset
TYPE: 8-bit unsigned integer
This variable is defined in 77.2.2.3 and denotes the offset (in octet times) of the current actual time from the *localTime* variable.

tqSizeC
TYPE: unsigned integer
This variable represents the number of octets that can be transmitted in 128 *time_quantum* units, i.e., 2048 ns.

transmitAllowed
TYPE: Boolean
This variable is defined in 77.2.2.3 and used to control PDU transmission at the CNU.

transmitEnable
TYPE: Boolean array
This array is defined in 77.2.2.3 and used to control the transmit path in the Multipoint MAC Control instance at the CLT.

transmitInProgress
TYPE: Boolean array
This array is defined in 77.2.2.3 and indicates that the Multipoint MAC Control instance is in a process of transmitting a frame.

transmitPending
TYPE: Boolean array
This array is defined in 77.2.2.3 and indicates that a Multipoint MAC Control instance is ready to transmit a frame.

103.2.2.4 Functions

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

Opcode–specific function(opcode)
These functions are defined in 77.2.2.4 and exported from opcode specific blocks that are invoked on the arrival of a MAC Control message of the appropriate opcode.

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

```
Function CheckGrantSizeC( length ){
   GntSize = 0;
   length += fecOffsetC;
   while (length > fecPldSz[0]) {      // Calc # of full Lg CW's
      length -= fecPldSz[0];
      GntSize += DS_FEC_CW_Sz_FRAC[0];
   }
   // Calc remaining 1 part Lg CW <OR> 2 Med CW
   // (2 full <OR> 1 full & 1 part) <OR> 1 Sm CW (full <OR> part)
```

```
    while (length > 0) {
       if (length > 6464) {
          GntSize += length + ceiling(length/64) + fecPrtySz[0];
       }
       else if (length >= fecPldSz[1]) {   // full Med CW
          GntSize += DS_FEC_CW_Sz_FRAC[1];
       }
       else if (length >= 1600) {          // partial Med CW
       GntSize += length + ceiling(length/64) + fecPrtySz[1];
       }
       else if (length >= fecPldSz[2]) {   // full Small CW
          GntSize += DS_FEC_CW_Sz_FRAC[2];
       }
       else {                              // partial Small CW
          GntSize += length + ceiling(length/64) + fecPrtySz[2];
       }
    }
    return GntSize – fecOffsetC;
 }
```

PHY_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 and to adjust the data rate to the effective data rate supported by the PCS and PMD. PHY_Overhead() returns the number of octets that the PHY inserts during transmission of a particular packet. Parameter *length* represents the size of an entire frame including preamble, SFD, DA, SA, Length/Type, FCS, and IPG which are accounted for in *tailGuard*. The following formula is used to calculate the overhead:

$$PMD\_Overhead(length) =$$

$$FEC\_Overhead(length) + Derating\_Overhead(length, \frac{XGMII\_Rate}{PCS\_Rate})$$

$$FEC\_Overhead(length) = 12 + DS\_FEC\_Prty\_Sz \times \left\lfloor \frac{fecOffset + length}{DS\_FEC\_Pld\_Sz} \right\rfloor$$

$$Derating\_Overhead(length) =$$

$$\left\lceil \left(\frac{XGMII\_Rate}{PCS\_Rate} - 1 \right) \times DS\_FEC\_CW\_Sz\_FRAC \times \left\lfloor \frac{fecOffset + length}{DS\_FEC\_Pld\_Sz} \right\rfloor \right\rceil$$

select()

This function is defined in 77.2.2.4 and selects the next Multipoint MAC Control instance allowed to initiate transmission of a frame. The function returns an index to the *transmitPending* array for which the value is not FALSE. The selection criteria in the presence of multiple active elements in the list is implementation-dependent.

SelectFrame()

This function is defined in 77.2.2.4 and 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)
: This function is defined in 77.2.2.4 and returns the size of the sdu in octets.

transmissionPending()
: This function is defined in 77.2.2.4 and returns TRUE if any of the Multipoint MAC Control instances has a frame waiting to be transmitted.

103.2.2.5 Timers

derating_timer:
: This timer is used to suspend the advancing of *fecOffsetC* in order to derate the MAC frame transmission to be able to match the PMD rate.

packet_initiate_timerC
: This timer is used to delay frame transmission from MAC Control to avoid variable MAC delay while MAC enforces IPG after a previous frame and is defined in 77.2.2.5.

103.2.2.6 Messages

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

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

103.2.2.7 State diagrams

The Multipoint transmission control function in the CLT shall implement state diagrams shown in Figure 103–8 and Figure 103–9. Control parser function in the CLT shall implement state diagram shown in Figure 103–10. Control parser function in the CNU shall implement state diagram shown in Figure 103–11. Control multiplexer function in the CLT shall implement state diagram shown in Figure 103–12. Control multiplexer function in the CNU shall implement state diagram shown in Figure 103–13.

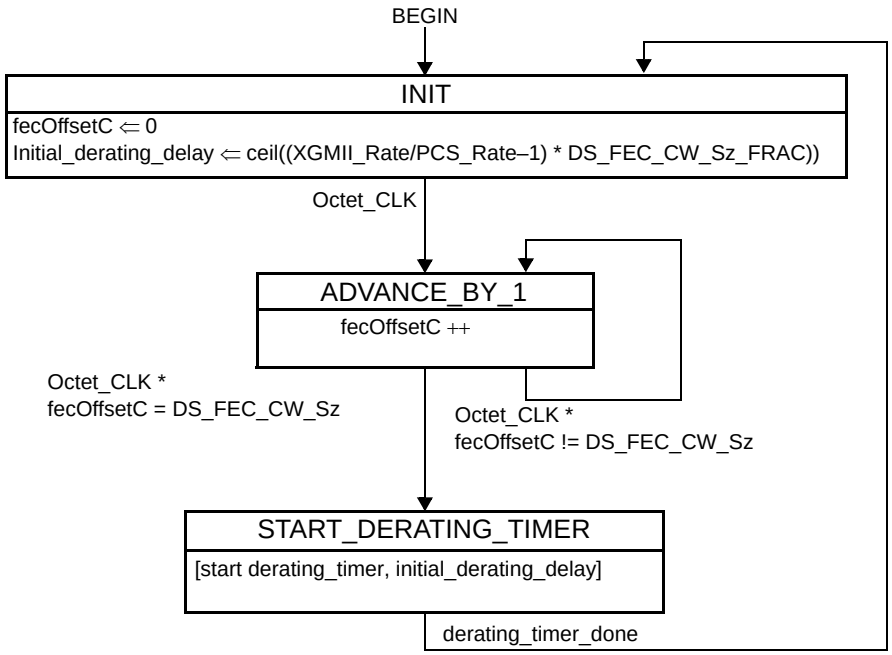


Figure 103–8—CLT fecOffsetC state diagram

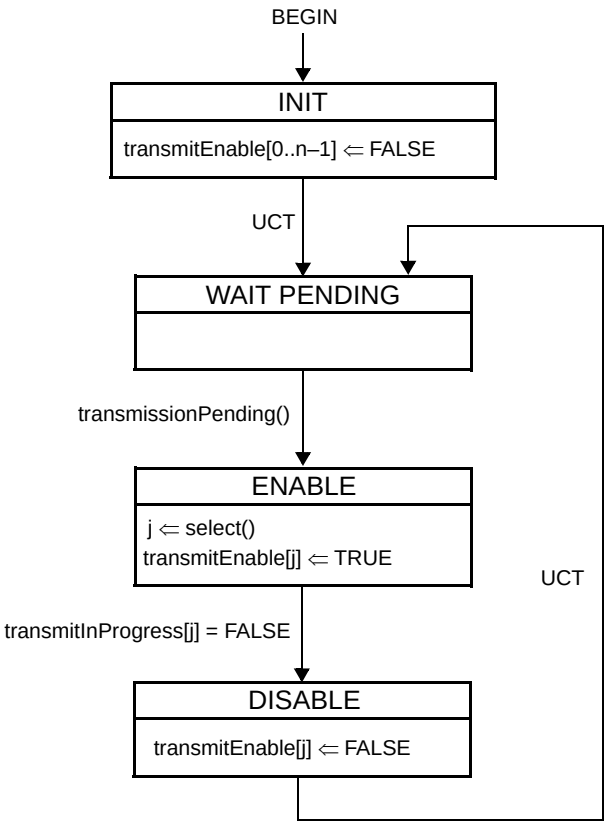
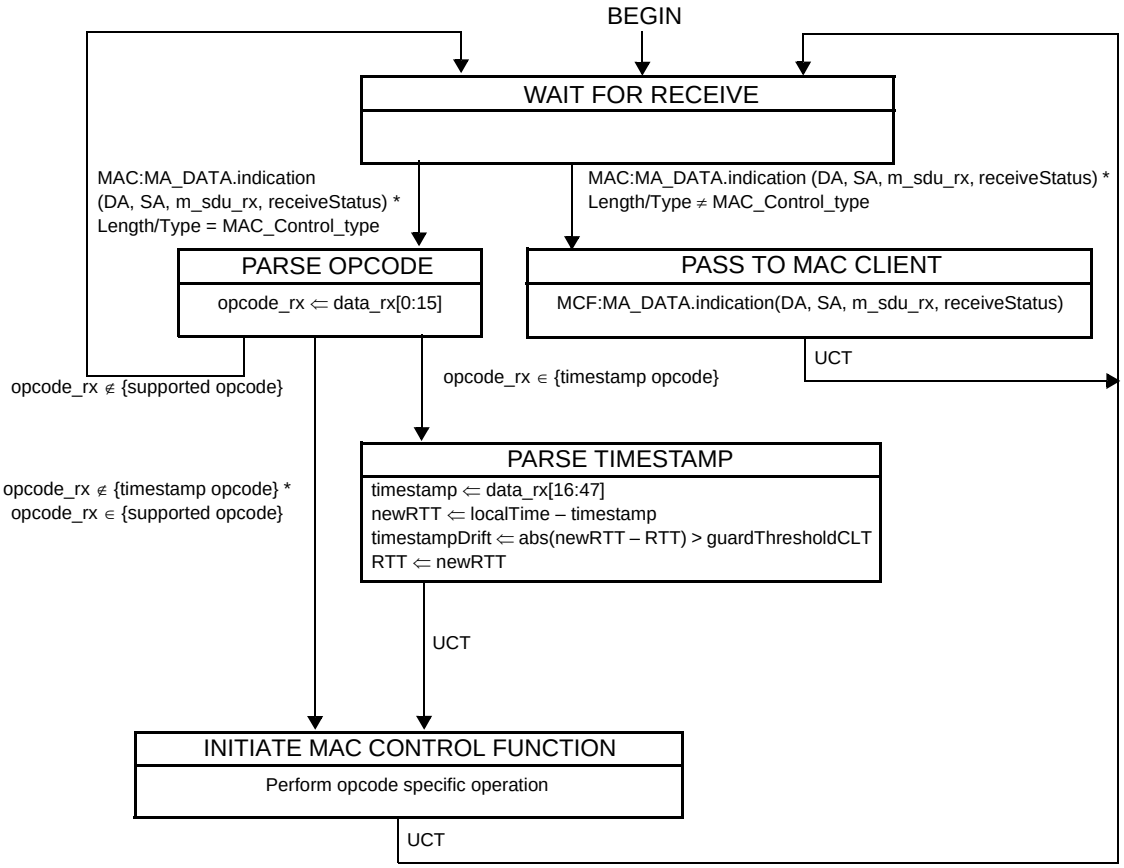


Figure 103–9—CLT Multipoint Transmission Control state diagram

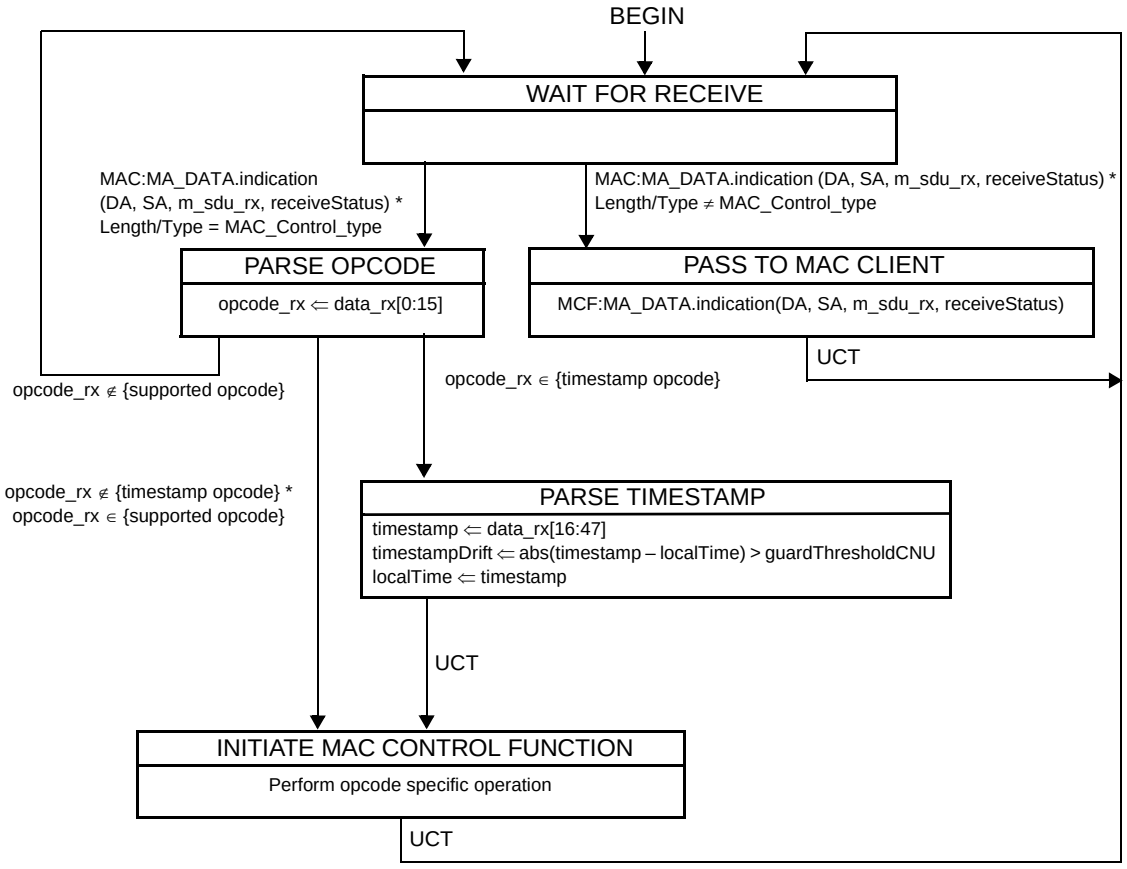


Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

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 103–10—CLT Control Parser state diagram

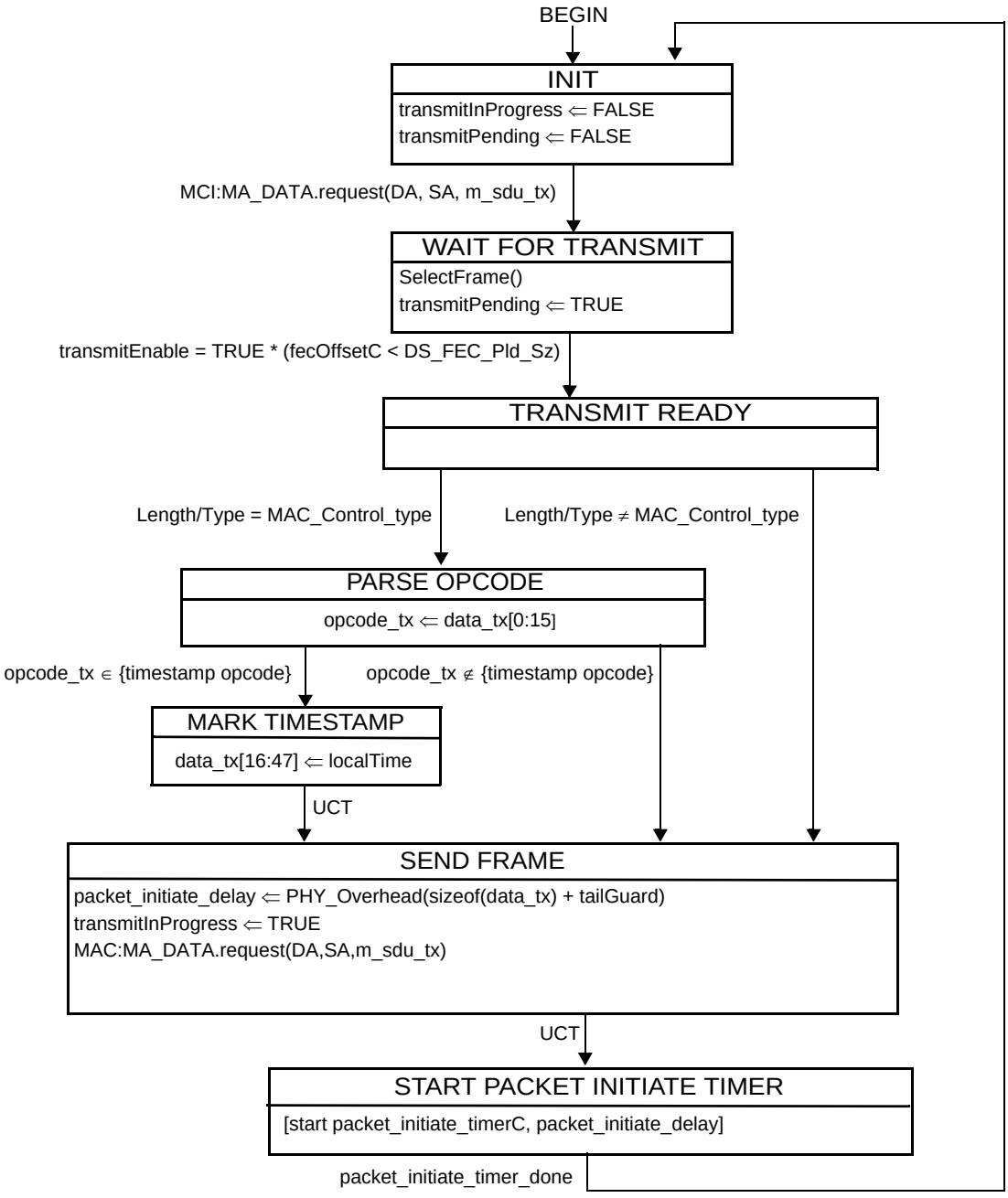


Instances of MAC data service interface:
MAC=interface to subordinate sublayer
MCF=interface to MAC Control client

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.

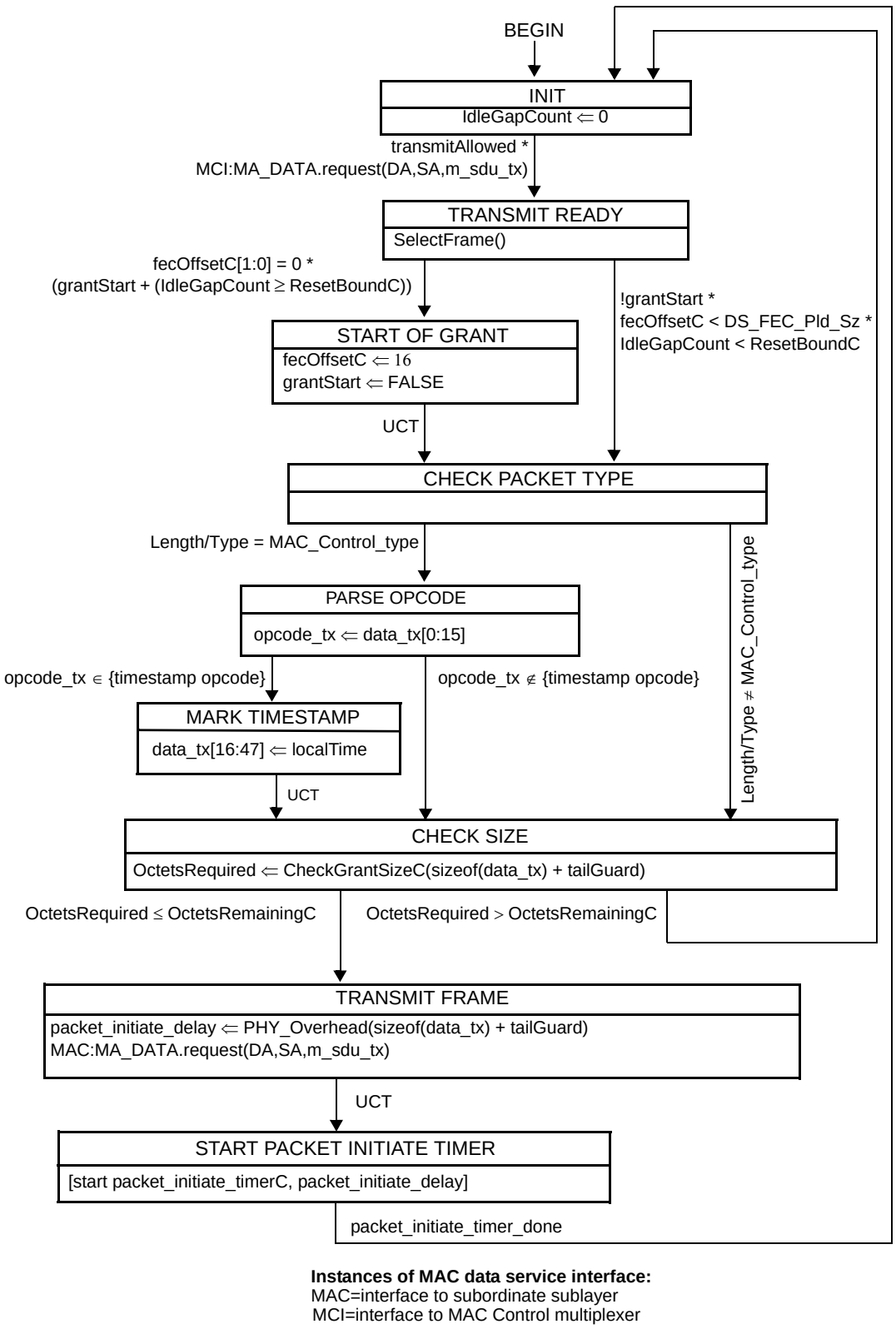
Refer to Annex 31A for list of supported opcodes and timestamp opcodes.

Figure 103–11—CNU Control Parser state diagram



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

Figure 103–12—CLT Control Multiplexer state diagram



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

Figure 103–13—CNU Control Multiplexer state diagram

103.3 Multipoint Control Protocol (MPCP)

As depicted in Figure 103–3, the Multipoint MAC Control functional block comprises nearly the same functions and layering system as that described in 77.3. In EPoC the CLT replaces the OLT and the CNU replaces the ONU. All other significant differences are noted in the following sections.

103.3.1 Principles of Multipoint Control Protocol

The principles of the Multipoint Control Protocol are the same as those found in 77.3.1 except the EPoC system uses an Orthogonal Frequency Division Multiple Access (OFDMA) method in the upstream direction. In EPON the Multipoint Control Protocol, one and only one MAC is allowed to transmit at any given time. In EPoC the Multipoint Control Protocol allows multiple MACs to transmit in any given time but coincident transmitters are separated in frequency.

103.3.2 Compatibility considerations

103.3.2.1 PAUSE operation

See 77.3.2.1 and time constraints found at 103.3.2.4.

103.3.2.2 Optional Shared LAN emulation

Optional Shared LAN emulation for EPoC is the same as described in 77.3.2.2 except the specific behavior of the filtering layer at the RS is specified in 76.2.6.1.3.2.

103.3.2.3 Multicast and single copy broadcast support

Multicast and single copy broadcast support in EPoC is the same as described in 77.3.2.3 except the 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.

103.3.2.4 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 CLT shall not issue a gate message more than 1024 time_quanta into the future. The CNU shall process all messages in less than this period. The CLT shall not issue more than one message every 1024 time_quanta to a single CNU. The unit of *time_quantum* is defined in 64.2.2.1. The CLT shall ensure that a minimum gap time between bursts from any two CNUs equal to *RB_time_quanta* (see Equation (101–33)).

103.3.3 Discovery processing

Discovery processing in the EPoC is the same as described in 77.3.3 with the following exceptions. In EPoC, CNUs that have not completed PHY Discovery process (see 102.4.1) will not respond to Discovery GATE MPCPDUs. In the EPoC coax cable distribution network only one upstream data rate is allowed for a given configuration. The *laserOnTime* and *laserOffTime* parameters in 77.3.3 are replaced in EPoC with *rfOnTime* and *rfOffTime*, respectively.

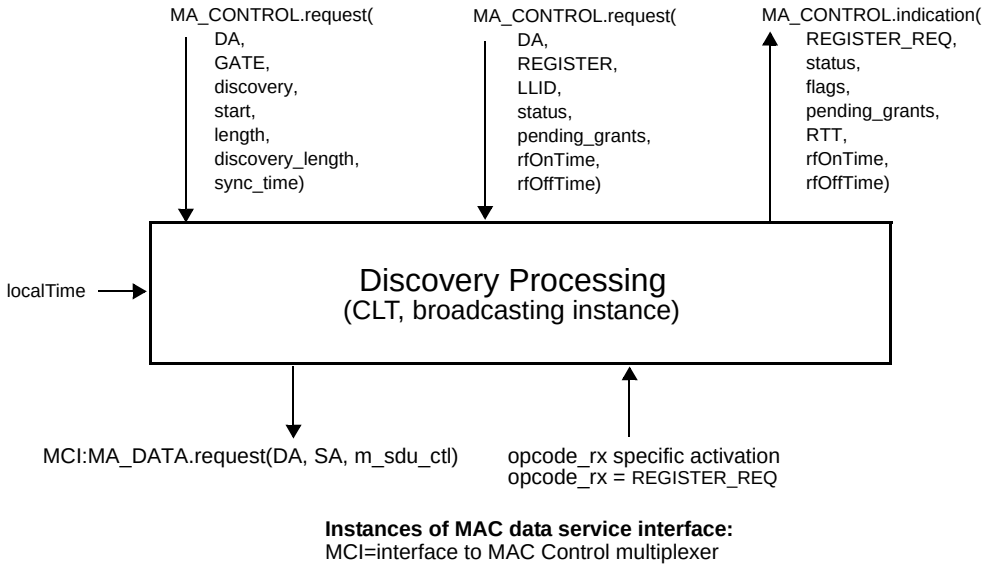


Figure 103–14—Discovery Processing service interfaces (CLT, broadcast instance)

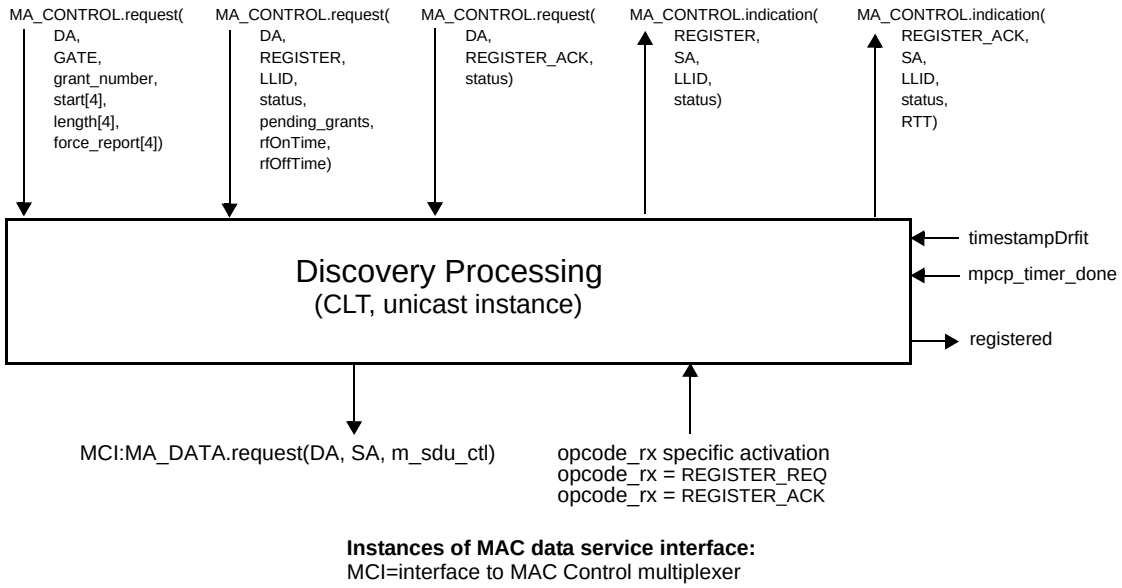


Figure 103–15—Discovery Processing service interfaces (CLT, unicast instance)

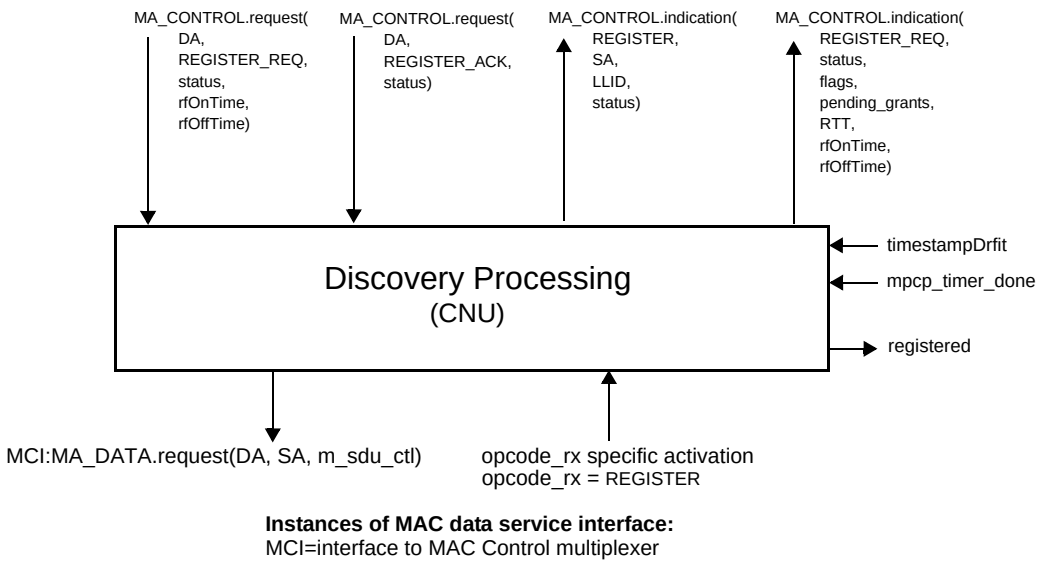


Figure 103–16—Discovery Processing service interfaces (CNU)

103.3.3.1 Constants

rfOffTime

TYPE: 8-bit unsigned integer
This variable holds the time required to terminate the RF and is included for consistency with Clause 77.
VALUE: zero

rfOnTime

TYPE: 8-bit unsigned integer
This variable holds the time required to initiate the PMD and is included for consistency with Clause 77.
VALUE: zero

103.3.3.2 Variables

BEGIN

This variable is defined in 103.2.2.3.

data_rx

This variable is defined in 103.2.2.3.

data_tx

This variable is defined in 103.2.2.3.

grantEndTime

TYPE: 32-bit unsigned integer
This variable is defined in 77.3.3.2 and holds the time at which the CLT expects the CNU grant to complete.

insideDiscoveryWindow

TYPE: Boolean
This variable is defined in 77.3.3.2 and holds the current status of the discovery window.

localTime
: This variable is defined in 103.2.2.2.

m_sdu_ctl
: This variable is defined in 103.2.2.3.

opcode_rx
: This variable is defined in 103.2.2.3.

pendingGrants
: TYPE: 16-bit unsigned integer
This variable is defined in 77.3.3.2 and holds the maximum number of pending grants that a CNU is able to queue.

registered
: TYPE: Boolean
This variable is defined in 77.3.3.2 and holds the current result of the Discovery Process.

RB_time_quanta
: see Equation (101–33).

syncTime
: TYPE: 16-bit unsigned integer
This variable is defined in 77.3.3.2 and holds the time required to stabilize an EPON receiver at the OLT (see 76.3.2.5.3). The EPoC CLT OFDMA receiver is synchronized and stablized during PHY Discovery and does not use a synchronization preamble as part of the upstream burst (see Figure 101–10). This variable is present to maintain compatibility with the EPON MPCP and always has a value of zero in EPoC PHYs.

timestampDrift
: This variable is defined in 103.2.2.3.

103.3.3.3 Timers

discovery_window_size_timer
: This timer is defined in 77.3.3.4 and used to wait for the event signaling the end of the discovery window.

mpcp_timer
: This timer is defined in 77.3.3.4 and used to measure the arrival rate of MPCP frames in the link.

103.3.3.4 Messages

MA_DATA.indication(DA, SA, m_sdu, receiveStatus)
: This service primitive is defined in 2.3.2.

MA_DATA.request (DA, SA, m_sdu)
: This service primitive is defined in 2.3.2.

MA_CONTROL.request(DA, GATE, discovery, start, length, discovery_length, sync_time)
: This service primitive is used by the MAC Control client at the 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.
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 CLT.

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 CLT to issue the GATE message to a CNU. This primitive takes the following parameters:

DA: Multicast MAC Control address as defined in Annex 31B.
GATE: Opcode for GATE MPCPDU as defined in Table 31A–1.
grant_number: Number of grants issued with this GATE message. The number of grants ranges from 0 to 4.
start[4]: Start times of the individual grants. Only the first grant_number elements of the array are used.
length[4]: Lengths of the individual grants. Only the first grant_number elements of the array are used.
force_report[4]: Flags indicating whether a REPORT message should be generated in the corresponding grant. Only the first grant_number elements of the array are used.

MA_CONTROL.request(DA,REGISTER_REQ,status,*rfOnTime*,*rfOffTime*)

This service primitive is used by a client at the 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.
REGISTER_REQ: opcode for REGISTER_REQ MPCPDU as defined in Table 31A–1.
status: This parameter takes on the indication supplied by the flags field in the REGISTER_REQ MPCPDU as defined in Table 77-3.
rfOnTime: This parameter holds the *rfOnTime* value, expressed in units of time_quanta, as reported by MAC client and specified in 103.3.6.3.
rfOffTime: This parameter holds the *rfOffTime* value, expressed in units of time_quanta, as reported by MAC client and specified in 103.3.6.3.

MA_CONTROL.indication(REGISTER_REQ, status, flags, pending_grants, *RTT*, *rfOnTime*, *rfOffTime*)

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.
status: This parameter holds the values incoming or retry. Value incoming is used at the CLT to signal that a REGISTER_REQ message was received successfully. The value retry is used at the 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 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 CLT.
RTT: The measured round trip time to/from the 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 CLT.
rfOnTime: This parameter holds the contents of the *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 CLT.
rfOffTime: This parameter holds the contents of the *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 CLT.

MA_CONTROL.request(DA, REGISTER, LLID, status, pending_grants, *rfOnTime*, *rfOffTime*)

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

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

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

LLID: This parameter holds the logical link identification number assigned by the MAC Control client.

status: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in Table 77-4.

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

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

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

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

This service primitive is issued by the Discovery Process at the CLT or a 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.

SA: This parameter represents the MAC address of the 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)

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

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

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

status: This parameter takes on the indication supplied by the flags field in the REGISTER MPCPDU as defined in Table 77-5.

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

This service primitive is issued by the Discovery Process at the 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.

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

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.

103.3.3.5 State Diagrams

The Discovery Process in the CLT shall implement the discovery window setup state diagram shown in Figure 103–17, request processing state diagram as shown in Figure 103–18, register processing state diagram as shown in Figure 103–19, and final registration state diagram as shown in Figure 103–20. The discovery process in the CNU shall implement the registration state diagram as shown in Figure 103–21.

Instantiation of state diagrams as described in Table 103–17, Table 103–18, and Table 103–19 is performed only at the Multipoint MAC Control instances attached to the broadcast LLID (0x7FFE). Instantiation of state diagrams as described in Table 103–20 and Table 103–21 is performed for every Multipoint MAC Control instance, except the instance attached to the broadcast channel.

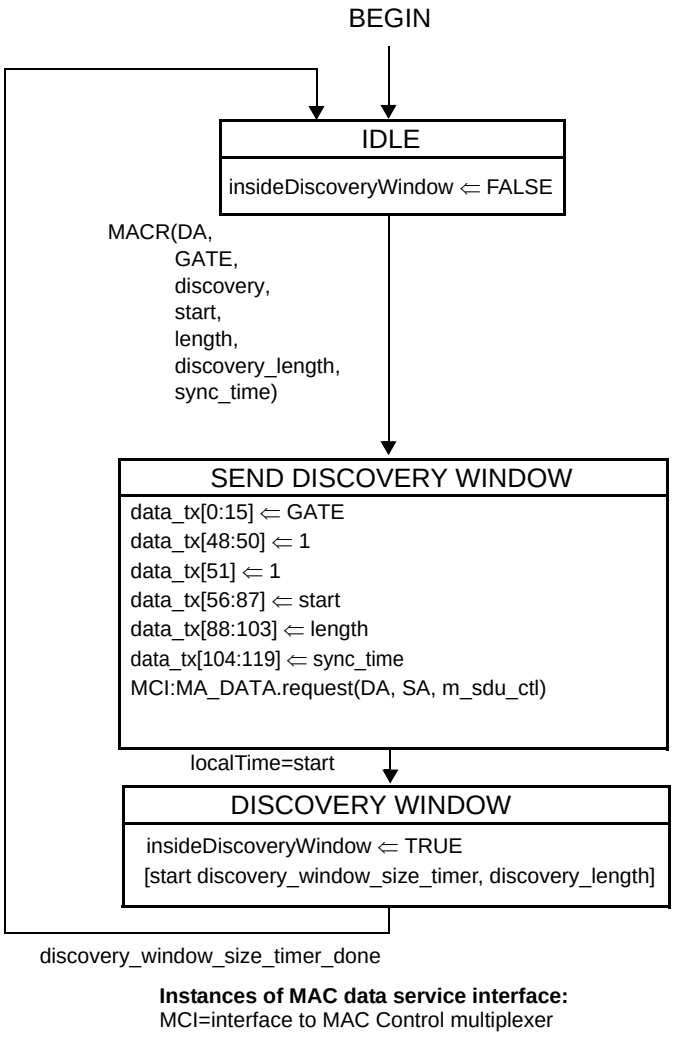


Figure 103–17—Discovery Processing CLT Window Setup state diagram

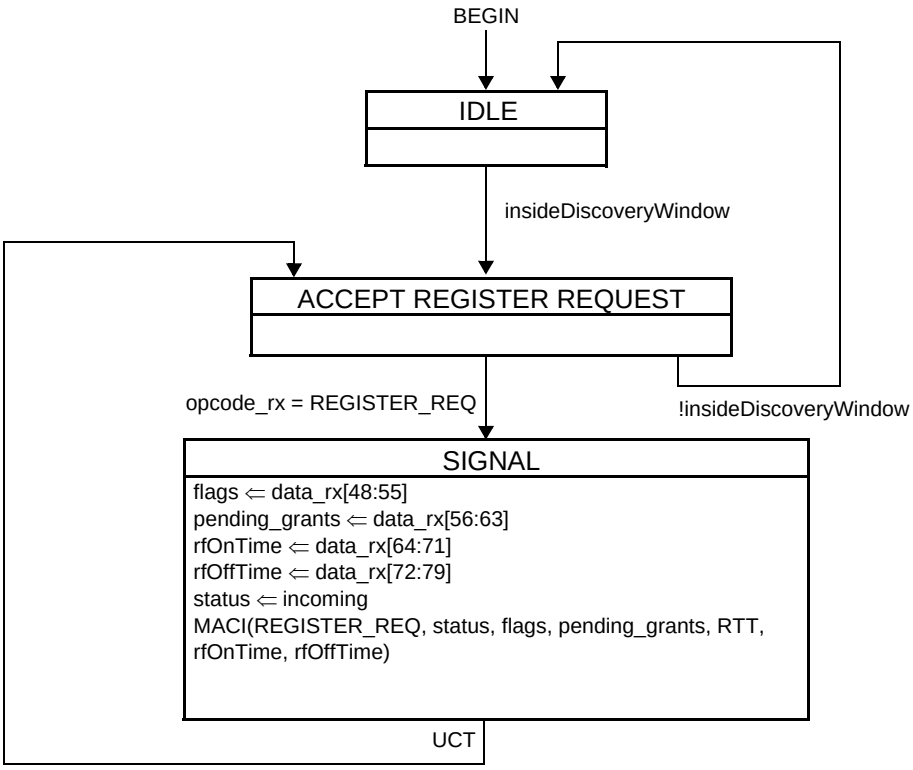
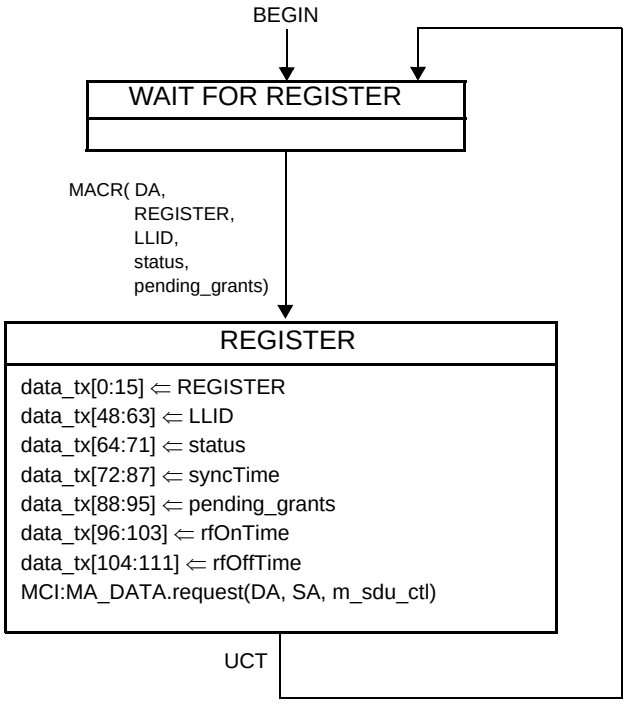
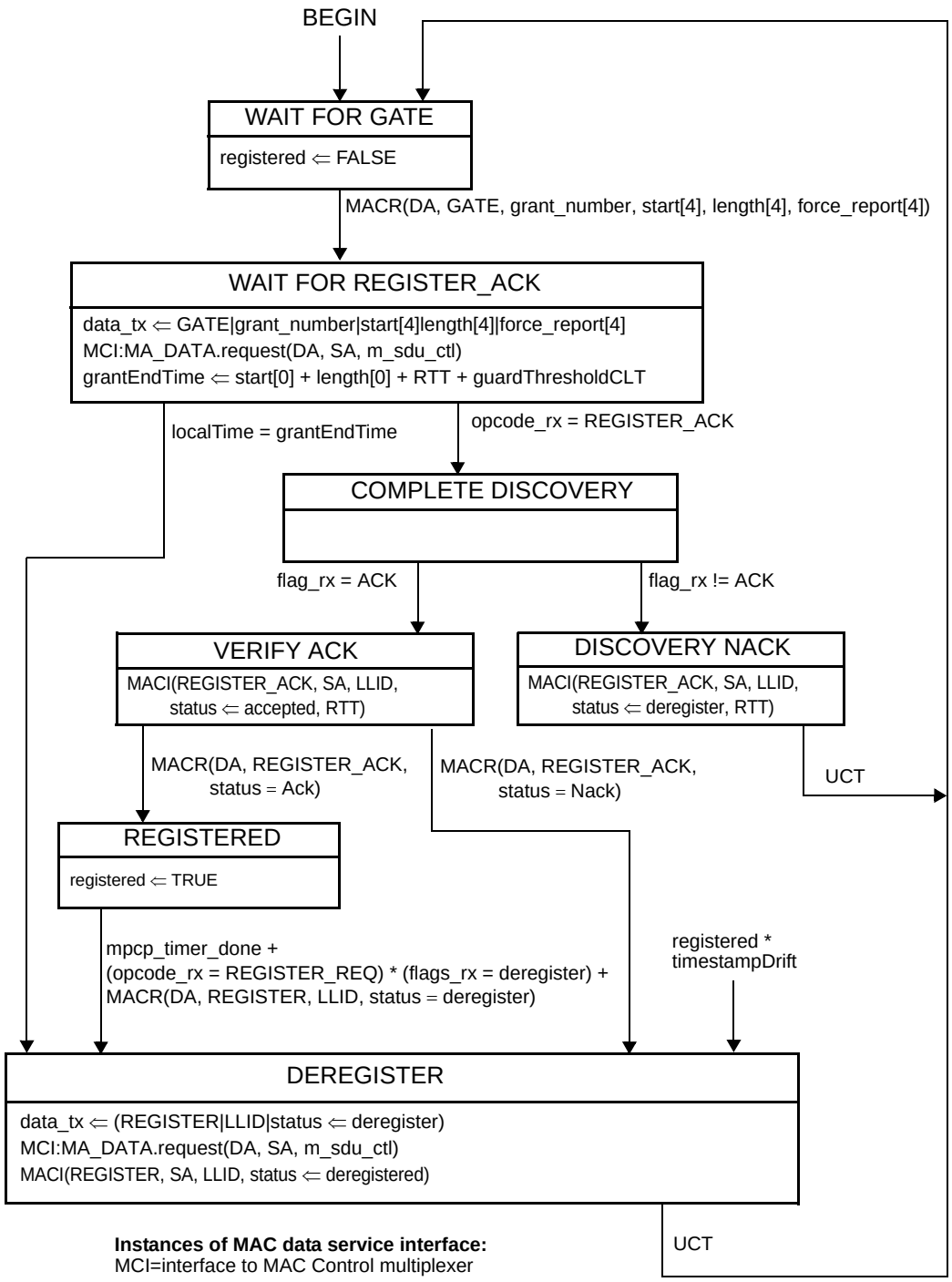


Figure 103–18—Discovery Processing CLT Process Requests state diagram



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

Figure 103–19—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 103–20—Discovery Processing CLT Final Registration state diagram

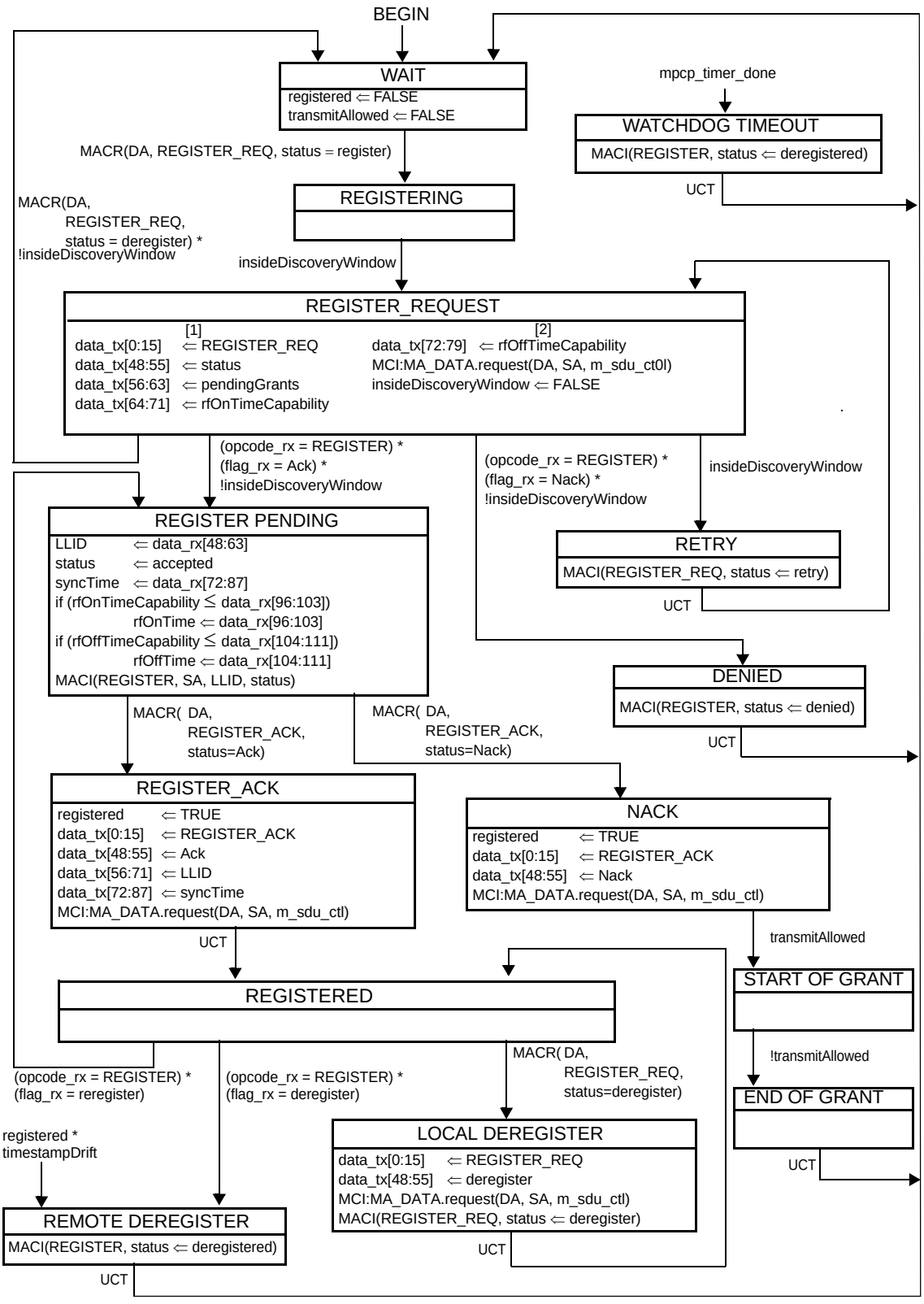


Figure 103–21—Discovery Processing CNU Registration state diagram

103.3.4 Report Processing

Report processing in EPoC is as described in 77.3.4.

103.3.5 Gate Processing

Gate processing in EPoC is as described in 77.3.5 with the exception being that EPoC uses an RF transmitter rather than a laser.

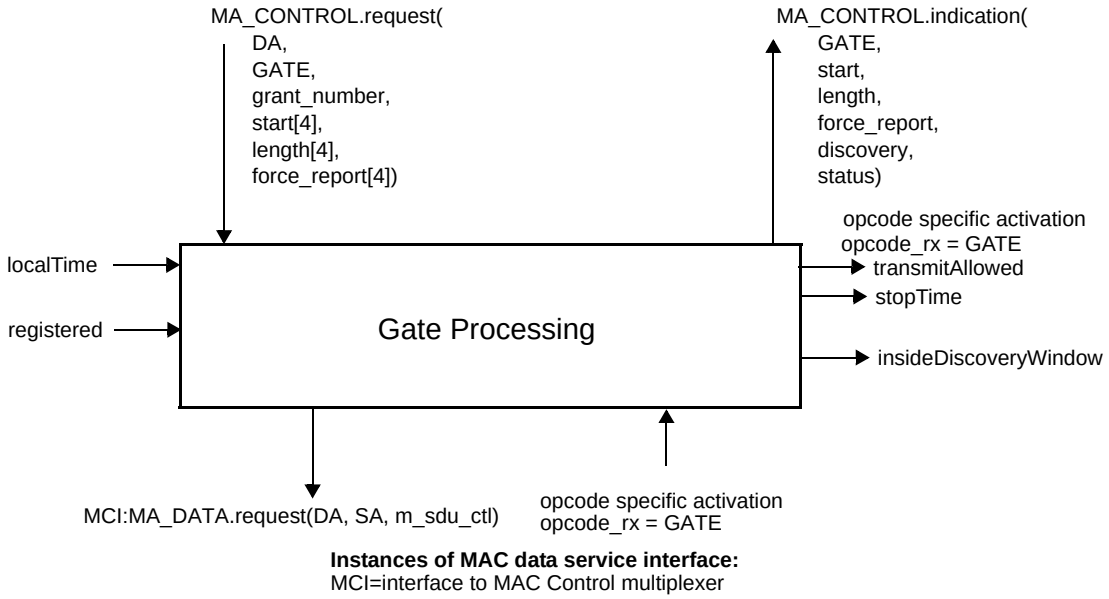


Figure 103–22—Gate Processing service interface

103.3.5.1 Constants

max_future_grant_time
: TYPE: 32-bit unsigned integer
This constant is defined in 77.3.5.1 and holds the time limiting the future time horizon for a valid incoming grant.
VALUE: 0x03-B9-AC-A0 (1 s)

min_processing_time
: TYPE: 32-bit unsigned integer
This constant is defined in 77.3.5.1 and holds the time required for the CNU processing.
VALUE: 1024

103.3.5.2 Variables

BEGIN
: This variable is defined in 103.2.2.3.

BurstOverhead
: TYPE: integer
This variable represents the burst overhead and equals the sum of *BurstTimeHeader()* (see

101.3.2.5.7) and two leading IDLE vectors of the payload. This variable is expressed in units of time_quanta.

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

}

This variable is defined in 77.3.5.2 and 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.

data_rx

This variable is defined in 103.2.2.3.

data_tx

This variable is defined in 103.2.2.3.

effectiveLengthC

TYPE: 32-bit unsigned integer
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.

gate_timeout

TYPE: 32-bit unsigned integer
This variable is defined in 77.3.5.2 and represents the maximum allowed interval of time between two GATE messages generated by the CLT to the same CNU, expressed in units of time_quanta.
VALUE: 0x00-2F-AF-08 (50 ms, default value)

grantList

TYPE: list of elements having the structure define in *currentGrant*
This variable is defined in 77.3.5.2 and 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

This variable is defined in 103.2.2.3.

insideDiscoveryWindow

This variable is defined in 103.3.3.2.

m_sdu_ctl

This variable is defined in 103.2.2.3.

maxDelay

TYPE: 16-bit unsigned integer
This variable is defined in 77.3.5.2 and holds the maximum delay that can be applied by a

CNU before sending the REGISTER_REQ MPCPDU. This delay is calculated such that the 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 RF before the end of the discovery grant.

minGrantLengthC

TYPE: 32-bit unsigned integer
This variable represents the minimum data portion of a grant expressed in units of time_quanta. The *minGrantLengthC* is related to the minimum size Ethernet frame plus EPON overhead of 72 bytes (576 bits) processed as per 101.3.2.5.2 using a short codeword plus two IDLE blocks and one block for Burst Time Header. This results in a shortened codeword length of 1091 bits. The minimum grant length will include stop and end burst marker overhead based on *RBsize* and *RB_time_quanta* (see Equation (101–33)):

Number_of_Burst_RBs is 8 for *RBsize* = FALSE or 4 for *RBsize* = TRUE

RB_total_time = *RB_time_quanta* * *Number_of_Burst_RBs*.

The value of *minGrantLengthC* is based on the time_quanta required for the number of Resource Blocks needed to transmit the 1091 data bits based on *US_DataRate* plus *RB_total_time*:

Data_TQ = 1091 / *US_DataRate* / *time_quantum*

minGrantLengthC = ceiling(*Data_TQ* / *RB_time_quanta*) * *RB_time_quanta* + *RB_total_time*

nextGrant

TYPE: element having same structure as defined in *currentGrant*
This variable is defined in 77.3.5.2 and 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 integer
This variable is defined in 77.3.5.2 and holds the value of the *localTime* counter corresponding to the end of the next grant.

opcode_rx

This variable is defined in 103.2.2.3.

registered

This variable is defined in 103.3.3.2.

stopTime

This variable is defined in 103.2.2.3.

syncTime

This variable is defined in 103.3.3.2.

transmitAllowed

This variable is defined in 103.2.2.3.

103.3.5.3 Functions

empty(list)

This function is defined in 77.3.5.3 and used 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.

InsertInOrder(sorted_list, inserted_element)

This function is defined in 77.3.5.3 and used to queue an element inside a sorted list. The queu-

ing 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)
: This function is defined in 77.3.5.3 and used to check whether its argument represents a broadcast grant, i.e., grant given to multiple 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, and FALSE otherwise.

PeekHead(sorted_list)
: This function is defined in 77.3.5.3 and used to check the content of a sorted list. It returns the element at the head of the list without dequeuing the element.

Random(r)
: This function is defined in 77.3.5.3 and used to compute a random integer number uniformly distributed between 0 and *r*. The randomly generated number is then returned by the function.

RemoveHead(sorted_list)
: This function is defined in 77.3.5.3 and used to dequeue an element from the head of a sorted list. The return value of the function is the dequeued element.

103.3.5.4 Timers

gntWinTmr
: This timer is defined in 77.3.5.4 and 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
: This timer is defined in 77.3.5.4 and counts down time remaining before a forced generation of a GATE message in the CLT. The CLT is required to generate GATE MPCPDUs with a periodicity of less than *gate_timeout* value.

mpcp_timer
: This timer is defined in 103.3.3.3.

rndDlyTmrC
: 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.
VALUE: A random value less than the net discovery window size less the REGISTER_REQ MPCPDU frame size less the idle period and 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.

103.3.5.5 Messages

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

MA_CONTROL.request(DA, GATE, grant_number, start[4], length[4], force_report[4])
: This service primitive is defined in 103.3.3.4.

MA_CONTROL.indication(GATE, start, length, force_report, discovery, status)
: This service primitive issued by the Gate Process at the CNU to notify the MAC Control client and higher layers that a grant is pending. This primitive is invoked multiple times when a sin-

gle GATE message arrives with multiple grants. It is also generated at the start and end of each grant as it becomes active. This primitive uses the following parameters:

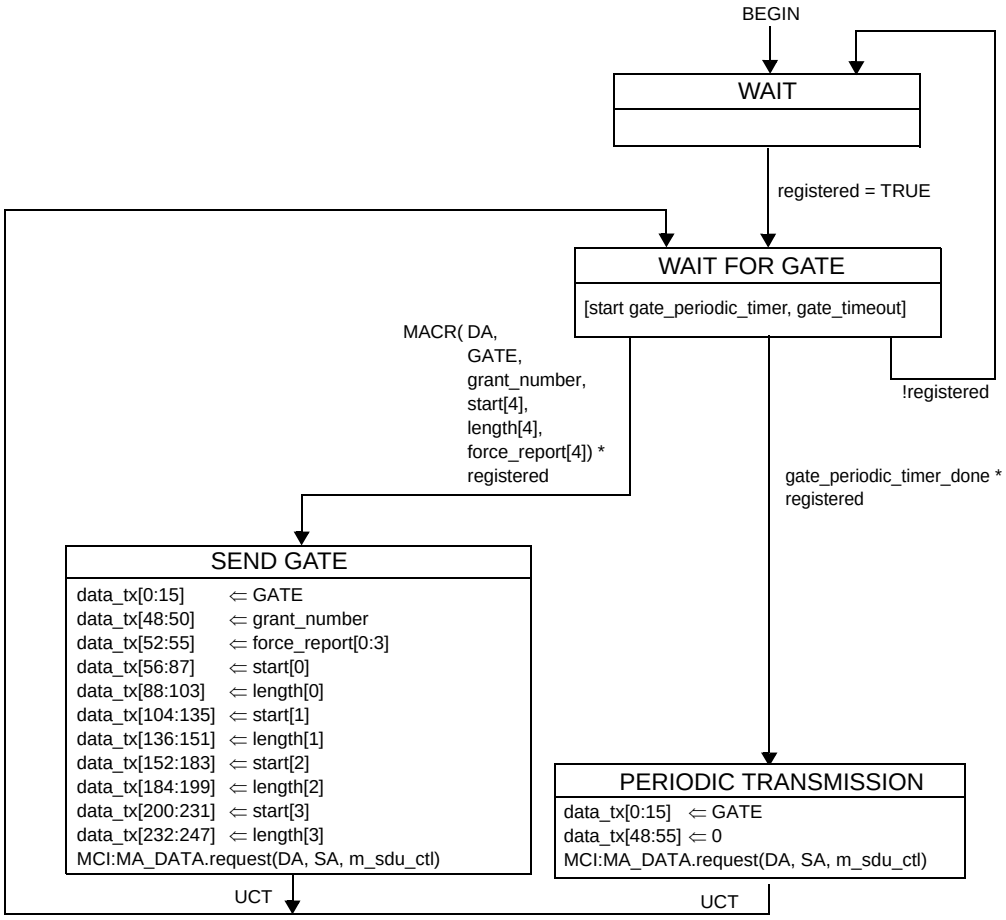
GATE:	Opcode for GATE MPCPDU as defined in Table 31A–1.
start:	start time of the grant. This parameter is not present when the parameter status value is equal to deactive.
length:	Length of the grant. This parameter is not present when the parameter status value is equal to deactive.
force_report:	Flags 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)

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

103.3.5.6 State diagrams

The gating process in the CLT shall implement the Gate processing state diagram as shown in Figure 103–23. The gating process in the CNU shall implement the Gate processing state diagram as shown in Figure 103–24 and Figure 103–25. 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 103–23—Gate Processing state diagram at CLT

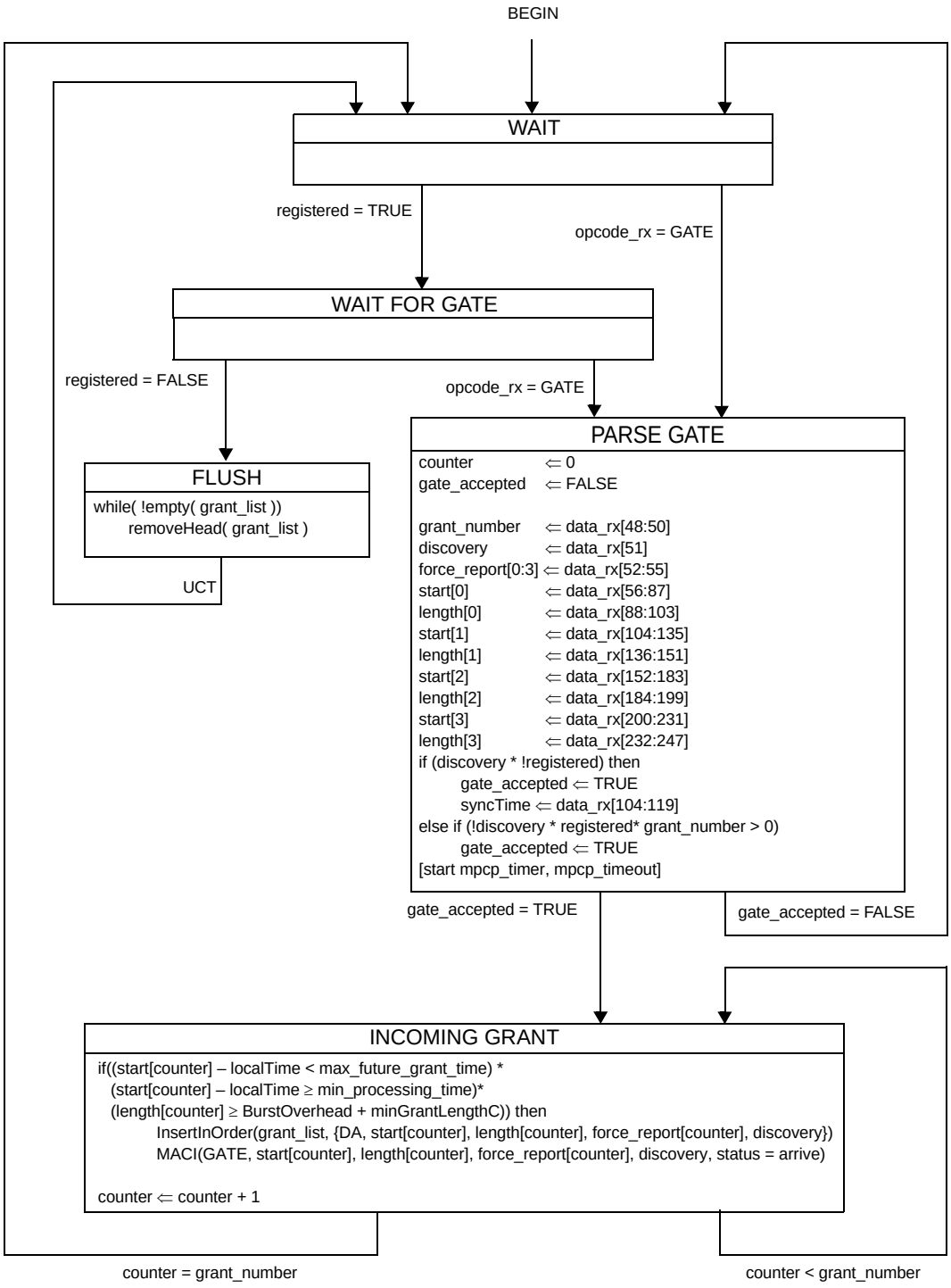


Figure 103–24—Gate Processing CNU Programing state diagram

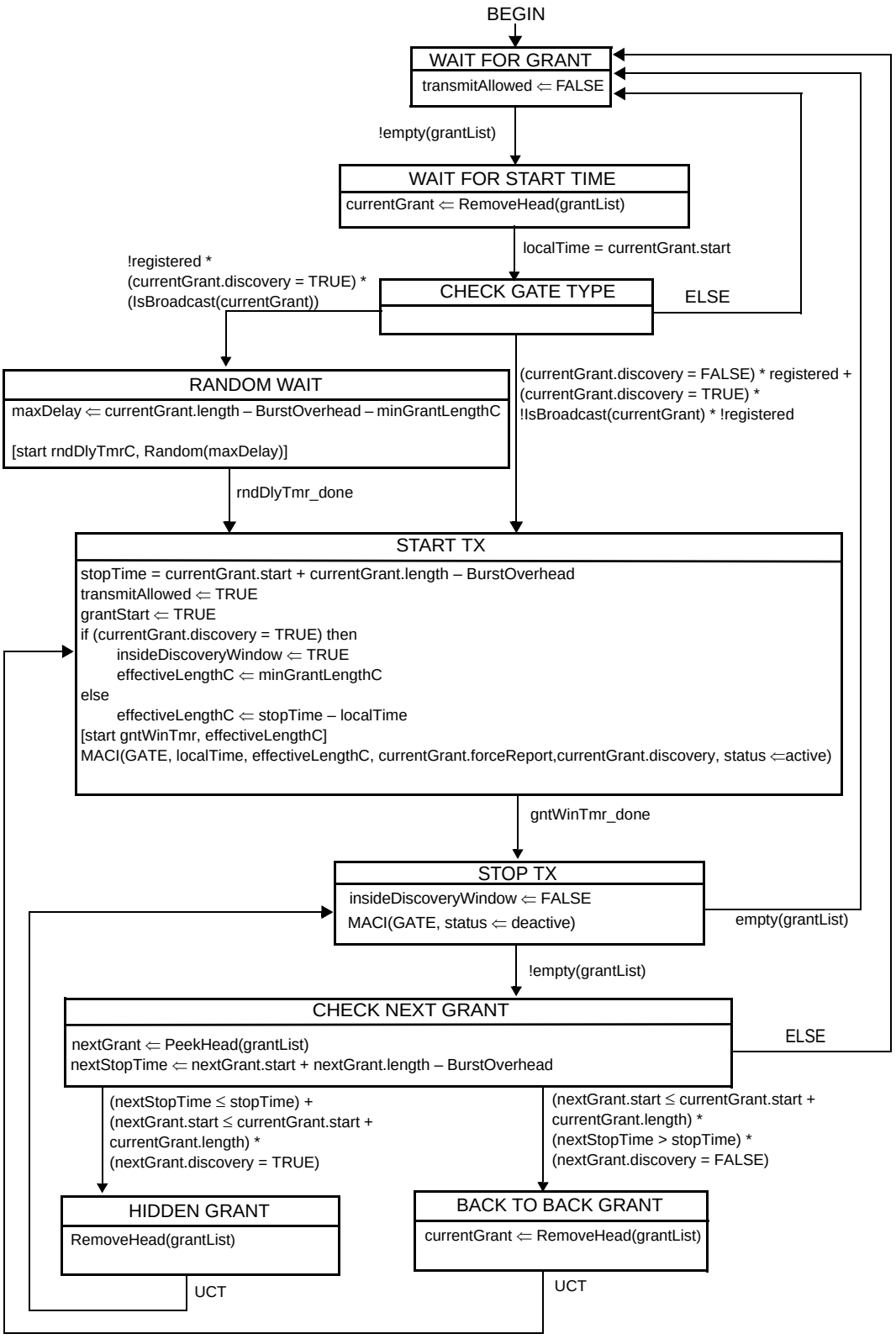


Figure 103–25—Gate Processing CNU Activation state diagram

103.3.6 MPCPDU structure and encoding

The generic MPCPDU structure and encoding in EPoC is as described in 77.3.6. The MPCPDU structure shall be as shown in Figure 77–31. Changes to specific MPCPDU messages are noted below.

103.3.6.1 GATE description

The GATE MPCPDU used in EPoC is the same as that described in 77.3.6.1 with the following exceptions. In EPoC *rfOnTime* and *rfOffTime* replace *laserOnTime* and *laserOffTime*, respectively. The Sync Time and Discovery Information fields described in 77.3.6.1 are not used in EPoC and shall be set to zero on transmit and ignored on reception.

103.3.6.2 REPORT description

The REPORT MPCPDU used in EPoC is the same as that described in 77.3.6.2.

103.3.6.3 REGISTER_REQ description

The REGISTER_REQ MPCPDU used in EPoC and illustrated in Figure 103–26 is the same as that described in 77.3.6.3 with the following exceptions. The Discovery Information field described in 77.3.6.3 is not used in EPoCand shall be set to zero on transmit and ignored on reception.

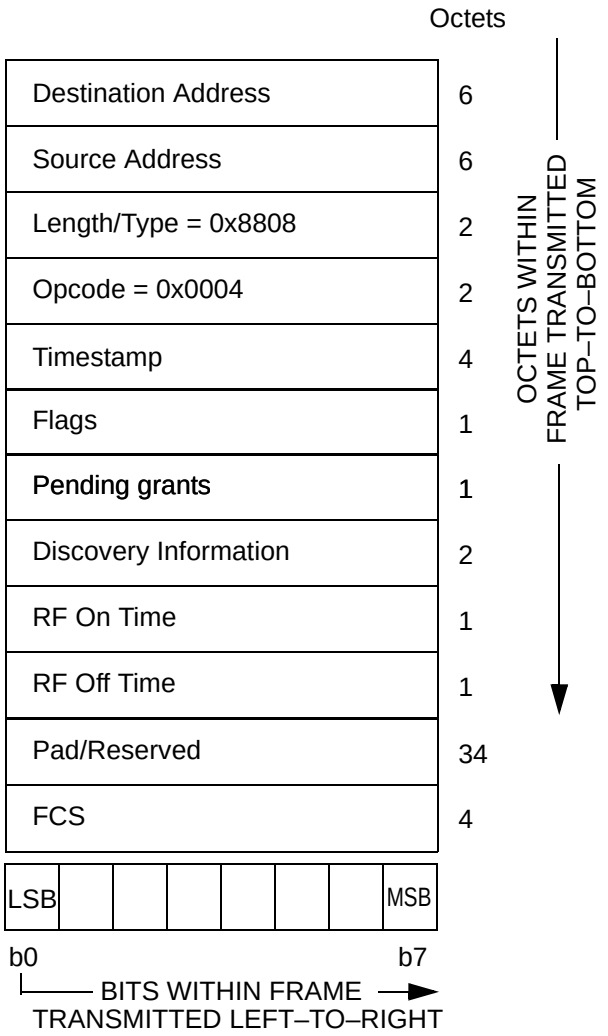


Figure 103–26—REGISTER_REQ MPCPDU

103.3.6.4 REGISTER description

The REGISTER MPCPDU used in EPoC and illustrated in Figure 103–27 is the same as that described in 77.3.6.4 with the following exceptions. In EPoC the Sync Time field is calculated using rfOnTime, rfOff-Time rather than the laserOnTime and laserOffTime used in 77.3.6.4.

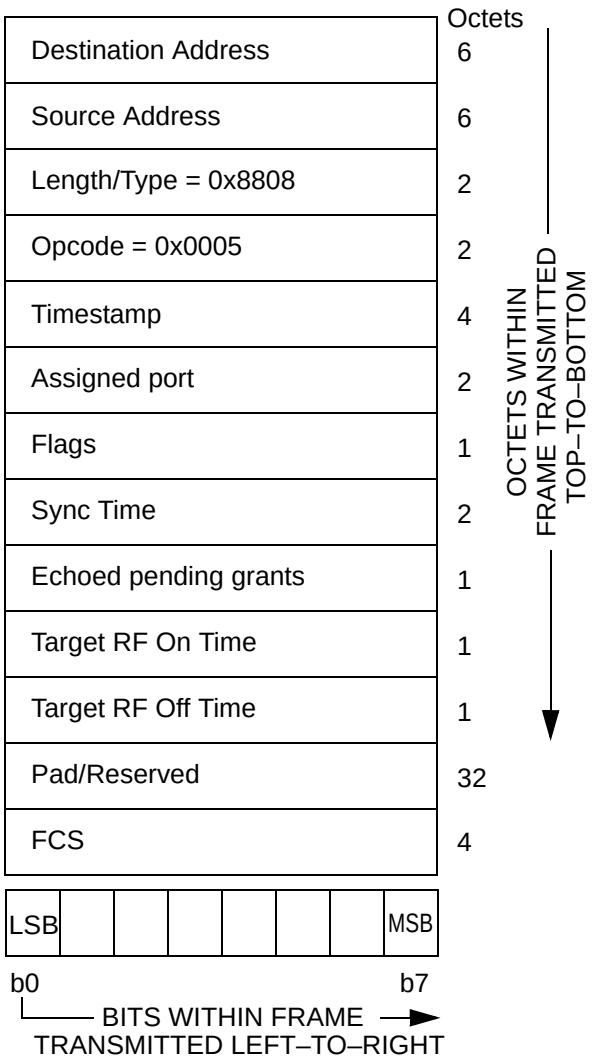


Figure 103–27—REGISTER MPCPDU

103.3.6.5 REGISTER_ACK description

The REGISTER_ACK MPCPDU used in EPoC is the same as that described in 77.3.6.5.

103.4 Protocol implementation conformance statement (PICS) proforma for Clause 103, Multipoint MAC Control for EPoC[6]

103.4.1 Introduction

The supplier of a protocol implementation that is claimed to conform to Clause 103, Multipoint MAC Control for EPoC, 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.

103.4.2 Identification

103.4.2.1 Implementation identification

Supplier[1]	
Contact point for inquiries about the 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 2—May be completed as appropriate in meeting the requirements for the identification. NOTE 3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).	

103.4.2.2 Protocol summary

Identification of protocol standard	IEEE Std 802.3bn-201x, Clause 103, Multipoint MAC Control for EPoC
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 802.3bn-201x.)	

Date of Statement	

[6]*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.

103.4.3 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
*CLT	CLT functionality	103.1	Device supports functionality required for CLT	O/1	Yes [ ] No [ ]
*CNU	CNU functionality	103.1	Device supports functionality required for CNU	O/1	Yes [ ] No [ ]

103.4.4 PICS proforma tables for Multipoint MAC Control

103.4.4.1 Compatibility considerations

Item	Feature	Subclause	Value/Comment	Status	Support
CC1	Delay through MAC	103.3.2.4	Maximum delay variation of 1 *time_quantum*	M	Yes [ ] No [ ]
CC2	CLT grant time delays	103.3.2.4	Not grant nearer than 1 time_quanta into the future	CLT:M	Yes [ ] No [ ] N/A [ ]
CC3	CNU processing delays	103.3.2.4	Process all messages in less than 1024 time_quanta	CNU:M	Yes [ ] No [ ] N/A [ ]
CC4	CLT grant issuance	103.3.2.4	Not grant more than one message every 1024 time_quanta to a single CNU	CLT:M	Yes [ ] No [ ] N/A [ ]
CC5	CLT gap time	103.3.2.4	Minimum gap time between bursts to any two CNUs is equal to *RB_time_quanta*	CLT:M	Yes [ ] No [ ] N/A [ ]

103.4.4.2 Multipoint MAC Control

Item	Feature	Subclause	Value/Comment	Status	Support
OM1	CLT *localTime*	77.2.2.2	Track transmit clock	CLT:M	Yes [ ] No [ ] N/A [ ]
OM2	CNU *localTime*	77.2.2.2	Track receive clock	CNU:M	Yes [ ] No [ ] N/A [ ]
OM3	Random wait for transmitting REGISTER_REQ messages	77.3.3	Shorter than length of discovery window	CNU:M	Yes [ ] No [ ] N/A [ ]
OM4	Periodic report generation	77.3.4	Reports are generated periodically	CNU:M	Yes [ ] No [ ] N/A [ ]
OM5	Periodic granting	77.3.4	Grants are issued periodically	CLT:M	Yes [ ] No [ ] N/A [ ]
OM6	Issuing of grants	77.3.5	Not issue more than maximum supported grants	CLT:M	Yes [ ] No [ ] N/A [ ]

103.4.4.3 State diagrams

Item	Feature	Subclause	Value/Comment	Status	Support
SM1	Multipoint Transmission Control	103.2.2.7	Meets the requirements of Figure 103–9	M	Yes [ ] No [ ]
SM2	CLT Control Parser	103.2.2.7	Meets the requirements of Figure 103–10	M	Yes [ ] No [ ]
SM3	CNU Control Parser	103.2.2.7	Meets the requirements of Figure 103–11	M	Yes [ ] No [ ]
SM4	CLT Control Multiplexer	103.2.2.7	Meets the requirements of Figure 103–12	CLT:M	Yes [ ] No [ ] N/A [ ]
SM5	CNU Control Multiplexer	103.2.2.7	Meets the requirements of Figure 103–13	CLT:M	Yes [ ] No [ ] N/A [ ]
SM6	Discovery Processing CLT Window Setup	103.3.3.5	Meets the requirements of Figure 103–17	CLT:M	Yes [ ] No [ ] N/A [ ]
SM7	Discovery Processing CLT Process Requests	103.3.3.5	Meets the requirements of Figure 103–18	CLT:M	Yes [ ] No [ ] N/A [ ]
SM8	Discovery Processing CLT Register	103.3.3.5	Meets the requirements of Figure 103–19	CNU:M	Yes [ ] No [ ] N/A [ ]
SM9	Discovery Processing CLT Final Registration	103.3.3.5	Meets the requirements of Figure 103–20	CLT:M	Yes [ ] No [ ] N/A [ ]
SM10	Discovery Processing CNU Registration	103.3.3.5	Meets the requirements of Figure 103–21	CNU:M	Yes [ ] No [ ] N/A [ ]
SM11	Report Processing at CLT	77.3.4.6	Meets the requirements of Figure 77-25	CLT:M	Yes [ ] No [ ] N/A [ ]
SM12	Report Processing at CNU	77.3.4.6	Meets the requirements of Figure 77-26	CNU:M	Yes [ ] No [ ] N/A [ ]
SM13	Gate Processing at CLT	103.3.5.6	Meets the requirements of Figure 103–23	CLT:M	Yes [ ] No [ ] N/A [ ]
SM14	Gate Processing at CNU	103.3.5.6	Meets the requirements of Figure 103–24	CNU:M	Yes [ ] No [ ] N/A [ ]
SM15	Gate Processing CNU Activation	103.3.5.6	Meets the requirements of Figure 103–25	CNU:M	Yes [ ] No [ ] N/A [ ]

103.4.4.4 MPCP

Item	Feature	Subclause	Value/Comment	Status	Support
MP1	MPCPDU structure	77.3.6	As in Figure 77-31	M	Yes [ ] No [ ]
MP2	LLID for MPCPDU	77.3.6	RS generates LLID for MPCPDU	M	Yes [ ] No [ ]
MP3	Grants during discovery	77.3.6.1	Single grant in GATE message during discovery	CLT:M	Yes [ ] No [ ] N/A [ ]
MP4	Grant start time	77.3.6.1	Grants within one GATE MPCPDU are sorted by their Start time values	CLT:M	Yes [ ] No [ ] N/A [ ]
MP5	GATE generation	77.3.6.1	GATE generated for active CNU except during discovery	CLT:M	Yes [ ] No [ ] N/A [ ]
MP6	GATE LLID	77.3.6.1	Unicast LLID except for discovery	CLT:M	Yes [ ] No [ ] N/A [ ]
MP7	GATE *rfOnTime* and *rfOffTime*	103.3.6.1	Always zero	CLT:M	Yes [ ] No [ ] N/A [ ]
MP8	REGISTER_REQ Discovery Information	103.3.6.3	Always zero	CNU:M	Yes [ ] No [ ] N/A [ ]
MP9	REPORT issuing	77.3.6.2	Issues REPORT periodically	CNU:M	Yes [ ] No [ ] N/A [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
MP8	REPORT generation	77.3.6.2	Generated by active CNU	CNU:M	Yes [ ] No [ ] N/A [ ]
MP9	REPORT queue #n	77.3.6.2	REPORT Queue #n length rounding	CNU:M	Yes [ ] No [ ] N/A [ ]
MP10	REPORT LLID	77.3.6.2	REPORT has unicast LLID	CNU:M	Yes [ ] No [ ] N/A [ ]
MP11	REGISTER_REQ generation	77.3.6.3	Generated by undiscovered CNU	CNU:M	Yes [ ] No [ ] N/A [ ]
MP12	REGISTER_REQ LLID	77.3.6.3	Use broadcast LLID	CNU:M	Yes [ ] No [ ] N/A [ ]
MP13	REGISTER DA address	77.3.6.4	Use individual MAC address	CLT:M	Yes [ ] No [ ] N/A [ ]
MP14	REGISTER generation	77.3.6.4	Generated for all CNUs	CLT:M	Yes [ ] No [ ] N/A [ ]
MP15	REGISTER_ACK generation	77.3.6.5	Generated by active CNU	CNU:M	Yes [ ] No [ ] N/A [ ]
MP16	REGISTER_ACK LLID	77.3.6.5	Use unicast LLID	CNU:M	Yes [ ] No [ ] N/A [ ]
MP17	MAC enable	74.2.2.4	MAC Control interface has pri-oroty over other clients	CLT:M	Yes [ ] No [ ] N/A [ ]