Annex A OAM Over UMT

A.1 Introduction

IEEE Std. 802.3 Clause 57 defines the Operations, Administration, and Maintenance (OAM) protocol that is used for monitoring link operation. Clause 57 also defines a method for extending the protocol. IEEE Std. 1904.1 extends OAM for ONU management in EPON networks. IEEE Std. 802.3 Clause 57 requires that the destination address of OAMPDUs be set to the Slow-Protocols-Multicast address. IEEE Std. 802.3 Clause 57 also limits the scope of an OAMPDU to a single link.

Some EPON use cases require that the controlling OAM implementation be located on a host that is not directly connected to the PON (i.e. not in the OLT system). The addressing requirements in IEEE Std. 802.3 Clause 57 effectively prohibit such an implementation.

This annex defines an adaptation layer that conforms to the specification of a UMT Client Adaptation entity as defined in IEEE Std. 1904.2 Clause 5. This adaptation layer, the OAM to UMT Adaptation Layer (OUAL), mediates between the OAM entity and the UMT layer. This mediation with UMT enables OAM to overcome the limitations defined in IEEE Std. 802.3 Clause 57 while maintaining backward compatibility with IEEE Std. 802.3 Clause 57.

A.2 Topology of OAM over UMT

Figure A-1 depicts the topology of a network over which a pair of OAM peers wish to communicate using UMT.

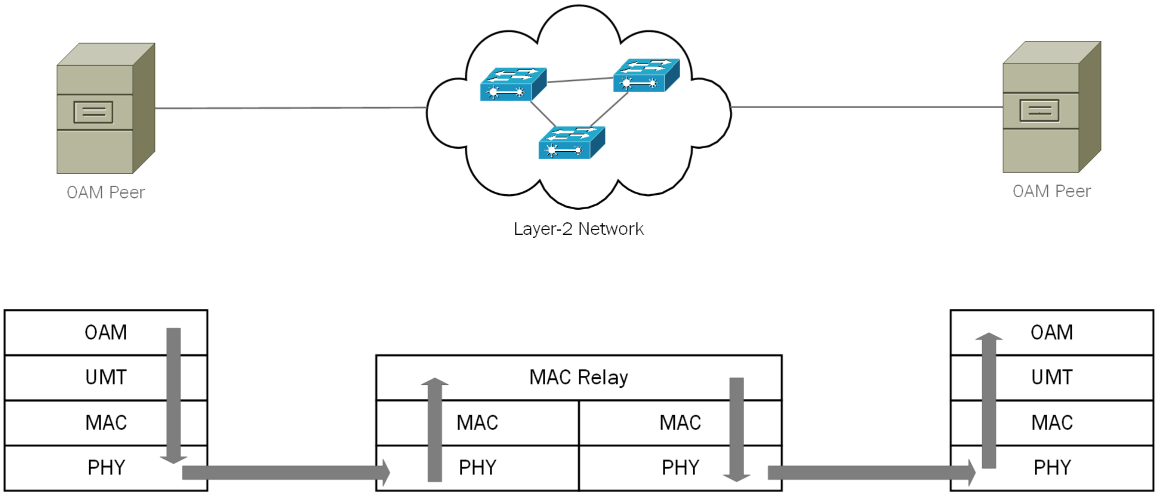

Figure A-1 - Topology of OAM over UMT

In this generalized topology, an OAM client wishes to discover and communicate with another OAM client that is located one or more MAC Relay hops away. The requirements in IEEE Std. 802.3 Clause 57 prevent the OAMPDU from being forwarded across the layer-2 network.

OAM is intended for use on point-to-point and emulated point-to-point links. In the case of point-to-point links, OAM functionality is uninhibited because the OAM peers are directly connected and there is no device to interfere with delivery of the OAMPDU.

In the case of emulated point-to-point links, the medium is typically a PON (point-to-multipoint). In a point-to-multipoint topology, a single OAMPDU sent from the root will be seen by all leaf nodes, so a method is required to segregate OAM peer relationships. This segregation is accomplished on an EPON through the logical link. On an EPON, therefore, an OAM peer relationship is created on each logical link requiring multiple instances of the OAM layer to reside on the root node – one for each leaf node participating in an OAM session.

Operation of OAM over UMT can be accomplished in a fashion similar to that used in EPON. Figure A-2 illustrates the concept of emulated point-to-point links over a generalized layer-2 network.

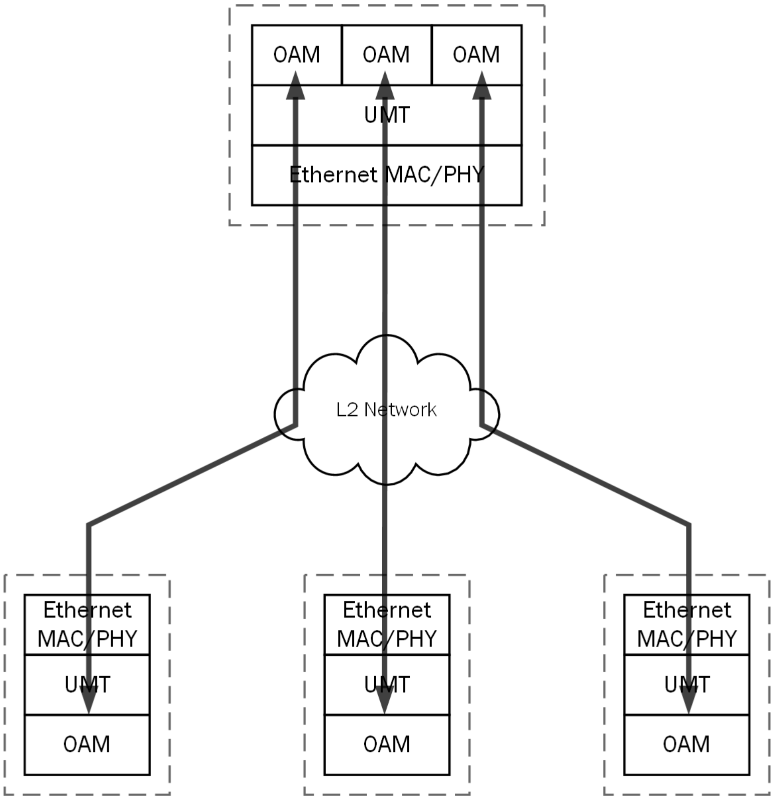

Figure A-2 - Emulated Point-to-Point over a generalized Layer-2 Network

In this model, a point-to-point link exists between each instance of OAM. This model conforms to the assumptions in IEEE Std. 802.3 Clause 57.

A.3 OAM Discovery over UMT

IEEE Std. 802.3 Clause 57 requires that an active OAM peer initiate OAM discovery by sending an OAM Info PDU to the Slow-Protocols-Multicast address, 01-80-C2-00-00-02, and that the Type/Length field of the frame contain the Slow_Protocols_Type value, 88-09.

By definition, this addressing will cause the frame to be consumed or discarded by the first MAC entity that receives the frame. To avoid this, the OAM layer can use UMT.

Instead of passing the OAMPDU to the MAC layer, the OAMPDU will be passed to a local UMT Tunnel Adapter via the OUAL. The OUAL will pass the subtype and the OAM Data field to the UMT Tunnel Adapter. The UMT Tunnel Adapter and UMT Tunnel Multiplexer set the MAC destination address to the MAC address of the UMT peer, and set the source address to the local station’s MAC address.

The OAMPDU will be encapsulated as described in IEEE Std. 1904.2 and as exemplified in Figure A-7.

The resulting MAC frame will be forwarded across the network according to the rules defined in IEEE Std. 802.3, IEEE Std. 802.1Q and IEEE Std. 802.1D.

Upon receipt at the remote UMT Peer, the MAC layer will forward the UMTPDU to the UMT Sublayer. The UMT Sublayer will parse the PDU to find that the received PDU is a UMTPDU and pass the UMTPDU to the UMT Tunnel Multiplexer. The UMT Tunnel Multiplexer will parse the UMTPDU to determine the UMT Tunnel Adapter and pass the decapsulated data to the OUAL for delivery to the OAM layer.

Figure A-3 shows an example of the message exchange described above.

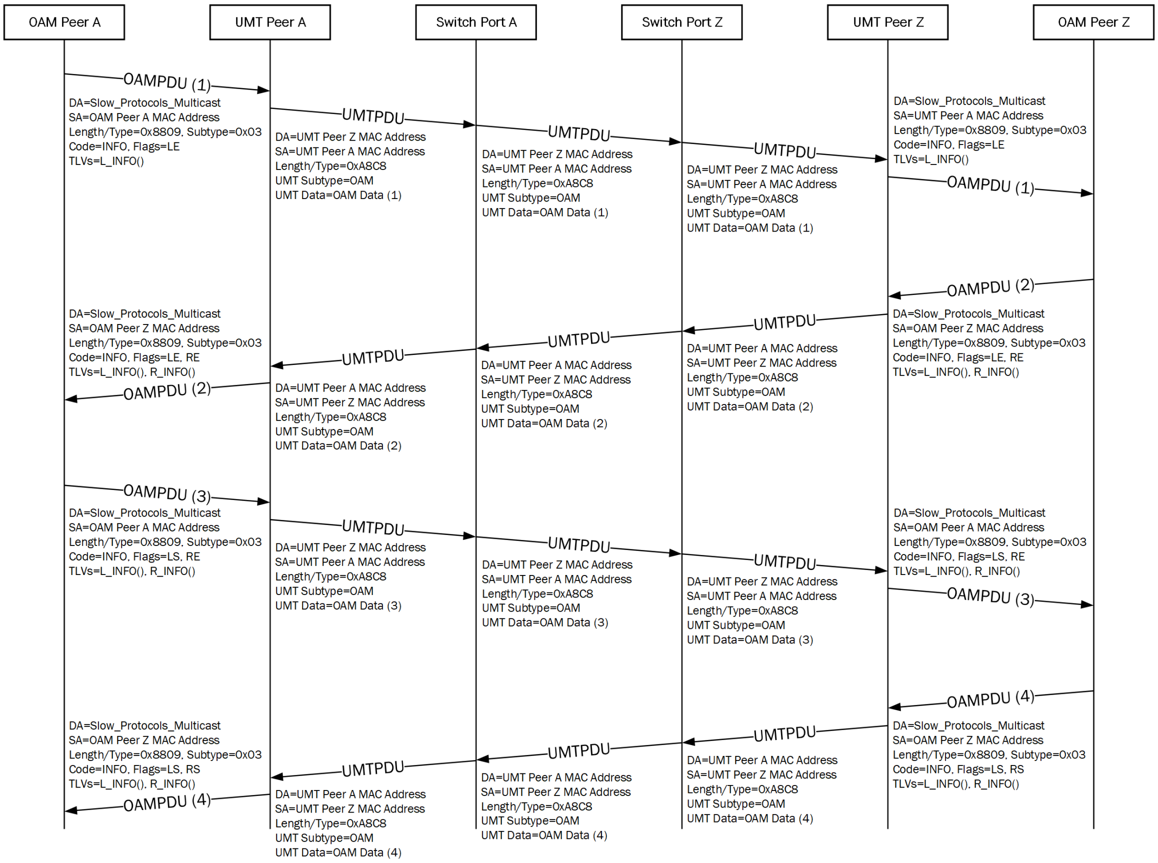
OAM Peer A
UMT Peer A
Switch Port A
Switch Port Z
UMT Peer Z
OAM Peer Z
OAMPDU (1)
DA=Slow_Protocols_Multicast
SA=OAM Peer A MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LE
TLVs=L_INFO()
UMTPDU
DA=UMT Peer Z MAC Address
SA=UMT Peer A MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (1)
UMTPDU
DA=UMT Peer Z MAC Address
SA=UMT Peer A MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (1)
UMTPDU
DA=UMT Peer Z MAC Address
SA=UMT Peer A MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (1)
DA=Slow_Protocols_Multicast
SA=UMT Peer A MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LE
TLVs=L_INFO()
OAMPDU (1)
OAMPDU (2)
DA=Slow_Protocols_Multicast
SA=OAM Peer Z MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LE, RE
TLVs=L_INFO(), R_INFO()
UMTPDU
DA=UMT Peer A MAC Address
SA=UMT Peer Z MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (2)
UMTPDU
DA=UMT Peer A MAC Address
SA=UMT Peer Z MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (2)
UMTPDU
DA=UMT Peer A MAC Address
SA=UMT Peer Z MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (2)
DA=Slow_Protocols_Multicast
SA=OAM Peer Z MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LE, RE
TLVs=L_INFO(), R_INFO()
OAMPDU (2)
OAMPDU (3)
DA=Slow_Protocols_Multicast
SA=OAM Peer A MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LS, RE
TLVs=L_INFO(), R_INFO()
UMTPDU
DA=UMT Peer Z MAC Address
SA=UMT Peer A MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (3)
UMTPDU
DA=UMT Peer Z MAC Address
SA=UMT Peer A MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (3)
UMTPDU
DA=UMT Peer Z MAC Address
SA=UMT Peer A MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (3)
DA=Slow_Protocols_Multicast
SA=OAM Peer A MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LS, RE
TLVs=L_INFO(), R_INFO()
OAMPDU (3)
OAMPDU (4)
DA=Slow_Protocols_Multicast
SA=OAM Peer Z MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LS, RS
TLVs=L_INFO(), R_INFO()
UMTPDU
DA=UMT Peer A MAC Address
SA=UMT Peer Z MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (4)
UMTPDU
DA=UMT Peer A MAC Address
SA=UMT Peer Z MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (4)
UMTPDU
DA=UMT Peer A MAC Address
SA=UMT Peer Z MAC Address
Length/Type=0xA8C8
UMT Subtype=OAM
UMT Data=OAM Data (4)
DA=Slow_Protocols_Multicast
SA=OAM Peer Z MAC Address
Length/Type=0x8809, Subtype=0x03
Code=INFO, Flags=LS, RS
TLVs=L_INFO(), R_INFO()
OAMPDU (4)

Figure A-3 - Packet Flow for OAM Discovery over UMT

A.4 Functional Specifications

A.4.1 Interlayer Service Interfaces

Figure A-4 depicts the usage of interlayer interfaces by the OAM to UMT Adaptation Layer (OUAL).

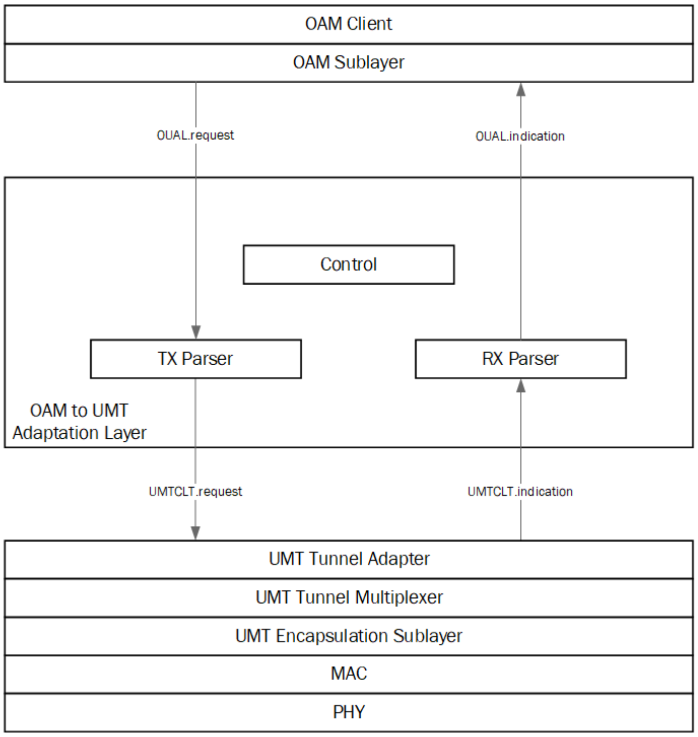

Figure A-4 - OAM to UMT Adaptation service interfaces

A.4.2 Principles of Operation

OAM to UMT Adaptation employs the following principles and concepts:

a) The OUAL presents the equivalent of an IEEE 802.3 MAC service interface to the superior layer, which is the OAM sublayer.

b) The OUAL is intended only for adapting an OAM entity to an UMT entity. Non-OAM entities that wish to use UMT will employ their own protocol-specific adaptation layer.

c) In compliance to IEEE Std. 1904.2 Clause 4, the OUAL is an implementation of the UMT Client Adapatation entity and employs the UMTCLT.request and UMTCLT.indication interfaces to the UMT Tunnel Adapter entity.

d) The OUAL drops the destination and source MAC address from the requested OAMPDU and transforms the remaining parameters into the format required by the UMTCLT.request primitive. Similarly, for indicated UMTPDUs destined for an OAM entity, the OUAL drops the source and destination MAC address received in the UMTPDU and adds the destination and source MAC address expected by the OAM entity before asserting the UMTUAL.indication primitive.

A.4.3 OAM Sublayer

The OAM Sublayer is fully defined in IEEE Std. 802.3 Clause 57.

A.4.4 OAM to UMT Client Adaptation Layer (OUAL)

A.4.4.1 OUAL Interactions

The OUAL communicates with the OAM sublayer using the following interlayer service interfaces:

OUAL.request

OUAL.indication

The OUAL.request and OUAL.indication service primitives are specific implementations of the abstract primitives UMTUAL.request and UMTUAL.indication (respectively) described in IEEE Std. 1904.2 Clause 4. The following subclauses will define the OUAL.request and OUAL.indication primitives *as they apply to adapt an OAM entity to the UMT via OUAL.*

The OUAL communicates with the UMT Client using the following interlayer service interfaces:

UMTCLT.request

UMTCLT.indication

The UMTCLT.request and UMTCLT.indication service primitives are described in IEEE Std. 1904.2 Clause 4.

A.4.4.1.1 OUAL.request

A.4.4.1.1.1 Function

This primitive defines the transfer of data from a UMT Client entity to the UMT Client Adaptation entity. This primitive is intended to mimic the MAC MA_DATA.request service primitive.

A.4.4.1.1.2 Semantics of the service primitive

The semantics of the primitive are as follows:

```
OUAL.request (
                    destination_address,
                    source_address,
                    umt_client_sdu
                    )
```

The destination_address parameter may specify either an individual or a group MAC entity address. The source_address parameter, if present, must specify an individual MAC address. The umt_client_sdu parameter is equivalent to the mac_service_data_unit parameter and is used to create the Data field within the UMTPDU to be transmitted.

When considering compatibility between existing OAM implementations and UMT, an OAM entity must not supply the field_check_sequence parameter to the OUAL.request primitive as it might to the MA_DATA.request service primitive.

A.4.4.1.1.3 When Generated

This primitive is generated by the OAM entity whenever an OAMPDU is to be transferred to a peer entity over the UMT.

A.4.4.1.1.4 Effect of Receipt

The receipt of this primitive will cause the UMT Client Adaptation entity to perform any required parsing and transformations of the destination_address, source_address, and umt_client_sdu parameters necessary to send the OAM data over the UMT. After performing these actions, the UMT Client Adaptation entity asserts the UMTCLT.request primitive to the UMT Tunnel Adapter according to the procedures described in IEEE Std. 1904.2 Clause 4.2.5.2.1.

A.4.4.1.2 OUAL.indication

A.4.4.1.2.1 Function

This primitive defines the transfer of data from a UMT Client Adaptation entity to the OAM entity. This primitive is intended to mimic the MAC MA_DATA.indication service primitive.

A.4.4.1.2.2 Semantics of the service primitive

The semantics of the primitive are as follows:

```
OUAL.indication (
                        destination_address,
                        source_address,
                        umt_client_sdu,
                        reception_status
                        )
```

Generally, the destination_address parameter is the MAC destination address of the incoming UMTPDU and the source_address parameter is the MAC source address of the incoming UMTPDU. The UMT Adaptation entity shall modify the destination_address parameter and/or the source_address parameter to maintain compatibility with the OAM implementation. The umt_client_sdu parameter is derived from the Data field of the incoming UMTPDU and may include transformations performed in the UMT Client Adaptation entity.

When considering compatibility between existing MAC clients and UMT, OUAL.indication will not supply the field_check_sequence parameter to the MAC client as MA_DATA.indication might. The OUAL.request primitive, to maintain compatibility with the MA_DATA.indication primitive, will supply the reception_status parameter and will always set the value to receiveOK.

A.4.4.1.2.3 When Generated

This primitive is passed from the UMT Client Adaptation entity to the UMT Client entity to indicate the arrival of a UMTPDU to the local UMT Client. Such UMTPDUs are reported only if they are validly formed and received without error. In the specific case of OAM as the UMT Client, the OUAL shall assert this primitive upon arrival of a UMTPDU containing OAM data as indicated by the Subtype field in the UMTPDU.

A.4.4.1.2.4 Effect of Receipt

The effect of receipt of this primitive by the UMT Client is unspecified.

A.5 Detailed functions and state diagrams

A.5.1 State diagram variables

A.5.1.1 Constants

Slow_Protocols_Multicast

The value of the Slow Protocols Multicast Address. See IEEE Std. 802.3 Table 57A-1.

Slow_Protocols_Type

The value of the Slow Protocols Length/Type Field. See IEEE Std. 802.3 Table 57A-2.

OAM_Subtype

The value of the Subtype field for OAMPDUs. See IEEE Std. 802.3 Table 57A-3.

OAM_UMT_Subtype

The value of the UMT Subtype field for OAM data in UMTPDUs. See IEEE Std. 1904.2 Table 4-2.

A.5.1.2 Variables

req_DA

req_SA

req_oam_sdu

The parameters of the OUAL.request service primitive, as defined in A.4.4.1.1, and passed from the OAM layer to the OUAL.

req_umt_subtype

req_umt_client_sdu

The parameters of the UMTCLT.request service primitive, as defined in IEEE Std. 1904.2 Clause 4.2.5.2.1, and passed from the OUAL to the UMT Tunnel Adapter.

ind_DA

ind_SA

ind_oam_sdu

ind_reception_status

The parameters of the OUAL.indication service primitive, as defined in A.4.4.1.2, and passed from the OUAL to the OAM layer.

A.5.1.3 Counters

No counters are defined.

A.5.1.4 Timers

No Timers are defined.

A.5.1.5 Functions

length(binary_data)

This function returns the length, in bits, of the binary_data parameter.

A.5.1.6 Messages

UMTCLT.indication

The service primitive used to pass a received UMTPDU to a client with the specified parameters.

UMTCLTIND

Alias for UMTCLT.indication(ind_DA, ind_SA, ind_umt_subtype, ind_umt_client_sdu)

UMTCLT.request

The service primitive used to transmit a UMTPDU with the specified parameters.

UMTCLTREQ

Alias for UMTCLT.request(req_umt_subtype, req_umt_client_sdu)

OUAL.indication

The service primitive used to pass a received OAM SDU to an OAM entity with the specified parameters.

OUALIND

Alias for OUAL.indication(ind_DA, ind_SA, ind_oam_sdu, ind_reception_status).

OUAL.request

The service primitive used to request transmission of an OAMPDU with the specified parameters via the OUAL.

OUALREQ

Alias for OUAL.request(req_DA, req_SA, req_oam_sdu).

A.5.2 OUAL Control

A.5.3 OUAL TX Parser

The OUAL entity shall implement the TX Parser state diagram shown in Figure A-5.

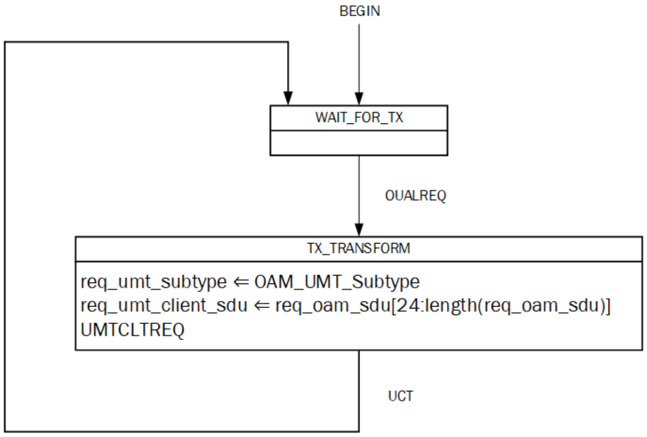
BEGIN
WAIT_FOR_TX
OUALREQ
TX_TRANSFORM
req_umt_subtype ⇐ OAM_UMT_Subtype
req_umt_client_sdu ⇐ req_oam_sdu[24:length(req_oam_sdu)]
UMTCLTREQ
UCT

Figure A-5 - OUAL TX Parser State Diagram

A.5.3.1 WAIT_FOR_TX state

Upon initialization, the WAIT_FOR_TX state is entered. While in the WAIT_FOR_TX state, the TX Parser waits for the occurrence of an OUAL.request. Upon assertion of the OUAL.request primitive, the TX Parser moves to the TX_TRANSFORM state.

A.5.3.2 TX_TRANSFORM state

When the TX Parser enters the TX_TRANSFORM state, the TX Parser extracts the concatenated OAM Flags, Code and Data fields from the req_oam_sdu parameter and puts the resulting bits into the req_umt_client_sdu parameter. The TX Parser then asserts the UMTCLT.request primitive with the required parameters. Upon completion of the TX_TRANSFORM state, the TX Parser moves to the WAIT_FOR_TX state. CHECK_DISC_TIMER state

A.5.4 OUAL RX Parser

The OUAL entity shall implement the RX Parser state diagram shown in Figure A-6.

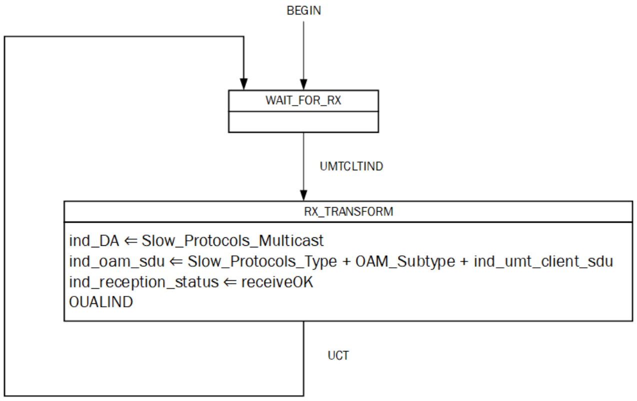
BEGIN
WAIT_FOR_RX
UMTCLTIND
RX_TRANSFORM
ind_DA ⇐ Slow_Protocols_Multicast
ind_oam_sdu ⇐ Slow_Protocols_Type + OAM_Subtype + ind_umt_client_sdu
ind_reception_status ⇐ receiveOK
OUALIND
UCT

Figure A-6 - OUAL RX Parser State Diagram

A.5.4.1 WAIT_FOR_RX state

Upon initialization, the WAIT_FOR_RX state is entered. While in the WAIT_FOR_RX state, the parser waits for the occurrence of an UMTCLT.indication. Upon assertion of UMTCLT.indication the parser enters the RX_TRANSFORM state.

A.5.4.2 RX_TRANSFORM state

In the RX_TRANSFORM state, the RX Parser constructs an OAM SDU that conforms to the requirements in IEEE Std. 802.3 Clause 57. The RX Parser does this by concatenating the value of Slow_Protocols_Type, the value of OAM Subtype, and the received ind_umt_client_sdu. The RX Parser then asserts the OUAL.indication primitive with the required parameters. Upon completion of the RX_TRANSFORM state, the RX Parser moves to the WAIT_FOR_RX state.

A.5.5 OUAL Control

The OUAL entity must implement the Control process. The control process is responsible for identifying the available UMT Tunnel Adapters and registering to use the desired UMT Tunnel Adapter. The operation of the Control function is out of scope of this annex.

A.6 OAMPDU Encapsulation in UMT

Figure A-7 depicts the steps of encapsulating an OAMPDU into a UMTPDU.

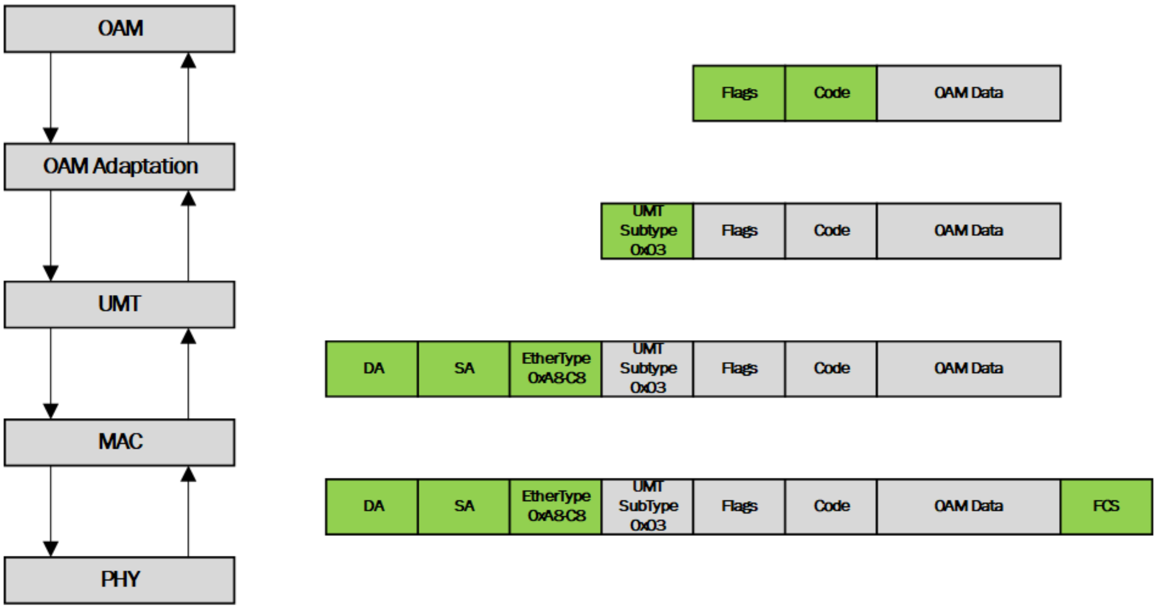

Figure A-7 - OAMPDU Encapsulation in UMTPDU