6.5 Support for OAM remote loopback

6.5.1 Overview

OAM defined in 802.3, 57.2.11 provides an optional data link layer frame-level loopback mode, which can be used for fault localization and link performance testing.

The OAM entity that initiates the loopback mode is called the *local* OAM entity. The OAM entity on the opposite end of a link is called the *remote* OAM entity. In the OAM remote loopback mode, the local and remote OAM entities operate as follows:

a) The local OAM entity transmits frames from the MAC client and OAMPDUs from the local OAM client or OAM sublayer.
b) Within the OAM sublayer of the remote OAM entity, every received OAMPDU is passed to the OAM client, while non-OAMPDUs, including other Slow Protocol frames, are looped back without altering any field of the frame.
c) Frames received by the local OAM entity are parsed by the OAM sublayer. OAMPDUs are passed to the OAM client and all other frames are discarded.

Both OAM entities continue exchanging OAMPDUs in order to keep the OAM discovery process from restarting and to perform other management tasks.

6.5.2. OAM loopback over UMT tunnel

When the OAM loopback is initiated over a UMT tunnel, the behavior of the local and remote OAM entities remains as it is described in 6.5.1. Specifically, the remote OAM sublayer loops back all non-OAMPDUs (i.e., generates an *MA_DATA.request()* primitive in response to every *MA_DATA.indication()* primitive that does not contain an OAMPDU). The local OAM sublayer discards all received non-OAMPDU frames.

However, to ensure that the non-OAMPDUs transmitted by the local MAC client are delivered to the remote OAM sublayer, an additional UMT tunnel needs to be established from the local DTE to the remote DTE. Similarly, to deliver the looped-back frames from the remote DTE back to the local DTE, a UMT tunnel operating in the opposite direction also needs to be established.

Since the OAM is a link-level protocol (i.e., operates over a single-span link), either a DTE itself or a bridge immediately adjacent to that DTE must be UMT-aware. A network configuration with both the local and the remote DTE being UMT-unaware is illustrated in Figure 6-4.

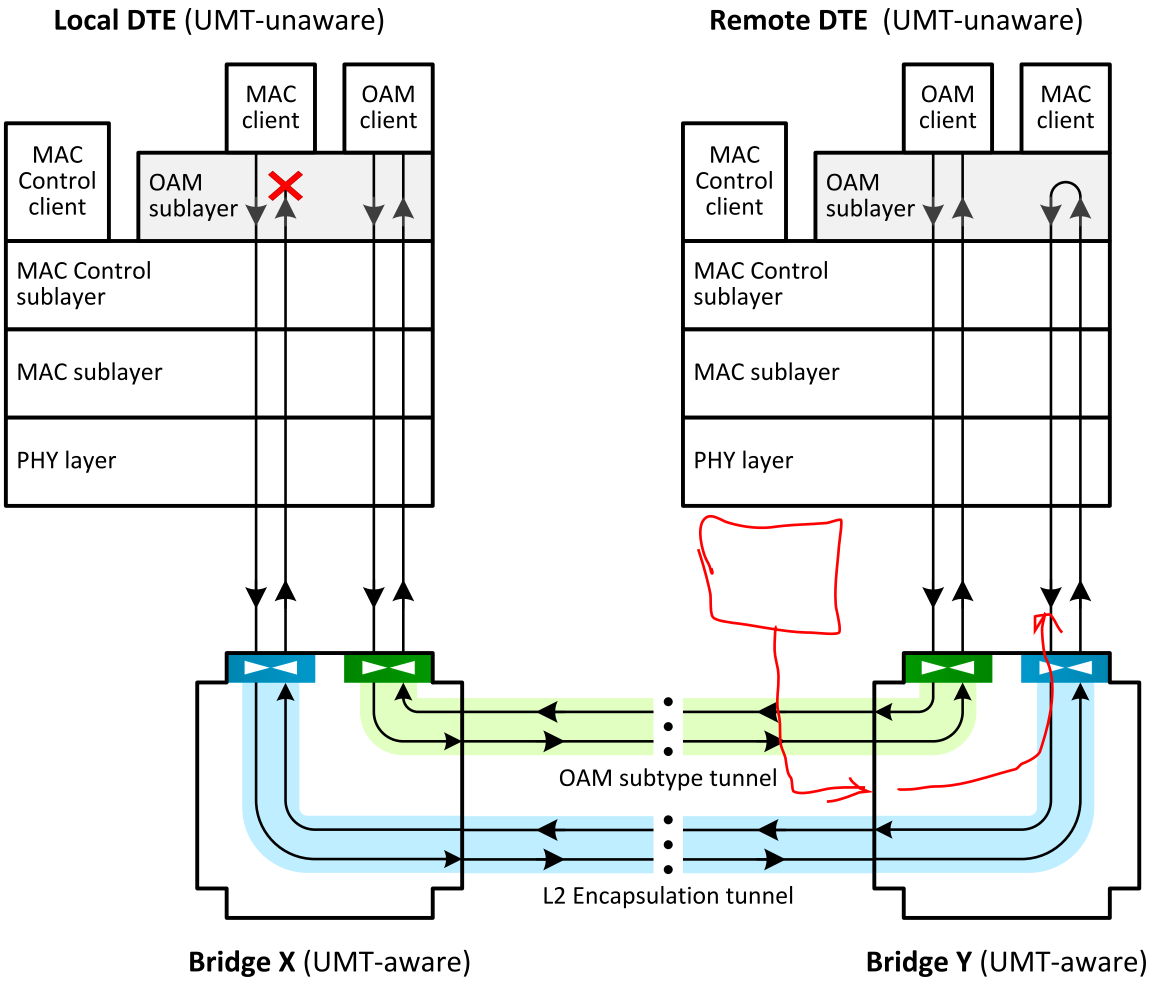

Figure 6-4 — Remote OAM loopback over UMT tunnel with UMT-unaware loacl DTE and UMT-unaware remote DTE.

The remote OAM loopback can also be established when one of the DTEs is UMT-aware and the other is not. Figure 6-5 illustrates a network configuration with the local DTE being UMT-aware and the remote DTE being UMT-unaware.

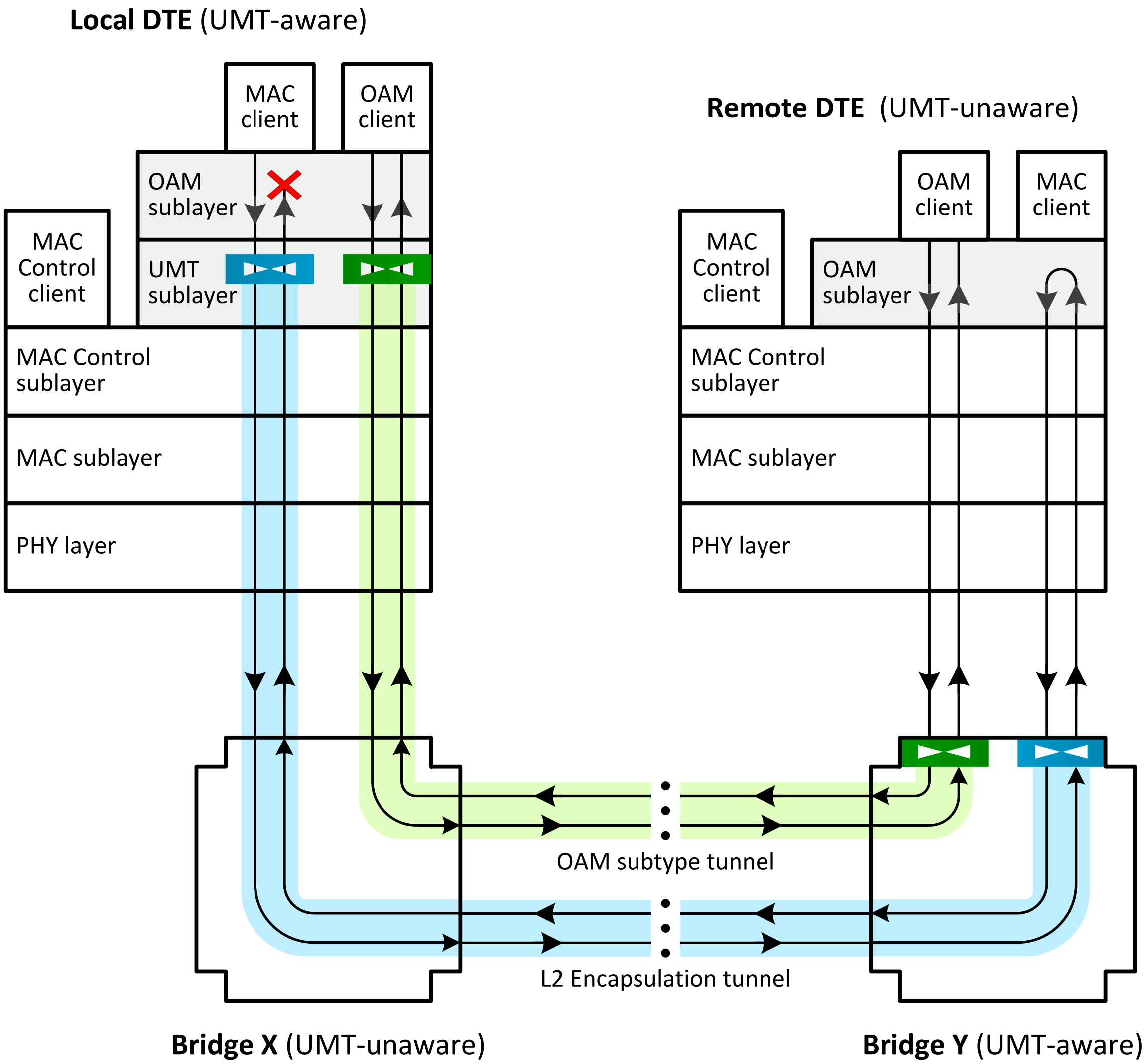

Figure 6-5 — Remote OAM loopback over UMT tunnel with UMT-aware loacl DTE and UMT-unaware remote DTE.

Figure 6-6 represents a similar network configuration, but with both the local and the remote DTEs being UMT-aware.

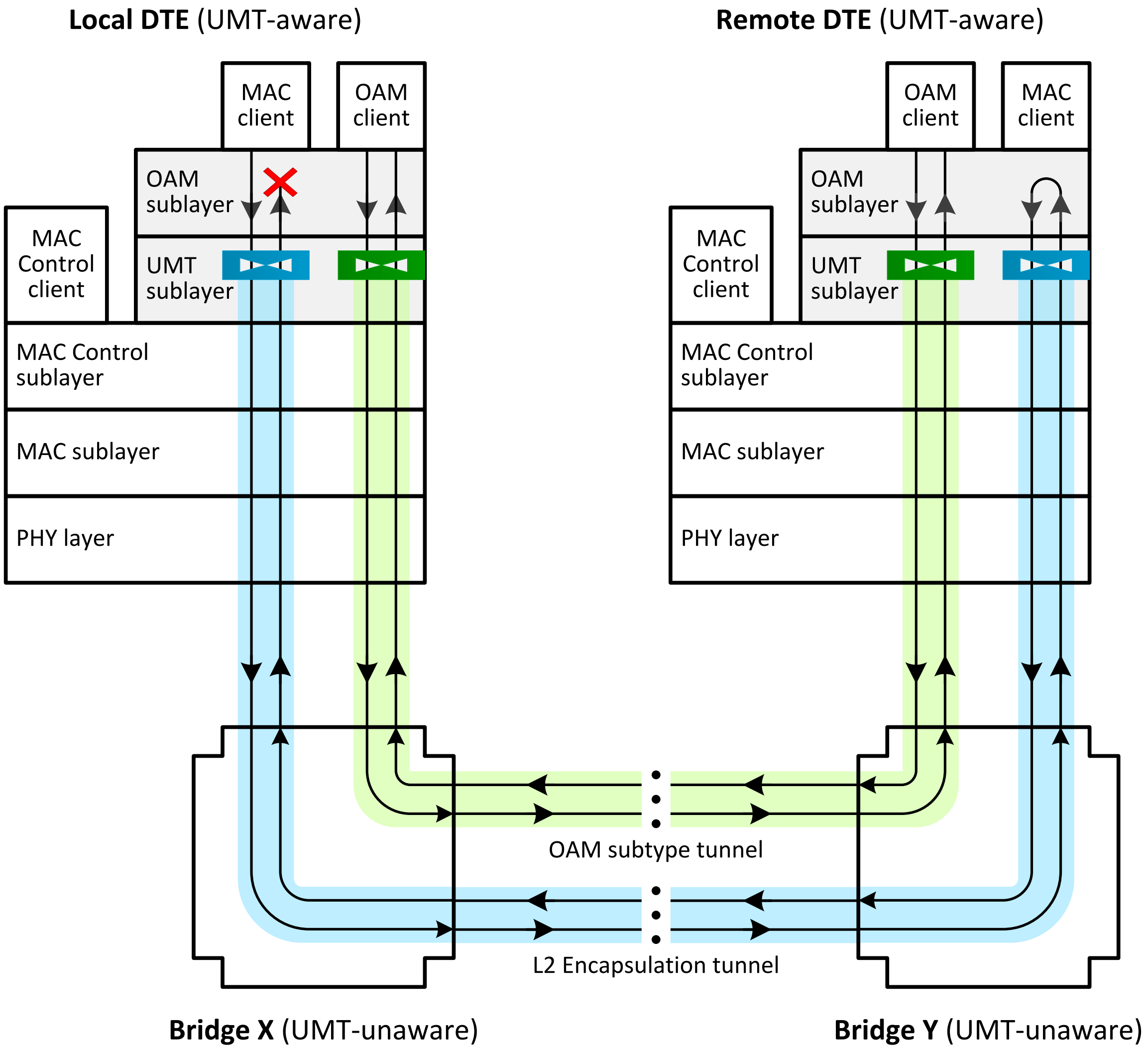

Figure 6-5 — Remote OAM loopback over UMT tunnel with UMT-aware loacl DTE and UMT-aware remote DTE.

While the OAM subtype tunnel between the local and remote DTEs persists permanently to ensure that the two OAM entities are able to exchange OAMPDUs, the bidirectional tunnel for the looped-back data only needs to be established for the duration of the loopback mode. This tunnel has L2 encapsulation subtype in order to deliver any non-OAMPDU (regardless of their Source and destination MAC addresses) to from the local DTE to the remote DTE and in the reverse direction, from the remote DTE to the local DTE.

Table 6-8 illustrates the tunnel entrance rules for the UMT L2 encapsulation tunnel from the local DTE to the remote DTE. The table shows two rules that have different conditions, but identical actions. If these rules are provisioned in the bridge adjacent to the local DTE, as illustrated in Figure 6-4, these are ingress tunnel entrance rules. If the rules are provisioned in the local DTE itself, as illustrated in Figure 6-5 and Figure 6-6, these are egress tunnel entrance rules.

Table 6-8—Tunnel entrance rule for non-OAMPDU traffic from local DTE to remote DTE

<table>
<tr><th>Conditions</th><th>Actions</th></tr>
<tr><td>1. ETYPE_LEN != SP_TYPE</td><td rowspan="2">1. ADD(UMT_DST_ADD, <remote_MAC>)
2. ADD(UMT_SRC_ADD, <local_MAC>)
3. ADD(UMT_ETH_TYPE, UMT_TYPE)
4. ADD(UMT_SUBTYPE, L2_subtype)</td></tr>
<tr><td>1. ETYPE_LEN == SP_TYPE
2. XPDU_SUBTYPE != OAM_subtype</td></tr>
<tr><td colspan="2">NOTE:
<local_MAC > - MAC address associated with the loopback port in the local DTE
<remote_MAC > - MAC address associated with the loopback port in the remote DTE

SP_TYPE – Slow Protocol Ethertype value (see IEEE Std 802.3, 57A.4)
UMT_TYPE – Ethertype value identifying UMTPDUs (see 5.1)

OAM_subtype – UMT subtype value identifying OAMPDU payload (see 5.2)
L2_subtype – UMT subtype value identifying L2 encapsulation payload (see 5.2)</td></tr>
</table>

Table 6-9 illustrates the tunnel exit rule for the UMT L2 encapsulation tunnel from the local DTE to the remote DTE. If this rule is provisioned in the bridge adjacent to the remote DTE, as illustrated in Figure 6-4 and Figure 6-5, this rule is an egress tunnel exit rule. If the rule is provisioned in the remote DTE itself, as illustrated in Figure 6-6, this rule is an ingress tunnel exit rule.

Table 6-9—Tunnel exit rule for non-OAMPDU traffic from local DTE to remote DTE

<table>
<tr><th>Conditions</th><th>Actions</th></tr>
<tr><td>1. DST_ADDR == <remote_MAC>
2. SRC_ADDR == <local_MAC>
3. ETH_TYPE == UMT_TYPE
4. UMT_SUBTYPE == L2_subtype</td><td>1. REMOVE(UMT_DST_ADDR)
2. REMOVE(UMT_SRC_ADDR)
3. REMOVE(UMT_ETH_TYPE)
4. REMOVE(UMT_SUBTYPE)</td></tr>
<tr><td colspan="2">NOTE:
<local_MAC > - MAC address associated with the loopback port in the local DTE
<remote_MAC > - MAC address associated with the loopback port in the remote DTE

UMT_TYPE – Ethertype value identifying UMTPDUs (see 5.1)
L2_subtype – UMT subtype value identifying L2 encapsulation payload (see 5.2)</td></tr>
</table>

The entrance rules for the return tunnel (from the remote DTE back to the local DTE), the rules are similar to the rules shown in Table 6-8, but with <*local_MAC*> and <*remote_MAC*> values swapped. Similarly, the tunnel exit rule is as shown in Table 6-9, but also with <*local_MAC*> and <*remote_MAC*> values swapped.