Contents

6 UMT sublayer

6.1 UMT Classification and Translation Engine

The function of the UMT Classification and Translation Engine (CTE) is to classify frames by certain criteria and to perform specific modification on the frames that match the criteria. The classification criteria together with the associated modification action comprise an entity called a *rule*. The concept of a rule is similar to that defined in IEEE 1904.1, 6.5.2.1. By matching frames to specific rules, the CTE is able to translate UMTPDUs into xPDUs (i.e., into frames with different Ethertype values) and vice versa. ~~A frame that does not match any CTE rules traverses the UMT sublayer without any modifications.~~

There are separate CTE instances in the transmit path and in the receive path of each physical or virtual port. The CTE located in the receive path is called *Ingress CTE* and the CTE located in the transmit path is called *Egress CTE* (see Figure 6-1). Fundamentally, a CTE instance is simply a table that stores multiple rules. Some of the rules are statically pre-configured (i.e., available and active at all times); other rules are dynamically added/deleted by NMS when tunnels are established or destroyed.

<Replace figure 6-1 with the following figure>

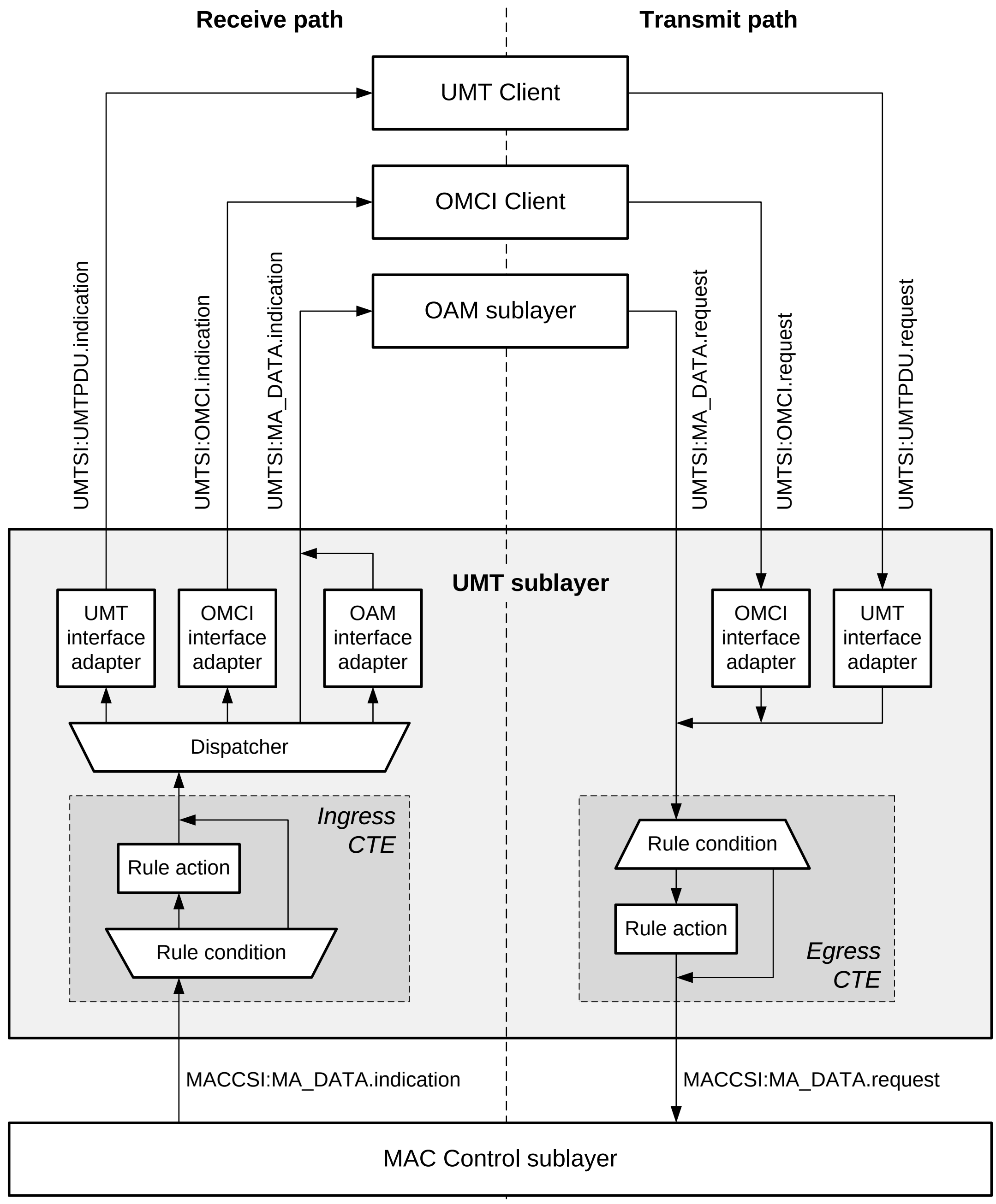

Figure 6-1—UMT sublayer functional block diagram

6.1.1 CTE rule structure

<as is>

6.1.1.1 CTE rule classification conditions

<a is>

6.1.1.1.1 Comparison operators

<a is>

6.1.1.1.2 Classification fields

<a is>

6.1.1.2 CTE rule modification actions

<a is>

6.1.2 CTE rule categories

<a is>

6.1.3 CTE rules involving operations on the VLAN tags

The classification clauses in the CTE rules may classify the incoming xPDUs and UMTPDUs based on *VLAN0* or *VLAN1* fields, or based on some sub-fields of these fields (see Table 6-2).

The action clauses in the CTE rules may add VLAN0 and VLAN1 tags to UMTPDUs or delete these tags from UMTPDUs. When performing a translation of an xPDU into a UMTPDU, and if the original xPDU includes any VLAN tags, the action clauses may also copy these tags from xPDU into UMTPDU. The COPY operation leaves the VLAN tags in the original xPDU intact.

Even though the UMT sublayer may be configured to manipulate VLAN tags in UMTPDUs, it does not imply that a given UMT-aware device is also VLAN-aware and that it is a participant in Multiple VLAN Registration Protocol (MVRP). The VLAN manipulation applied by the UMT sublayer is entirely based on the provisioned CTE rules and not on any higher-layer protocol behavior or device configuration. In a VLAN-enabled L2 network, the management entity responsible for UMT port configuration and provisioning is expected to be aware of VLAN topology and to participate in MVRP if necessary.

6.2 Receive path specification

6.2.1 Principles of operation

The receive path of the UMT sublayer includes the Receive process. The Receive process waits for a frame to be received on MACCSI:MA_DATA interface (via `MACCSI:MA_DATA.request()` primitive, as defined in 4.4). When a frame is received, it is processed by the ingress Classification and Translation Engine (CTE) and if match is found, the frame is modified according to the matched rule action. If the frame does not match any rules, it is passed through the CTE block unmodified.

After traversing the ingress CTE block (highlighted in Figure 6-4), the frame is dispatched to one of the UMTSI interfaces: (UMTSI:UMTPDU, UMTSI:OMCI, or UMTSI:MA_DATA). The dispatching decision is based on the values of the MAC destination address, Ethertype, and UMT subtype.

The UMTPDUs with the destination address matching the local MAC address and the UMT subtype equal to `UMT_SUBTYPE` (see Table 5.1) are modified to match the parameters expected by the `UMTSI:UMTPDU.indication()` primitive (see 4.4.x) and are passed to the UMTSI:UMTPDU interface.

The UMTPDUs with the destination address matching the local MAC address and the UMT subtype equal to `OAM_SUBTYPE` (see Table 5.1) are converted into OAMPDUs and are passed to the UMTSI:MA_DATA interface.

The UMTPDUs with the destination address matching the local MAC address and the UMT subtype equal to `OMCI_SUBTYPE` (see Table 5.1) are modified to match the parameters expected by the `UMTSI:OMCI.indication()` primitive (see 4.4.y) and are passed to the UMTSI:OMCI interface.

All other xPDUs are passed unmodified to the UMTSI:MA_DATA interface. Note that there still may be other local clients that will intercept/consume these xPDUs at a higher layer.

The Receive process does not discard any frames, i.e., every `MACCSI:MA_DATA.indication()` primitive results in a generation of a single indication primitive on either UMTSI:UMTPDU, UMTSI:OMCI, or UMTSI:MA_DATA interface.

Note that no provisioning of the ingress tunnel exit rules is required in situations where the tunnel is terminated at the same port where the xPDUs are to be consumed by their respective clients. The functionality to convert UMTPDUs into xPDUs is built-in into the Receive process.

6.2.2 Constants

`DST_ADDR`

This constant identifies a field in a frame, as defined in Table 6.1.

`ETH_TYPE_LEN`

This constant identifies a field in a frame, as defined in Table 6.1.

`LOCAL_MAC_ADDR`

TYPE: 48-bit MAC address

This constant holds the value of the MAC address associated with the port where the Receive process state diagram is instantiated. Some devices may associate the same MAC address value with multiple ports. The format of MAC address is defined in IEEE Std 802.3, 3.2.3.

VALUE: device-specific

`OMCI_SUBTYPE`

This constant represents a UMTPDU subtype as defined in Table 5.1.

`SP_ADDR`

This constant holds the value of the destination MAC address associated with Slow Protocols (see IEEE Std 802.3, 57A.3).

`SP_TYPE`

This constant holds the value of the Ethertype identifying the Slow Protocol (see IEEE Std 802.3, 57A.4).

`SRC_ADDR`

This constant identifies a field in a frame, as defined in Table 6.1.

`SUBTYPE`

This constant identifies a field in a frame, as defined in Table 6.1.

`UMT_ETHERTYPE`

TYPE: 16-bit Ethertype

This constant holds the Ethertype value identifying the UMTPDUs.

VALUE: 0xA8-C8

`UMT_SUBTYPE`

This constant represents a UMTPDU subtype as defined in Table 5.1.

6.2.3 Variables

`IngressRuleId`

TYPE: 16-bit unsigned integer

This variable identifies one of the provisioned CTE ingress rules. It also may have a special value `none` that does not identify any of the provisioned rules.

`RxInputPdu`

TYPE: structure containing an Ethernet frame

This variable holds an Ethernet frame received from the MACCSI:MA_DATA interface. The fields of this structure correspond to the parameters of the `MA_DATA.indication()` primitive, as defined in IEEE Std 802.3, 2.3.2.

`RxOutputPdu`

TYPE: structure containing an Ethernet frame

This variable holds an Ethernet frame to be passed to one of the the UMTSI interfaces (UMTSI:UMTPDU, UMTSI:OMCI, or UMTSI:MA_DATA). The fields of this structure correspond to the parameters of the `MA_DATA.indication()` primitive, as defined in IEEE Std 802.3, 2.3.2.

Additionally, the `RxOutputPdu` structure supports the `RemoveField(field_code)` method, which removes a field identified by the `field_code` from the structure. Thus, unlike the `RxInputPdu` structure, the `RxOutputPdu` may contain only a partial Ethernet frame. The `field_code` parameter takes values as defined in Table 6.1.

6.2.4 Functions

`CheckIngressRules(input_pdu)`

This function returns the identification of an ingress rule that matched the frame contained in `RxInputPdu` structure. If multiple rules macthed the frame, the function returns an identification of any of these rules. If none of the rules matched the frame, a special value `none` is returned.

`Modify(rule_id, input_pdu)`

This functions returns a frame that is a result of applying the modification action(s) of the rule identified by the `rule_id` parameter to the frame contained in the `input_pdu` parameter.

6.2.5 Primitives

The primitives referenced in this state diagram are defined in 4.4.

6.2.6 State Diagram

UMT sublayer shall implement the Receive process as defined in the state diagram in Figure 6-4.

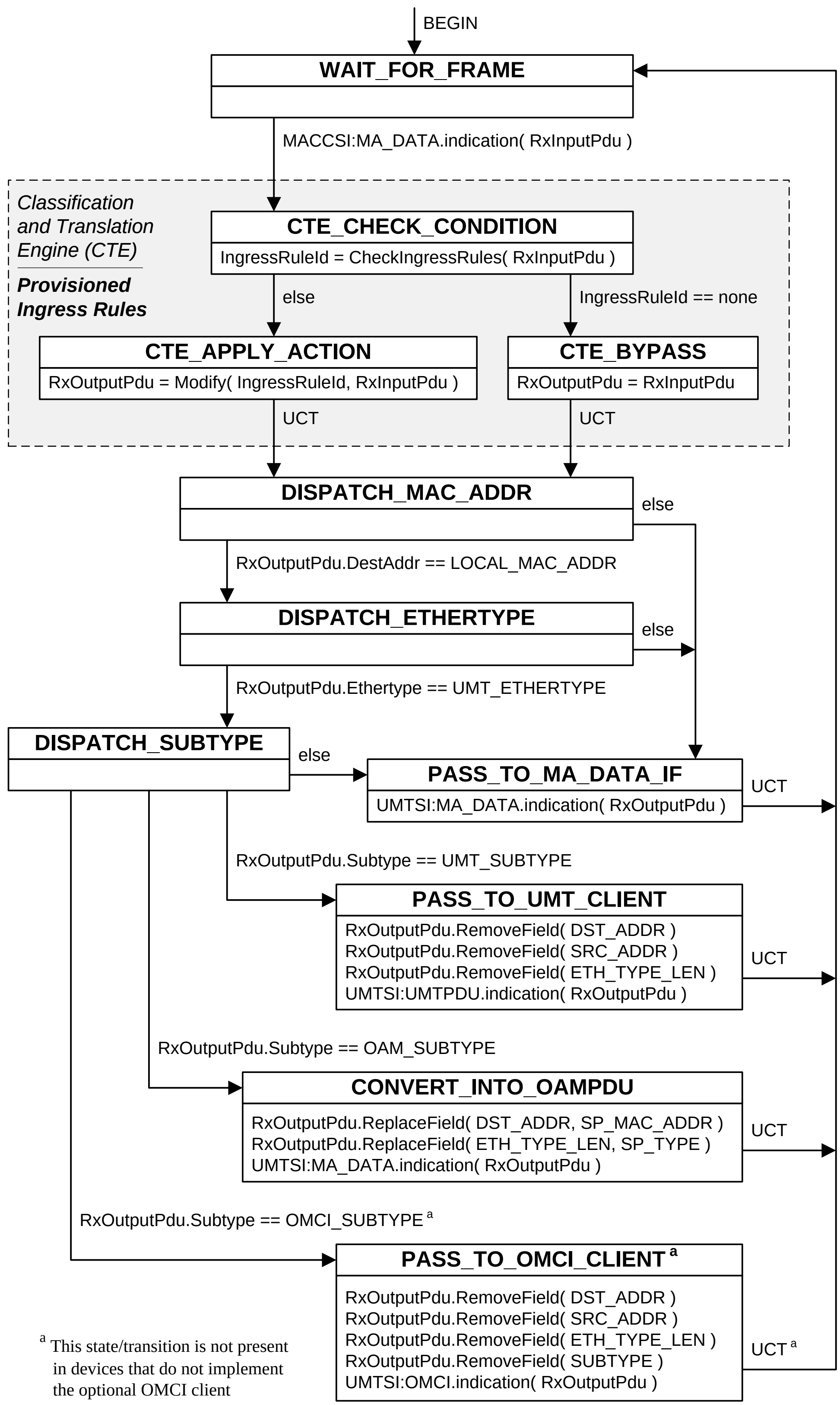

Figure 6-4—Receive process state diagram

6.3 Transmit path specification

6.3.1 Principles of operation

The transmit path of the UMT sublayer includes the Transmit process. The Transmit process waits for an xPDU to be received from one of the UMTSI interfaces: (UMTSI:MA_DATA, UMTSI:UMTPDU, or UMTSI:OMCI).

If an UMT xPDU is received from the UMTSI:UMTPDU interface, it is converted into UMTPDU with subtype `UMT_CONFIG` (see Table 5.1) by prepeding a UMTPDU header to the UMT xPDU payload. The header cosnsists of the destination address, source address, and Ethertype fields. Note that both the destination and the source addresses are equal to the local MAC address assigned to the given port.

If an OMCI xPDU is received from the UMTSI:OMCI interface, it is converted into UMTPDU with subtype `OMCI_SUBTYPE` (see Table 5.1) by prepeding a UMTPDU header to the UMT xPDU payload. The header cosnsists of the destination address, source address, Ethertype, and subtype fields. Note that both the destination and the source addresses are equal to the local MAC address assigned to the given port.

After the above modifications, the UMT or OMCI xPDU is formed into a complete frame, which is then processed by the Egress Classification and Translation Engine (CTE). If match is found, the frame is modified according to the matched rule action. If the frame does not match any rules, it is passed through the CTE block unmodified.

Note that to enter a tunnel, the UMT xPDU or the OMCI xPDU require a matching egress CTE rule that, as a minimum, overwrites the local MAC address value in the UMTPDU destination address field with the MAC address associated with the xPDU destination for the given tunnel.

6.3.2 Constants

The constants referenced in this state diagram are defined in 6.2.2.

6.3.3 Variables

`EgressRuleId`

TYPE: 16-bit unsigned integer

This variable identifies one of the provisioned CTE egress rules. It also may have a special value `none` that does not identify any of the provisioned rules.

`TxInputPdu`

TYPE: structure containing an Ethernet frame

This variable holds a PDU received from one of the the UMTSI interfaces (UMTSI:UMTPDU, UMTSI:OMCI, or UMTSI:MA_DATA). When received from the UMTSI:MA_DATA interface, the `TxInputPdu` structure contains a complete and properly-formed Ethernet frame. When received from UMTSI:UMTPDU or UMTSI:OMCI interfaces, the `TxInputPdu` structure contains a partial frame, that only includes the parameters defined for the respective `request()` primitive (see 4.4).

Additionally, the `TxInputPdu` structure supports the `AddField(field_code, field_value)` method, which adds a field identified by the `field_code` and having the value `field_value` to the structure. The `field_code` parameter takes values as defined in Table 6.1.

`TxOutputPdu`

TYPE: structure containing an Ethernet frame

This variable holds an Ethernet frame to be passed to the MACCSI:MA_DATA interface. The fields of this structure correspond to the parameters of the `MA_DATA.request()` primitive, as defined in IEEE Std 802.3, 2.3.1. A CTE egress rule is considered misconfigured if applying this rule to the `TxInputPdu` results in a malformed Ethernet frame being stored in the `TxOutputPdu` structure.

6.3.4 Functions

`CheckEgressRules(input_pdu)`

This function returns the identification of an egress rule that matched the the frame contained in `TxInputPdu` structure. If multiple rules macthed the frame, the function returns an identification of any of these rules. If none of the rules matched the frame, a special value `none` is returned.

`Modify(rule_id, input_pdu)`

This functions is defined in 6.2.4.

6.3.5 Primitives

The primitives referenced in this state diagram are defined in 4.4.

6.3.6 State Diagram

UMT sublayer shall implement the Transmit process as defined in the state diagram in Figure 6-5.

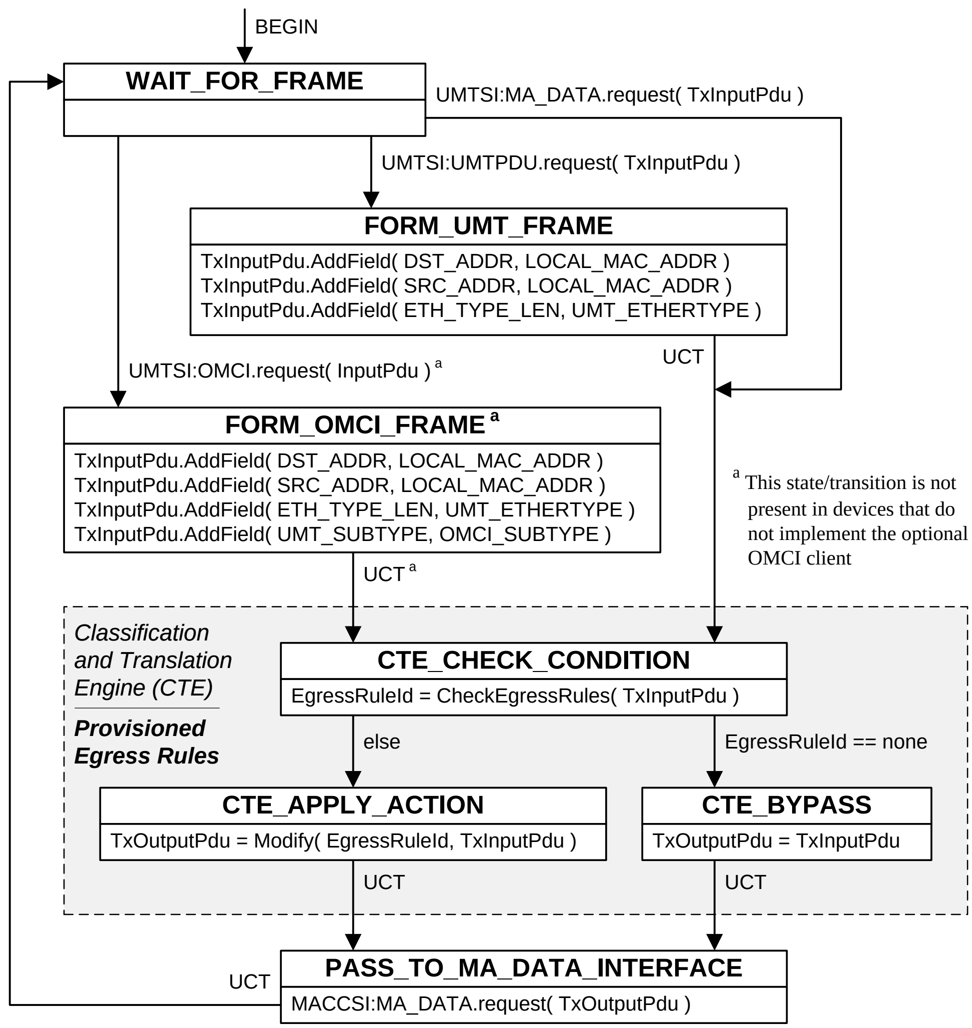

Figure 6-5—Transmit process state diagram