# 12. Media-dependent layer specification for IEEE 802.11 links

## 12.1 Overview

### 12.1.1 General

Accurate synchronized time is distributed across a domain through time measurements between adjacent PTP Instances in a packet network. Time is communicated from the root of the clock spanning tree (i.e., the Grandmaster PTP Instance) to the leaves of the tree, by recursively propagating time from a leaf-facing "master" port to some number of root-facing "slave" ports in PTP Instances at the next level of the tree through measurements made across the links connecting the PTP Instances. While the time semantics are consistent across the time-aware packet network, the method for communicating synchronized time from a master port to the immediate downstream link partner varies depending on the type of link interconnecting the two systems.

This clause specifies the interface primitives and state machines that provide accurate synchronized time across wireless IEEE 802.11 links as part of a packet network. This clause builds upon time measurement features defined in IEEE Std 802.11-2016 and makes no distinction between stations with an access point function and stations without an access point function.

### 12.1.2 IEEE 802.11 Timing Measurement and Fine Timing Measurement procedures

#### 12.1.2.1 General

IEEE Std 802.11-2016 defines a family of wireless measurements, including both "Timing Measurement" (TM) and "Fine Timing Measurement" (FTM), which captures timestamps of the transmit time and receive time of a round-trip message exchange between associated wireless local area network (WLAN) stations.

In contrast to the protocol defined for full-duplex point-to-point links, this clause does not define any new frames nor the transmission of any frames. Rather, it makes use of a MAC layer management entity (MLME) interface, which causes the IEEE 802.11 layer to not only take timestamps of measurement frames as they are transmitted and received, but to also *generate* and *consume* the measurement frames, all within the IEEE 802.11 MLME layer, and then to provide timestamp information from the MLME to this media-dependent layer through a set of well-defined service primitives. However, as an aid to the reader, the protocol and frames used by the IEEE 802.11 MLME for both Timing Measurement and Fine Timing Measurement are described briefly as follows and illustrated in Figure 12-1 and Figure 12-2, respectively.

Both Timing Measurement and Fine Timing Measurement are accomplished through a round-trip frame exchange. For Timing Measurement, the first frame of the round-trip measurement is generated by the master within the IEEE 802.11 MLME when the MLME-TIMINGMSMT.request primitive is invoked. For Fine Timing Measurement, an initial Fine Timing Measurement request frame is generated by the slave within the IEEE 802.11 MLME when the MLME-FINETIMINGMSMTRQ.request primitive is invoked. After this frame is successfully received by the master, the first frame of the round-trip measurement is generated by the master within the IEEE 802.11 MLME when the MLME-FINETIMINGMSMT.request primitive is invoked. As defined by IEEE Std 802.11-2016, upon receipt of the resulting Timing Measurement or Fine Timing Measurement frame, the slave station transmits an IEEE 802.11 Ack control frame to the master station. Four timestamps are captured during this two-frame exchange, as follows:

a) t1 is when (in the master station's time base) the request frame is transmitted
b) t2 is when (in the slave station's time base) the request frame is received
c) t3 is when (in the slave station's time base) the Ack control frame is transmitted
d) t4 is when (in the master station's time base) the Ack control frame is received

When the master sends either a Fine Timing Measurement or a Timing Measurement frame, it passes the t1 and t4 timestamps (and other end-to-end synchronization information) and FollowUpInformation, from the previous measurement to the slave. A pair of tokens is passed in each timing or Fine Timing Measurement frame, one to identify the current measurement and the other to allow the slave to associate the timestamp information with the previous measurement.

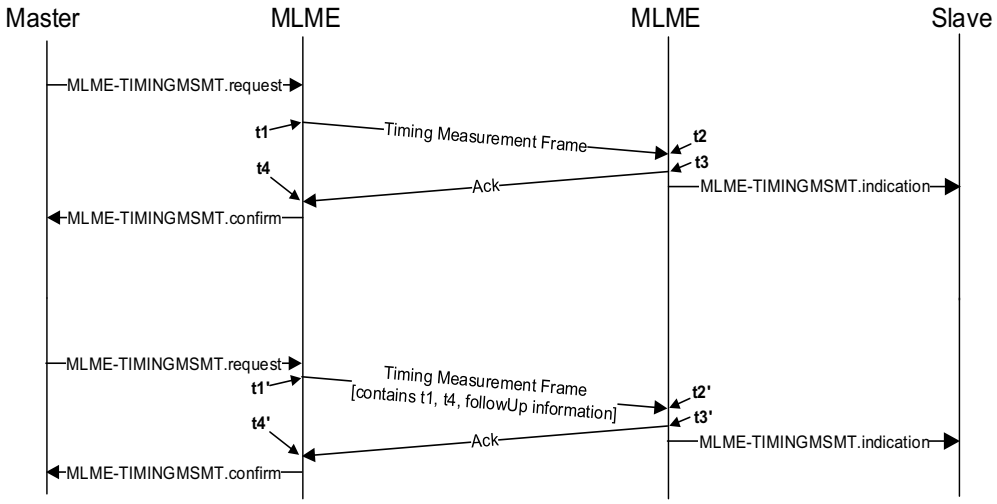


**Figure 12-1—Timing measurement procedure for IEEE 802.11 links**

NOTE 1—TM also can include a Timing Measurement Request Frame; however, this frame type is not used by this standard.

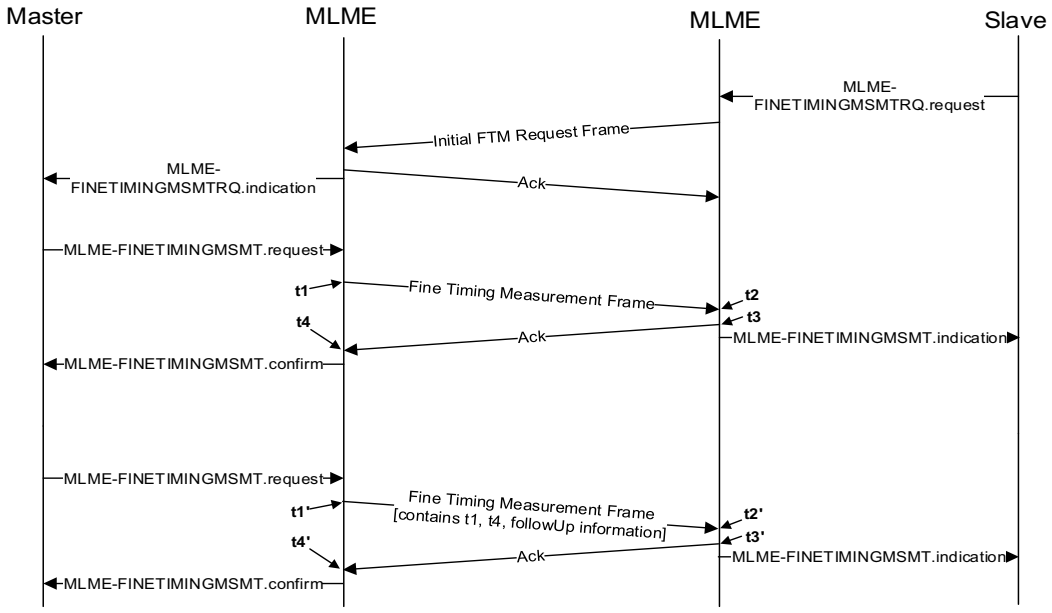


**Figure 12-2—Fine Timing Measurement procedure for IEEE 802.11 links**

Note that, unlike full-duplex point-to-point ports, IEEE 802.11 ports do not compute the link delay measurements in both directions since only a PTP Port in the Slave state makes use of that information. A PTP Port that transitions from the Master state to the Slave state (e.g., due to selection of a new Grandmaster PTP Instance) can collect a number of link delay measurements and perform averaging or other filtering to achieve the desired accuracy.

NOTE 2—Fine Timing Measurement can be used for time synchronization as described in this standard; however, it can also be used for location services as defined in IEEE Std 802.11-2016. Since FTM supports only one configuration at a time, it is possible that setting that single configuration for time synchronization might disable its previous configuration for use by another application for location services or, conversely, an application that configures FTM for location services could disable this standard's use for time synchronization. Implementations that use FTM for time synchronization and other applications need to coordinate usage of the FTM protocol.

The master generates MLME-TIMINGMSMT.request primitives for Timing Measurement, as described in this standard, in a manner such that the requirements of 10.7.2.3 and 12.8 for the time synchronization message interval are satisfied. Timing measurement frames are then sent from the master to the slave continuously and at a rate that satisfies the requirements of those two subclauses. It is not necessary for the slave to continually request timing information from the master. In contrast, the slave must request timing information from the master for Fine Timing Measurement. A Fine Timing Measurement frame carries timestamp information for a previous measurement. The slave requests a burst of Fine Timing Measurement frames from the master. Figure 12-2 shows an example of a burst of Fine Timing Measurement frames (there are two frames in that example). The Fine Timing Measurement process is described in more detail in 12.1.2.2 and is illustrated in Figure 12-3. In that discussion, the focus is on the transmission of the frames. In the simplified example, service primitives are omitted from Figure 12-3.

#### 12.1.2.2 Detailed description of Fine Timing Measurement (FTM)

Figure 12-3 is adapted from Figure 11-37 of IEEE Std 802.11-2016. Additional details on the FTM procedure are given in 11.24.6 of IEEE Std 802.11-2016. The example of this figure is for when the initiating station (STA), i.e., the slave, requests a single burst of three FTM frames from the responding STA, i.e., the master, as soon as possible. The slave makes this request by sending an initial FTM Request to the master with respective parameters set to appropriate values. The FTM parameters that are relevant to time synchronization in this standard are described in 12.6, and all the FTM parameters are described in more detail in 9.4.2.168 of IEEE Std 802.11-2016. However, in the example here, the parameter ASAP is set to 1 to indicate to the master that the FTM frames are desired as soon as possible, and the Number of Bursts Exponent parameter is set to 0 to indicate a single burst. Figure 12-3 is a simplified view; the slave causes the frame to be sent by invoking the MLME-FINETIMINGMSMTRQ.request primitive, which includes the FTM parameter values. The master sends an acknowledgment (Ack) frame to the slave to indicate it received the initial FTM request. The master then sends an initial FTM frame at a time that is recommended to be no more than 10 ms later than the receipt of the initial FTM request. The initial FTM frame indicates to the slave whether the master was able to grant the values of the FTM parameters that the slave requested. If the requested parameters are granted, the procedure continues (Figure 12-3 illustrates this case). If the requested parameters are not granted, the slave sends a new initial FTM request for a burst of two FTM frames. If the new request is granted, the procedure continues. If the new request is not granted, the slave and master use TM if they both support TM. If, at this point, the slave or the master, or both, do not support TM, the procedure terminates and asCapable is set to FALSE (see 12.4).

NOTE—IEEE Std 802.11-2016 allows various options in case the master does not grant the request. The above procedure is used in this subclause.

The initial FTM frame (initial FTM_1 in Figure 12-3) sent by the master is timestamped with the value t1_1 on transmission from the master and timestamped with the value t2_1 on receipt by the slave. The initial FTM frame has fields that carry the t1 and t4 timestamps of the previous FTM frame and corresponding Ack; however, since this is the first FTM frame of the burst, these fields are set to zero. The slave responds to the master with an Ack, which is timestamped with the value t3_1 on transmission from the slave and with the value t4_1 on receipt by the master.

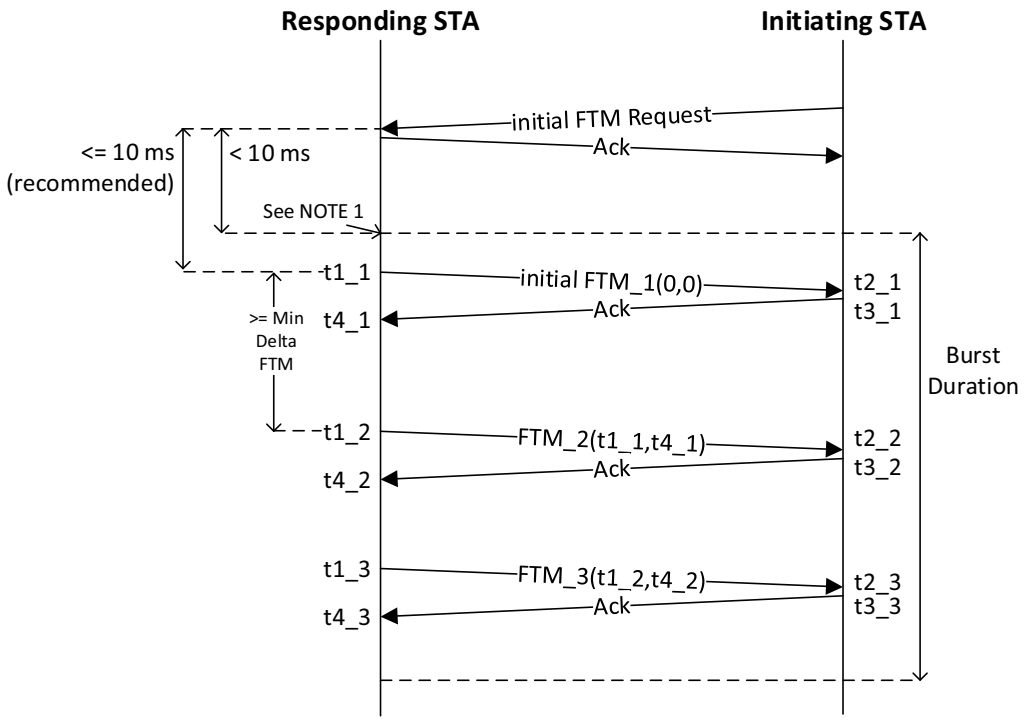


**Figure 12-3—Illustration of Fine Timing Measurement burst**

The master sends the second FTM frame (FTM_2 in Figure 12-3) after a time interval since sending the initial FTM frame that is greater than or equal to the Min Delta FTM parameter that was requested by the slave. This FTM frame is timestamped with t1_2 on transmission and t2_2 on reception. This FTM frame also carries the values of the timestamps t1_1 and t4_1 of the initial FTM frame and corresponding Ack. On receipt of the second FTM frame, the slave sends an Ack frame to the master; this frame is timestamped with t3_2 on transmission and t4_2 on reception. Finally, the master sends the third FTM frame (FTM_3 in Figure 12-3) to the slave, also at a time interval since sending FTM_2 that is greater than or equal to the Min Delta FTM parameter that was requested by the slave. As with FTM_2, this frame is timestamped with t1_3 on transmission and t2_3 on reception. This FTM frame also carries the values of the timestamps t1_2 and t4_2 of the second FTM frame and corresponding Ack. On receipt of the third FTM frame, the slave sends an Ack frame to the master; this frame is timestamped with t3_3 on transmission and t4_3 on reception.

On completion of the above exchanges of FTM frames and corresponding acknowledgments, the slave knows the transmission and reception times for the initial FTM frame (t1_1, t2_1, t3_1, and t4_1) and second FTM frame (t1_2, t2_2, t3_2, and t4_2) and the reception time for the third FTM frame (t2_3) and transmission time for the corresponding Ack (t3_3). The slave can use this information (along with FollowUpInformation contained in the VendorSpecific information element; see 12.7) to synchronize to the master. In this standard, timestamps for the minimum delay FTM frames are used. Specifically, the slave computes the quantities D1 = t2_1 – t1_1, and D2 = t2_2 – t1_2, and uses the timestamps t1_*i* and t2_*i*, where *i* = 1 if D1 < D2 and *i* = 2 if D1 ≥ D2, to compute the respective members of the MDSyncReceive structure. The timestamps of FTM_3 and its corresponding Ack are not used; FTM_3 is used only to convey the timestamps of FTM_2 and its corresponding Ack.

If the master does not grant the parameters requested initially by the slave, i.e., for a burst of three FTM frames, but it does grant the subsequent request for a burst of two FTM frames, the slave has a full set of timestamps for only the initial FTM_1 frame. In this case, the slave uses the timestamps t1_1, t2_1, t3_1, and t4_1 to compute the respective members of the MDSyncReceive structure.

With the above procedure for FTM, the slave controls the rate at which time synchronization information is sent from the master. This is different from TM, full-duplex IEEE 802.3, IEEE 802.3 EPON, and CSN transports. In those cases, the sending of time synchronization information from the master to the slave is controlled by the master; this is true for syncLocked (see 10.2.5.15) TRUE, in which case the information is sent as soon as it is received from further upstream, and syncLocked FALSE, in which case it is sent independently of information received from further upstream. For FTM, the slave requests time synchronization information from the master at an average rate equal to the inverse of the current synchronization message interval currentLogSyncInterval (see 12.8 and 14.8.18). In addition, the actual intervals between successive requests by the slave for time synchronization information meet the requirements of 10.7.2.3. Also, the value of syncLocked at the master port will not affect the sending of time synchronization information from the master to the slave; the requests for time synchronization information from the slave are asynchronous to the receipt of time synchronization information from upstream at the node that contains the master port.

### 12.1.3 Layering for IEEE 802.11 links

The *media-dependent* (MD) entity is tailored to the link technology and is responsible for translating the PortSync entity's media-independent actions to media-dependent PDUs or primitives as necessary for communicating synchronized time from the master port over the link to a single slave port. For an IEEE 802.11 link, this one-to-one relationship between the MD entities of the master and slave implies that if the one physical IEEE 802.11 port is associated with multiple stations, each association requires its own instantiation of the IEEE 802.1AS PortSync entity and MD entity. The MLME-TIMINGMSMT and MLME-FINETIMINGMSMT service primitives defined in IEEE Std 802.11-2016 are used to perform Timing Measurements and Fine Timing Measurements, respectively, between a master IEEE 802.11 station and associated IEEE 802.11 slave stations. Figure 12-4 illustrates how the MD entity interacts with the higher and lower layers.

## 12.2 Messages

All media-dependent frames are generated and consumed by the lower-layer IEEE 802.11 MLME and thus none are defined here. Also, since the IEEE 802.11 event messages are timestamped by the MAC/PHY, the timestamp point is defined in IEEE Std 802.11-2016 as well. Media-independent messages, i.e., Announce and Signaling messages, are transmitted using the unicast address of the WLAN station instead of the group address defined in 10.5.3.

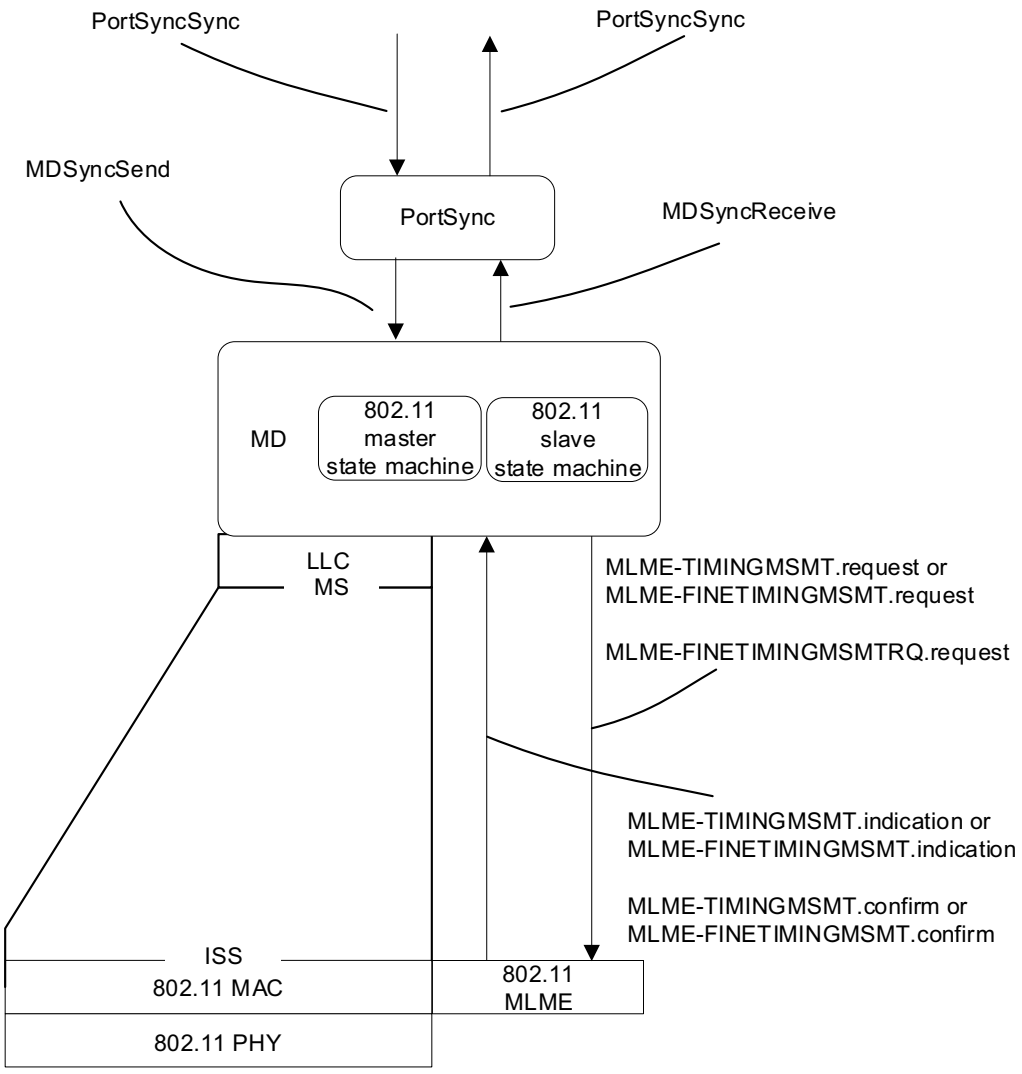


**Figure 12-4—Media-dependent and lower entities in stations with IEEE 802.11 links**

## 12.3 Determination of Timing Measurement and Fine Timing Measurement capability

The bits of the per-PTP Port global variable tmFtmSupport (see 12.5.1.5) shall be set as indicated in Table 12-1.

**Table 12-1—Values of bits of tmFtmSupport**

| Bit | Value |
|---|---|
| 0 | TRUE if :<br>a) The port supports Timing Measurement and<br>b) The Timing Measurement bit in the Extended Capabilities information element defined in Table 9-135 of IEEE Std 802.11-2016 indicates that the peer IEEE 802.11 station is capable of participating in the Timing Measurement protocol.<br><br>FALSE otherwise. |
| 1 | TRUE if :<br>a) The port supports Fine Timing Measurement and<br>b) The Fine Timing Measurement responder and initiator bits in the Extended Capabilities information element defined in Table 9-135 of IEEE Std 802.11-2016 indicate that the peer IEEE 802.11 station is capable of participating in the Fine Timing Measurement protocol.<br><br>FALSE otherwise. |
| 2–7 | Reserved as FALSE. |

## 12.4 Determination of asCapable

The per-PTP Port, per-domain instance of the global variable asCapable (see 10.2.5.1) is set to TRUE if the following conditions hold (see 12.5.1 and 12.5.2):

a) The value of tmFtmSupport is not zero.
b) neighborGptpCapable is TRUE.
c) At least one of the following conditions hold:
    1) Bit 0 of tmFtmSupport is TRUE.
    2) Bit 1 of tmFtmSupport is TRUE and, if the PTP Port is a master port, it can support (i.e., grant) the parameters requested by the slave with either FTMs per burst equal to 3 or FTMs per burst equal to 2.
    3) Bit 1 of tmFtmSupport is TRUE and, if the PTP Port is a slave port, the master port at the other end of the link can support (i.e., grant) the parameters requested by the slave with either FTMs per burst equal to 3 or FTMs per burst equal to 2.

If the value of domainNumber is zero (which is required for support of the 2011 edition of this standard) and bit 0 of tmFtmSupport is TRUE, asCapable can be set to TRUE. In all other instances, asCapable shall be set to FALSE.

NOTE—The above conditions ensure backward compatibility with the 2011 edition of this standard. A time-aware system that is compliant with the 2011 edition of this standard will not process the gPTP capable TLV, and asCapable will be determined as specified in the 2011 edition. A PTP Instance of a time-aware system compliant with the current edition of this standard that is attached, via an IEEE 802.11 link, to a node compliant with the 2011 edition of this standard will not receive Signaling messages that contain the gPTP capable TLV and will not set neighborGptpCapable to TRUE. However, item c) 1) in this subclause ensures that asCapable for this PTP Port and domain (i.e., domain 0) will still be set in a manner consistent with the 2011 edition of this standard.

## 12.5 State machines

### 12.5.1 Media-dependent master state machines

#### 12.5.1.1 Overview

The MD entity of an IEEE 802.11 port whose port state is MasterPort (see Table 10-2) shall behave in a manner that is indistinguishable, relative to an observer external to a system, from a strict implementation of the master state machines in Figure 12-5 and Figure 12-6 (denoted as master state machine A and master state machine B, respectively, in 12.5.1.2), the local variables specified in 12.5.1.3, the functions specified in 12.5.1.4, the shared variables specified in 12.5.1.5, and the primitives defined in 12.5.1.6.

For Timing Measurement, master state machine A is responsible for initiating a time measurement whenever the PortSync entity requests it do so, as indicated by the rcvdMDSyncDot11MasterA Boolean (see 12.5.1.3.8). Master state machine A invokes the IEEE 802.11 MLME-TIMINGMSMT.request primitive and waits for the subsequent MLME-TIMINGMSMT.confirm primitive. It collects local timestamp information from the measurement (t1 and t4, provided by the confirm primitive) and includes the information in the subsequent request. See 8.4.3 for more information on timestamps. Master state machine B is not used for Timing Measurement.

For Fine Timing Measurement, master state machine A receives and stores information from the PortSync entity. Master state machine B receives the MLME-FINETIMINGMSMTRQ.indication caused by the initial FTM request from the slave. It sets asCapable as specified in 12.4. It then generates successive MLME-FINETIMINGMSMT.request primitives to indicate to the slave whether it can grant the requested parameters and also to cause information saved by master state machine A to be sent to the slave. It receives MLME-FINETIMINGMSMT.confirm primitives caused by Acks received from the slave. It collects local timestamp information from the current measurement (t1 and t4, provided by the confirm primitive) and includes the information in the subsequent MLME-FINETIMINGMSMT.request.

#### 12.5.1.2 State diagrams

NOTE—In the computation of the burstDuration in master state machine B, the burst duration parameter from IEEE Std 802.11-2016 is converted to UScaledNs (i.e., units of $2^{-16}$ ns; see 6.3.3.2 of IEEE Std 802.11-2016). The quantity initReqParamsDot11MasterB.burstDuration–2 is the logarithm to base 2 of the burst duration, in microseconds. Also, it is assumed that the burst duration starts when the initial FTM request is received. In actuality, the timer begins by the partial TSF timer value indicated in the initial FTM frame, which is slightly after the initial FTM request is received.

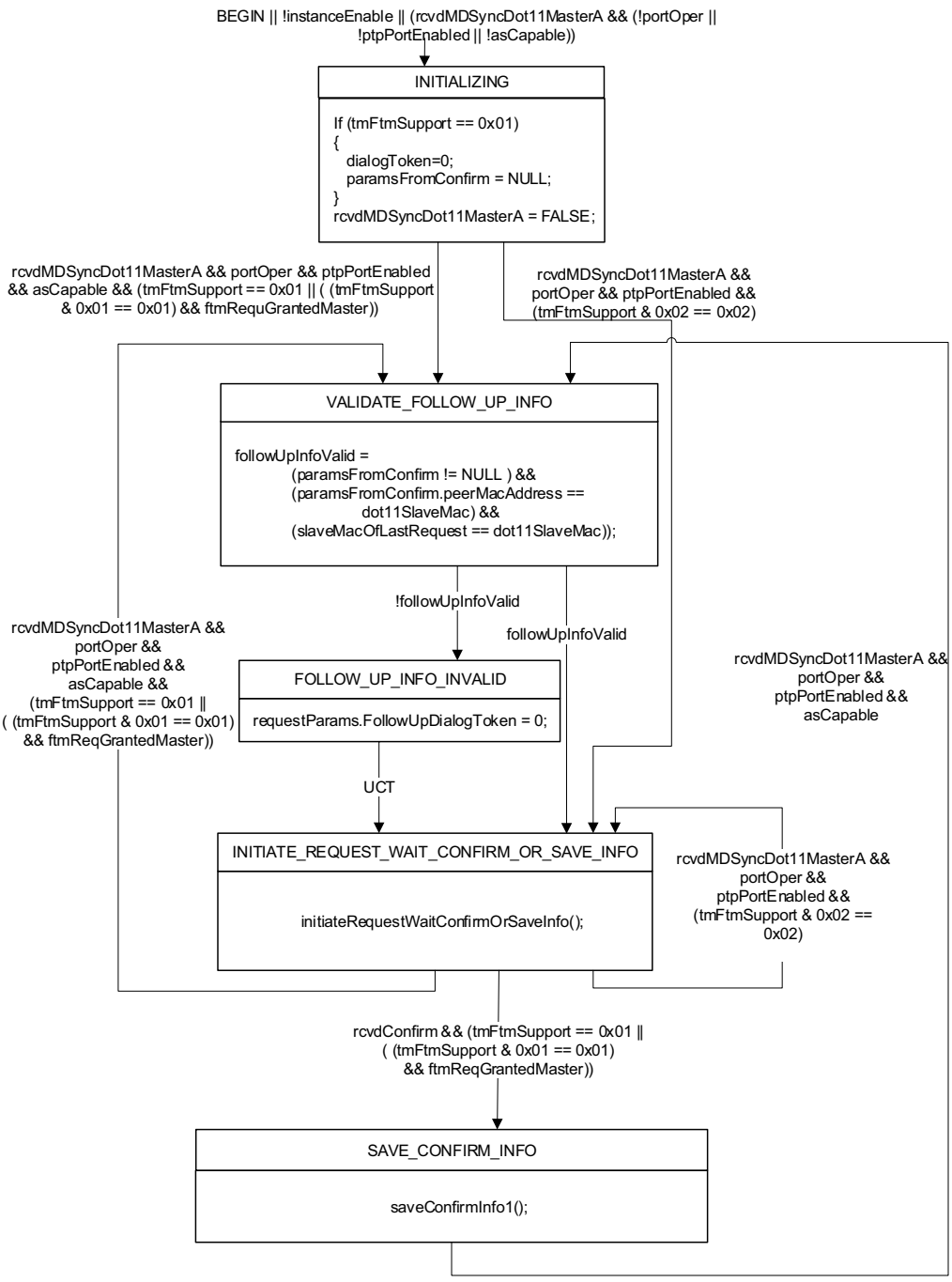


**Figure 12-5—Master state machine A**
**(a) For TM, receives information from the PortSync entity and sends to slave, and**
**(b) for FTM, receives and stores information from the PortSync entity**

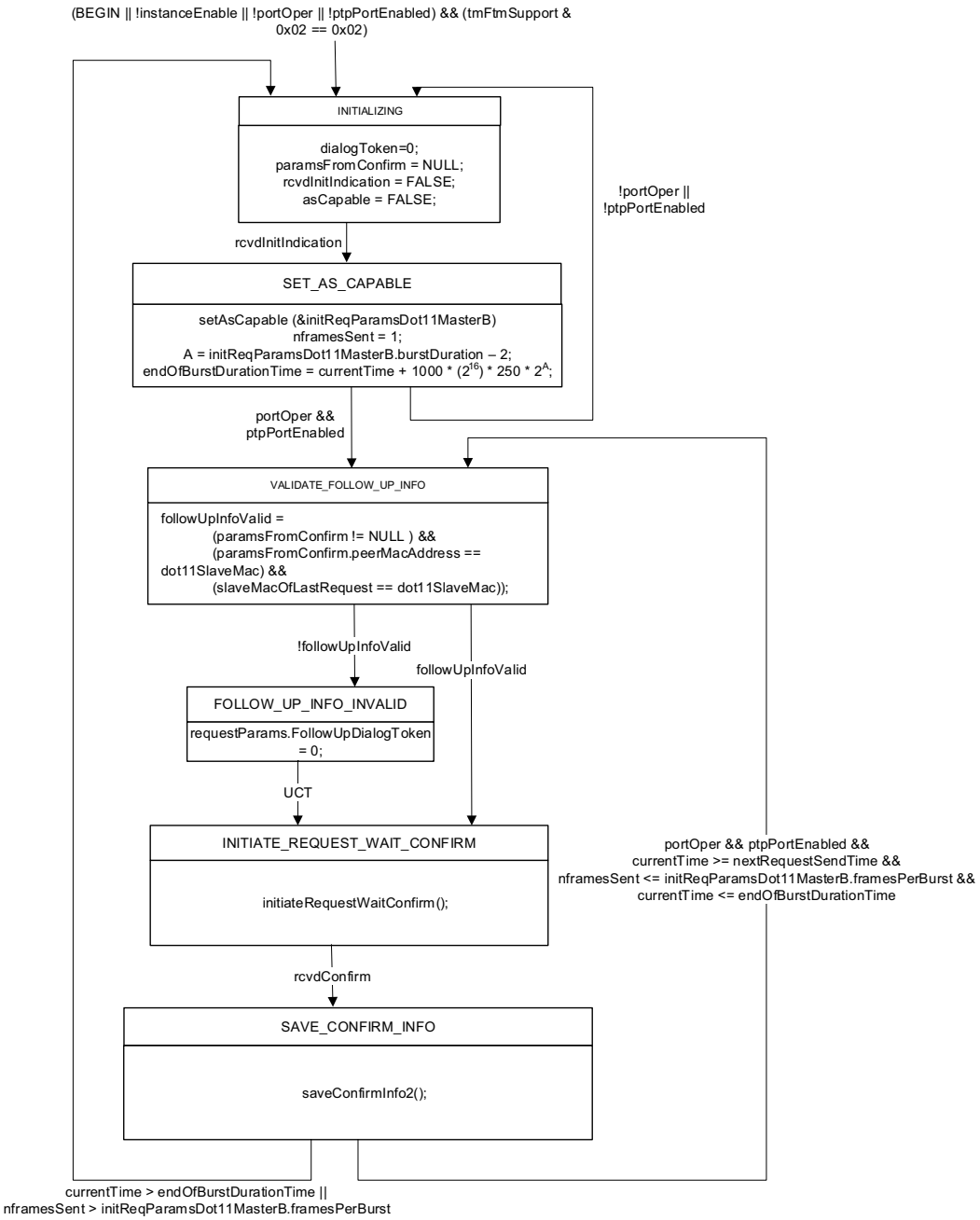


**Figure 12-6—Master state machine B**
**(a) For TM, not invoked and**
**(b) for FTM, receives initial FTM request from slave and**
**sends information received from upstream to slave in successive FTM frames**

#### 12.5.1.3 State machine local variables

**12.5.1.3.1 dialogToken:** An unsigned 8-bit integer used to identify a measurement from among those preceding and following it.

**12.5.1.3.2 followUpInfoValid:** A Boolean variable indicating whether the FollowUpInformation (e.g., timestamps and rateRatio; see 12.7) and the link partner are unchanged since the last Timing Measurement or Fine Timing Measurement.

**12.5.1.3.3 requestParams:** A structure whose members contain the values of the fields of either the MLME-TIMINGMSMT.request or MLME-FINETIMINGMSMT.request primitive.

**12.5.1.3.4 paramsFromConfirm:** A structure whose members contain the values of the fields of either the MLME-TIMINGMSMT.confirm or MLME-FINETIMINGMSMT.confirm primitive.

**12.5.1.3.5 dot11SlaveMac:** The MAC address of the station associated with the current port.

**12.5.1.3.6 slaveMacOfLastRequest:** The MAC address of the station of the previous request, used to validate FollowUpInformation.

**12.5.1.3.7 residenceTime:** A temporary variable that holds the computation of the time between receipt of the last synchronization information and transmission of synchronization information.

**12.5.1.3.8 rcvdMDSyncDot11MasterA:** A Boolean variable that is set to TRUE when an MDSyncSend structure is provided by the PortSync entity.

**12.5.1.3.9 rcvdConfirm:** A Boolean variable that is set to TRUE when either the MLME-TIMINGMSMT.confirm or MLME-FINETIMINGMSMT.confirm primitive is received.

**12.5.1.3.10 nframesSent:** An unsigned 8-bit integer used to count the number of frames sent by the slave (the number of indications received is counted).

**12.5.1.3.11 rcvdInitIndication:** A Boolean variable that is set to TRUE when the initial MLME-FINETIMINGMSMTRQ.indication primitive is received.

**12.5.1.3.12 initReqParamsDot11MasterB:** A structure whose members contain the values of the fields of an MLME-FINETIMINGMSMTRQ.indication primitive.

**12.5.1.3.13 endOfBurstDurationTime:** A UScaledNs variable whose value is the time at which the current burst ends.

**12.5.1.3.14 nextRequestSendTime:** A UScaledNs variable whose value is the expected time that the next MLME-FINETIMINGMSMTRQ.indication primitive, for the next request from the slave, will be received.

#### 12.5.1.4 State machine functions

**12.5.1.4.1 setRequestParams(&requestParams, MDSyncSend):** Assigns values to the parameters of the request primitive of either MLME-TIMINGMSMT or MLME-FINETIMINGMSMT (see 12.5.1.6) as follows:

a) Members of the FollowUpInformation member of the VendorSpecific information element, as defined in 12.7, are assigned as defined in 11.4.4 with the exception of the correctionField, which is assigned, for TM, by the function saveConfirmInfo1() in Master state machine A (see Figure 12-5) and, for FTM, by the function saveConfirmInfo2() in Master state machine B (see Figure 12-6).

b) The other fields of the VendorSpecific information element are assigned as follows:
   1) ElementID is assigned the value 221 as defined in Table 9-77 (Element IDs) of IEEE Std 802.11-2016, indicating that the information element is of type Vendor Specific.
   2) The Length field is set to 80.

NOTE—This is equal to the length of the Follow_Up payload defined in 11.4.4 (including the common header) plus the length of the OUI or CID field and the Type field (see Figure 12-8).

   3) The OUI or CID field is set to 00-80-C2.
   4) The Type field is set to 0.
c) Max t1 Error, Maxt4 Error are set to zero.
d) For Fine Timing Measurement frames, the location configuration information (LCI) Report and Location Civic Report are not present. All other members are left unchanged.

**12.5.1.4.2 setAsCapable (&initReqParamsDot11MasterB):** Determines the value of asCapable consistent with 12.4 and whether the master is able to grant the parameters requested by the slave. This function is used only for FTM.

**12.5.1.4.3 initiateRequestWaitConfirmOrSaveInfo():** This function is defined as indicated below. It is used in Master state machine A. It is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
initiateRequestWaitConfirmOrSaveInfo()
{
        rcvdMDSyncDot11MasterA = FALSE;

        If (tmFtmSupport == 0x01)
        {
                if ((++dialogToken % 256) == 0) dialogToken++;
                requestParams.DialogToken=dialogToken;
                requestParams.PeerMACAddress = dot11SlaveMac;
                setRequestParams(&requestParams, MDSyncSend);
                MLME-TIMINGMSMT.request(requestParams);
                requestParams.FollowUpDialogToken = 0;
                                        //In case no confirm is received
                slaveMacOfLastRequest = dot11SlaveMac;
        }
}
```

**12.5.1.4.4 saveConfirmInfo1():** This function is defined as indicated below. It is used in Master state machine A. It is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
saveConfirmInfo1()
{
        MLME-TIMINGMSMT.confirm(&paramsFromConfirm);

        requestParams.FollowUpDialogToken = paramsFromConfirm.DialogToken;
        requestParams.T1 = paramsFromConfirm.T1;
        requestParams.T4 = paramsFromConfirm.T4;

        // NOTE: In Timing Measurement, T1 is in units of 10 ns.
        // upstreamTxTime is units of 2^-16 ns.

        K = 1;
```

```
            // K is 1 for Timing Measurement.
            residenceTime = MDSyncSend.rateRatio *
                    (paramsFromConfirm.T1 * 10^K*(2^16) - MDSyncSend.upstreamTxTime);

            requestParams.VendorSpecific.correctionField =
                    residenceTime + MDSyncSend.followUpCorrectionField;
            // NOTE: T1 and T4 are timestamps from a single
            // local clock source. The roll-over of the 32-bit timestamps returned by
            // MLME-TIMINGMSMT.request and MLME-TIMINGMSMT.indication
            //      must be accounted for.
    }
```

**12.5.1.4.5 initiateRequestWaitConfirm():** This function is defined as indicated below. It is used in Master state machine B. It is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
    initiateRequestWaitConfirm()
    {
            If ((++dialogToken % 256) == 0) dialogToken++;
            If (nframesSent == initReqParamsDot11MasterB.framesPerBurst)
                    dialogToken = 0;

            requestParams.DialogToken=dialogToken;
            requestParams.PeerMACAddress = dot11SlaveMac;
            setRequestParams(&requestParams, MDSyncSend);
            // In the following statement, MinDeltaFTM, which is in units of 100
            //    microseconds, is converted to UScaledNs (i.e., units of 2^-16 ns; see 6.3.3.2)
            nextRequestSendTime = currentTime +
                    initReqParamsDot11MasterB.MinDeltaFTM * (65536 x 10^5);
            MLME-FINETIMINGMSMT.request(requestParams);
            requestParams.FollowUpDialogToken = 0;    //In case no confirm is received
            slaveMacOfLastRequest = dot11SlaveMac;
    }
```

**12.5.1.4.6 saveConfirmInfo2():** This function is defined as indicated below. It is used in Master state machine B. It is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
    saveConfirmInfo2()
    {
            MLME-FINETIMINGMSMT.confirm(&paramsFromConfirm);

            requestParams.FollowUpDialogToken = paramsFromConfirm.DialogToken;
            requestParams.T1 = paramsFromConfirm.T1;
            requestParams.T4 = paramsFromConfirm.T4;

            // NOTE: In Fine Timing Measurement, T1 is in units of 0.1 ns.
            // upstreamTxTime is units of 2^-16 ns.

            K = -3;
            // K is 1 for Timing Measurement and -3 for Fine Timing Measurement.
            residenceTime = MDSyncSend.rateRatio *
                    (paramsFromConfirm.T1 * 10^K*(2^16) - MDSyncSend.upstreamTxTime);

            requestParams.VendorSpecific.correctionField =
            residenceTime + MDSyncSend.followUpCorrectionField;
            // NOTE: T1 and T4 are timestamps from a single
            // local clock source. The roll-over of the 48-bit timestamps returned by
```

```
                // MLME-FINETIMINGMSMT.request and MLME-FINETIMINGMSMT.indication
                // must be accounted for.

                // A frame is only counted as being sent, for purposes of number of frames in a
                // burst, if a confirm is received. It is up to IEEE Std 802.11-2016 to handle the
                // case where a confirm is not received.
                nframesSent += 1;
        }
```

#### 12.5.1.5 Shared variables

**12.5.1.5.1 MDSyncSend:** A structure as defined in 10.2.2.1.

**12.5.1.5.2 portOper:** A Boolean as defined in 10.2.5.12.

**12.5.1.5.3 ptpPortEnabled:** A Boolean as defined in 10.2.5.13.

**12.5.1.5.4 asCapable:** A Boolean whose value is specified in 12.4 and 10.2.5.1.

**12.5.1.5.5 tmFtmSupport:** An Octet whose bits are interpreted as Booleans and whose values are specified in Table 12-1.

**12.5.1.5.6 ftmReqGrantedMaster:** A Boolean whose value is TRUE if the master grants the current initial FTM request and FALSE otherwise.

#### 12.5.1.6 Master primitives

##### 12.5.1.6.1 MLME-TIMINGMSMT.request

The MLME-TIMINGMSMT.request primitive is used by a master station to initiate a Timing Measurement and also communicates timestamps t1 and t4 captured by the master during a previous measurement. The primitive and its parameters are specified in 6.3.57.2 of IEEE Std 802.11-2016.

##### 12.5.1.6.2 MLME-TIMINGMSMT.confirm

The MLME-TIMINGMSMT.confirm primitive indicates that a Timing Measurement request has completed. The primitive and its parameters are specified in 6.3.57.3 of IEEE Std 802.11-2016.

##### 12.5.1.6.3 MLME-FINETIMINGMSMT.request

The MLME-FINETIMINGMSMT request primitive is used by a master station to initiate a Fine Timing Measurement and also communicates timestamps t1 and t4 captured by the master during a previous measurement. The primitive and its parameters are specified in 6.3.58.2 of IEEE Std 802.11-2016.

##### 12.5.1.6.4 MLME-FINETIMINGMSMT.confirm

The MLME-FINETIMINGMSMT request primitive indicates that a Fine Timing Measurement request has completed. The primitive and its parameters are specified in 6.3.58.3 of IEEE Std 802.11-2016.

##### 12.5.1.6.5 MLME-FINETIMINGMSMTRQ.indication

The MLME-FINETIMINGMSMTRQ.indication primitive indicates to a master station that the slave is requesting a burst of FTM frames with the indicated parameters. The primitive and its parameters are specified in 6.3.70.3 of IEEE Std 802.11-2016.

### 12.5.2 Media-dependent slave state machine

#### 12.5.2.1 Overview

The MD entity of an IEEE 802.11 port whose PTP Port state is SlavePort or PassivePort (see 10.3.6) shall behave in a manner that is indistinguishable, relative to an observer external to a system, from a strict implementation of the slave state machine in 12.5.2.2, the local variables specified in 12.5.2.3, the functions specified in 12.5.2.4, the shared variables specified in 12.5.2.5, and the primitives defined in 12.5.2.6.

The slave state machine is responsible for collecting information from the Timing measurement or Fine Timing measurement indications, constructing an MDSyncReceive structure with the relevant information, and passing the structure to the PortSync entity for further processing. In order to do this, the state machine saves locally captured timestamps (i.e., t2 and t3) received in the indication and associates them with the timestamps sent from the master port in a future indication (i.e., t1 and t4). In addition, for Fine Timing measurement, the slave state machine is responsible for generating the MLME-FINETIMINGMSMTRQ.request primitive, which causes the initial FTM request frame to be sent to the master.

#### 12.5.2.2 State diagram

Figure 12-7 presents the slave state machine. While quantities are shown to be computed from information in consecutive indications, an implementation can choose to compute over longer intervals as long as the clock performance requirements of Annex B are met.

#### 12.5.2.3 State machine local variables

**12.5.2.3.1 indParams:** A structure whose members contain the values of the fields of the MLME-TIMINGMSMT.indication primitive or MLME-FINETIMINGMSMT.indication primitive, as defined in 12.5.2.6, depending on whether TM or FTM, respectively, is used.

**12.5.2.3.2 previousIndParams:** A structure with members identical to those of indParams, used to save parameters from the previous indication.

**12.5.2.3.3 rcvdIndication:** A Boolean that is set to TRUE when either the MLME-TIMINGMSMT.indication or MLME-FINETIMINGMSMT.indication primitive is received.

**12.5.2.3.4 RESTART:** A Boolean that indicates, when FTM is being used, that a new burst should be initiated.

**12.5.2.3.5 rcvdIndicationTimeoutTime:** A UScaledNs variable whose value is the time after which the state machine will not wait any longer for the next MLME-FINETIMINGMSMT.indication, and a new burst is initiated.

**12.5.2.3.6 nframesRcvd:** An unsigned 8-bit integer used to count the number of frames received by the master in the burst (the number of indications received from the master are counted).

**12.5.2.3.7 initReqParamsDot11Slave:** A structure whose members contain the values of the fields of an MLME-FINETIMINGMSMTRQ.indication primitive.

**12.5.2.3.8 ftmsPerBurst:** The value of the FTM parameter ‘FTMs per burst’ (see 12.6), i.e., the number of FTM frames in the burst granted by the master.

**12.5.2.3.9 ftmReqGrantedSlave:** A Boolean that is TRUE if the master has granted the respective request for a burst and FALSE otherwise.

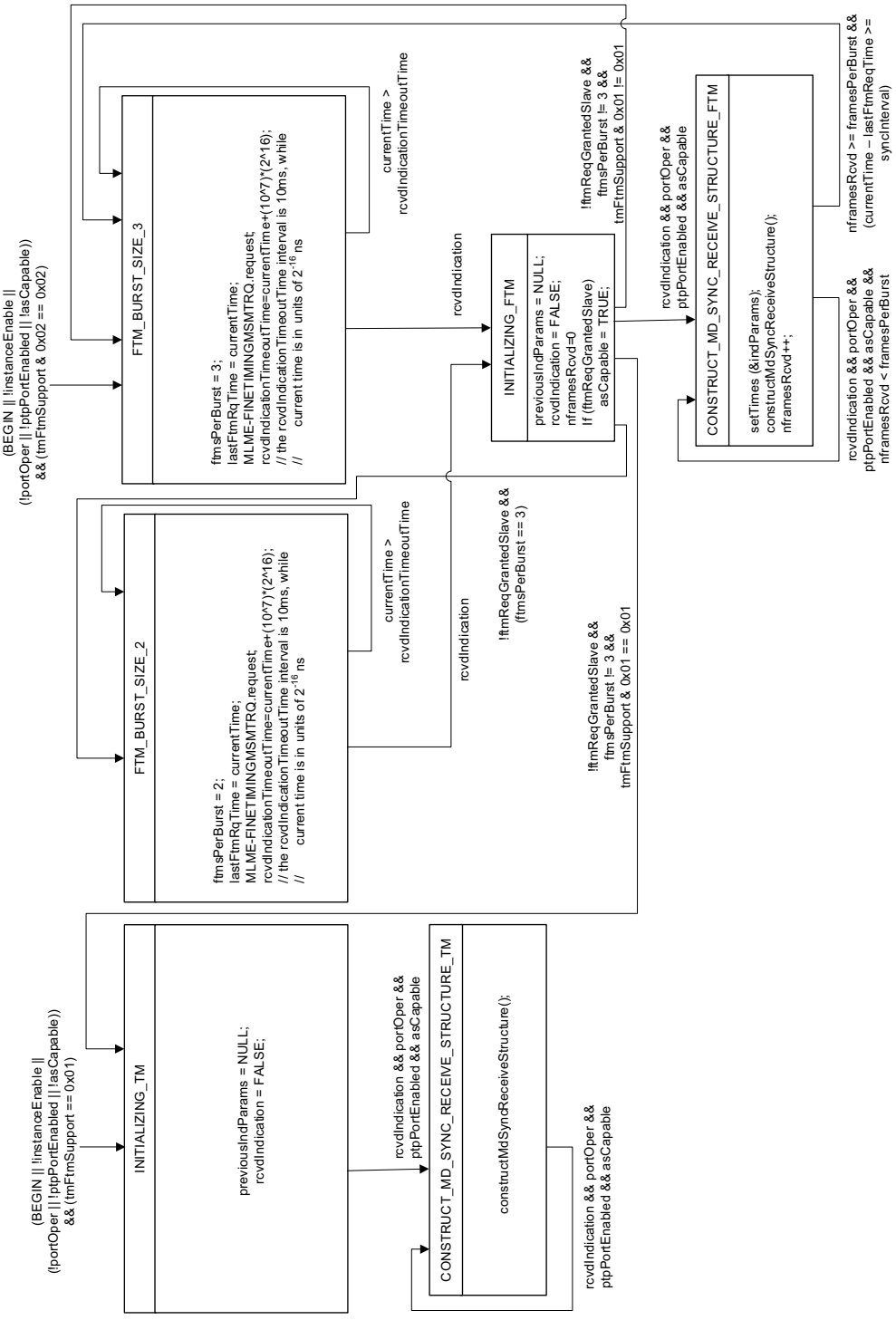


**Figure 12-7—Slave state machine**

**12.5.2.3.10 t1_1, t1_2, t2_1, t2_2, t2_3, t3_1, t3_2, t3_3, t4_1, t4_2:** Temporary local variables used to hold the values of the timestamps t1, t2, t3, and t4 returned by the MLME-FINETIMINGMSMT.indication primitive in the structure indParams.

**12.5.2.3.11 D1, D2:** Temporary local variables used to hold the values of delay computed for the first two FTM frames and corresponding Acks when the master grants the request for three FTM frames.

### 12.5.2.4 State machine functions

**12.5.2.4.1 setMDSyncReceiveDot11Slave(indParams):** Creates an MDSyncReceive structure and returns the structure. All fields are assigned from FollowUpInformation (contained in the VendorSpecific information element) of indParams as in 11.2.14.2.1.

**12.5.2.4.2 passMDSyncReceiveToPortSync():** Passes an MDSyncReceive structure to the PortSync entity of this PTP Port.

**12.5.2.4.3 setTimes (&indParams):** extracts the timestamp values from the successive indParams structures returned by the multiple FTM frame exchanges of a burst, and places the correct times (corresponding to minimum delay frames and Acks) in the final indParams structure. This final indParams structure is then used in the function constructMdSyncReceiveStructure(). This procedure is needed when the master grants three FTM frames for the burst. If the master grants only two FTM frames for the burst, the timestamp values returned in the indication primitive of the second frame are used. The function setTimes is used in the Slave state machine and is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
setTimes (&indParams)
{
        if (nframesRcvd == 1)
        {
                t2_1 = indParams.T2;
                t3_1 = indParams.T3;
        }
        if (nframesRcvd == 2)
        {
                t1_1 = indParams.T1;
                t2_2 = indParams.T2;
                t3_2 = indParams.T3;
                t4_1 = indParams.T4
        }
        if (nframesRcvd == 3)
        {
                t1_2 = indParams.T1;
                t2_3 = indParams.T2;
                t3_3 = indParams.T3;
                t4_2 = indParams.T4;
                if (ftmsPerBurst == 3)
                {
                        D1 = t2_1 - t1_1;
                        D2 = t2_2 - t1_2
                        if (D2 <= D1)
                        {
                                indParams.T2 = t2_2;
                                indParams.T1 = t1_2;
                        }
                        else
                        {
```

```
                                        indParams.T2 = t2_1;
                                        indParams.T1 = t1_1;
                                }
                                D1 = t4_1 - t3_1;
                                D2 = t4_2 - t3_2
                                if (D2 <= D1)
                                {
                                        indParams.T4 = t4_2;
                                        indParams.T3 = t3_2;
                                }
                                else
                                {
                                        indParams.T4 = t4_1;
                                        indParams.T3 = t3_1;
                                }
                        }
                }
        }
```

**12.5.2.4.4 constructMdSyncReceiveStructure():** This function constructs the MDSyncReceive structure and is defined as indicated below. It is used in the Slave state machine. It is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
constructMdSyncReceiveStructure()
{
        if (tmFtmSupport == 0x01)
                MLME-TIMINGMSMT.indication(&indParams);
        else if (tmFtmSupport & 0x02 == 0x02)
        {
                MLME-FINETIMINGMSMT.indication(&indParams);
                nframesRcvd++;
                if (nframesRcvd == initReqParamsDot11Slave.framesPerBurst) ||
                        (currentTime > endofBurstDurationTime)
                                RESTART=1;
        }

        if ((previousIndParams != NULL) &&
                (previousIndParams.PeerMacAddress == dot11SlaveMac) &&
                (indParams.FollowUpDialogToken != 0))
        {

                neighborRateRatio =
                        (indParams.T1-previousIndParams.T1) /
                        (indParams.T2-previousIndParams.T2);
                //NOTE: Other methods of computing neighborRateRatio
                                can be used.

                if (tmFtmSupport == 0x01)
                        K = 1;
                else if (tmFtmSupport & 0x02 == 0x02)
                        K = -3;
                //K = 1 for Timing Measurement and K = -3 for Fine Timing Measurement
                meanLinkDelay =
                        (((indParams.T4 - indParams.T1) -
                        neighborRateRatio * (indParams.T3 - indParams.T2)) /
                                        (2.0)) * (10^K);

                //NOTE: Other methods of computing meanLinkDelay
```

```
                                    can be used.

                        MDSyncReceive = setMDSyncReceiveDot11Slave(indParams);
                        MDSyncReceive.VendorSpecific.rateRatio +=
                                (neighborRateRatio – 1);
                        MDSyncReceive.VendorSpecific.upstreamTxTime =
                                        indParams.T2* (2^16) *(10^K) -
                                        meanLinkDelay*(2^16)/neighborRateRatio;
                        //NOTE: Actions performed with the timestampError
                                parameters of indParams are implementation independent.

                        passMDSyncReceiveToPortSync(&MDSyncReceive);
                }
                previousIndParams = indParams;
                rcvdIndication = FALSE;
        }
```

#### 12.5.2.5 State machine shared variables

**12.5.2.5.1 MDSyncReceive:** A structure used for passing information between MD and PortSync, as defined in 10.2.2.2.

**12.5.2.5.2 portOper:** A Boolean as defined in 10.2.5.12.

**12.5.2.5.3 ptpPortEnabled:** A Boolean as defined in 10.2.5.

**12.5.2.5.4 asCapable:** A Boolean as defined in 12.4 and 10.2.5.1.

**12.5.2.5.5 meanLinkDelay:** The delay over the link to the associated WLAN station as defined in 10.2.5.8.

**12.5.2.5.6 neighborRateRatio:** The measured ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this PTP Port, to the frequency of the LocalClock entity of this time-aware system, as defined in 10.2.5.7.

**12.5.2.5.7 tmFtmSupport:** An Octet whose value is specified in 12.3.

#### 12.5.2.6 Slave primitives

##### 12.5.2.6.1 MLME-TIMINGMSMT.indication

The MLME-TIMINGMSMT.indication primitive is received by a slave station as the natural result of the peer master station issuing the corresponding request primitive and carries the same parameters plus local timestamp information. The primitive and its parameters are specified in 6.3.57.4 of IEEE Std 802.11-2016.

##### 12.5.2.6.2 MLME-FINETIMINGMSMT.indication

The MLME-FINETIMINGMSMT.indication primitive is received by a slave station as the natural result of the peer master station issuing the corresponding request primitive and carries the same parameters plus local timestamp information. The primitive and its parameters are specified in 6.3.58.4 of IEEE Std 802.11-2016.

#### 12.5.2.6.3 MLME-FINETIMINGMSMTRQ.request

The MLME-FINETIMINGMSMTRQ.request primitive is used by the slave to request a burst of FTM frames from the master, with respective FTM parameters. The primitive and its parameters are specified in 6.3.70.2 of IEEE Std 802.11-2016.

## 12.6 FTM parameters

The values of the FTM parameters that are relevant to time synchronization transport in this standard are specified in Table 12-2, along with a brief description of each parameter. These parameter values are carried in the initial FTM Request. More detailed descriptions of these and other FTM parameters are given in 9.4.2.168 of IEEE Std 802.11-2016. The parameters Burst Duration and Min Delta FTM depend on the time synchronization message interval, i.e., on currentLogSyncInterval (see 10.7.2.3 and 12.8). The values for these parameters are specified in Table 12-3. In this table, the Burst Duration encoding (i.e., value and corresponding duration in ms) is taken from Table 9-257 of IEEE Std 802.11-2016. The Min Delta FTM encoding values are in multiples of 100 μs (see 9.4.2.168 of IEEE Std 802.11-2016).

The values of the FTM parameters given in Table 12-2 shall be used in the MLME-FINETIMINGMSMTRQ.request invoked by the slave STA (i.e., in the initial FTM Request). The values for Burst Duration and Min Delta FTM given in Table 12-3 shall be used in the MLME-FINETIMINGMSMTRQ.request invoked by the slave STA.

Background on the derivation of the FTM parameters of Table 12-2 and Table 12-3 is given in Garner [B4].

**Table 12-2—FTM parameters relevant to time-synchronization transport**

| Parameter | Value | Description |
|---|---|---|
| Number of Bursts Exponent | 0 | Log to base 2 of the number of bursts requested by the slave (value of 0 indicates that one burst is requested) |
| Burst Duration | See Table 12-3 | Duration of the burst of FTM frames and their corresponding Acks |
| Min Delta FTM | See Table 12-3 | Minimum time between consecutive FTM frames |
| Partial TSF Timer | 1 | See 9.4.2.168 of IEEE Std 802.11-2016 |
| Partial TSF Timer No Preference | Reserved in the initial FTM request | See 9.4.2.168 of IEEE Std 802.11-2016 |
| ASAP | 1 | ASAP = 1 indicates that the slave would like the master to respond as soon as possible |
| ASAP Capable | Reserved in the initial FTM request | See 9.4.2.168 of IEEE Std 802.11-2016 |
| FTMs per burst | 3 in the first initial FTM Request<br><br>2 in the first retry, if the first initial FTM Request is not granted | Desired number of FTM frames and corresponding Acks in the requested burst |
| Burst Period | Reserved when Number of Bursts Exponent is zero | See 9.4.2.168 of IEEE Std 802.11-2016 |

**Table 12-3—Values of Burst Duration and Min Delta FTM, for each value of currentLogSyncInterval**

| currentLogSyncInterval | Nominal message rate (messages/s) | Burst duration | Min delta FTM |
|---|---|---|---|
| –24 through –6, inclusive | 64 through 16 777 216 | 6 (4 ms) | 6 (0.6 ms) |
| –5 | 32 | 8 (16 ms) | 25 (2.5 ms) |
| –4 | 16 | 9 (32ms) | 50 (5 ms) |
| –3 | 8 | 10 (64 ms) | 100 (10 ms) |
| –2 through 24, inclusive | 4 | 11 (128 ms) | 200 (20 ms) |

## 12.7 Format of VendorSpecific information element

The IEEE 802.11 MLME request and indication primitives for Timing Measurement and Fine Timing Measurement support an ability to carry data transparently between stations using the VendorSpecific information element. The Type field within the VendorSpecific Content identifies the type of information that follows the Type field. See Figure 12-8 and Table 12-4.

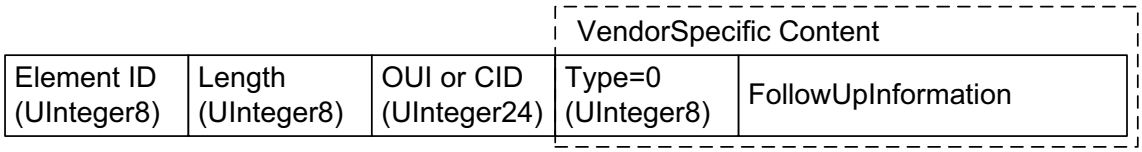


**Figure 12-8—Format of VendorSpecific information element when Type = 0**

**Table 12-4—Values of the Type field in the VendorSpecific information element**

| Value | Description |
|---|---|
| 0 | The Type field is followed by FollowUp-Information |
| 1–255 | Reserved |

This mechanism shall be used to carry end-to-end link-independent timing information from the master port to the associated slave port, including preciseOriginTimestamp, rateRatio, correctionField, and other fields of the Follow-Up message, as described in 12.5.1.4. For consistency, all of these fields are packed into the FollowUpInformation field using exactly the same format as used for full-duplex point-to-point links. In other words, the master state machine communicates an entire Follow_Up message [i.e., including all the fields of the common header (see 11.4.2 and 10.6.2), the preciseOriginTimestamp, and all the fields of the Follow_Up information TLV (see 11.4.4)] using this mechanism. The Type field, illustrated in Figure 12-8, identifies this use of the OUI or CID within the VendorSpecific information element. Table 12-4 lists values for the Type field.

## 12.8 Synchronization message interval

### 12.8.1 General synchronization message interval specification

The mean time interval between successive synchronization messages shall be as specified in 10.7.2.1, 10.7.2.3, and 12.8.2.

### 12.8.2 Synchronization message interval default value

The default value of initialLogSyncInterval (see 10.7.2.3) is –3. Every PTP Port supports the value 127; the PTP Port does not send Sync messages when currentLogSyncInterval has this value (see 10.3.18). A PTP Port may support other values, except for the reserved values indicated in Table 10-16. A PTP Port ignores requests (see 10.3.18) for unsupported values.

Processing of the message interval request TLV carried in a Signaling message (see 10.6.4) shall be supported, as specified by the SyncIntervalSetting state machine of 10.3.18 (see Figure 10-20), except that the logLinkDelayInterval (which is not relevant to IEEE 802.11 ports) is set to –128 by the sender of the Signaling message, the logLinkDelayInterval is ignored by the receiver, and unsupported values of logTimeSyncInterval are ignored by the receiver.

NOTE 1—For TM, a slave port that requests (using a Signaling message that contains a message interval request TLV; see 10.6.4 and 10.3.18) that the PTP Port at the other end of the attached link set its currentLogSyncInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of a FollowUpInformation contained in the VendorSpecific information element of a subsequent MLME indication primitive.

NOTE 2—The time interval between every pair of adjacent timing or Fine Timing Measurements is not guaranteed to be precisely the same. Some variation is expected, due to factors such as the MAC protocol (e.g., delay in accessing the medium and/or packet retries) and the power state of the associated station. However, timestamp t1 is not captured when either TIMINGMSMT.request or FINETIMINGMSMT.request is invoked, but only after the frame resulting from the request is actually transmitted.