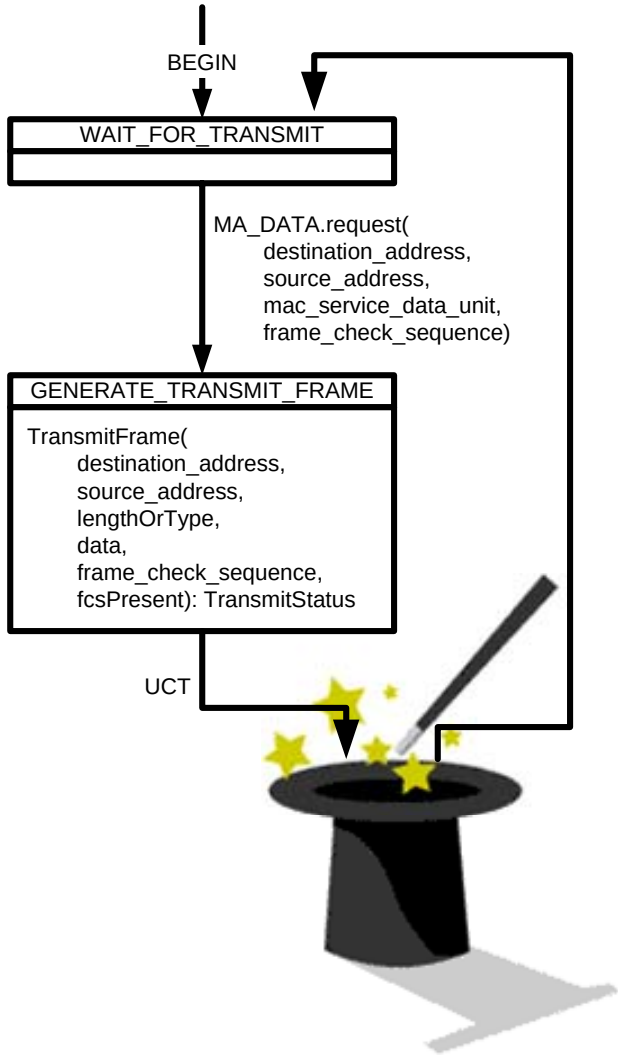
MAC Service Interface

Glen Kramer, Teknovus, Inc.
glen.kramer@teknovus.com

Two problems identified

1. Ambiguity of rules when state diagrams are combined with Pascal

2. MAC Clients need to resort to “magic” to find out when a MAC_DATA.request can be generated or whether it succeeded



Problem 1: Ambiguity of rules when state diagrams are combined with Pascal

Combining State Diagrams and Pascal

- **1.2.1:** State transitions and sending and receiving of messages occur instantaneously. When a state is entered and the condition to leave that state is not immediately fulfilled, the state executes continuously, sending the messages and executing the actions contained in the state in a continuous manner.

- **21.5.1 (extensions to 1.2.1):** The actions inside a state block execute instantaneously. Actions inside state blocks are atomic (i.e., uninterruptible). After performing all the actions listed in a state block one time, the state block then continuously evaluates its exit conditions until one is satisfied, at which point control passes through a transition arrow to the next block. While the state awaits fulfillment of one of its exit conditions, the actions inside do not implicitly repeat.

- **4.2.8:** The TransmitFrame operation is synchronous. Its duration is the entire attempt to transmit the frame; when the operation completes, transmission has either succeeded or failed, as indicated by the TransmitStatus status code.

Need to clarify the conventions

- If a state calls TransmitFrame() function, should 21.5.1 or 4.2.8 take precedence?

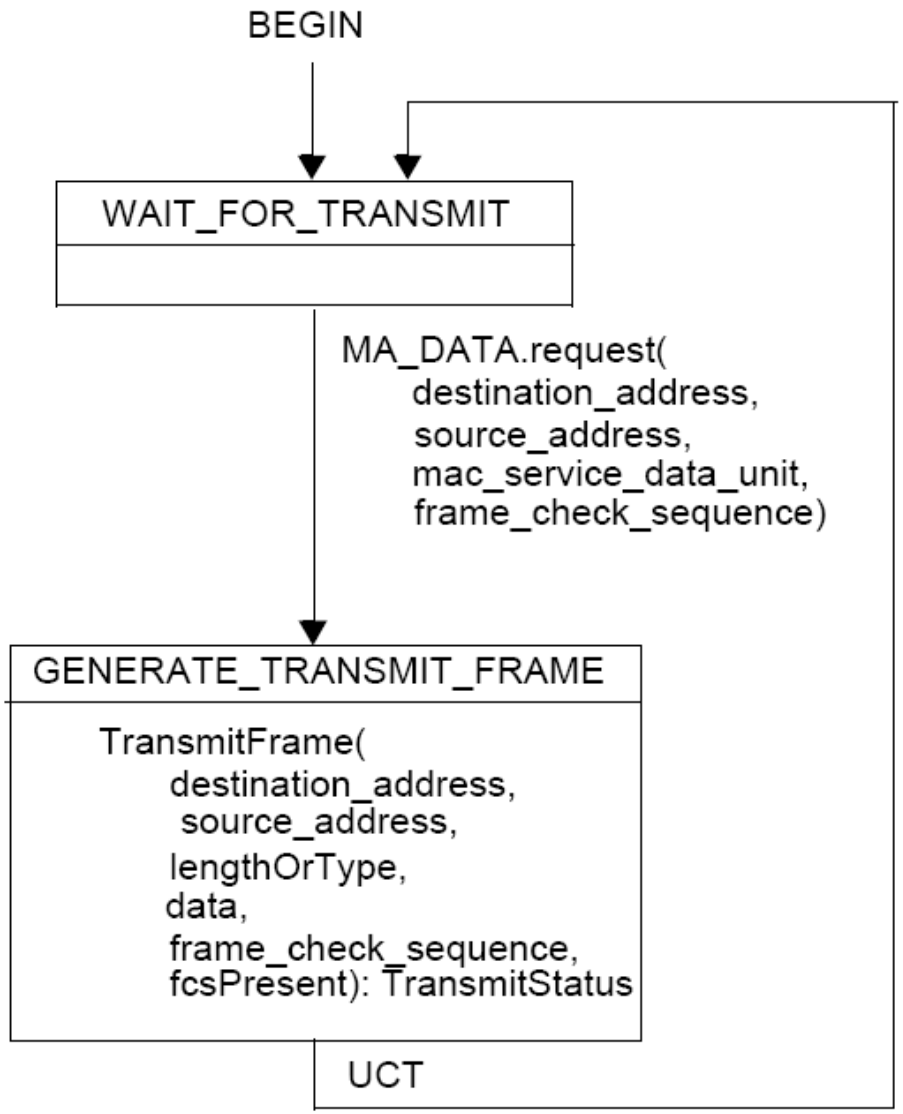
- If 4.2.8 takes precedence, then, the state machine will remain in the state until TransmitFunction() returned, even if the exit condition is satisfied immediately
  - This is analogous to a serial execution in any single-threaded (single-stack) finite automata

- If 21.5.1 takes precedence, then calling TransmitFrame() function will be an instantaneous event and the execution will immediately proceed with the state code that follows this function call.
  - This is analogous to a multi-threaded model, where a call to TransmitFrame() simply spawns another thread (process) which executes asynchronously from the parent process.

- Both approaches are justified. We need to make a choice and document it.

If 4.2.8 takes precedence…

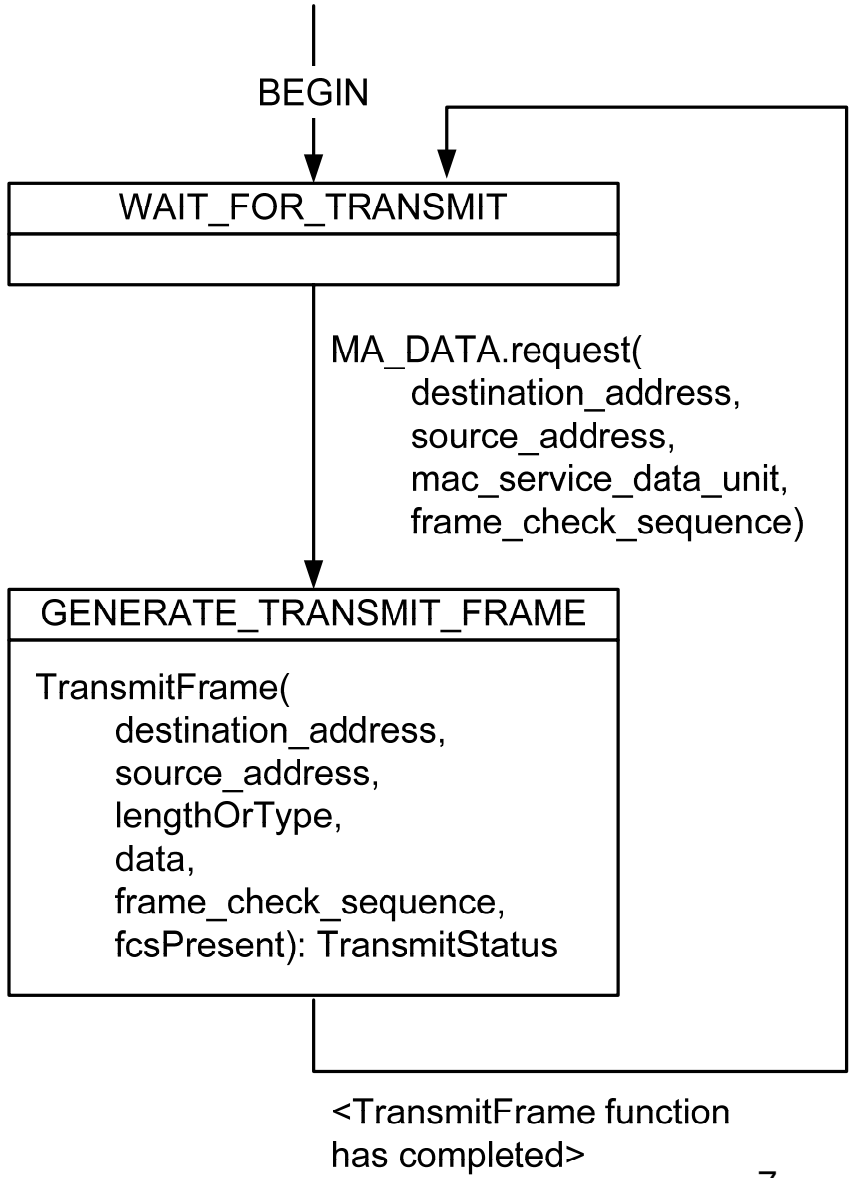
- Existing state diagram in Figure 4-6 assumes 4.2.8 takes precedence.



If 21.5.1 takes precedence…

- … then, the transition from the state should wait for a condition.

- Few ways to specify the condition:
  1. Simply describe it in a note
  2. Add a boolean TransmitFrameCompleted
  3. Wait for specific value(s) of TransmitStatus



Option 1: Add a note

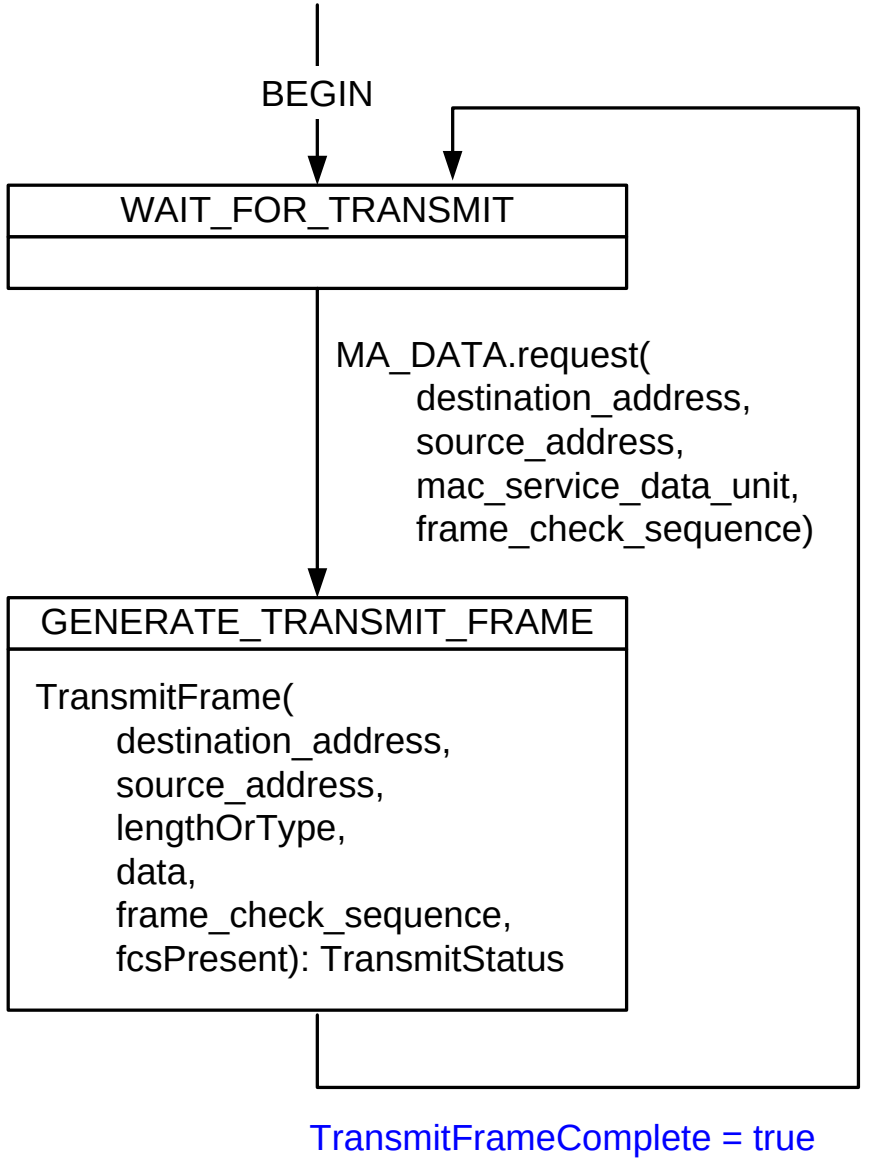
Simply describe it in a note

> “Transition from state GENERATE_TRANSIT_FRAME occurs immediately upon completion of TransmitFrame() function.”

Option 2: Add a new Boolean

1. Add a boolean TransmitFrameCompleted

TransmitFrameCompleted
This boolean is set to false immediately after TransmitFrame is called and is set to true when TransmitFrame function completes.
TYPE: Boolean
DEFAULT: false



Option 3: Wait for a specific TransmitStatus

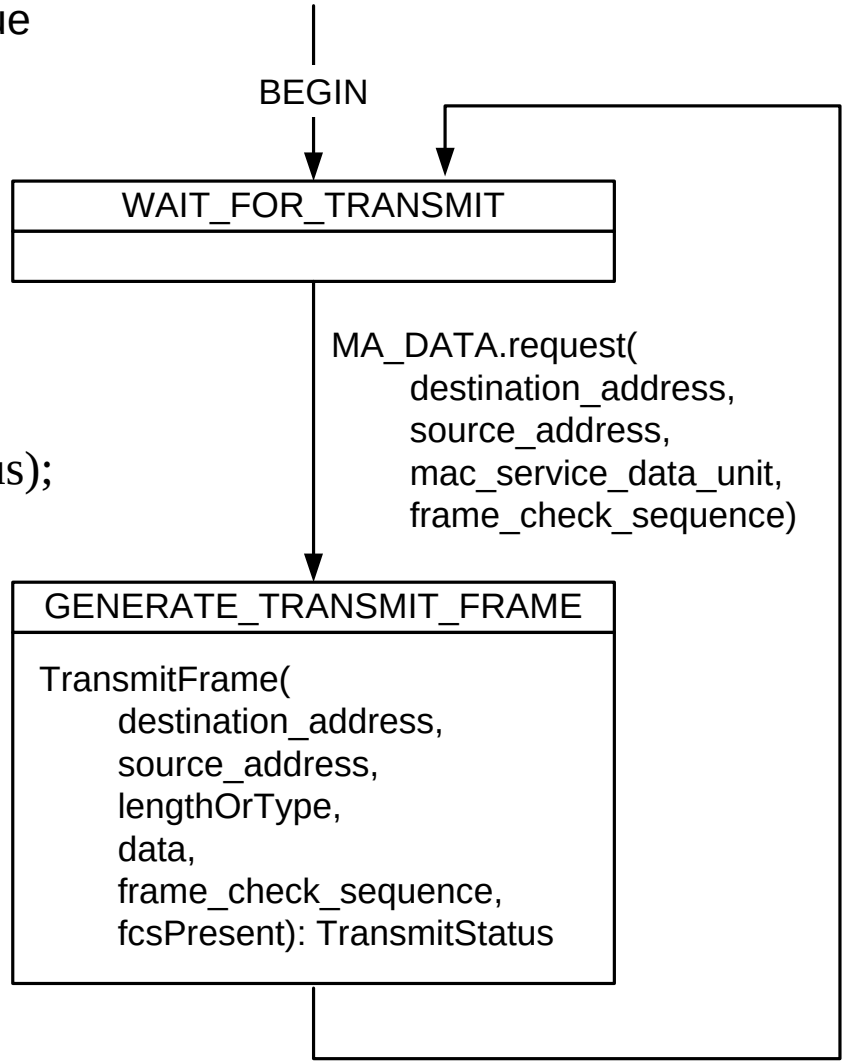
- Wait for specific value(s) of TransmitStatus
  - With this option, we need to specify what value the TransmitStatus variable has before TransmitFrame completes (it should not hold the status of the previous TransmitFrame)

- **Some changes needed to Pascal:**

- TransmitStatus = (transmitInProgress, transmitDisabled, transmitOK, excessiveCollisionError, lateCollisionErrorStatus);

- *function* TransmitFrame (

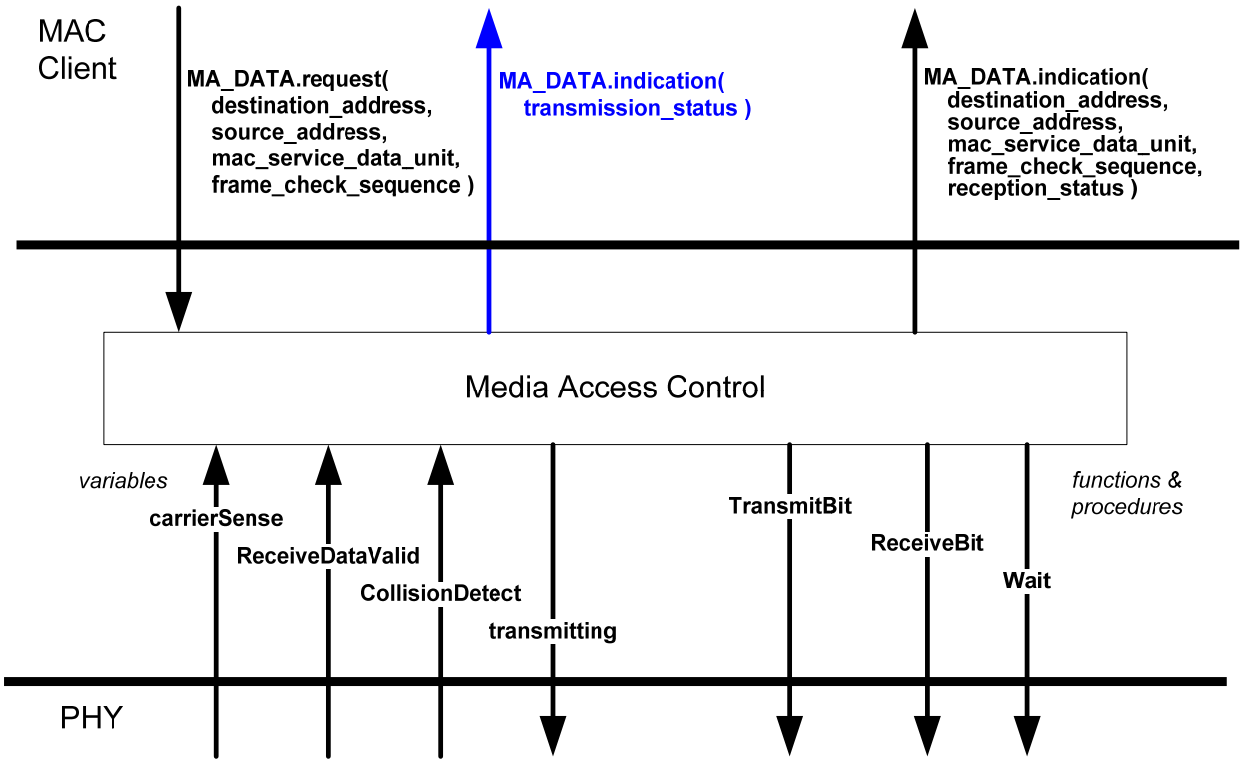
```
…
begin
   TransmitFrame := transmitInProgress
   if transmitEnabled then
      begin
         TransmitDataEncap;
         TransmitFrame := TransmitLinkMgmt
      end
  else TransmitFrame := transmitDisabled
end; {TransmitFrame}
```



Problem 2: MAC Clients doesn't know when a MAC_DATA.request can be generated or whether it succeeded

Three-legged MAC Service Interface

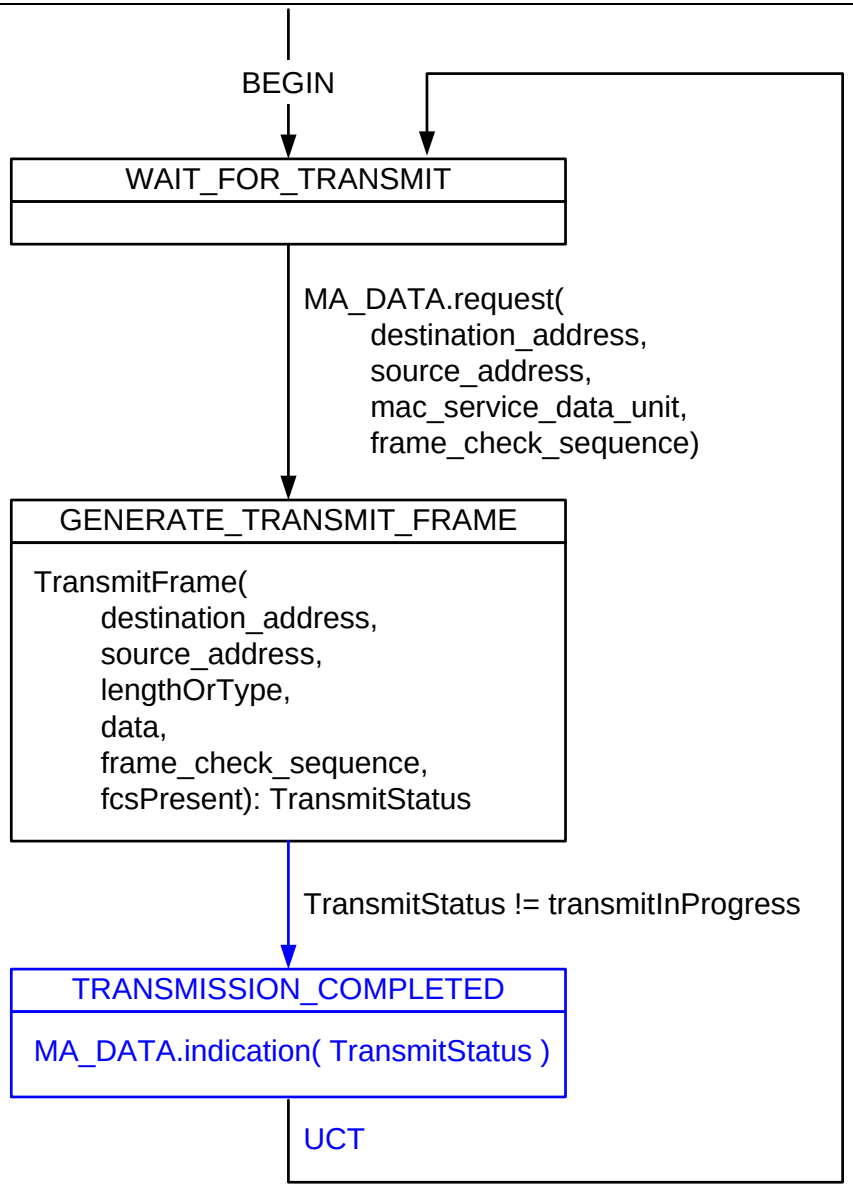
Figure 2-1



- To let the MAC client know when frame transmission ended, another indication can be added to the MAC Service Interface.

MAC Client Transmit Interface

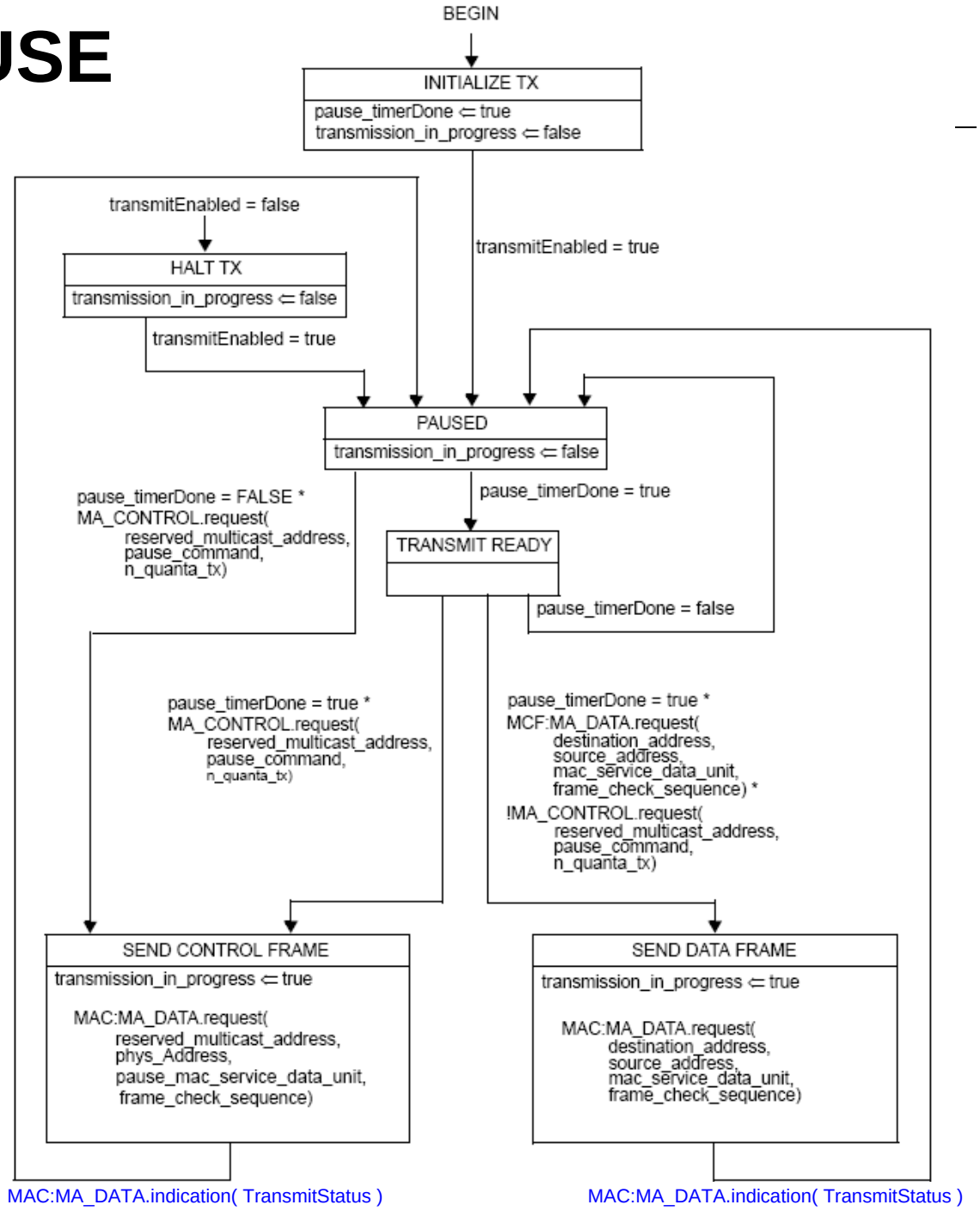
- Modification to Figure 4-6 – MAC Client transmit interface state diagram.
- (This example assumes option 3 is selected, but this is not necessary)



MAC Client: PAUSE

- To pace itself, a MAC client after issuing MA_DATA.request(…) would wait for MA_DATA.indication(TransmitStatus)

- Example shown for PAUSE Operation Transmit state diagram (Figure 31B-1) →



MAC Client: OAM

- Example shown for OAM Multiplexer state diagram (Figure 57-7) →