Dealing with fragile bridge implementations

Mick Seaman

This note describes modifications to RSTP and MSTP to deal with fragile bridge implementations whose control plane spanning tree protocol operation (including BPDU transmission) can stop while their data planes continue to relay frames. At present such brainless bridges can cause frame forwarding loops, as other bridges naturally transition their own ports to Forwarding.

The problem can be addressed in a number of quite different ways, independently of xSTP, each with its own disadvantages. MAC Security (IEEE Std 802.1AE, P802.1X-rev) will provide a first class defence against unauthorized snooping, ensuring that silent bridges will not acquire the necessary cryptographic keys to forward frames around a loop that includes a secured bridge. However we need a solution for bridges that are already deployed. Early loop detection protocols were initially retained as a backstop to STP operation, but are unattractive in today's more complex networks as loop detect frames would have to be sent by each Bridge Port on each and every VLAN. Some early bridges were capable of monitoring repeated and rapid movement, from port to port, of learnt addresses—behaviour that is usually associated with a loop—but this functionality may not be possible for a given high speed design. This note therefore limits itself to RSTP/MSTP enhancements.

At present RSTP (802.1D-2004) and MSTP (802.1Q-2005) can use the absence of BPDUs from another bridge on a LAN to conclude that the attached Bridge Port is an Edge Port (i.e. only attached to end stations[1]) thus allowing rapid transitions to Forwarding. This plug-and-play functionality is important in a number of deployment scenarios and has to be retained, though it can be overridden by manual configuration. Equally it is now desired that a Bridge Port that is known not to be an Edge Port automatically transition to Discarding if the other bridge attached to the same LAN does not continue to transmit BPDUs[2] to show that it has not become brainless.

RSTP and MSTP do not normally transmit BPDUs periodically through Root or Alternate Ports. It is highly desirable that new bridges be able to deal with existing fragile bridges: (a) without requiring coordinate configuration (always error prone) of several bridges; and (b) without requiring an upgrade to the fragile bridge's software. Instead the Designated Bridge Port that requires regular BPDU receipt to hold off a transition to Discarding can solicit those BPDUs by setting the Proposal flag for the CIST. Existing RSTP/MSTP Bridges already respond to Proposals—this is behavior is essential for rapid healing of temporary cuts in the active topology—and the fact that they do so is used to make rapid determination of operEdge reliable.

Brainless bridge detection can be configured using the existing Admin Edge and Auto Edge controls, as setting both False for a port specifies that it can never be operEdge. One new per port variable is required. This note specifies the necessary enhancements to the RSTP/MSTP state machines[3,4] and provides state diagrams for the relevant machines.

[1] End stations from the point of view of the MAC Service. These may include stations supporting router interfaces.

[2] Some other type of frame might be used, but using a BPDU provides a greater assurance of continued spanning tree protocol operation.

[3] It is not possible to provide a BPDU based capability to protect against fragile bridges that only implement the original STP.

[4] The capability is intentionally limited to point-to-point connections to fragile bridges. Defending against brainless bridges on shared media would require something so similar to the MAC Security protocols that it is not worth providing an alternative.

1. Summary

This note begins (2) by reviewing the current AdminEdge, AutoEdge, operEdge functionality in some detail. This due diligence is necessary to make sure that we are not about to break something.

The desired functionality is then described (3), together with details of variable use and state machine changes to realize that functionality. A full set of state machine changes is provided (5).

While working on this note I encountered a number of editorial problems with the existing 802.1Q and some serious technical problems with the 802.1ah specification (4). To the extend they impinge upon our ability to maintain the standard in general and to make the changes proposed by this note I have suggested corrections, and incorporate these in the detailed state machines.

2. Current functionality

At present (802.1Q–2008) rapid detection of 'Edge Ports' i.e. Bridge Ports that are believed to be attached to LANs that have no other attached bridges is facilitated by the following variables, timers, and state machine transitions/actions/procedures.

Variables:

— AdminEdge (default False): The initial value of operEdge when the port is first enabled.

— AutoEdge (default True): True if BDM (the Bridge Detection state Machine) is allowed to set operEdge = True.

— operEdge: Initialized to the value of AdminEdge by BDM, set True by BDM if AutoEdge is True and edgeDelayWhile expires while proposing and sendRstp are True. Note that we should not overload operEdge since it is used for various purposes, including the TCN machine.

— proposing: Set in PRT:DESIGNATED_PROPOSE, (and nowhere else) and used to determine the value of the Proposal flag in transmitted BPDUs. Reset whenever the Port Role will no longer be Designated (receipt of superior information in PIM, the Port Information state Machine).

— proposed: Set by PIM (the Port Information Machine) by recordProposal() in PIM:SUPERIOR_DESIGNATED and PIM:REPEATED_DESIGNATED when a Proposal flag is received (for the relevant tree). If the receiving port is a Root or Alternate Port then proposed will result in a BPDU transmission, possibly delayed if ports need to be blocked to sync the tree (see PRT:ROOT_PROPOSED, PRT:ROOT_AGREED, PRT:ALTERNATE_PROPOSED, PRT:ALTERNATE_AGREED. Note that this solicitation works even if the Bridge Port's are attached to shared media. It is only use of the returned Agreement that is affected by shared media.

Timers and timeout values:

— edgeDelayWhile: The delay before BDM makes a port operEdge.

— EdgeDelay: A conditional variable in 802.1D, returning MigrateTime if operPointToPointMAC is True and MaxAge otherwise. Only used in PRT:DESIGNATED_PROPOSE.

— MigrateTime: A fixed value (3 seconds). Used for various purposes, but principally for migration/ interoperability between RSTP/MSTP and the original STP. Used as the initial value of edgeDelayWhile in PRX, and in L2GPRX (layer 2 Gateway Port Receive[1]) state machine.

State Machines:

— BDM (Bridge Detection state Machine): Sets the initial value of operEdge to AdminEdge, and allows operEdge to transition True if the edgeDelayWhile timer runs from its initial value to expiry while both proposing and sendRstp are continuously True. Note that proposing is only ever True for a Designated Port.

— PRX (Port Receive state machine): Sets operEdge False and restarts edgeDelayWhile (with its initial value of MigrateTime) whenever a BPDU is received.

— PRT (Port Role Transitions state machine): Sets edgeDelayWhile = EdgeDelay in PRT:DESIGNATED_PROPOSE, and uses operEdge as one of the conditions that allows rapid transitions to Learning and Forwarding.

[1]Note—In the light of some problems with the 802.1ah documentation of L2GP I am suggesting a change of name for the Port Receive Pseudo-Information machine. The new name is (I believe) a more accurate description, but the real reason for suggesting the change is that there is a significant bug in a procedure used by the current machine and the name change might successfully distinguish implementations before and after the bug fix.

The functionality provided can be summarized as follows:

a) If no other bridges are transmitting or responding to proposals the Bridge Port will (by default) conclude that there are no other bridges attached to its LAN, and hence assert operEdge and transition to Forwarding, some time after it becomes a Designated Port (provided it is not Forwarding already, due to previously being Root Port).

b) That time can be short (MigrateTime—3 seconds) for point-to-point media, but can be much longer for shared media. The reason for the latter is that on shared media a previous, and superior, Designated Bridge could have just died (or been silently disconnected from the shared media) as our Bridge Port is added. As a result the bridges will have continued to believe that bridge is Designated and ignore our Proposals.

c) The initial setting of edgeDelayWhile to MigrateTime can be something of a red herring: any non-zero value would have done.

d) When the port is disabled and re-enabled operEdge is set back to AdminEdge.

e) The Bridge Port will not set operEdge if it was a Root Port and then becomes Designated without transitioning from Discarding.

3. Desired functionality

The most often described concern with potentially brainless bridges is that one of the bridge's ports becomes Designated, then stops sending BPDUs while remaining Forwarding. Because this first port stops sending BPDUs, a second port, on another bridge previously Discarding, and attached to the same LAN as the first becomes Designated and eventually Forwarding, thus creating a loop. To address this scenario without losing the spanning tree reconfiguration capability, the second port needs to be able to distinguish it from one in which the first port loses connectivity to the Root and simply becomes a Root Port.

Other brain death scenarios can also occur, starting with the brainless bridge's port becoming Forwarding without ever sending a BPDU. Brain death could have more subtle manifestations, but all have the following in common. A bridge that seeks to protect against a fragile, potentially brainless neighbour, needs some assurance that neighbour is indeed receiving its BPDUs, is capable of processing them, and is capable of sending BPDUs on its own account. Otherwise the bridge that requires that assurance will not transition to Forwarding, and indeed will transition from Forwarding. The latter is required because although the now brainless bridge's port may have been a Root Port when fit and well it is quite possible that additional loop creating connectivity has been created on its far side.

To get this coverage requires configuration, possibly with different shipping configuration defaults on different types of bridge ports, to distinguish between a LAN internal to the network (where lack of BPDUs indicates a brainless bridge) and a LAN at the edge of the network (where lack of BPDUs serves to confirm that only edge ports are present). The Proposal Agreement mechanism is an existing method, already supported in all RSTP/MSTP conformant bridges, by which a Designated Port can solicit a BPDU from another bridge (if a break in connectivity is not forced due a dispute, which would happen if there was one-way connectivity).

While preserving the current functionality, for ports that are likely to be Edge Ports, we would also like the lack of response to Proposals to be interpreted as symptomatic of a brain-death. Since current RSTP capable bridges already respond, this allows brain-death detection by configuring just the bridge that sends the Proposals. Since a fragile implementation is assumed to be capable of brain-death at any time we would like to configure the detecting ports to make each and every BPDU sent by a Designated Port a Proposal. It is however desirable to make an exception in the case of shared media, where continuous Proposals could result in an unpleasantly large number of BPDUs, and it would be difficult to detect one brainless bridge amongst many without a protocol mechanism aimed at explicitly identifying each participating system. Brainless bridge detection for shared media is best left to scenarios where MAC Security is to be deployed.

As well as retaining the current functionality, it is also desirable to introduce the new functionality with the minimum change or addition to management variables. There appears to be no downside to interpreting AdminEdge == False, AutoEdge == False as indicating that the Bridge Port will never be at the edge of the network and therefore always connected to at least one other bridge.

Note that we have to deal with the case where a peer Bridge Port is already brainless when the port is enabled, so we cannot simply rely on an initial period when the fragile bridge responds to Proposals to identify the LAN as being internal to the network. Moreover we would like the Bridge Port not to

automagically acquire a persistent memory of events that occurred prior to it last being enabled, as such a memory can make an error condition persistent and very hard to diagnose. The proposed AdminEdge, AutoEdge setting makes management configuration of the Bridge Port as being persistently internal to the network explicit. If an administrator wants persistent behavior after operEdge has been automatically cleared then he can change the AutoEdge setting at that time.

The following additional and changed variables, timer, and state machine transitions/actions/procedures are suggested.

Variables:

— isolate: (per port variable, initial value, following portEnabled False) True if the Port State is to be kept Discarding because no other bridge appears to be present, RSTP or MSTP is in use, and AdminEdge and AutoEdge are both False.

Conditions (of existing variables):

— operPointToPoint: Simply mentioned here because it was not previously declared as a variable used by the state machines.

Timers and timeout values:

— No new timers are required as edgeDelayWhile will do for this functionality (operEdge cannot occur with the proposed AdminEdge, AutoEdge settings).

— However MigrateTime is really too short to set isolate. Determination of operEdge when AutoEdge is set is a rather different case (initial transmissions by two ports). Define IsolateTime with a default of 5 seconds to allow for two lost messages.

— PRX needs a further modification (not included elsewhere in this note yet) to use an initial value of around 5 seconds when !AdminEdge && !AutoEdge.

State Machine Procedures:

— In recordAgreement() do not clear proposing for the CIST if AdminEdge and AutoEdge are both False.

State Machines:

— PIM: In PIM:UPDATE remove proposing from proposing = proposed = FALSE. Note that this change should be entirely harmless so far as other RSTP/MSTP functionality is concerned, clearing proposing here appears to have been quite unnecessary in the first place.

— PRT: In the transition conditions from PRT:DESIGNATED_PORT to DESIGNATED_LEARN and DESIGNATED:FORWARD add "&& !isolate".
In the transition condition from PRT:DESIGNATED_PORT to DESIGNATED_DISCARD to add "|| isolate" immediately after "|| disputed".
In the PRT:DESIGNATED_PORT actions add "if (cist) {proposing = proposing || (!AdminEdge && !AutoEdge && operPointToPoint);}"

— PRX: In the PRX:RECEIVE actions clear isolate, and make the initial value of edgeDelayWhile MigrateTime (as before) or IsolateTime (if AdminEdge and AutoEdge both False).

— BDM: Add another state BDM:ISOLATED. Clear operEdge and set isolate in this state.
In the current states BDM:EDGE and BDM:NOT_EDGE clear isolate.
Add || (!AdminEdge && !AutoEdge) to the transition condition from BDM:EDGE to BDM:NOT_EDGE. This addition allows for a change to these variables while the port is enabled.
Transition from BDM:NOT_EDGE to the new state BDM:ISOLATED on the condition "(edgeDelayWhile == 0) && (!AdminEdge && !AutoEdge) && sendRstp && proposing && operPointToPoint"
Transition from BDM:ISOLATED to BDM:NOT_EDGE on the condition AdminEdge || AutoEdge || !isolate || !operPointToPoint.
Change the transition conditions from BDM:NOT_EDGE to BDM:NOT_EDGE by replacing "!portEnabled" with "(!portEnabled || !AutoEdge)".

All these state machine changes are shown in the figures in this note.

4. Current editorial issues

In reviewing the current functionality (see 2 above) I have noticed a fair number of editorial issues[1]. The most significant is the MSTP specification's heavy dependence on the RSTP specification of 802.1D-2004. Casual use of references from 802.1Q to 802.1D has obscured the fact that sometimes the referenced material simply does not meet the obvious needs of the reference. This is particularly true where references have been truncated to point to the management

[1] I suspect that at least some of these are my fault, especially if acts of omission as well as commission are included.

chapter, which then says nothing apart from noting the existence of the reference. Incorporating the material from 802.1D that 802.1Q actually relies on will be needed to produce an intelligible specification.

Other oddities include:

a) The Port Receive state machine in 802.1Q is said to be identical to the Bridge Detection state machine in 802.1D, but is not, and 802.1Q also includes the Bridge Detection state machine from 802.1D (by explicit reference. In fact the 802.1Q PRX is identical to the 802.1D PRX (with the exception of providing for multiple trees).

b) Because BDM is specified in 802.1D I am not sure that it is clear that it uses proposing for the CIST as opposed to any other tree.

c) AdminEdge is renamed AdminEdgePort (in some places), adminEdgePort in others, and adminEdge in others. The root of the problem seems to have been changes introduced in the 802.1D management clause without checking the rest of the document. Since the state machines use AdminEdge and much grief could result from effectively 'changing the code' I suggest that we change all to AdminEdge with the possible exception of what is in the MIB, which needs to be mapped to that variable name anyway.

d) The Port Receive Pseudo Information machine is not referenced in Figure 13-10. Figure 13-23 does not use the normal diagrammatic state machine layout conventions (transitions enter a state from the top, transitions exit from below, transition conditions are above (not below) horizontal transition arrows.

e) 802.1Q 13.38.1 (introduced by 802.1ad) states that AdminEdge, AutoEdge, and operEdge are always false, true, and false for a Provider Edge Port but does not explain how operEdge is kept false in this circumstance.

f) Cramming the Port Receive Pseudo Information machine into a corner of Figure 13-9 has resulted in two quite unnecessary jumps in other lines. The variable list in the machine box duplicate those elsewhere, and there are no connections from the machine to any other machines.

g) Using the Port Receive Pseudo Information state machine to clear rcvdBpdu when will produce undefined results as the Port Receive state machine is operating in parallel (perhaps, this is what the specification says from a formal point of view but the intent would seem to be otherwise) and might equally consume rcvdBpdu first. Investigation of this apparently glaring flaw has led me to the observation that one of the changes that Francois Tallet proposed to the transition condition to PRX:RECEIVE actually got applied to the transition to PIM:RECEIVE (i.e. to the wrong state machine entirely).

5. Changes to 802.1Q

The functionality described here has already ben mentioned, in general terms, as something that should be done as part of P802.1aq. From the editor's point of view the changes would be much better carried out as an integrated part of that project rather than as a parallel activity with all the effort required for reconciling edits and the attendant risks that a parallel project would entail. A principle reason for writing this note was to share information about what I believe to be the best way of dealing with the brainless bridge problem sooner rather later.

Most of the editorial issues already mentioned (see 4 above) should be dealt with by P802.1aq, though a few minor points that have no possible impact on the rest of P802.1aq could be left to P802.1Q-REV to avoid scope arguments. The Q-REV PAR covers integrating 802.1D functionality into Q and the edits should have that end goal in mind, though the task is too large to do all at once. At the minimum it should be possible to look at the new text and ascertain how to upgrade an RSTP implementation previously based on 802.1D Clause 17. It is clear that the MSTP specification needs to be complete in and of itself, without having to refer to variables, procedures, and state machines in 802.1D.

Other edits, additional to those introduced with the desired functionality (see 4 above) include:

a) Add a point on brainless bridge detection to clause 13.1. A further point to cover the existing one way connectivity detection should also be added.

b) Possible minor updates to 13.4.

c) Update Figure 13-9. In particular BDM sends isolate to PRT.

d) Add an Informative Annex to draw attention to and explain the detail of this new capability. Would be a good idea to include one way connectivity detect as well. General scope of the annex being recent or unrecognized capabilities.

Figure 1 shows the proposed BDM.

Figure 2 and Figure 3 contain a corrected Port Receive (PRX) machine and a suggested L2 Gateway Port Receive (L2GPRX) machine as a replacement for the

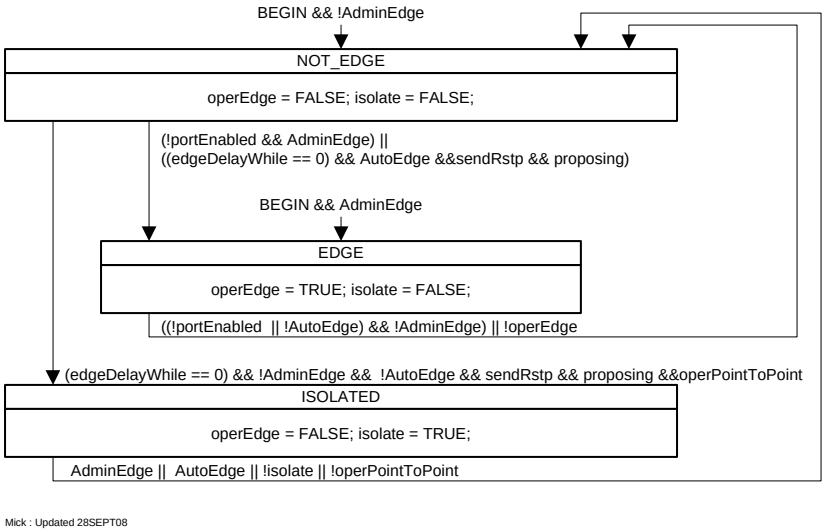

Figure 1—Replacement Bridge Detection state machine

existing Port Receive Pseudo Information state machine. The latter does not change the main functionality of the machine, but clears up some minor issues[1].

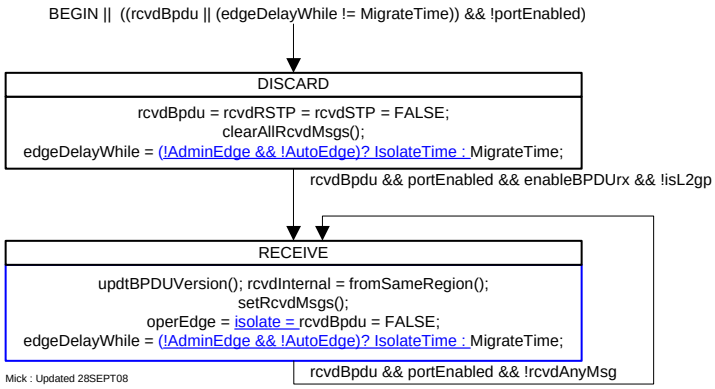

Figure 2—Changed Port Receive state machine based on correction to 802.1ah

Figure 4 is an updated PIM, based on the 802.1Q-2006 state machine (i.e. without the incorrect 802.1ah change).

Figure 5 is an updated PRT (Designated Port states only), based on the 802.1Q-2008 Edition in preparation state machine, which I don't believe has changed from the 802.1Q-2006 edition.

[1]Which should be the subject of a separate explanation.

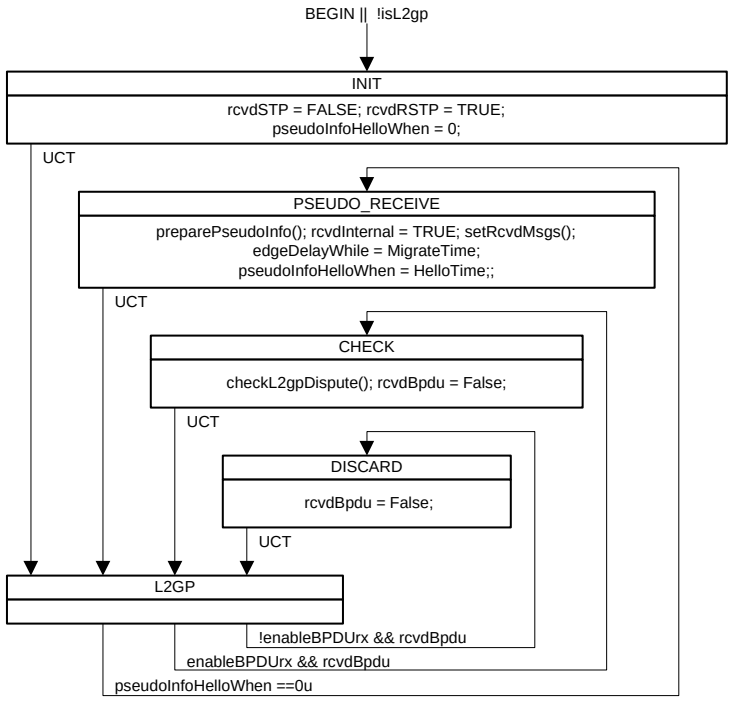

Figure 3—L2 Gateway Port Receive state machine

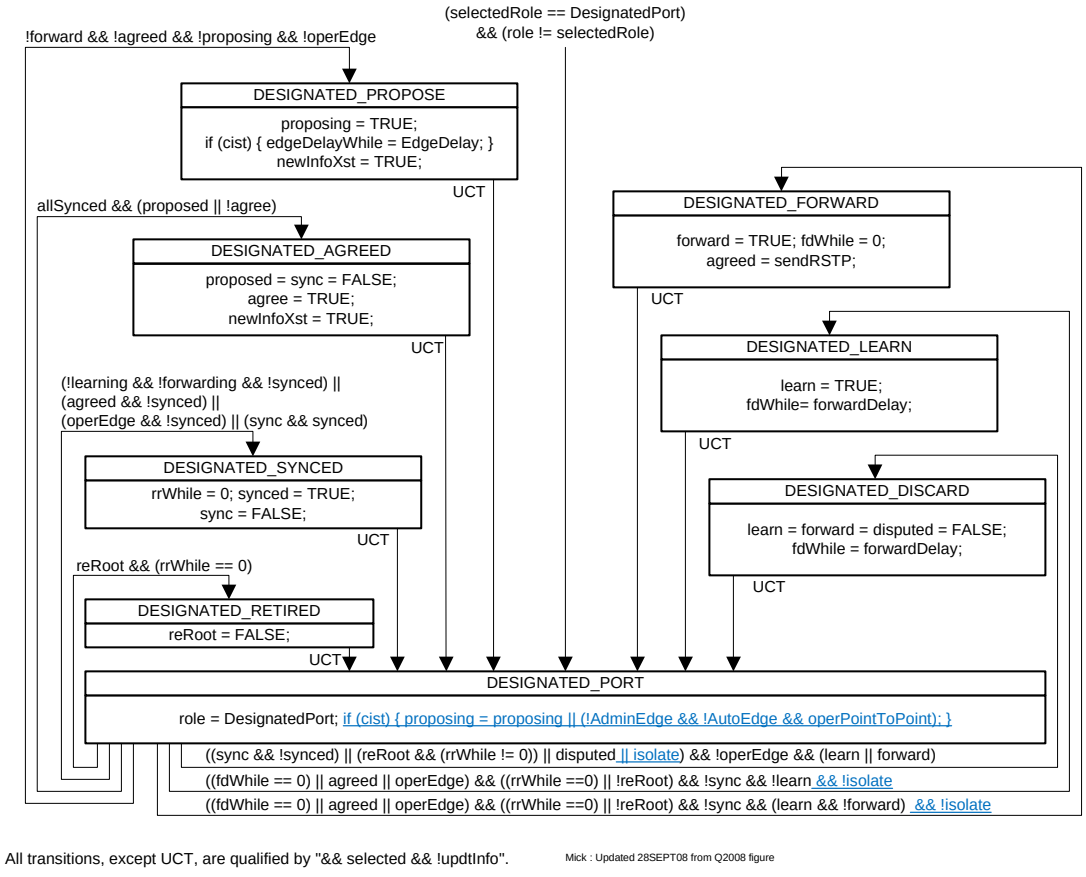

Figure 5—Changed Port Role Transitions state machine (change from 802.1Q-2006/2008)

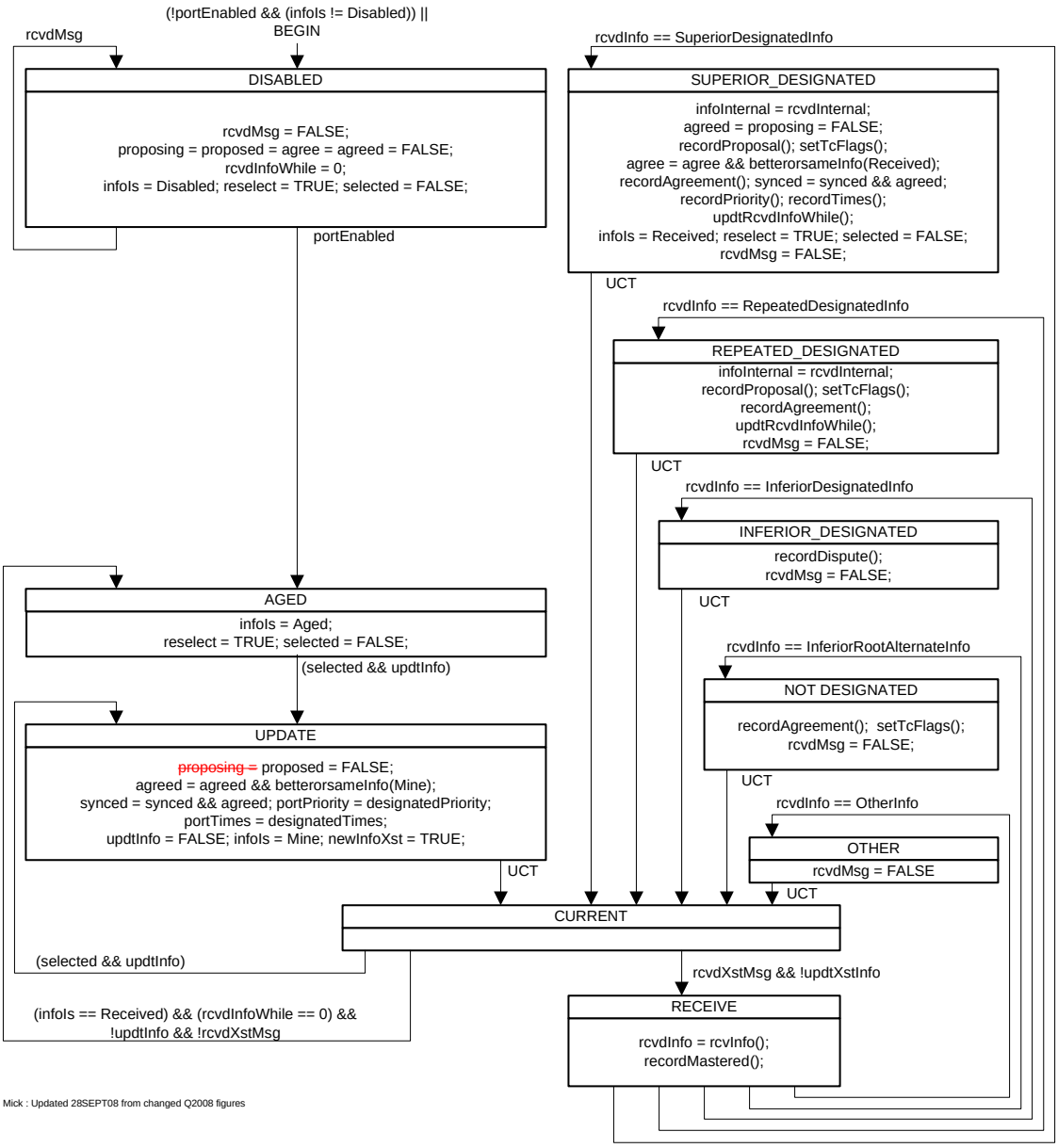

Figure 4—Changed Port Information state machine (change from 802.1Q-2006)