LACP state machines revisited

Mick Seaman

This note is a snapshot of a further critical examination of the current LACP machines. LACP was the first significant 802.1 effort using the current state machine conventions, and perhaps unsurprisingly did not follow them as strictly as later work (such as RSTP/MSTP for example). These machines may need further work, but should be of interest to others working on P802.1AX-Rev.

---

1. Contents

The figures (one per page, following this front page) and accompanying notes cover the following:

1) A LACP State Machine Overview

   This not a state machine, but shows how the state machines are meant to work together following the style used in the 802.1Q RST/MSTP description.

   The Churn machines have been removed. They were a nice idea but don’t do much or work as originally intended.

   The TRANSMIT and PERIODIC machines have been combined, as they clairvoyant (aware of each other’s state without using any communication variables). The combination is simpler and sends fewer unnecessary LACPDUs.

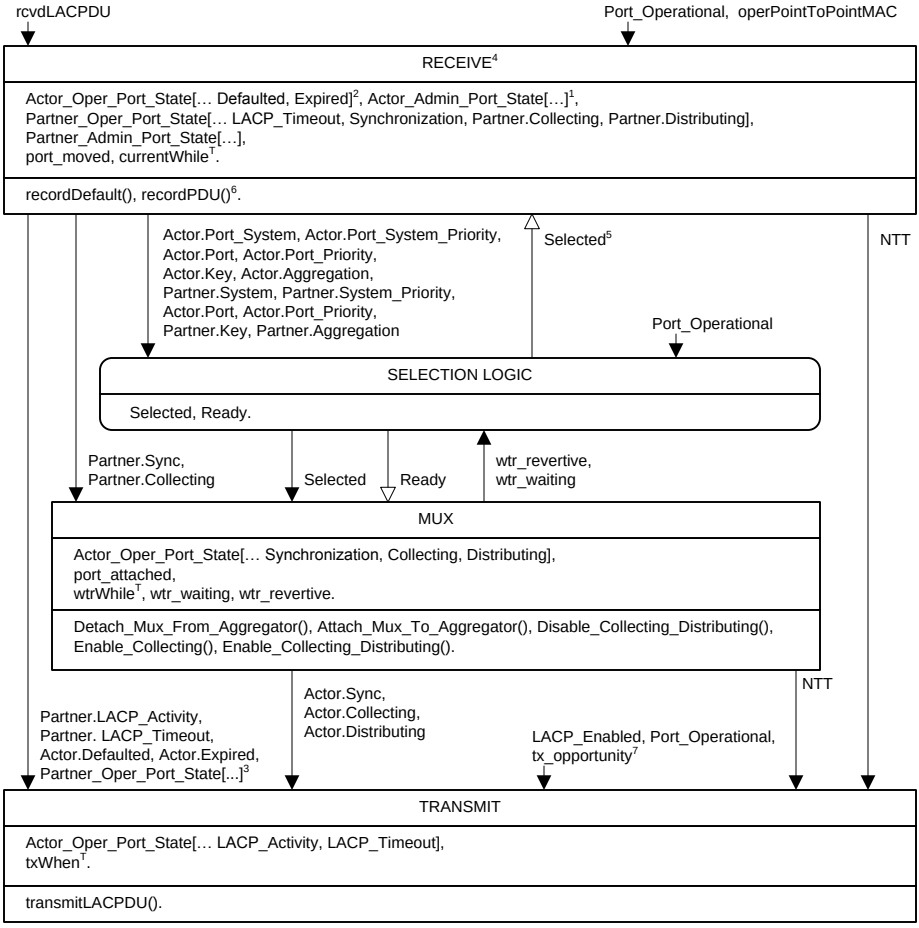
   We are left with RECEIVE, MUX, and TRANSMIT machines, plus the SELECTION LOGIC.

2) An updated/replacement RECEIVE machine.

3) The MUX machine. I believes that this agrees with the conclusion we reached when we last discussed the MUX machine.

4) The updated/replacement TRANSMIT machine, including the periodic functionality.

LACP State Machine Overview

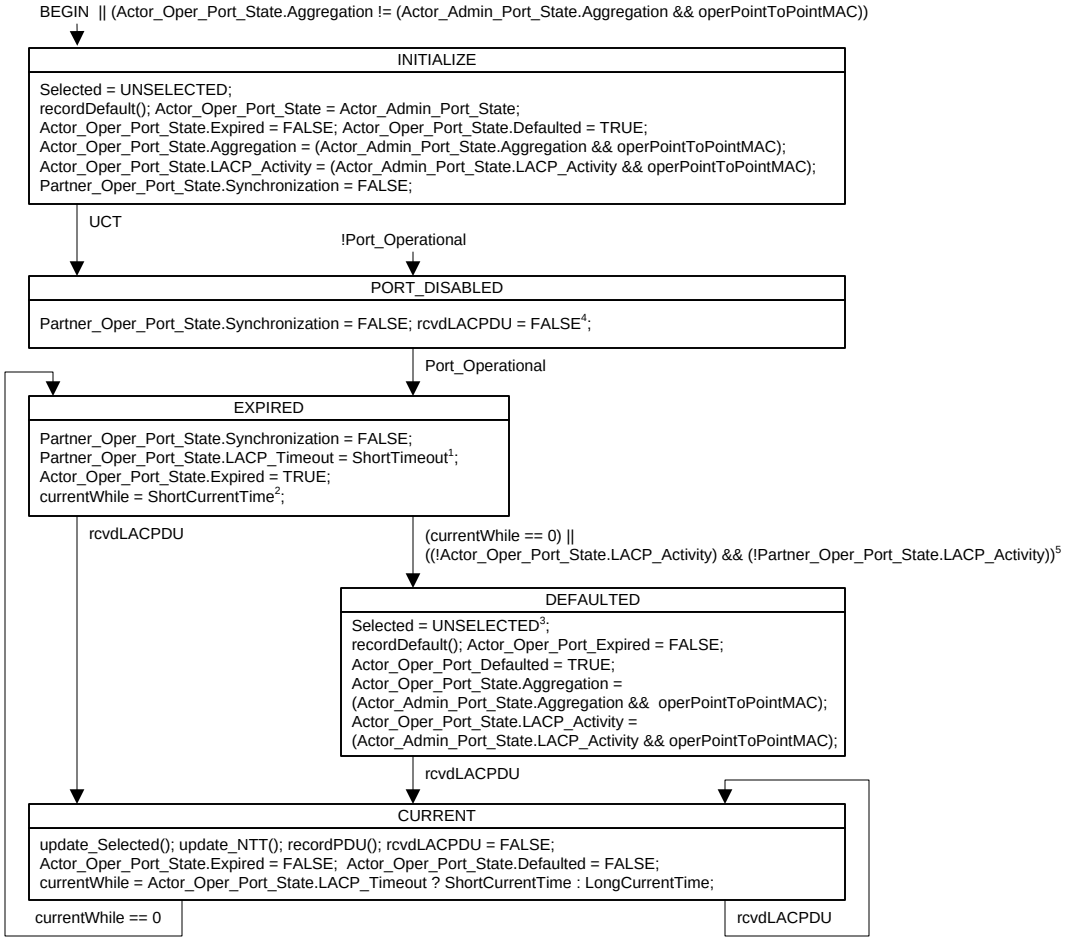


This figure is not a state machine, but an informal overview of the basic LACP state machines, their state variables, and the use of variables to communicate between machines. It follows the general style of IEEE 802.1Q Figure 13-13's overview of RSTP/MSTP, though is simpler and has conventions adapted for LACP description. It could be a more informative replacement for 802.1AX Figure 6-18, but was produced simply to aid the author's own understanding.

Three per port state machines (RECEIVE, MUX, and TRANSMIT) are shown [the 802.1AX-2014 PERIODIC state machine is considered to be part of the TRANSMIT state machine, for reasons detailed elsewhere] together with the SELECTION LOGIC (which operates over all ports, shown in the figure with the set of variables for a single port – omitting the identifier of that port and the identifier of the aggregator to which it might be attached).

The following are shown for each of the state machines: its name; the principal variables used in its operation – focusing on those that it initalizes, modifies, and that guide its operation; and the functions it uses. The assignment of some variables to a given machine are arbitrary, but aims at making changes to a variable by one machine that affect another explicit, while at the same time minimizing communication. that the components of Actor_Oper_State Timer variables (which countdown on a system tick until they reach zero, at which point a state machine transition condition might be satisfied) are subscripted[T]. Arrows show variables used to communicate between machines and the selection logic: solid arrows for those controlled by the source (or by management associated with the source) and not changed by the target machine; open arrows for those that can be modified by the target.

FOOTNOTES:
1. RECEIVE would seem to be the obvious place to use Actor_Admin_Port_State, no instance of use in the current revision draft, so currently unclear whether RECEIVE is natural home of the whole of Actor_Oper_Port_State.
2. Calling out components of Actor_Oper_Port_State used or modified by RECEIVE.
3. The whole of Partner_Oper_State[…] is provided to TRANSMIT, so it can be reflected to the partner.
4. Removed port_moved, see discussion with Receive Machine suggestion.
5. Can be set to UNSELECTED by the Receive machine, but is not used by it.
6. Remove update_Default_Selected(), see Receive Machine suggestion.
7. Set by the system when transmission is permitted, can be (but is not necessarily) cleared by the system following LACPDU transmission, or indeed transmissions by other protocols competing for limited and rate-limited transmit resources.



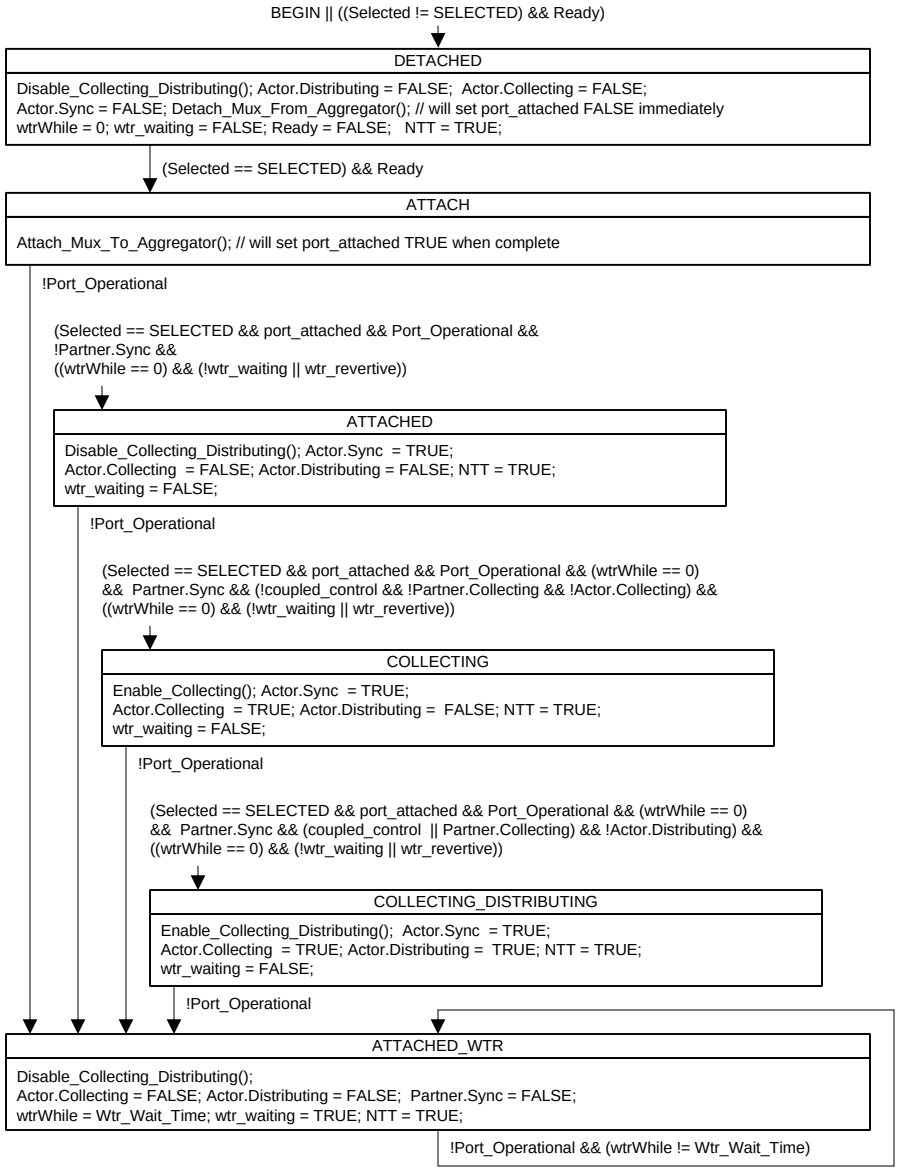
A suggested replacement for the 802.1AX-Rev D0.1 machine, may need further updating when that draft is updated. Intended to fix problems with the D0.1 port_moved variable (by removing it) and with LACP_Enabled (replaced by operPointToPointMAC), see discussion below.

D0.1 port_moved problems: 1. The !Port_operational && !port_moved transition to PORT_DISABLED would be defeated if port_moved could be set in any of the other states, either when Port_Operational is still true or before this state machine acts on the transition. The state machines do not provide any way to avoid the later possibility even if all the state machines for all ports operate mutually exclusively (an unrealistic constraint in a large system). 2. If the defaults for this port cause the conflict with the other port then the port_moved transition to INITIALIZE will be executed for every LACPDU received on the other , but without any useful effect. 3. Since Partner.Sync will be cleared on !Port_Operational and not set again until Port_Operational and receipt of a LACPDU, port_moved doesn't make any additional contribution to preventing unwanted connectivity, though the execution of INITIALIZE does remove potentially useful management information.

D0.1 LACP_Enabled problems: The name and description of this variable confused the possibility of link aggregation (deemed impossible, or at least undesirable to the extent it should be prevented, for non-point-to-point links) with sending and receiving LACPDUs. The description also implied that LACP_Enabled could combine further management controls. Curiously LACP_Enabled could be set without asserting BEGIN, but not cleared. This suggested machine adopts the point of view that management control is exerted by changing the value of Actor_Admin_Port_State.LACP_Enabled, while operPointToPointMAC clears the corresponding Oper state. The D0.1 machine assumed that both ends of the link saw the same value for operPointToPointMAC, which is not necessarily the case if there are intervening lower level bridges (such as TPMRs). This machine therefore clears Actor_Oper_Port_State.Aggregation (rather than Partner_Oper_Port_State.Aggregation) for non-point-to-point links. That change means the Actor's decision about aggregatability can be communicated to his Partner, which is useful as a LACP capable partner can then attach the link (as an individual link) without delay. Since the LACP_DISABLED has been removed there is no need to add an explaination how any received LACPDUs are handled in that state. To avoid sending LACPDUs on links which the Actor and all potential partners believe to be non-point-to-point it sets the Actor PASSIVE i.e. FALSE when operPointToPointMAC is FALSE.

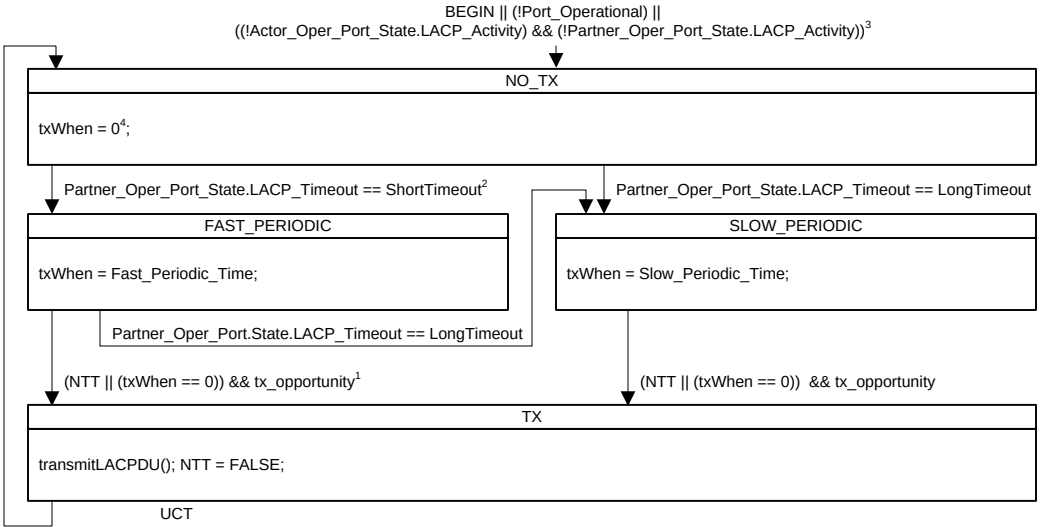
Assignments to Actor_Oper_Port_State.Defaulted are shown explicitly, rather than hidden in recordDefault(). The D0.1 state machine kept Defaulted TRUE in CURRENT.

Additional Footnotes:
1. ShortTimeout is the value TRUE, assigned to the flag ...Port_State.LACP_Timeout.
2. The value of the short time out in local ticks (ticks being of a granularity sufficient to meet the timer accuracy spec).
3. Set unconditionally, removed update_Default_Selected(), a judgment call but seemd to be turd polishing to me.
4. Machines not guaranteed to run instantaneously, so rcvdLACPDU could be set even if !Port_Operational on entry to this state.
5. Added 27AUG 15.30 PST.



This machine reflects what (I believe) was agreed in July, i.e. the machine waits in ATTACHED_WTR until the Selection Logic sets Selected to UNSELECTED even if wtr_revertive is FALSE.

This version of the machine uses a separate wtr_waiting variable in addition to wtr_revertive (selection of revertive mode vs non-reverrtive mode) and the wtrWhile timer (which can be zero in ATTACHED_WTR if Port_Operational has been set for the required time but non-revertive mode has been selected. The Selection Logic can use wtr_waiting as an indication that the machine is in ATTACHED_WTR, and stuck there is non-revertive. The mode variable wtr_revertive can be changed by management at any time without additional operations or disruption of machine operation.



The 802.1AX-2014 transmit machine uses the states of the Periodic machine, though the idea of one machine knowing the state of another without any explicit communication variables does not fit the declared paradigm for machine design. This suggested Transmit state machine is similar to and no more complex than the 802.1AX-2014 Periodic machine, and provides the functionality of both transmit and periodic machines. It can also transmit fewer LACPDUs, since the txWhen timer is reinitialized whenever transmission occurs, so a fast periodic transmission that conveys no new information will not immediately follow a transmission prompted by the MUX or Receive machines.

1. tx_opportunity is set by the system when transmission is permitted, can be (but is not necessarily) cleared by the system following LACPDU transmission, or indeed transmissions by other protocols competing for limited and rate-limited transmit resources.

2. ShortTimeout is the value TRUE, assigned to the flag ...Port_State.LACP_Timeout.

3. This machine allows transmission when operPointToPointMAC is FALSE (previously prevented by !LACP_Enabled. See Receive Machine suggestion for reasoning.

4. Clearing txWhen in this state is not strictly necessary, since it is initialized in FAST_PERIODIC and SLOW_PERIODIC prior to use, but some may want all variables initialized by BEGIN.