State Machines for Rapid Reconfiguration

Mick Seaman

This note specifies the Rapid Spanning Tree Protocol proposed for P802.1w using the state machine formalism used for P802.3ad. This description is a straight forward development and completion of the state machines discussed at the York, September '99 802.1 interim meeting, though a revised layout may make these appear unfamiliar. It assumes that the reader is familiar with the original specification of the Spanning Tree Algorithm and Protocol, with the prior papers on Rapid Reconfiguration circulated to 802.1, and with the state machine methodology of 802.3 (modified to use C-like syntax in this note).

The operation of each bridge port is represented by two state machines:

a) The Port Timers State Machine
b) The Port Transition State Machine

The process of BPDU reception and BPDU transmission are also candidates for state machine description. They interact with the above machines by setting flag variables (in the case of reception) or resetting them (on transmission of an appropriate BPDU). The transmission process is responsible for its own rate limiting: a limit of 3 transmissions in any two second period is suggested.

The operation of the bridge as a whole is represented by the interaction between Bridge Ports specified, and by parameters of the bridge stored in 'Port 0'. This removes the need for any 'per Bridge' specification elements, and helps ensure the minimum dependencies between bridge ports. This in turn supports the development of implementations that scale well with increasing numbers of bridge ports.

This shift of focus to 'per port operation' is supported by underlying technical changes from the legacy algorithm:

a) Transmission of Configuration BPDUS is prompted by:
    1) Changes to the information that a Designated Port derives from the current Root Port.
    2) A port based Hello Timer.
b) Topology Changes and Topology Change Notifications are propagated by setting a 'tc_while' timer on each port through which the change or notification is to be propagated. This in turn causes regular TCNs to be sent through a Root Port (for the duration of the tc_while timer, or until a topology change acknowledgment is received). There is no longer a need for a 'tc_detected' flag for the bridge as a whole.

Port timers are simple down counters, decremented on a per second 'tick' until they reach zero. The Port Timers State machine provides this functionality for the following timers:

a) **hello_when**. When this timer 'expires' (is decremented to zero) the Port Transition State Machine causes a Configuration BPDU to be queued for transmission if the port is a Designated Port. The initial value for the timer is Hello_time (2 seconds by default).
b) **tc_while**. The interval for which TCNs are sent through the Root Port (for ever in the legacy algorithm in the absence of an acknowledgment) and for which Config BPDUs are sent with the Topology Change flag set (determined by the Root in the legacy algorithm). For RSTP, tc_while is twice hello time on point-to-point links where the partner bridge port is RSTP capable, and 'Max_Age' otherwise (non-RSTP capable partners or shared media).
c) **fd_while**. The Forward Delay timer of the legacy algorithm, with an initial value of Forward Delay. Note: this can be reduced to twice Hello_time for bridges implementing RSTP. Why this is so may be the subject of a future paper.

d) **info_age**. The time remaining before the information held for this port expires, i.e. before message age equals or exceeds max age for received information on this port.
e) **rr_while**. ‘recent root while’, the elapsed time since this port was a Root Port forwarding frames. The initial value for this timer is Forward Delay, as communicated by the Root Bridge, and this value is maintained for a Root Port forwarding frames.
f) **rb_while**. ‘recent backup while’, the elapsed time since this port was a Backup Port. The initial value for this timer is twice Hello_time.

   Note: rr_while and rb_while provide the functionality of the ‘Forwards’ and ‘Forwarder’ states introduced in prior papers. This specification adds no new Port States or Port Roles to the original algorithm, and models Rapid Reconfiguration as a set of accelerated transitions between those states.

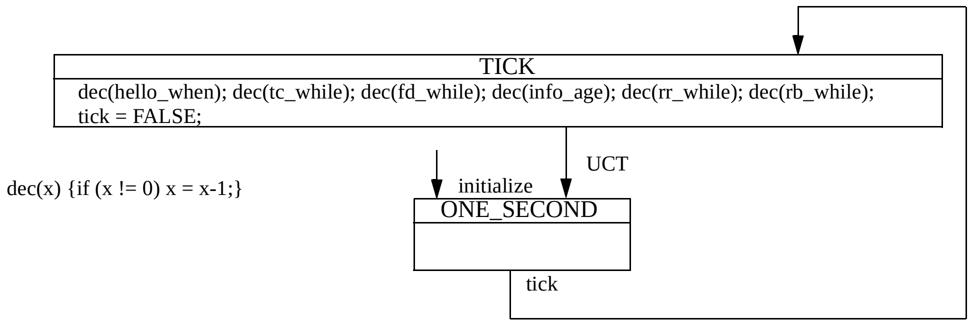

TICK
dec(hello_when); dec(tc_while); dec(fd_while); dec(info_age); dec(rr_while); dec(rb_while);
tick = FALSE;
UCT
initialize
ONE_SECOND
tick
dec(x) {if (x != 0) x = x-1;}

Figure 17-1—Port Timers

The Port Transition Machines assign Port Roles to Bridge Ports. It moves the Root Port and Designated Ports to the Forwarding Port State, and Alternate and Backup Ports to Blocking. It uses the following variables in addition to the timers already described:

a) **initialize**. This variable is common to all state machines, and is externally controlled. When asserted it forces all machines to their initial state. The Bridge Port becomes operational once initialize is deasserted.
b) **port_enabled**. This variable reflects the operational state of the MAC service supporting the bridge port and is TRUE if ‘port oper’ is ‘up’, and FALSE otherwise.
c) **role**. The assigned Port Role. The port is either a Disabled_port, a Root_port, a Designated_port, an Alternate_port, or a Backup_port.
d) **reselect**. This variable is TRUE if the Port Role has to be reassigned before any of the other variables or conditions can be processed.

   Note: Port Role reassignments can happen through changes in the spanning tree information received on other ports, so reselect is not always TRUE immediately prior to a role change. The scope and impact of reselect = = TRUE can be used to set the target for an implementation that aspires to optimal performance through minimizing interactions between the separate port machines. Good luck!

   Formally this state machine methodology calls for infinitely fast continuous execution of all machines, so reselect should not remain TRUE for any time. However a practical software implementation can use ‘reselect’ from any port to schedule a bridge wide recomputation of port roles. Until the recomputation is complete ‘reselect’ can remain TRUE to inhibit other processing.
e) **learn**. This is the administrative state for the source address learning function for this port provided by the Bridge Relay Entity.
f) **forward**. This is the administrative state for the packet forwarding function.
g) **learning**. This is the operational state for the source address learning function. This description assumes that some bridge port process, not explicitly described, monitors the administrative state

variables learn and forward, and rapidly though not instantaneously changes the operational states to correspond to the administrative states.

h) **forwarding**. This is the operational state for the packet forwarding function.

i) **info**. This variable describes the source of the spanning tree information for this port.
If **info** is **Received**, the port has received current (not aged out) information from the Designated Bridge for the attached LAN (a point-to-point bridge link being a special case of a LAN).
If **info** is **Mine**, information for the port has been derived from the Root Port for the Bridge (with the addition of root port cost information). This includes the possibility that the Root Port is 'Port 0', i.e. the bridge is the Root Bridge for the Bridged Local Area Network.
If **info** is **Aged**, information from the root port has been aged out. Just as for 'reselect' (see above), the state machine does not formally allow the 'Aged' state to persist. However if there is a delay in recomputing the new root port, correct processing of a received BPDU is specified.
Finally if the port is disabled, **info** is **Disabled**.

j) **retiring_roots**. A signal controlled by the Root Port. If set, it instructs any Designated Ports with the **rr_while** timer still running (recent roots) to revert to the Listening Port State (**learning** and **forwarding** both FALSE). The **rr_while** timer for that port is stopped when that has been accomplished.

k) **recent_roots**. This is not an independent variable, but is TRUE if any port other than the port whose state machine it appears in has **rr_while** running.
A simpler but imperfect description is that **recent_roots** is TRUE if any other port other than the Root Port has **rr_while** running. The first description allows for the possibility of more than one port considering itself to be the unique Root Port for an instant while **role** changes are taking place.
A practical software implementation of **recent_roots** that avoids repeatedly scanning all ports, is to maintain a bridge wide count of ports that have **rr_while** running, incrementing or decrementing this count as **rr_while** changes for each individual port. For a given port **recent_roots** is calculated by subtracting one from that count if **rr_while** for that port is running.

l) **rcvd_info**. True if spanning tree information (Root, Root Path Cost, Designated Bridge, Designated Port) has been received in a BPDU.

m) **rcvd_tcn**. True if a TCN BPDU has been received. Also true if any other BPDU, yet to be invented, has been received signalling a topology change notification.

n) **rcvd_tc_ack**. True if a configuration BPDU with a Topology Change Acknowledge flag has been received.

o) **rcvd_tc**. True if a configuration BPDU with a Topology Change flag has been received.

p) **rcvd_di**. True if a 'Rapid Configuration BPDU' indicating a Designated Port's desire to receive a confirmation of its role and permission to rapidly transition to forwarding has been received (however this is to be encoded).

q) **rcvd_dc**. True if a 'Rapid Configuration BPDU' giving a Designated Port permission to rapidly transition to forwarding has been received.

r) **txmt_info**. True if spanning tree information (Root, Root Path Cost, Designated Bridge, Designated Port) is to be transmitted.

s) **txmt_tcn**. True if a topology change notification is to be transmitted, either as a TCN BPDU or (possibly) as a flag in a 'Rapid Configuration BPDU'.

t) **txmt_tc**. True if a configuration BPDU with a topology change flag set is to be transmitted.

u) **txmt_tc_ack**. True if a configuration BPDU with a topology change acknowledge flag set is to be transmitted.

The Port Transition State Machine also makes use of the following procedures:

To avoid much repetition it is convenient to define the 'spanning tree priority' of spanning tree information as the number formed from the concatenation of the Root Priority, Root Identifier, Root Path Cost, Designated Bridge Priority, Designated Bridge Identifier, Designated Port Priority, and Designated Port Identifier, with each of these components always having greater significance than its successor. If the spanning tree priority of two sets of information are compared, the numerically lower has the 'higher' priority. The spanning tree priority of information plus a Port Cost component for the port on which it was received is also defined. The Port Cost component is added to the Root Path Cost.

a) **better_msg**. Returns TRUE if the spanning tree information received in a BPDU:
    1) has a higher priority than the information current for the port, just as for the legacy algorithm.
    2) is from the same Designated Port as the current information, i.e. from the same Designated Bridge and Designated Port priority and number.
b) **select_as_root_port**. Returns TRUE if (info == Received) for the port and its spanning tree information plus Port Cost has a higher priority than any other port with (info == Received), and a higher priority than the information for ‘Port 0’.
c) **select_as_alternate_port**. Returns TRUE if (info == Received) for the port, the port is not selected as the root port, but the information for the root port plus the root port’s port cost does not have a higher priority than the information for this port without the inclusion of its own port cost component, and the Designated Bridge identified by the received information is not this bridge.
d) **select_as_backup_port**. Returns TRUE if the port would otherwise be an alternate port but the Designated Bridge identified by the received information for the port is this bridge.
e) **select_as_designated_port**. Returns TRUE if the information for the root port plus the root port’s own port cost has a higher priority than the information for this port without the inclusion of its own port cost component.
f) **tc_prop(port)**. Starts **tc_while** on all other ports.
g) **flush(port)**. Removes all address entries learned on this port from the filtering database.
h) **flush_others(port**). Removes all address entries learned on all other ports from the filtering database.

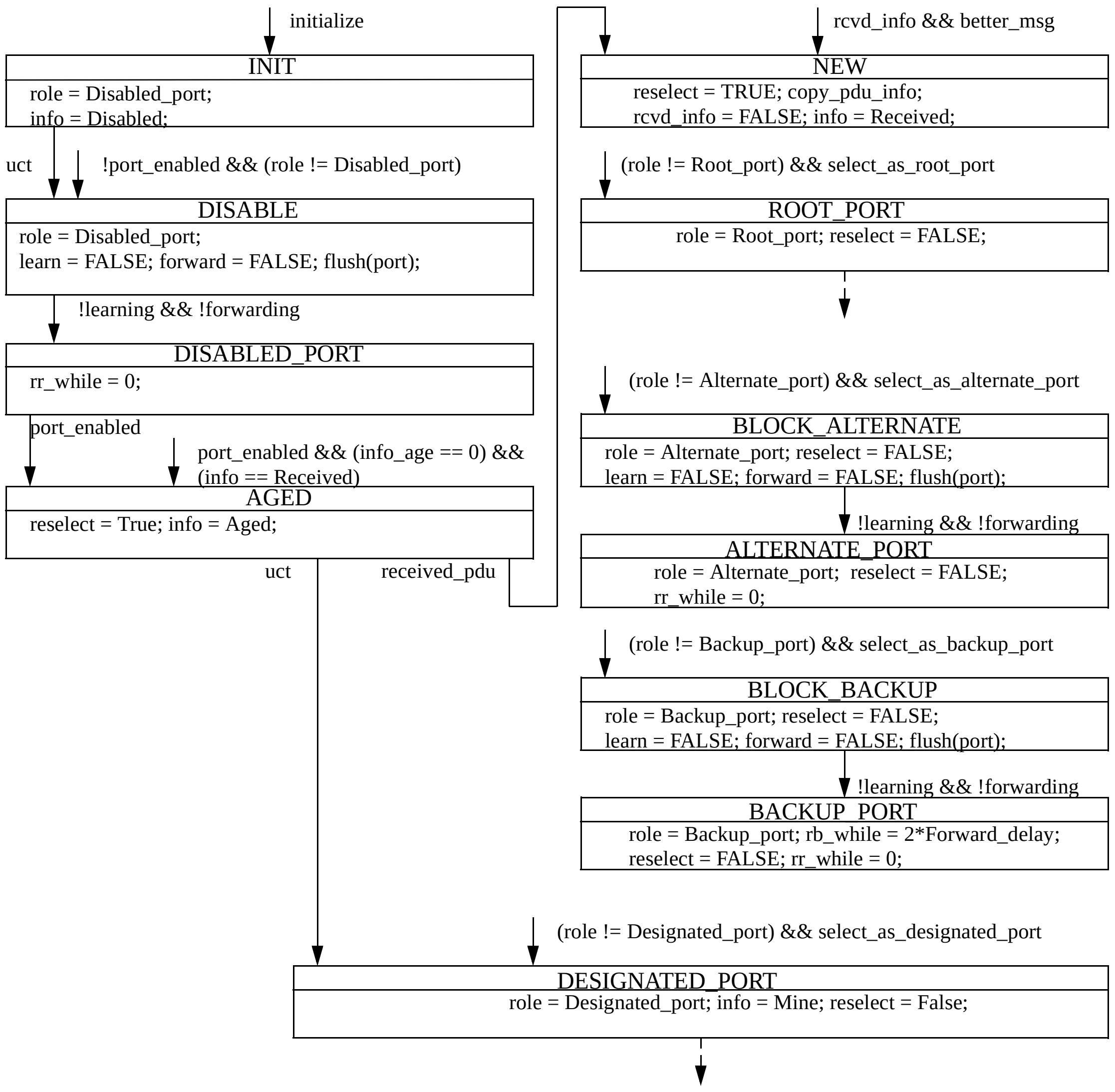
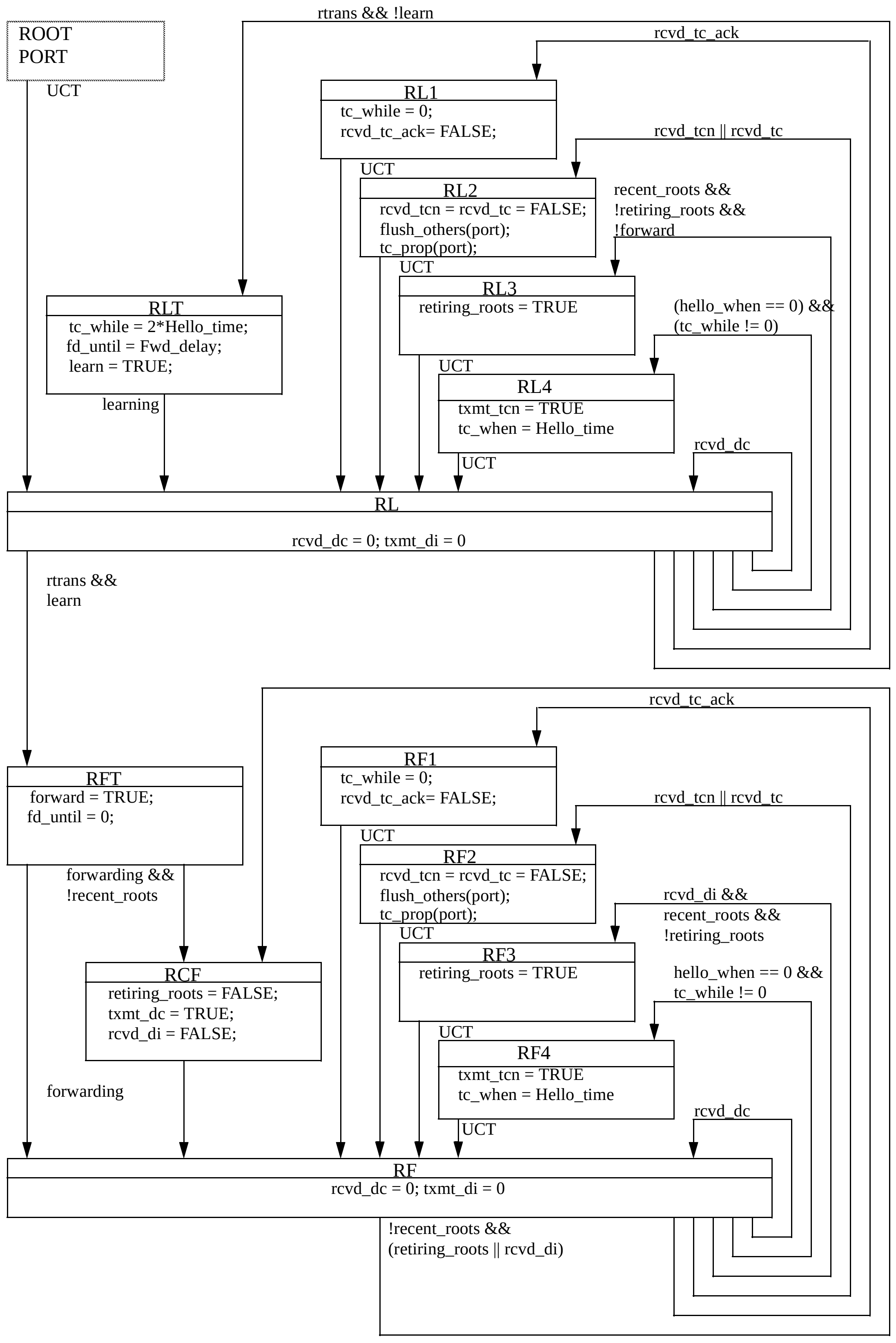
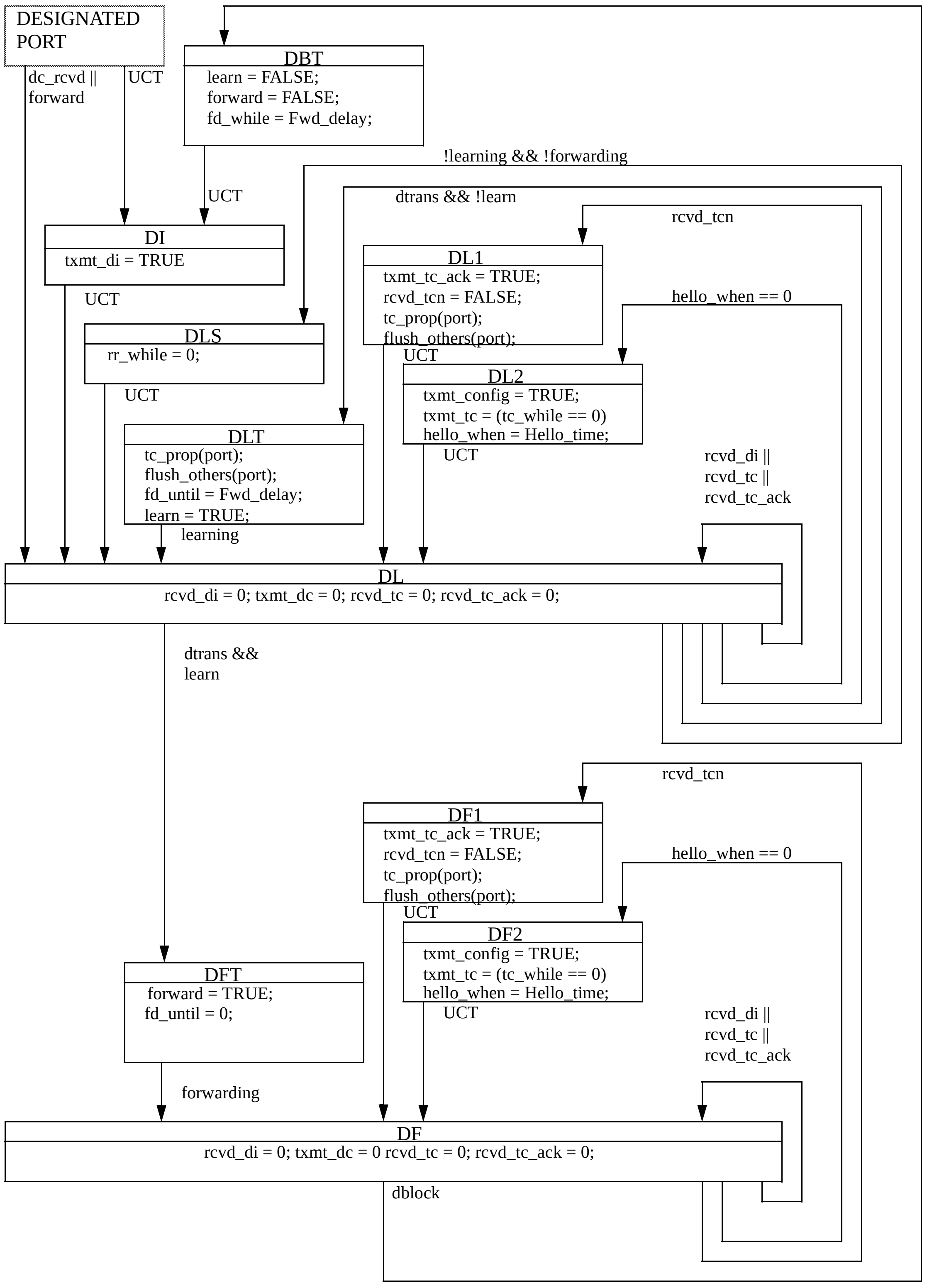

initialize
INIT
role = Disabled_port;
info = Disabled;
uct
!port_enabled && (role != Disabled_port)
DISABLE
role = Disabled_port;
learn = FALSE; forward = FALSE; flush(port);
!learning && !forwarding
DISABLED_PORT
rr_while = 0;
port_enabled
port_enabled && (info_age == 0) && (info == Received)
AGED
reselect = True; info = Aged;
uct
received_pdu
rcvd_info && better_msg
NEW
reselect = TRUE; copy_pdu_info;
rcvd_info = FALSE; info = Received;
(role != Root_port) && select_as_root_port
ROOT_PORT
role = Root_port; reselect = FALSE;
(role != Alternate_port) && select_as_alternate_port
BLOCK_ALTERNATE
role = Alternate_port; reselect = FALSE;
learn = FALSE; forward = FALSE; flush(port);
!learning && !forwarding
ALTERNATE_PORT
role = Alternate_port; reselect = FALSE;
rr_while = 0;
(role != Backup_port) && select_as_backup_port
BLOCK_BACKUP
role = Backup_port; reselect = FALSE;
learn = FALSE; forward = FALSE; flush(port);
!learning && !forwarding
BACKUP_PORT
role = Backup_port; rb_while = 2*Forward_delay;
reselect = FALSE; rr_while = 0;
(role != Designated_port) && select_as_designated_port
DESIGNATED_PORT
role = Designated_port; info = Mine; reselect = False;

Figure 17-1—Port Role Transitions



rtrans: ((fd_until == 0) || (!recent_roots && (rb_until == 0)))

Figure 17-2—Root Port Transitions



dblock: (rr_while && retiring_roots && !dc_rcvd)
dtrans: ((fd_until == 0) || dc_rcvd)

Figure 17-3—Designated Port Transitions