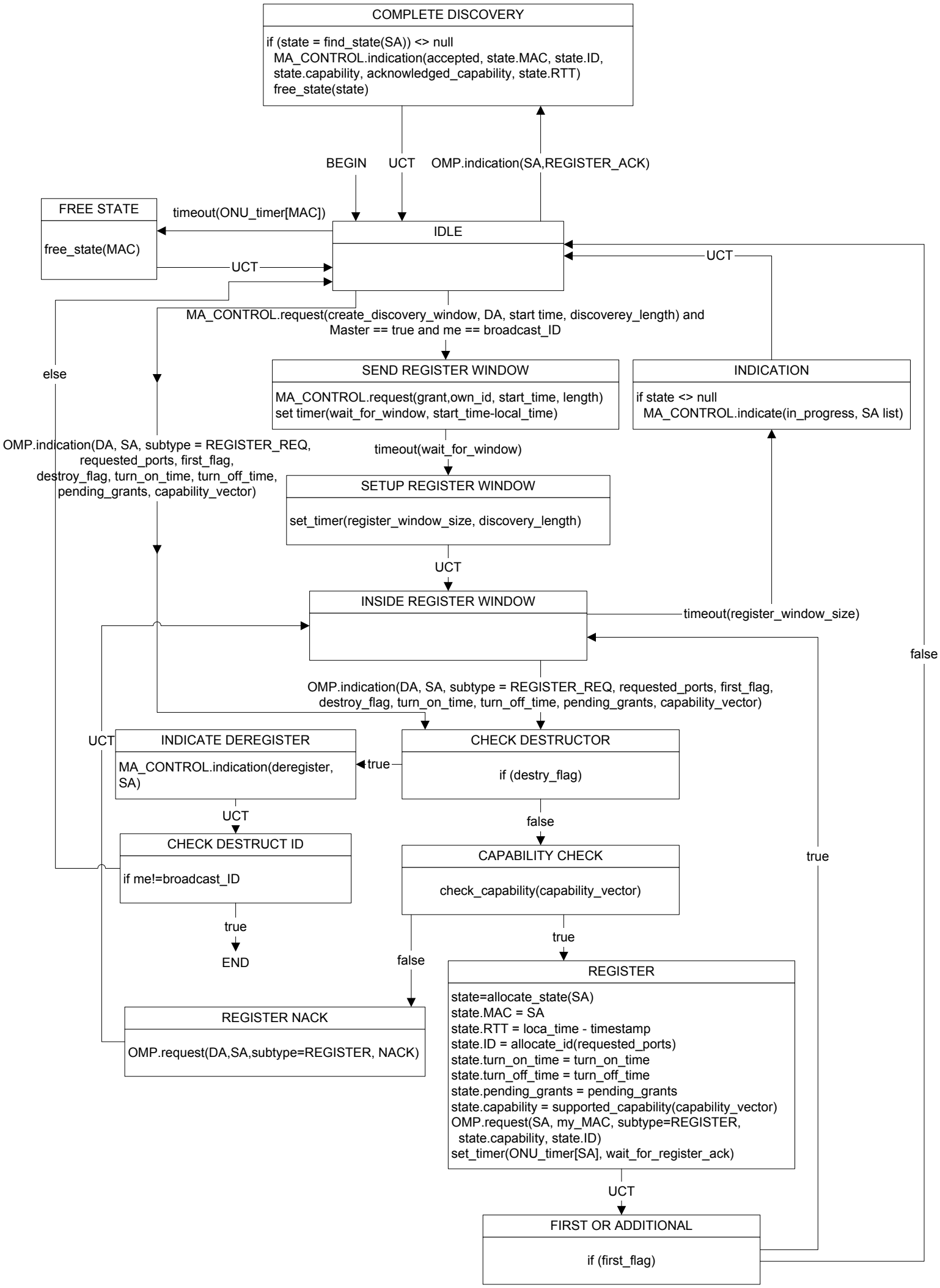
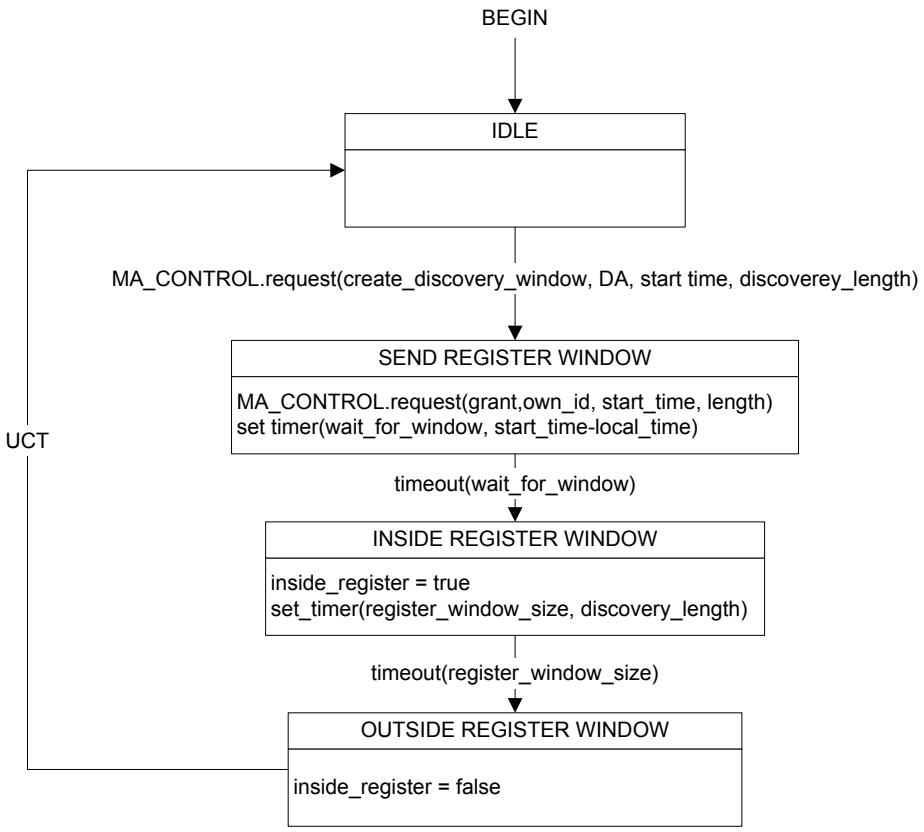
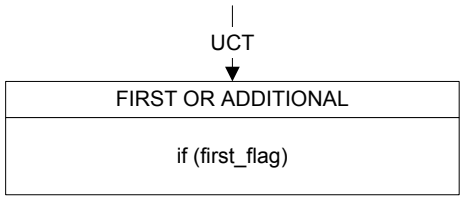
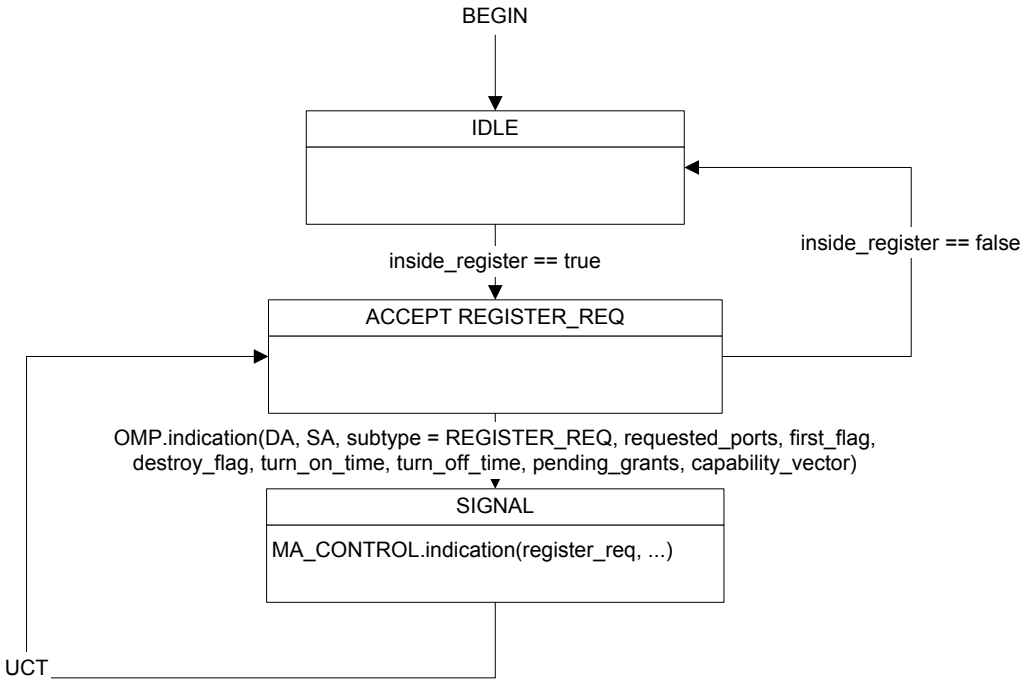
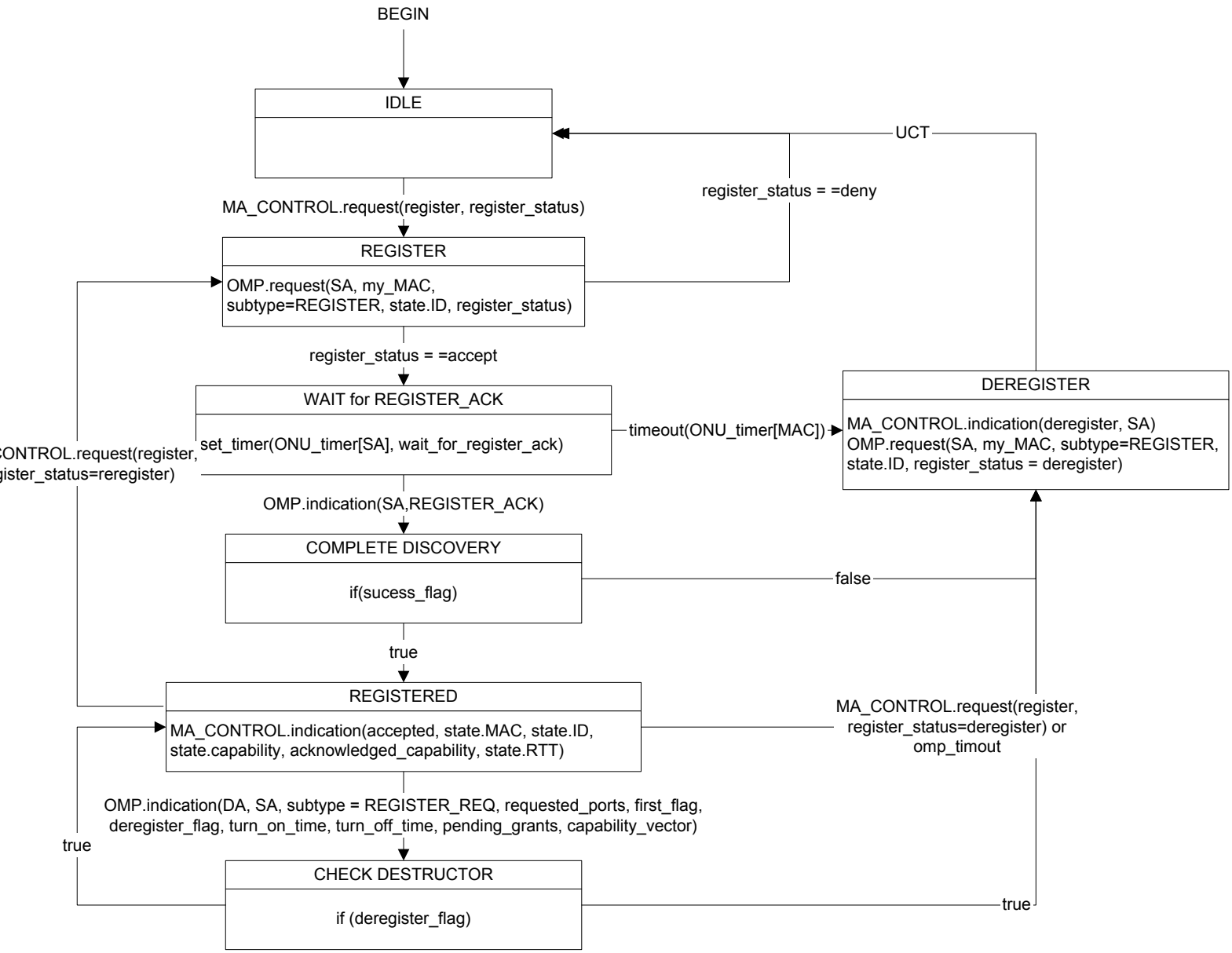

COMPLETE DISCOVERY
if (state = find_state(SA)) <> null
MA_CONTROL.indication(accepted, state.MAC, state.ID, state.capability, acknowledged_capability, state.RTT)
free_state(state)
BEGIN
UCT
OMP.indication(SA,REGISTER_ACK)
FREE STATE
free_state(MAC)
timeout(ONU_timer[MAC])
UCT
IDLE
UCT
MA_CONTROL.request(create_discovery_window, DA, start time, discoverey_length) and Master == true and me == broadcast_ID
else
SEND REGISTER WINDOW
MA_CONTROL.request(grant,own_id, start_time, length)
set timer(wait_for_window, start_time-local_time)
INDICATION
if state <> null
MA_CONTROL.indicate(in_progress, SA list)
OMP.indication(DA, SA, subtype = REGISTER_REQ, requested_ports, first_flag, destroy_flag, turn_on_time, turn_off_time, pending_grants, capability_vector)
timeout(wait_for_window)
SETUP REGISTER WINDOW
set_timer(register_window_size, discovery_length)
UCT
INSIDE REGISTER WINDOW
timeout(register_window_size)
false
OMP.indication(DA, SA, subtype = REGISTER_REQ, requested_ports, first_flag, destroy_flag, turn_on_time, turn_off_time, pending_grants, capability_vector)
UCT
INDICATE DEREGISTER
MA_CONTROL.indication(deregister, SA)
true
CHECK DESTRUCTOR
if (destry_flag)
UCT
false
CHECK DESTRUCT ID
if me!=broadcast_ID
CAPABILITY CHECK
check_capability(capability_vector)
true
true
END
false
REGISTER
state=allocate_state(SA)
state.MAC = SA
state.RTT = loca_time - timestamp
state.ID = allocate_id(requested_ports)
state.turn_on_time = turn_on_time
state.turn_off_time = turn_off_time
state.pending_grants = pending_grants
state.capability = supported_capability(capability_vector)
OMP.request(SA, my_MAC, subtype=REGISTER, state.capability, state.ID)
set_timer(ONU_timer[SA], wait_for_register_ack)
REGISTER NACK
OMP.request(DA,SA,subtype=REGISTER, NACK)
UCT
FIRST OR ADDITIONAL
if (first_flag)


BEGIN
IDLE
MA_CONTROL.request(create_discovery_window, DA, start time, discoverey_length)
SEND REGISTER WINDOW
MA_CONTROL.request(grant,own_id, start_time, length)
set timer(wait_for_window, start_time-local_time)
UCT
timeout(wait_for_window)
INSIDE REGISTER WINDOW
inside_register = true
set_timer(register_window_size, discovery_length)
timeout(register_window_size)
OUTSIDE REGISTER WINDOW
inside_register = false


UCT
FIRST OR ADDITIONAL
if (first_flag)


BEGIN
IDLE
inside_register == true
inside_register == false
ACCEPT REGISTER_REQ
OMP.indication(DA, SA, subtype = REGISTER_REQ, requested_ports, first_flag,
destroy_flag, turn_on_time, turn_off_time, pending_grants, capability_vector)
SIGNAL
MA_CONTROL.indication(register_req, ...)
UCT


BEGIN
IDLE
UCT
register_status = =deny
MA_CONTROL.request(register, register_status)
REGISTER
OMP.request(SA, my_MAC,
subtype=REGISTER, state.ID, register_status)
register_status = =accept
WAIT for REGISTER_ACK
set_timer(ONU_timer[SA], wait_for_register_ack)
timeout(ONU_timer[MAC])
DEREGISTER
MA_CONTROL.indication(deregister, SA)
OMP.request(SA, my_MAC, subtype=REGISTER,
state.ID, register_status = deregister)
CONTROL.request(register,
gister_status=reregister)
OMP.indication(SA,REGISTER_ACK)
COMPLETE DISCOVERY
if(sucess_flag)
false
true
REGISTERED
MA_CONTROL.indication(accepted, state.MAC, state.ID,
state.capability, acknowledged_capability, state.RTT)
MA_CONTROL.request(register,
register_status=deregister) or
omp_timout
OMP.indication(DA, SA, subtype = REGISTER_REQ, requested_ports, first_flag,
deregister_flag, turn_on_time, turn_off_time, pending_grants, capability_vector)
true
CHECK DESTRUCTOR
if (deregister_flag)
true