This (draft) state machine is the bridge side state machine for the VSI association (discovery) protocol. A state machine for the server (Hypervisor) side can be developed if needed. This state machine is intended to implement the following behaviors:

1) Idempotency – the repetition of a message has the same result as the first message; there are no damaging side effects to repeating a message.

2) Soft state – an association ages out after a time if not refreshed; this makes extra validation and lost message recovery unnecessary. (Currently the holding time is shown as a constant, but a requested value could be included in the protocol PDU if desired.)

3) Progression – the fetch, reservation, and association (activation) steps for configuring a VSI and port profile are allowed individually or together, with a request for a later step implying a request for the earlier step(s) if they have not yet been done.

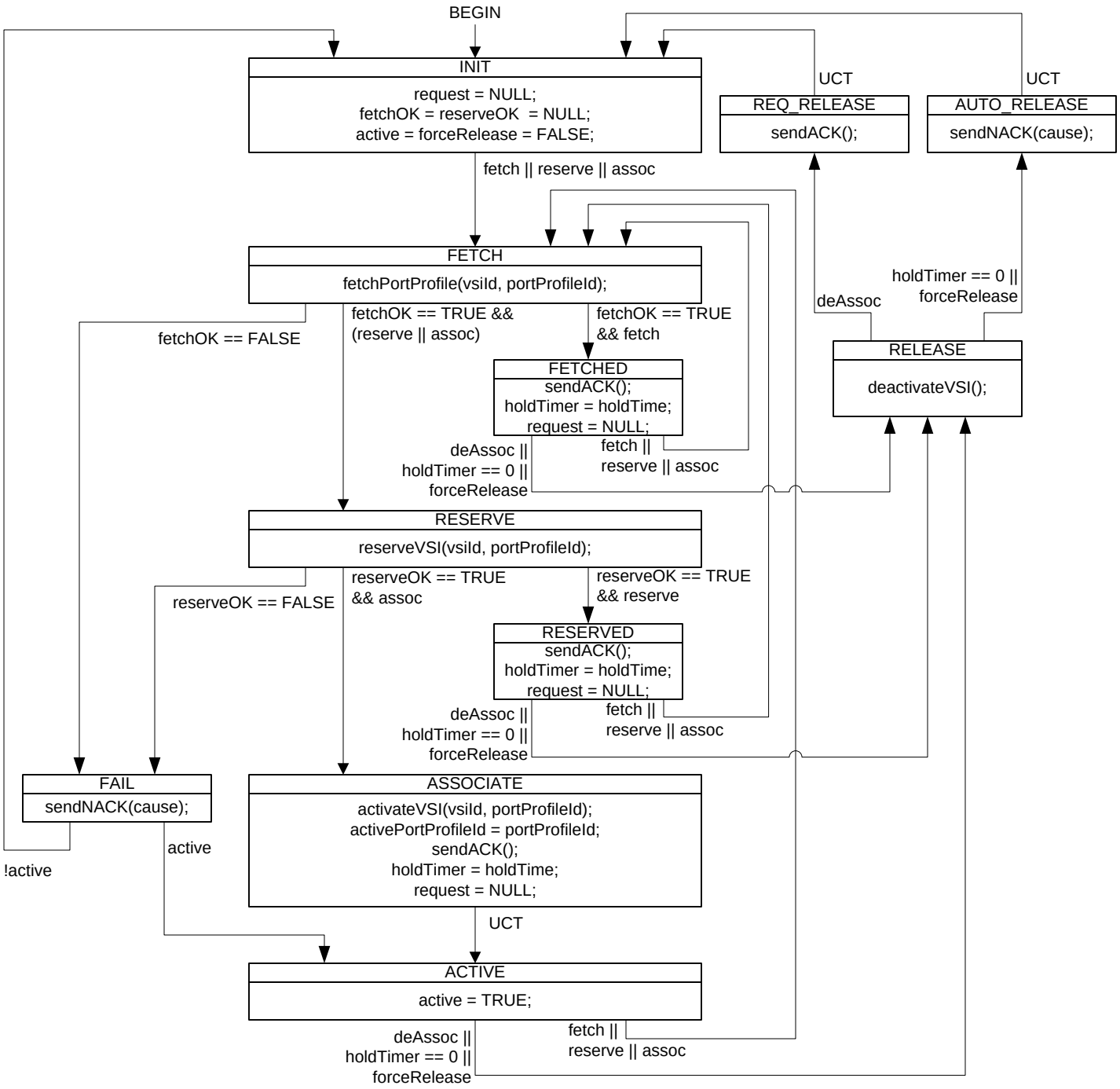
4) If the parameters conveyed in a PDU are changed in a subsequent PDU the functions invoked by the state machine are expected to make the necessary modifications to the configuration. If this is not possible (e.g., failure to fetch or reserve a new port profile) and the VSI association is active, the existing active association will continue to be in effect.

Control Variables:
holdTime – amount of time VSI association is kept without refresh
vsiId – identifier of the VSI carried in the PDU; there is a separate association state machine for each VSI
portProfileId – identifier of port profile for the VSI carried in the PDU, used to retrieve port provisioning information from port profile manager
activePortProfileId – identifier of currently active port profile for the VSI
request – request conveyed by the PDU being processed (one of: *NULL*, *fetch*, *reserve*, *assoc*, *deAssoc*)
forceRelease – enables forced (unsolicited) release of a VSI association by the bridge

Operational Variables:
holdTimer – timer for aging out VSI association
fetchOK – result of attempt to retrieve port profile data from port profile manager (one of: *NULL*, *TRUE*, *FALSE*)
reserveOK – result of attempt to reserve resources for a particular VSI and port profile (one of: *NULL*, *TRUE*, *FALSE*)

Functions:
fetchPortProfile() – initiates the process of retrieving port profile data from the port profile manager; when this process is completed or abandoned the value of *fetchOK* is set to TRUE or FALSE according to whether or not the port profile was successfully retrieved
reserveVSI() – reserves resources on the bridge for a particular VSI and port profile
activateVSI() – activates a particular VSI port profile
deactivateVSI() – deactivates a particular VSI and port profile and releases reserved resources
sendACK() – sends a VSI association protocol ACK PDU
sendNACK() – sends a VSI association protocol NACK PDU, including a cause

NOTES:
1) For this state machine three request messages are assumed: pre-associate, associate, and de-associate. To provide independence between fetching a port profile and reserving resources a **reserve bit** is assumed to be carried in the pre-associate message. When a pre-associate message is received, if the **reserve bit** is *set* **request** is set to *reserve*, otherwise **request** is set to *fetch*. When an associate or de-associate message is received, **request** is set to *assoc* and *deAssoc* respectively. In the state machine diagram the expressions *fetch*, *reserve*, *assoc*, and *deAssoc* correspond to **request** equaling that value (i.e., “assoc” means “request == assoc”).
2) The state machine assumes that the functions can determine if some information has changed from a previous invocation and perform the necessary work to update the bridge configuration to match the current request.
3) When a message is received **request** is set according to the message and the state machine runs until it clears **request** before processing the next message received (that is, before setting **request** once again according to the next message).
4) When a pre-associate or associate message is received, **portProfileId** is set to the value carried in the message.
5) The ACK and NACK PDUs are intended to communicate the current state of the bridge state machine for the VSI that is the subject of a request or state transition (e.g., hold timeout). These messages may be safely repeated (idempotency) but this version of the state machine will send only one message in response to a request or other event. A repeated request will result in a repeated response. This behavior should be sufficient to deal with lost messages.
6) To force the association be be released **forceRelease** may be set by management action.
7) The expression **UCT** is always true. That is, it creates an unconditional state transition that occurs immediately after the processing defined for a state is completed.


BEGIN
INIT
request = NULL;
fetchOK = reserveOK = NULL;
active = forceRelease = FALSE;
UCT
REQ_RELEASE
sendACK();
UCT
AUTO_RELEASE
sendNACK(cause);
fetch || reserve || assoc
FETCH
fetchPortProfile(vsiId, portProfileId);
holdTimer == 0 ||
forceRelease
deAssoc
fetchOK == FALSE
fetchOK == TRUE &&
(reserve || assoc)
fetchOK == TRUE
&& fetch
RELEASE
deactivateVSI();
FETCHED
sendACK();
holdTimer = holdTime;
request = NULL;
deAssoc ||
holdTimer == 0 ||
forceRelease
fetch ||
reserve || assoc
RESERVE
reserveVSI(vsiId, portProfileId);
reserveOK == FALSE
reserveOK == TRUE
&& assoc
reserveOK == TRUE
&& reserve
RESERVED
sendACK();
holdTimer = holdTime;
request = NULL;
deAssoc ||
holdTimer == 0 ||
forceRelease
fetch ||
reserve || assoc
FAIL
sendNACK(cause);
!active
active
ASSOCIATE
activateVSI(vsiId, portProfileId);
activePortProfileId = portProfileId;
sendACK();
holdTimer = holdTime;
request = NULL;
UCT
ACTIVE
active = TRUE;
deAssoc ||
holdTimer == 0 ||
forceRelease
fetch ||
reserve || assoc