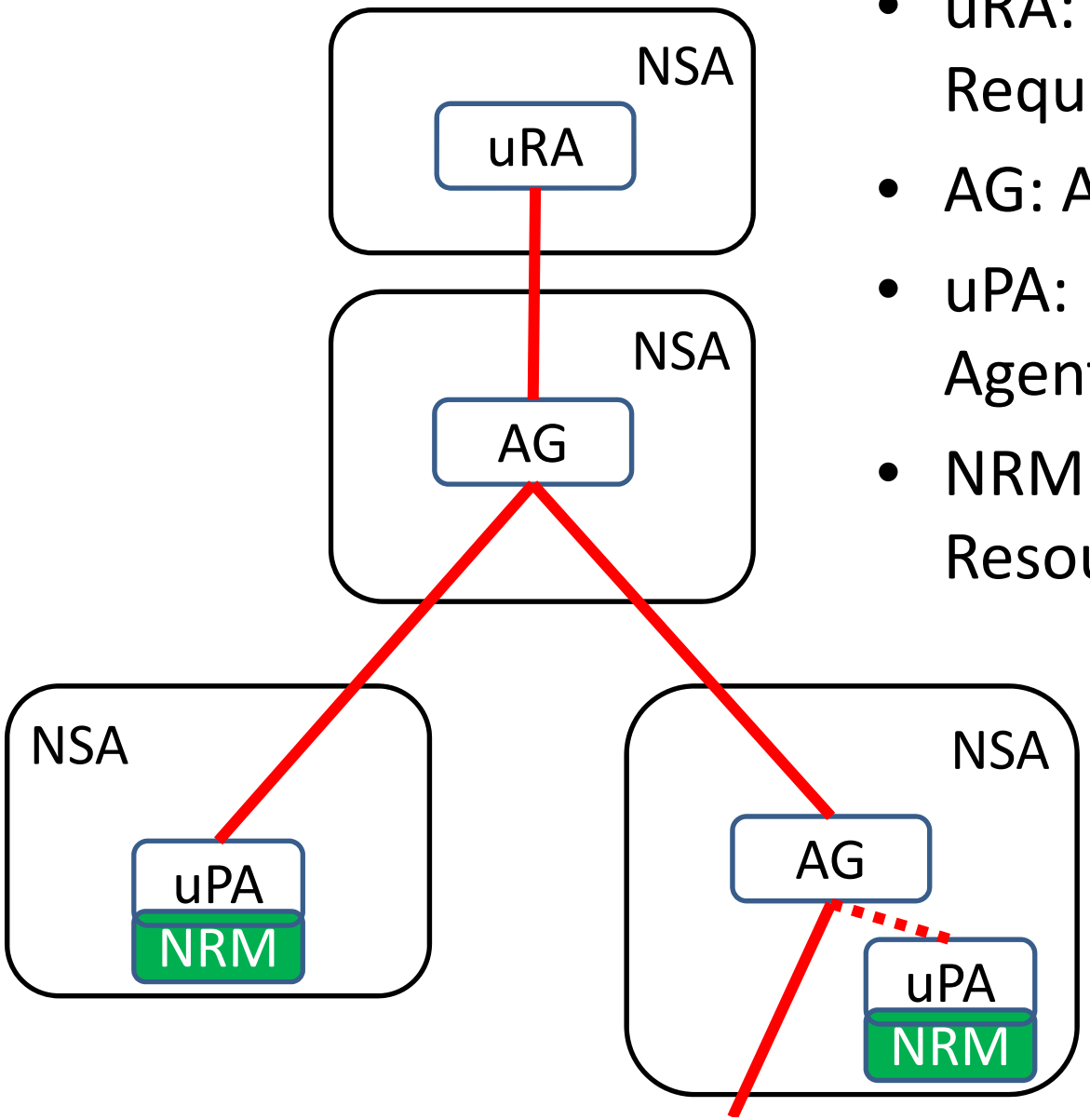
NSI CS Protocol State Machines and Message Handler

NSA: uRA, Aggregator and uPA

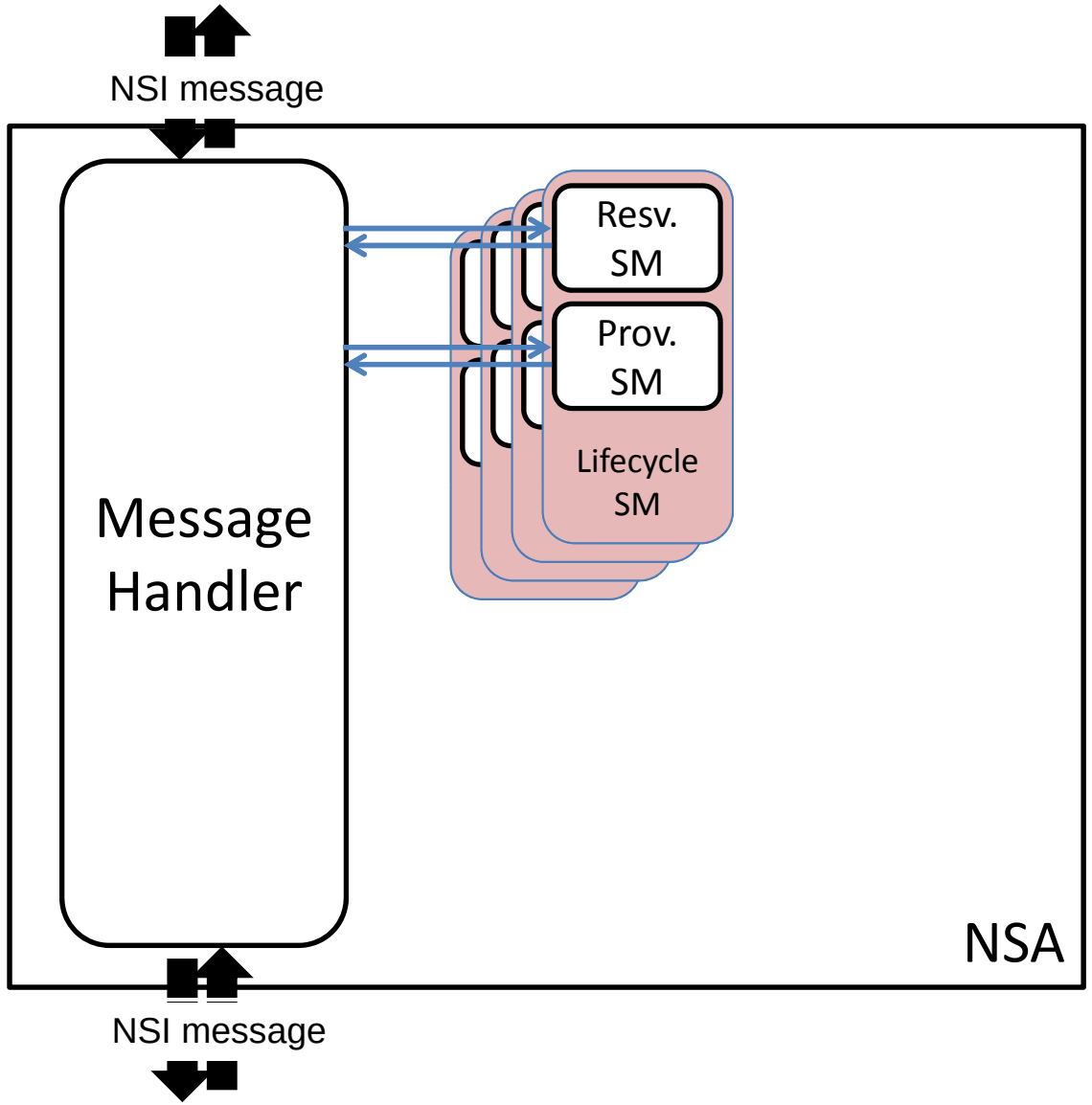


- uRA: Ultimate Requester Agent
- AG: Aggregator
- uPA: Ultimate Provider Agent
- NRM: Network Resource Manager

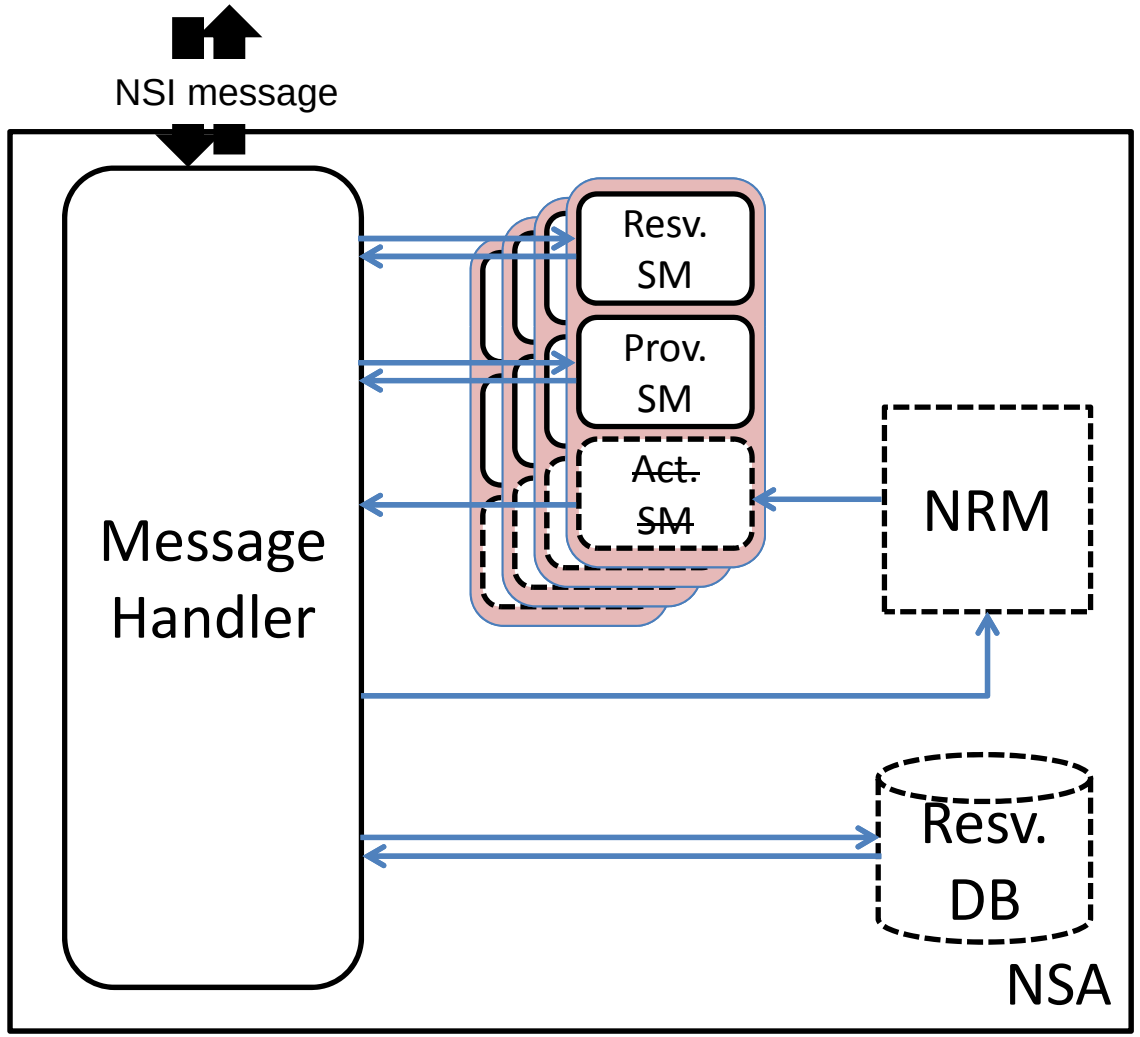
State Machines and Message Handler

- Behavior of NSI CS protocol is modeled as state machines and message handler
- State Machines:
  - RSM: Reservation State Machine
  - PSM: Provision State Machine
  - ~~ASM: Activation State Machine~~
  - LSM: Lifecycle State Machine
- Aggregator:
  - can talk to upstream and downstream NSAs
  - Has RSM, PSM and LSM
- uPA
  - Can talk to upstream NSAs only
  - Has RSM, PSM, ~~ASM~~ and LSM

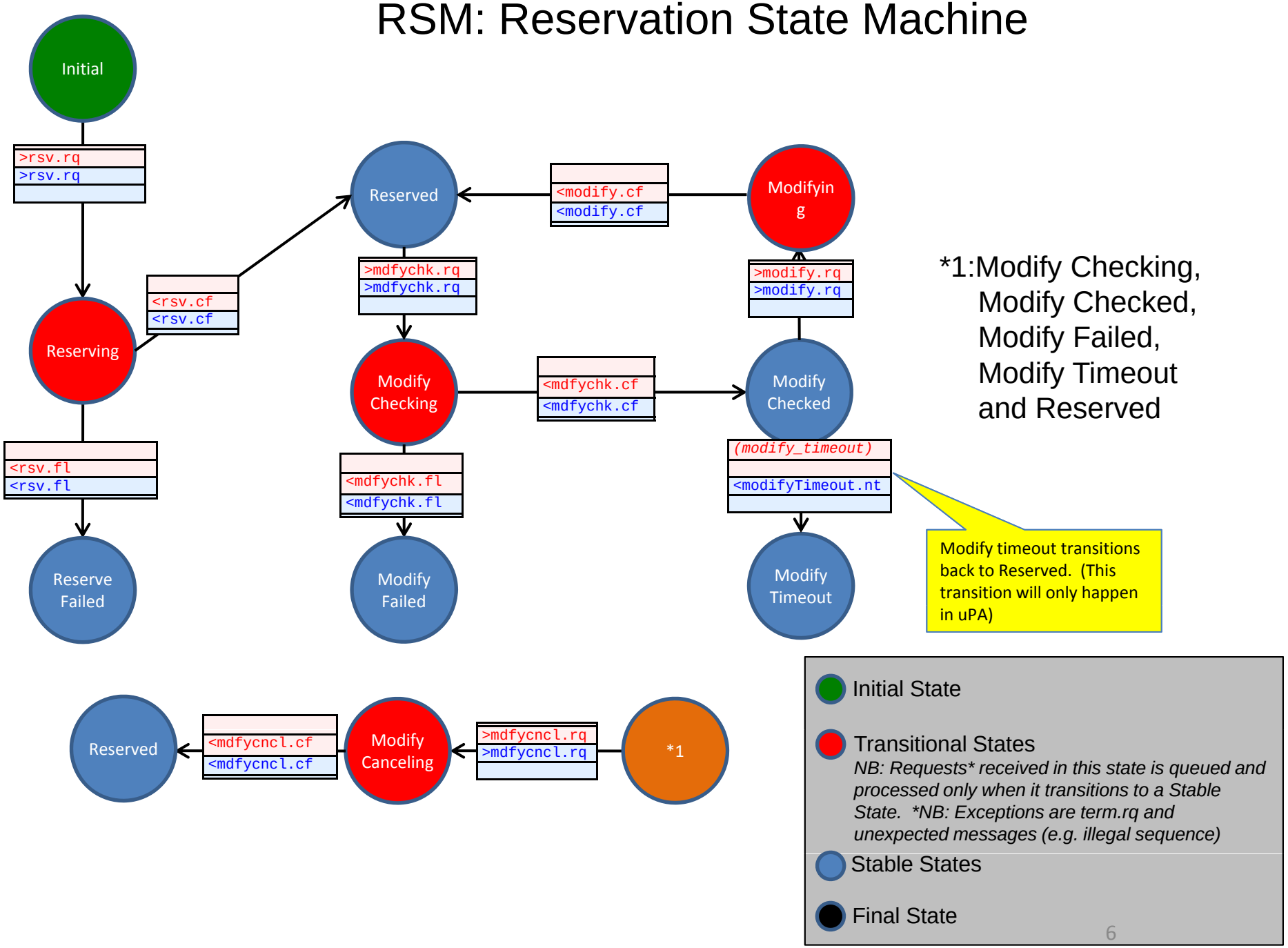
Aggregator


NSI message
Message Handler
Resv. SM
Prov. SM
Lifecycle SM
NSA
NSI message

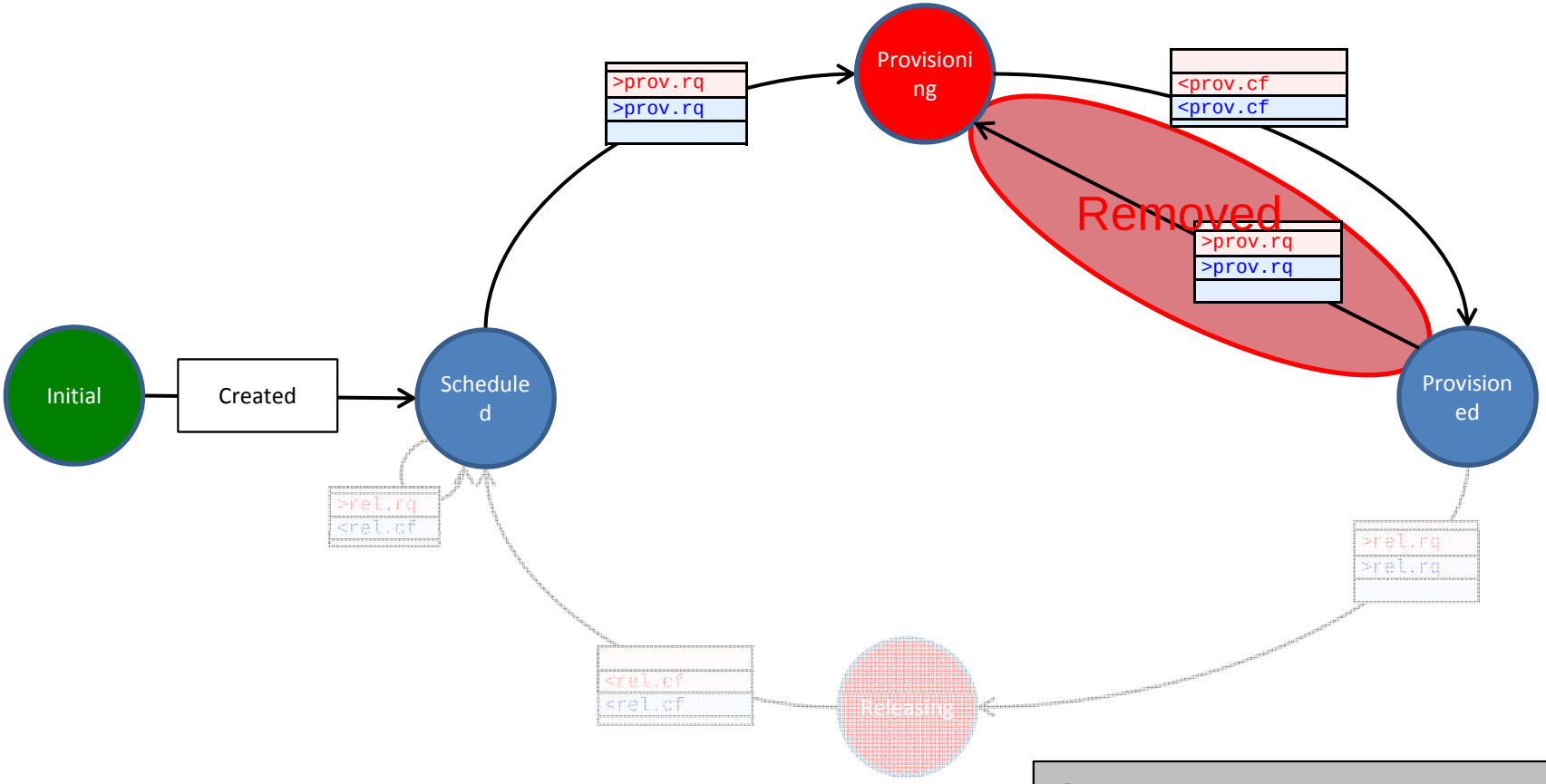
uPA


NSI message
Message Handler
Resv. SM
Prov. SM
Act. SM
NRM
Resv. DB
NSA

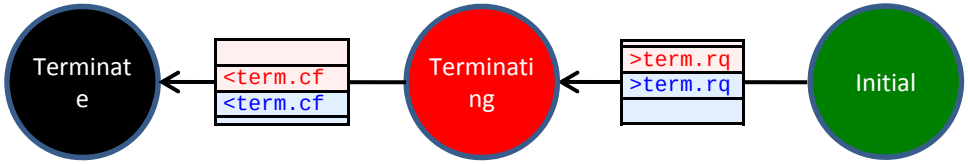
RSM: Reservation State Machine

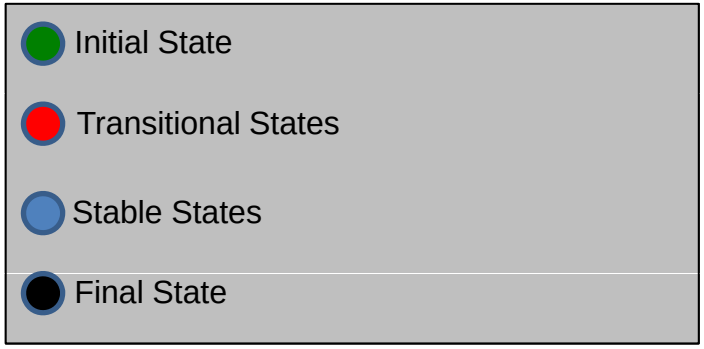


PSM : Provision State Machine

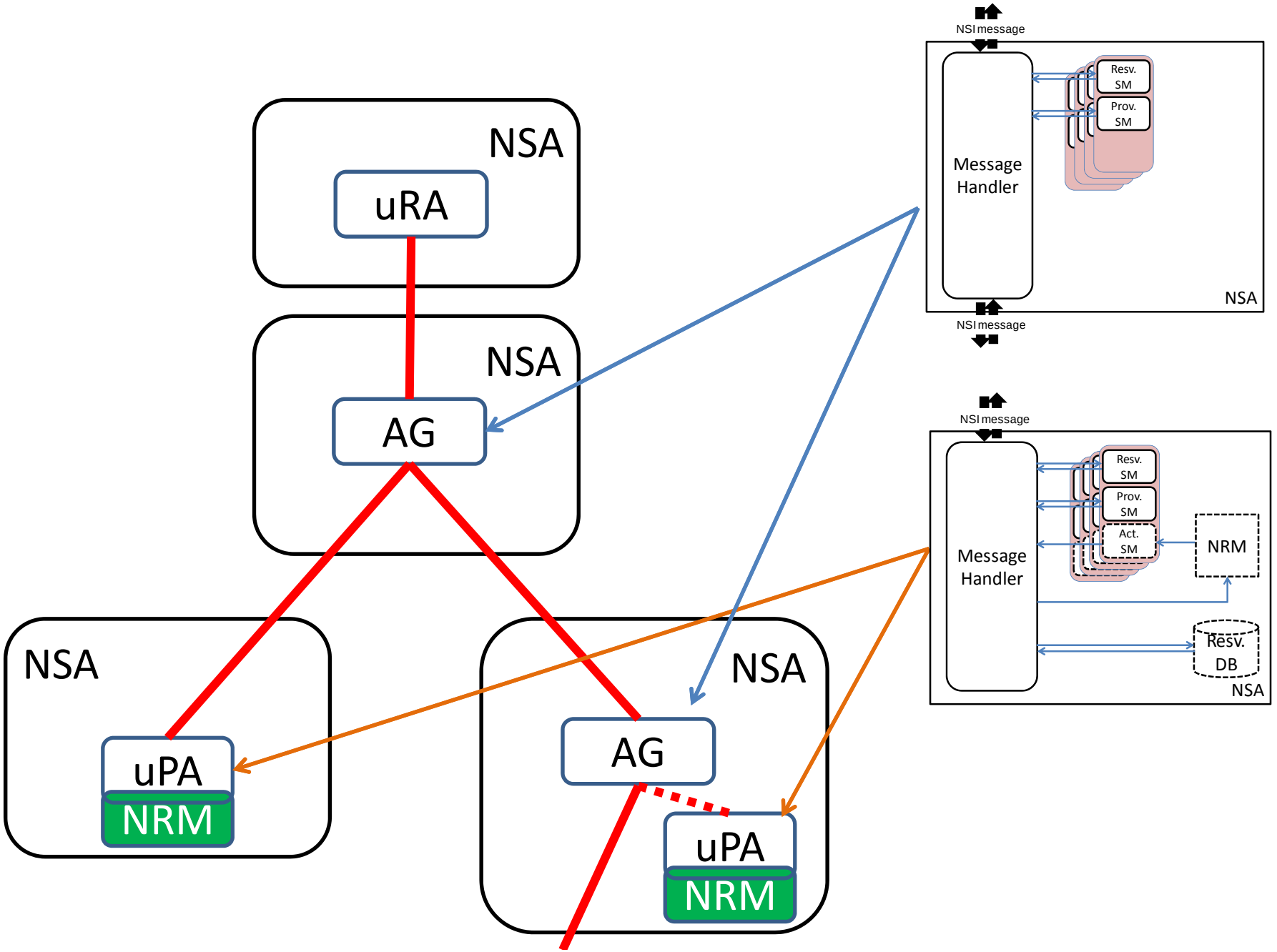


LSM : Lifecycle State Machine





NSA: uRA, Aggregator and uPA



Modify

- Modify operations modify reservation
- Currently, following two changes are supported
  - Change end time of a reservation
  - Change bandwidth
- Modify is a 2-phase operation
  - 1: check availability (ModifyCheck.rq)
    - Note: resources are held
  - 2: Commit (Modify.rq) or Abort (ModifyCancel.rq)
- When committed, the reservation is updated
  - Reservation has a version number assigned by uRA, and the version number is updated when committed (uniformly increasing)

Provision ~~and Release~~

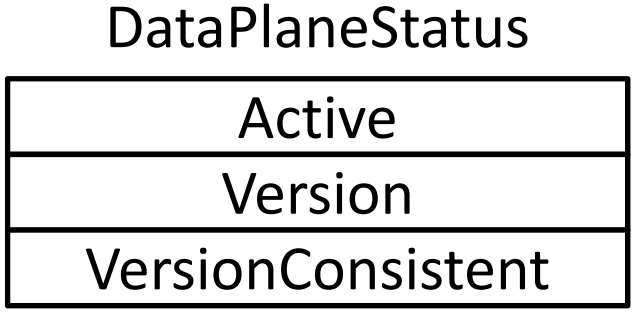
- Provision state machine is independent from the reservation state machine
- Provision state:
  - Data plane should be activated if the PSM is in “Provisioned” state AND start_time < current_time < end_time
  - Data plane shall not be activated before the start_time
- ~~A connection can be repeatedly provisioned and released~~

Data plane activation

- Data plane should be activated if the PSM is in “Provisioned” state AND start_time < current_time < end_time
- Activation is done at the timing of following events (if the above condition is met), using the latest reservation information
  - PSM transits to “Provisioned”
  - At the start_time
  - Reservation is updated (by commit of modify)
  - Data plane is recovered from an error
- Data plane activation/deactivation are notified by DataPlaneStateChange.nt notification messages.
- Errors are notified by a generic error message

DataPlaneStateChage.nt (1)

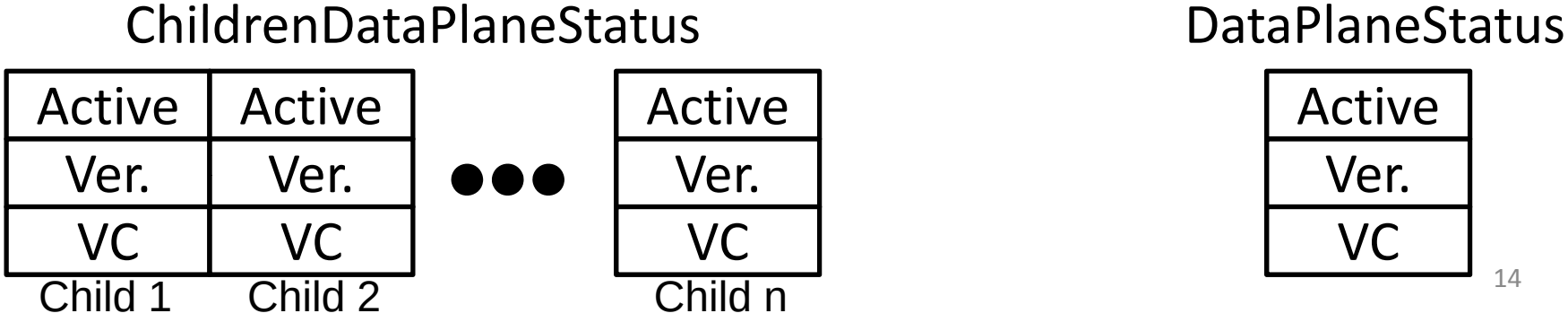
- PA and aggregator has DataPlaneStatus information
  - (Boolean) Active: True if data plane is active. For an aggregator, this flag is true when data plane is activated in all participating children
  - (Int) Version: For a uPA, current (latest) reservation version number. For an aggregator, the largest version number of the participating children. This field is valid when Active is true.
  - (Boolean) VersionConsistent: Always true for uPA. For an aggregator, If version numbers of all children are the same, This flag is true. This field is valid when Active is true.
- When a valid filed of DataPlaneStatus is changed, DataPlaneStatusChange.nt is sent up.



DataPlaneStateChage.nt(2)

- An aggregator keeps an array of statuses of its children, ChildrenDataPlaneStatus[1..n]
- Aggregator's DataPlaneStatus is determined by the following rule

```
if all of ChildrenDataPlaneStatus[1..n].Active are true
        then
        {
        DataPlaneStatus.Active = true
        DataPlaneStatus.Version =
            maximum_of(ChildrenDataPlaneStatus[1..n].Version)
        If all ChildrenDataPlaneStatus[1..n].Version are the same, and all
            of ChildrenDataPlaneStatus[1..n].VersionCosistent are true
                then DataPlaneStatus.VersionConsistent = true
                else DataPlaneStatus.VersionConsistent = false
        }
```



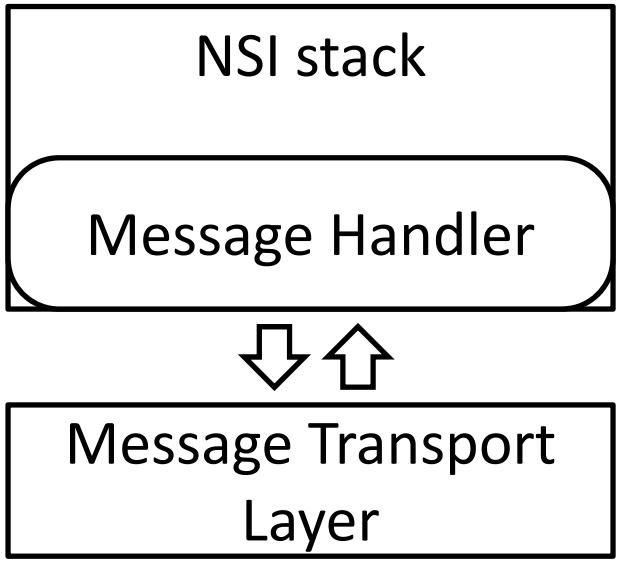
Notifications: Activation related

- There are no activateComplete.nt nor deactivateComplete.nt
- A general error message is used to notify following events. Those error are sent up the tree to uRA immediately
  - activateFailed: Activation failed at the time when uPA should activate its data plane
  - deactivateFailed: Deactivation failed at the time when uPA should deactivate its data plane
  - dataplaneError: Data plane is deactivate when deactivation is not expected. The error is recoverable.
  - forcedEnd: Something unrecoverable is happened in uPA/NRM

Requests which can fail

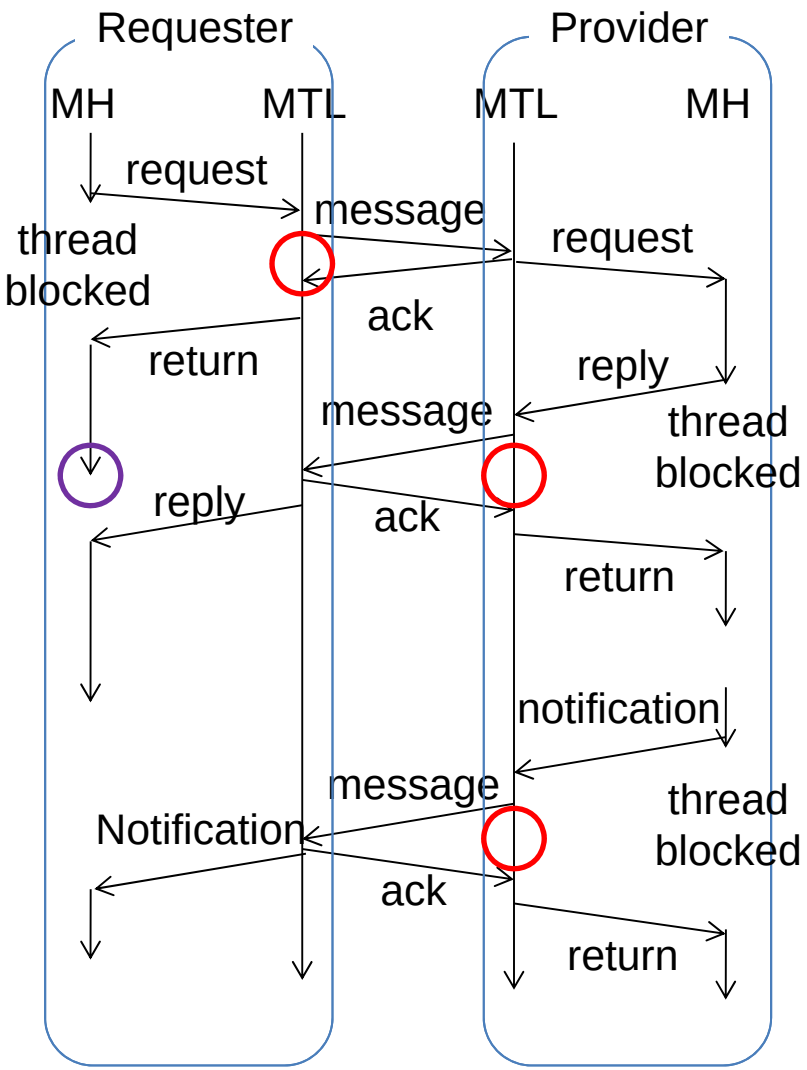
- Operations which can functionally fail are:
  - Reserve.rq
  - ModifyCheck.rq
  - Those requests fail when requested resources are not available.
- Other operation cannot fail. However, they can timeout in MTL/MH, or can be denied because they are invalid requests.
  - If a SM is at a state in which the request cannot be received, the request is denied.
    - *.na (not applicable) message is returned.

Message Handler (MH) and Message Transport Layer (MTL)



- MH is a part of NSI stack, and uses MTL to send/receive messages
- MH is primarily responsible for keeping track of messaging state, e.g.
  - Who was the message sent to
  - Was the message received (i.e. ack'ed or MTL timeout)
  - Who has not replied to the message (e.g. *.cf, *.fl, etc)
- MTL is primarily responsible for sending and receiving messages, and notifying MH if the message was received, or if a (MTL) timeout occurs
- MTL interface (to MH) has 2 simple operations:
  - Send: blocks until ack is returned by destination MTL, or timeout happens. Timeout value is implementation dependent. NB: The MTL may be implemented to retry sending messages, but this is opaque to the MH
  - Receive: a thread in MH is invoked when a message is received

Message ack, reply and timeouts



: MTL timeout may happen

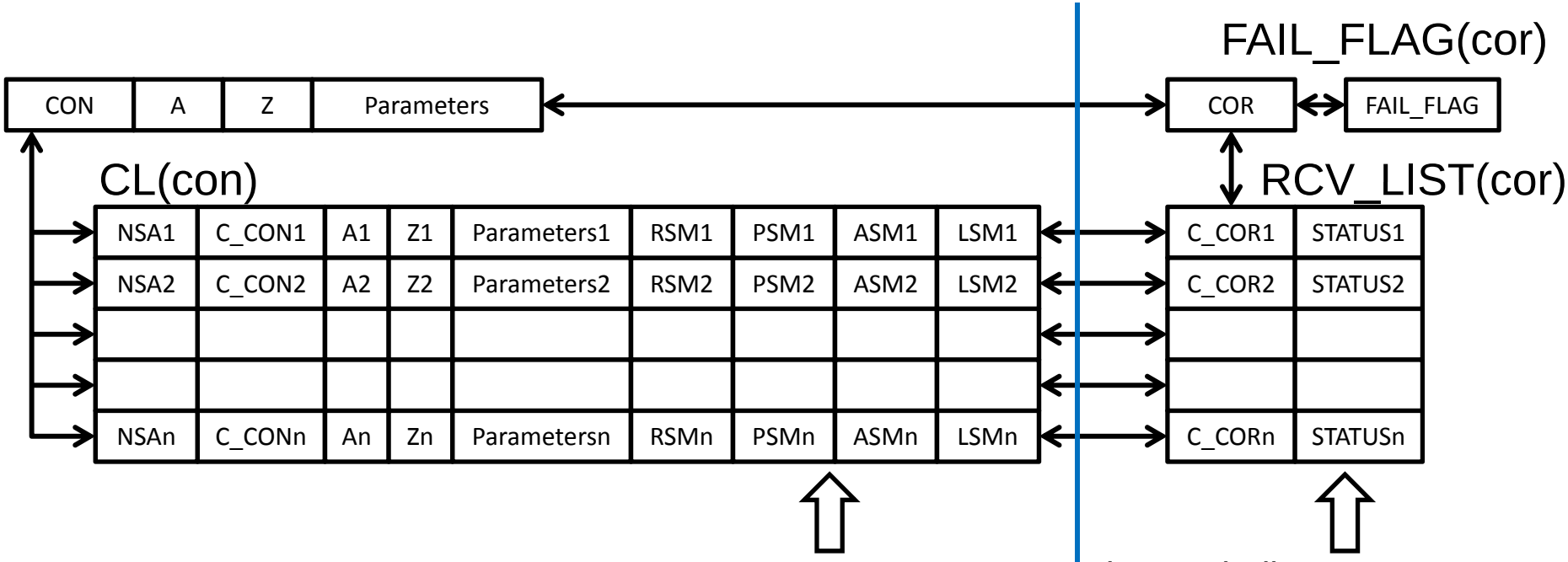
: MH timeout may happen

- Ack is sent by MTL for each message
  - If ack is not returned in a certain period of time, MTL timeout occurs
- Reply is sent by MH (via MTL) and is either confirm, fail or not_applicable
  - MH can timeout if expected reply is not received from a child

Timeouts

- Message transport layer (MTL) timeout
  - Underlying MTL (http/tcp) initiates a MTL timeout
  - Happens when an ack is not returned for a message.
- Message Handler (MH) timeout
  - MH can timeout if a reply message is not returned in a certain period of time
- MH notifies both MTL and MH timeouts to the parent RA
- When a MTL/MH timeout is notified, uRA can either retry or terminate the connection.
  - Retry is requested by NSI_messageRetry.rq, which has the original request message's id (correlation id) as a parameter
  - MH keeps not-yet-replied requests in a table, so that it can re-send the request.

Tables an aggregator MH maintains for each reservation (connection)



Generated for each reservation (connection)

Generated for each request

modifyCancel after timeout

- After modifying operations, if a NSA is already in RESERVED state, it can receive NSI_modifyCancel.rq and reply NSI_modifyCancel.cf, but the modification is not rolled back. The system may be in an inconsistent state (different versions across the system) after those operations.

Notifications: modify timeout and MTL failure

- NSI_modifyTimeout.nt
- NSI_genericEvent.nt
  - Message delivery failure will be notified by this message (to be defined)
- When a MTL/MH timeout is notified, uRA can either retry or terminate the connection.
  - Retry is requested by NSI_messageRetry.rq, which has the original request message's id (correlation id) as a parameter

Termination and LSM

- A connection lifecycle is terminated when NSI_terminate.rq is received.
- LSM (Lifecycle State Machine) handles the terminate request.
  - Terminate request will delete the RSM, PSM and ASM, but the LSM should be there to send/receive terminate request and confirm messages
  - uPA may delete RSM, PSM and ASM when it isseues fcd_end, but LSM cannot be deleted.