OGF NSI CS State Machine
The Skype Sessions v8

Sep 11, 2012

Tomohiro Kudoh, Vocals
Chin Guok, Lead Guitar
John MacAuley, Cowbell

Lest we forget

- The primary goal of this activity was to:
  1. Correct the behaviour of the Provision command where a long period of time (based on startTime) could occur between the Provision request and confirmation message.
  2. Add a basic reservation modification capability.
- We needed to stick to our primary principles:
  – As simple as possible - reduced complexity simplifies implementation and error paths.
  – Graceful handling of error and race conditions.
  – Separation of responsibility.
  – No loss of information and no hiding of information
    - No outside context required - an independent external observer should be able to view the current reservation information and understand the exact state.

The long and winding road

- In Pre-1.0 we collapsed two state machines into one for simplicity.
- We split them apart again in 2.0, removed a bunch of states for simplicity, and added the Message Delivery Layer (MDL).
- We added the states back in that we realized we needed.
- We added a modify command.
- We collapsed the two machines back together and removed some states.
- We added in new failed states.
- We created an activationState and removed activate messages from main state machine giving us separate reservation and data plane state machines.
- We added state change events, we argued about state change events versus notification messages, and we ended up removing the state change events.
- We added a unique identifier to Provision and Modify to correlate activation events.
- John wanted to rip auto-provision out of the protocol and make it an uRA issue but was beaten down.
- Tomohiro slipped off his noodle due to lack of sleep and exploded the single state machine into a separate reservation and provisioning state machines. Then there were three!

Key changes for Skype v8

- There are now three separate state machines for NSI Connection Services:
  - The Reservation State Machine models the lifecycle of a reservation, including modifications to the reservation.
  - The Provisioning State Machine models the provisioning state of the reservation's associated connection resources.
  - The Activation State Machine models the activation state of the data plane resources associated with the reservation.
- Start time has been removed from the Reservation and Provisioning state machines
  - A uPA and NRM are now responsible for management of the "(provision)" and "(activation)" behaviors locally.
  - The uPA and NRM must activate data plane during reserved period, until "(release)" or "(clean_up)" is received.
  - How to maintain timer is left to uPA and NRM implementation.
  - In this document we model the "(provision)" and "(activation)" individually.

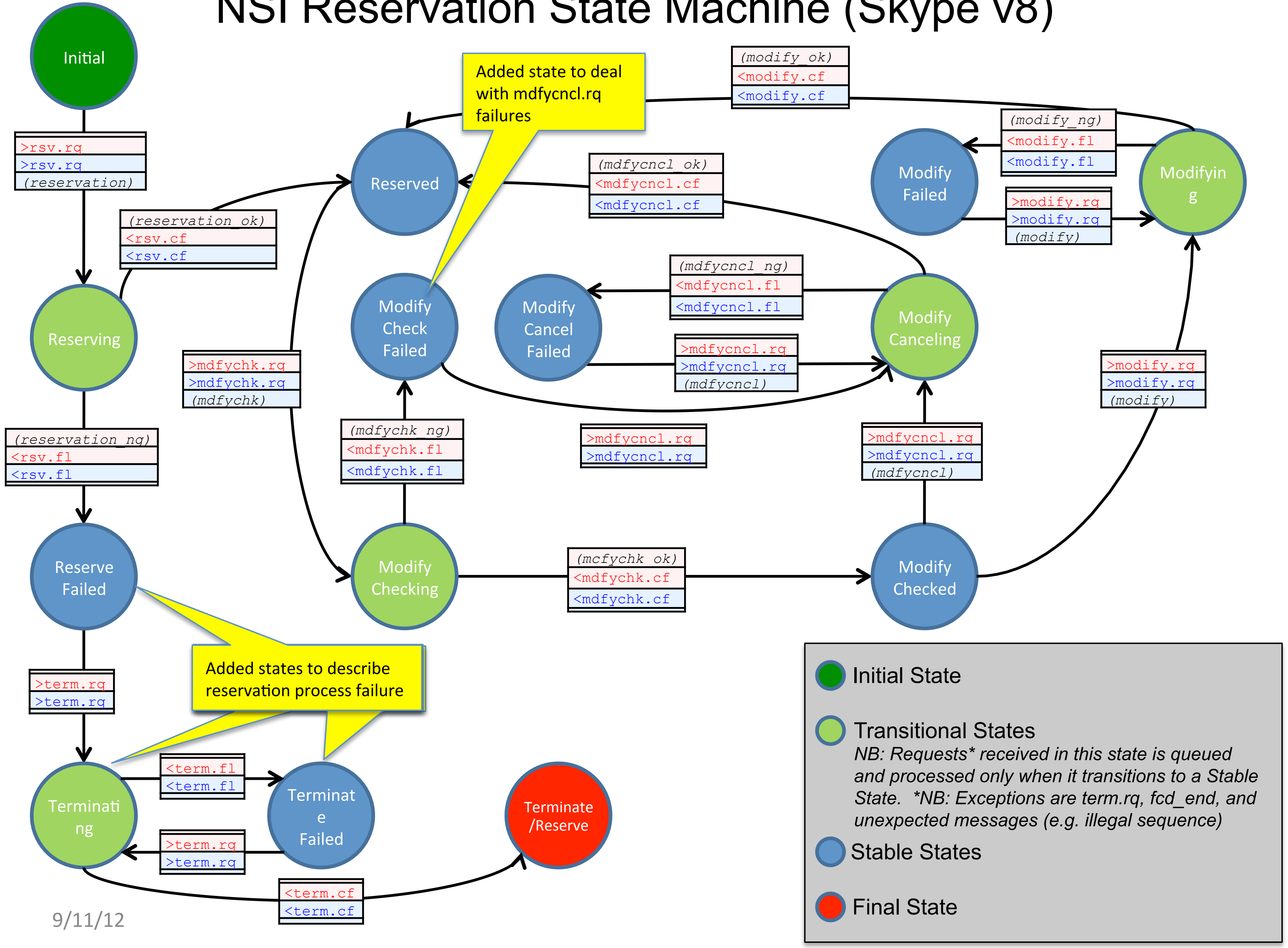
Key changes for Skype v8

- Data plane activation is not reflected in the reservation’s connectionState.
- Activation notification messages are handled separately and are modeled using the reservation’s new activationState.
- Partial failure states have been introduced
  - “Provision Failed”, “Release Failed” and “Modify Failed” to help model the inconsistent state of the connection within the control plane.

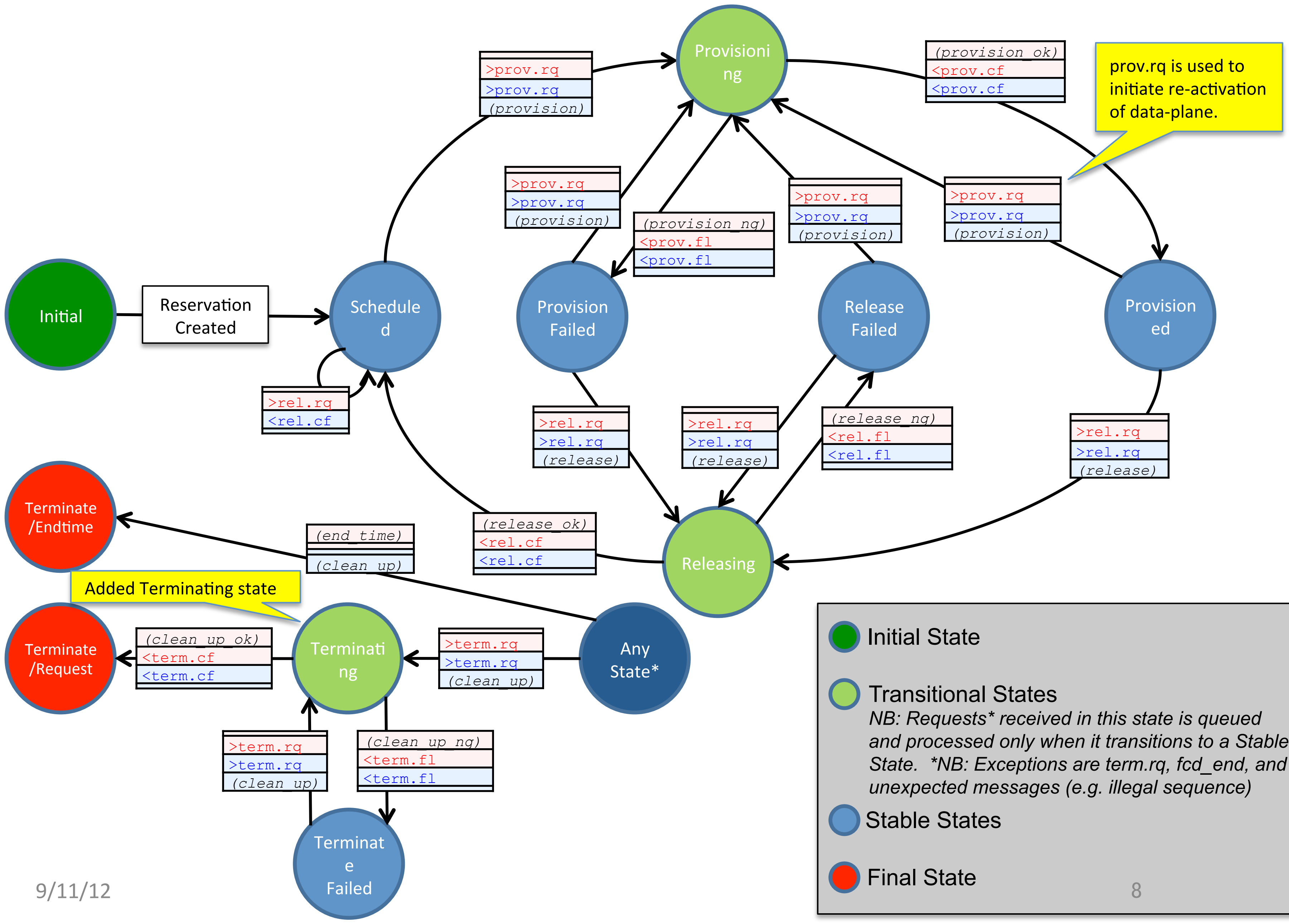
Key changes for Skype v8

- Three final terminated states are model:
  - TerminatedReserve – terminated state was reached as the result of a failed reservation request.
  - TerminatedEndTime – terminated state was reached as the result of reservation end time being reached.
  - TerminatedRequest – terminated state was reached as the result of a terminate request.

NSI Reservation State Machine (Skype v8)



NSI Control Plane State Machine – unified - (Skype v8)



● Initial State

● Transitional States
NB: Requests received in this state is queued and processed only when it transitions to a Stable State. *NB: Exceptions are term.rq, fcd_end, and unexpected messages (e.g. illegal sequence)*

● Stable States

● Final State

Modify Notes

- The modify.rq is equivalent to a provision.rq in behaviors for the newly modified reservation
  - When the `modify_ok` is returned to the uPA by the local NRM, the uPA must invoke the “uPA Activation Sequence” as described on slide 14 to active the newly modified reservation within the network.
- An NSA can choose to fail a request for modification (mdfychk.rq) if it deems the local state machine is in a state which my result in complications satisfying the request
  - If mdfychk.rq is received in Provisiond state, and data plane is not yet activated at that time, the data plane may be activated while in the modification sequence due to the occurrence of start time. An NRM which cannot support such modification should return mdfychk.fl.

activationState

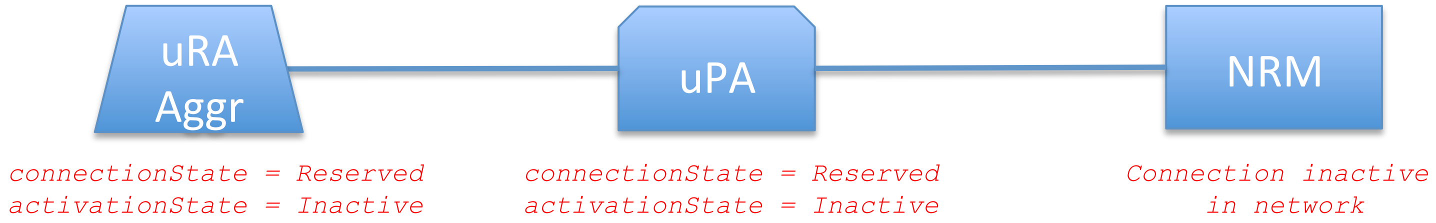
- activationState models the reservation's overall activation state within the network (i.e. data plane).
  - activationState = { Active, Inactive } with Inactive the default
- activationState is controlled through notifications out side of normal state machine interactions
  - activate_ok.nt is sent when an NSA has a connection that has transitioned to Active state (all child NSA or NRM segments included).
  - activation_fl.nt is generated by a uPA when the activation operation fails on the local NRM. This notification is propagated by NSA up the tree. There is no impact to the activationState since it should already be in an "Inactive" state.
  - deactivate_ok.nt is sent when an NSA has one or more components of a connection transition to Inactive state (a child NSA or NRM).
  - deactivate_fl.nt is generated by a uPA when the deactivate operation fails on the local NRM. This notification is propagated by NSA up the tree.
- uPA will generate "activate" or "deactivate" messages based on the reservation's connection activation state within the network.
- An aggregator will only propagates "activate" or "deactivate" events when there is a change in activationState on the connection.

activationState continued…

- For the uPA, activationState will only transition to “Active” when:
  - Current time is between start_time and end_time;
  - connectionState is “Provisoned”;
  - The local NRM has successfully activated the connection into the data plane;
  - For all other it is “Inactive”.
- For the uPA, activationState can be “Active” when:
  - Current time is between start_time and end_time;
  - connectionState is “Provisoned”;
- For the uRA or Aggregator, activationState will only transition to “Active” when:
  - All children NSA have reported an activationState of “Active”.
  - If any child connection segment becomes inactive, then the overall state transitions to “Inactive”.

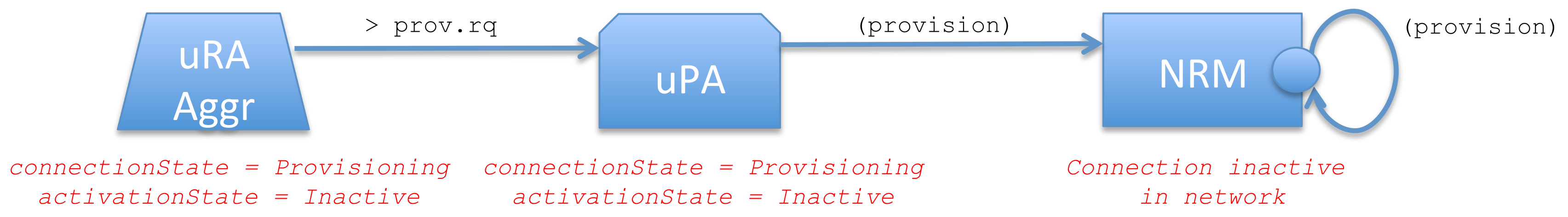
Reserved State

- Initial connection state within the network for a reservation:
    - connectionState is “Reserved”;
    - activationState is “Inactive”.

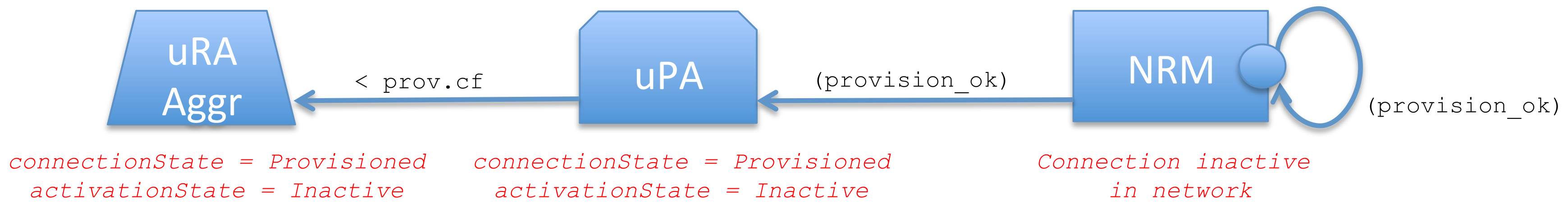

uRA Aggr
uPA
NRM
connectionState = Reserved
activationState = Inactive
connectionState = Reserved
activationState = Inactive
Connection inactive in network

uPA Provision Sequence

- Provision command issued down the request tree transitions connectionState to “Provisioning”.


uRA Aggr
> prov.rq
uPA
(provision)
NRM
(provision)
connectionState = Provisioning
activationState = Inactive
connectionState = Provisioning
activationState = Inactive
Connection inactive in network

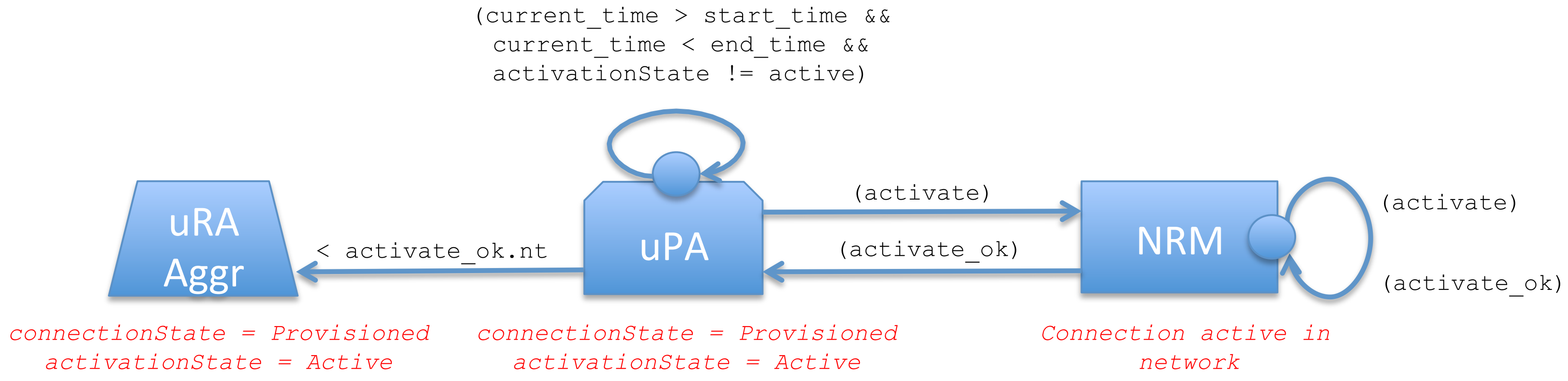
- Provision confirms returned up the tree transitioning connectionState to “Provisioned”, but activationState remains “Inactive” for now.


uRA Aggr
< prov.cf
uPA
(provision_ok)
NRM
(provision_ok)
connectionState = Provisioned
activationState = Inactive
connectionState = Provisioned
activationState = Inactive
Connection inactive in network

- Once the Provision operation is completed, if it is past reservation start_time then the uPA will activate the connection within the NRM. See next slide for this sequence.

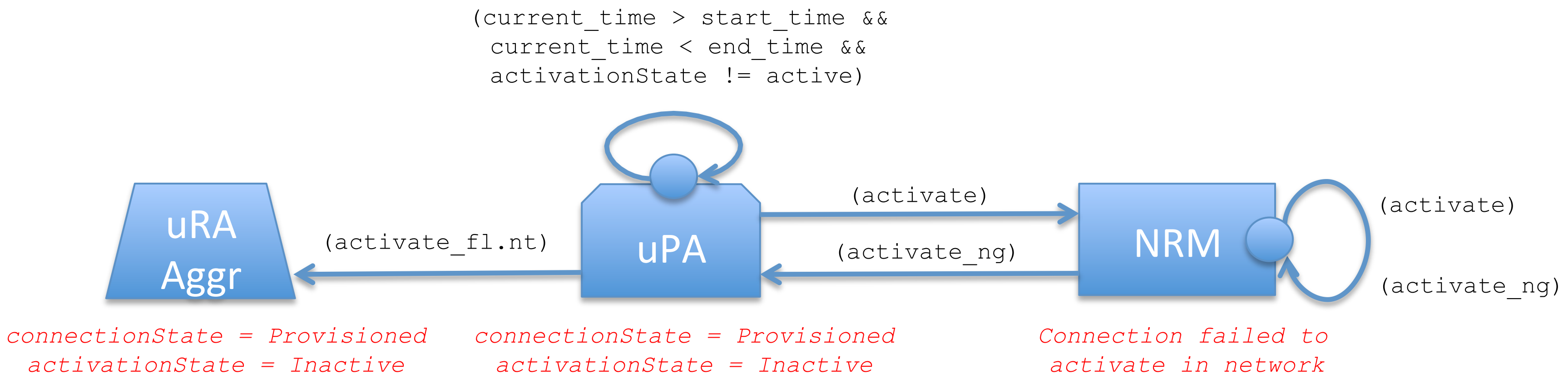
uPA Activation Sequence

- If connectionState is "Provisioned", and it is past reservation start_time but not end_time, the uPA will activate the connection within the NRM.
- Once activated the uPA transitions activationState to "Active" and generates an activate_ok.nt notification.


(current_time > start_time &&
current_time < end_time &&
activationState != active)
uRA Aggr
< activate_ok.nt
uPA
(activate)
(activate_ok)
NRM
(activate)
(activate_ok)
connectionState = Provisioned
activationState = Active
connectionState = Provisioned
activationState = Active
Connection active in network

uPA Activation Failed Sequence

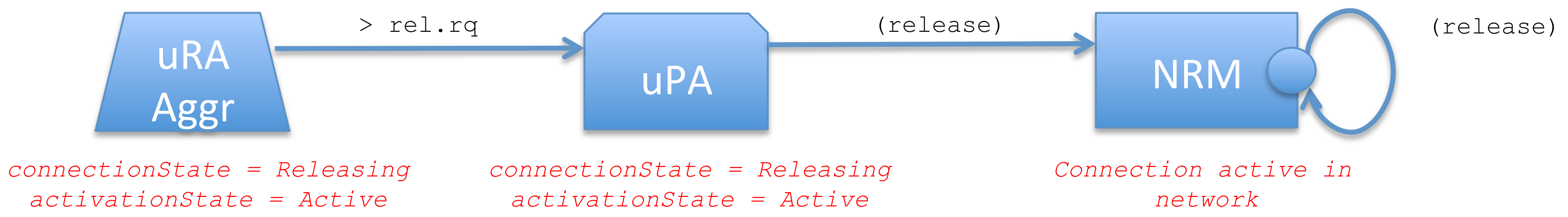
- If connectionState is "Provisioned", and it is past reservation start_time but not end_time, the uPA will activate the connection within the NRM.
- If activation fails with the NRM there is no need to transition to "Inactive" since the activationState is already "Inactive".
- The NRM will generate an activation failure notification (activate_fl.nt) to inform parent NSA of the condition.



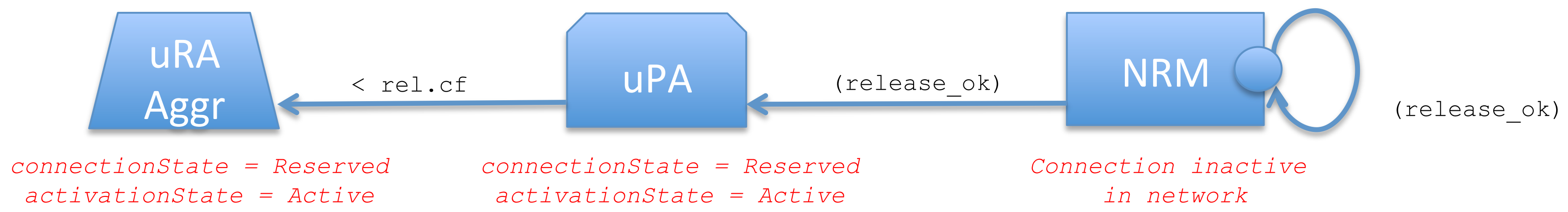
(current_time > start_time && current_time < end_time && activationState != active)
uRA Aggr
uPA
NRM
(activate)
(activate_ng)
(activate_fl.nt)
(activate)
(activate_ng)
connectionState = Provisioned activationState = Inactive
connectionState = Provisioned activationState = Inactive
Connection failed to activate in network

uPA Release Sequence

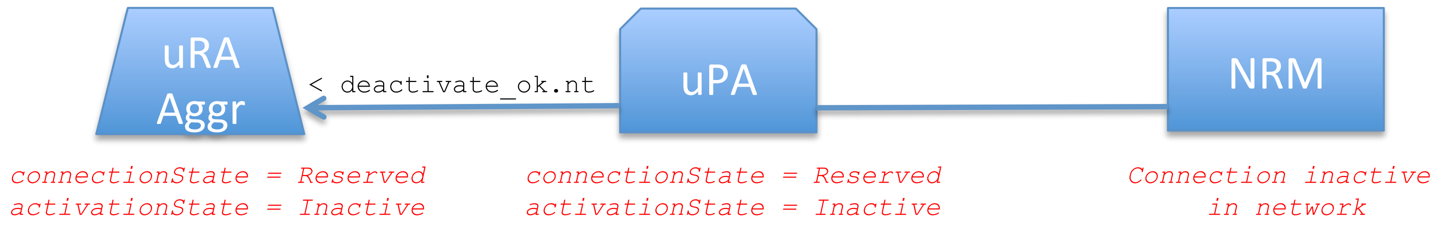
- Release command issued down the request tree transitions connectionState to “Releasing”.


uRA Aggr
> rel.rq
uPA
(release)
NRM
(release)
connectionState = Releasing activationState = Active
connectionState = Releasing activationState = Active
Connection active in network

- Release confirms returned up the tree transition connectionState to “Reserved”, but activationState remains “Active” for now.

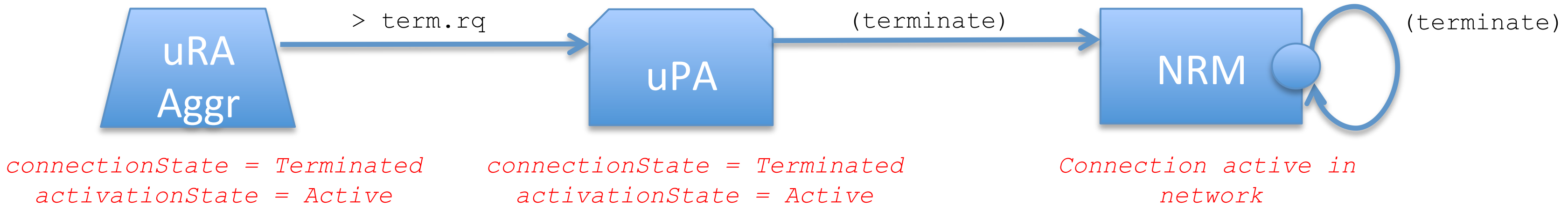

uRA Aggr
< rel.cf
uPA
(release_ok)
NRM
(release_ok)
connectionState = Reserved activationState = Active
connectionState = Reserved activationState = Active
Connection inactive in network

- The uPA immediately transitions activationState to “Inactive” and sends a deactivate_ok.nt notification to the parent NSA.

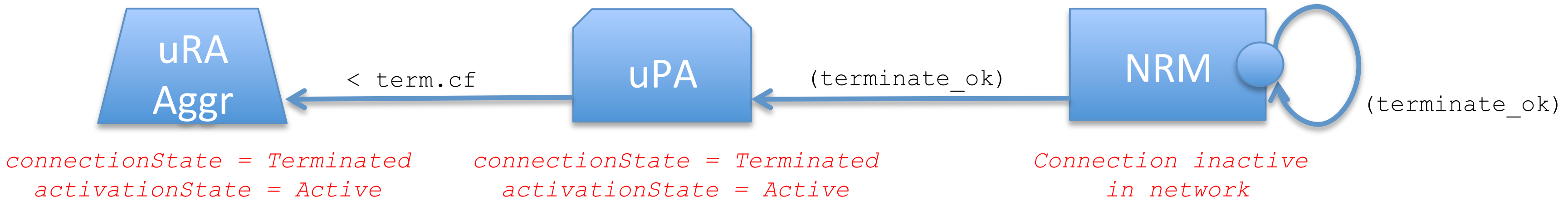

uRA Aggr
< deactivate_ok.nt
uPA
NRM
connectionState = Reserved activationState = Inactive
connectionState = Reserved activationState = Inactive
Connection inactive in network

uPA Terminate Sequence

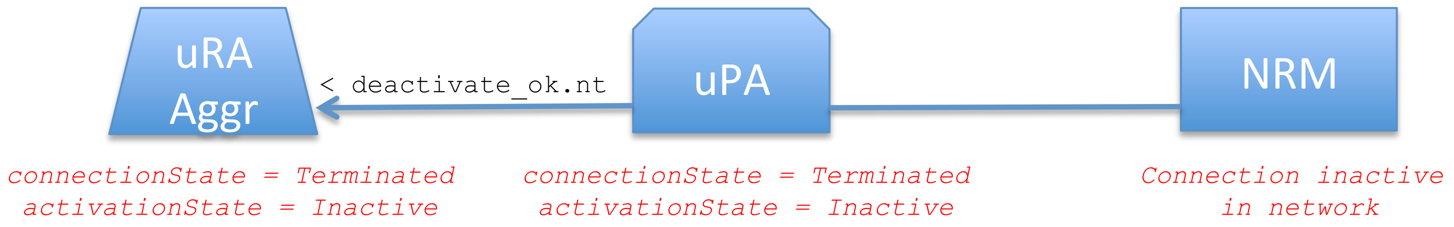
- Terminate command issued down the request tree transitions connectionState to “Terminated”.



- Terminate confirms returned up the tree does not transition state because we have already forced the connectionState to Terminated.



- The uPA immediately transitions activationState to “Inactive” and sends deactivate_ok.nt notification to the parent NSA.

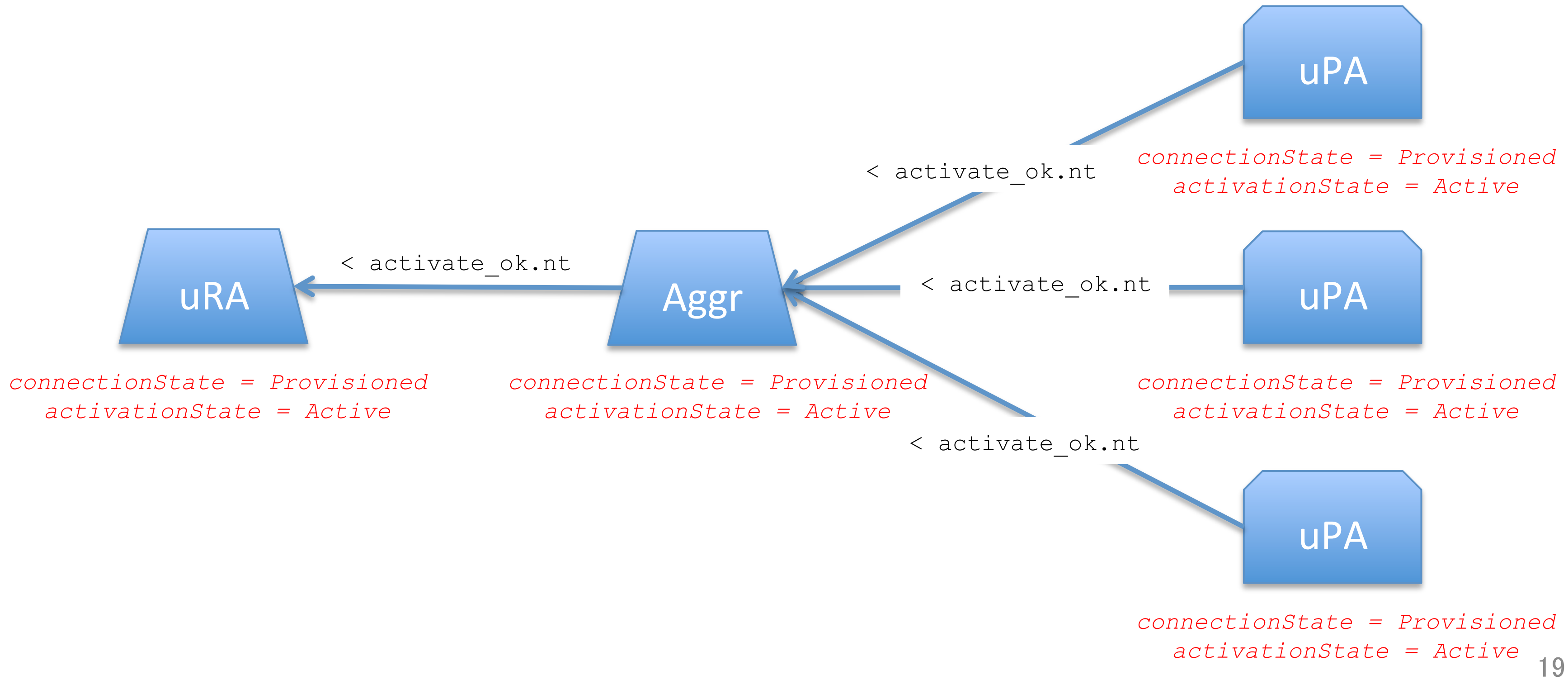

uRA Aggr
< deactivate_ok.nt
uPA
NRM
connectionState = Terminated
activationState = Inactive
connectionState = Terminated
activationState = Inactive
Connection inactive in network

Activation Failure Recovery

- There are two thoughts on this topic:
  1. Push recovery as low in the tree as possible to localize messaging, and to the place (uPA/NRM) that would best know how to recover the localized failure.
  2. Do nothing lower in the tree and let the uRA recover based on the application's needs.
- #1 may eventually need to fail, and therefore, #2 will be the fall back
  – How long would the local uPA/NRM retry before finally failing and generating the activation_fl.nt?
- #2 simplifies processing lower down in the tree but means more messaging to recover from the failure
  – The activation_fl.nt is immediately send up the tree by the uPA.
  – The uPA/NRM localized to the failure does nothing to recover.
  – The uRA can implement specific recovery strategies as needed. For example, immediately issue a new provision request, or perhaps, initiate a call to network operations for support on the failure.

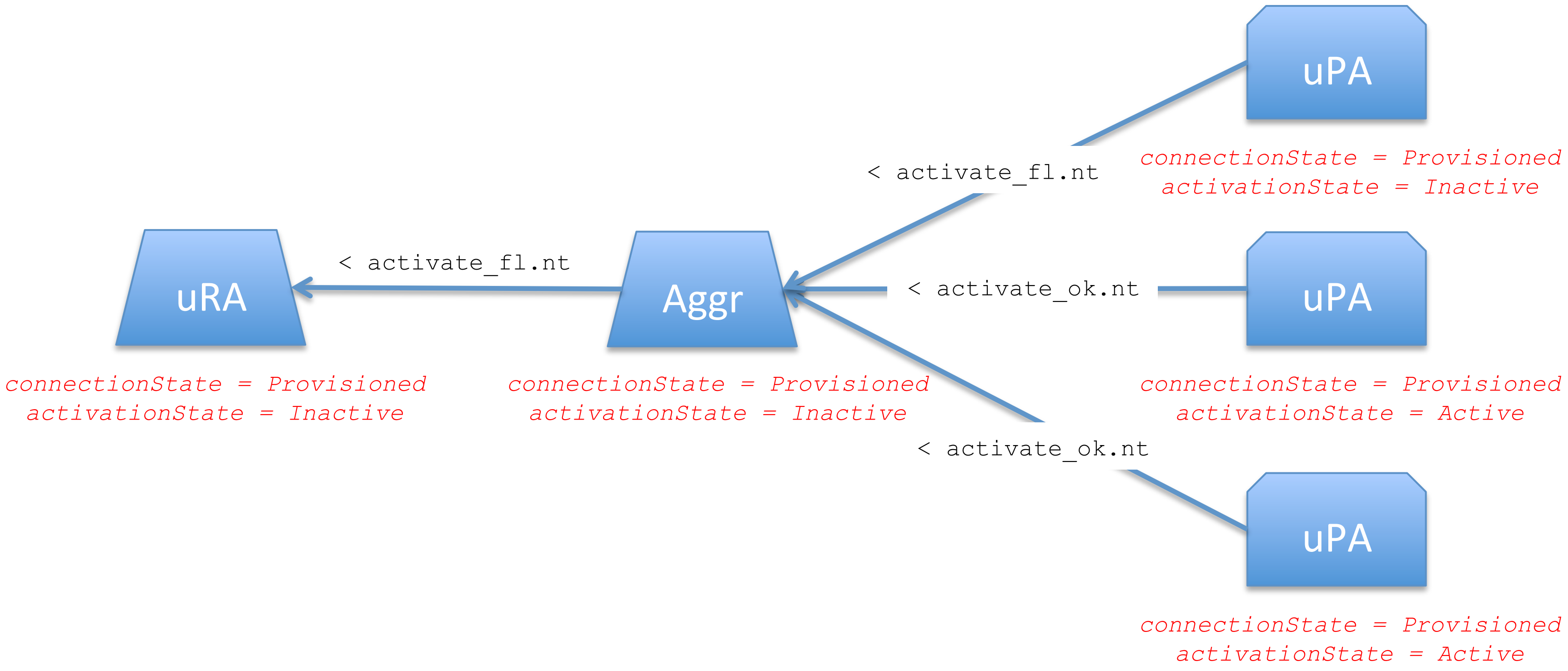
uRA/Aggregator Activation Sequence

- If connectionState is “Provisioned”, and it is past reservation start_time but not end_time, the uPA will activate the connection within the NRM, and generate an activate_ok.nt notification.
- The Aggregator NSA must receive an activate_ok.nt notification from all children before transitioning activationState to “Active” and generating an activate_ok.nt notification to the parent NSA.



uRA/Aggregator Failed Activation Sequence

- If the Aggregator NSA receives an "Activation Failed" notification (activate_fl.nt) from a child NSA, then it leaves the activationState "Inactive" and propagates an activate_fl.nt to the parent NSA.
- The Aggregator NSA takes no corrective action due to the failed activation.

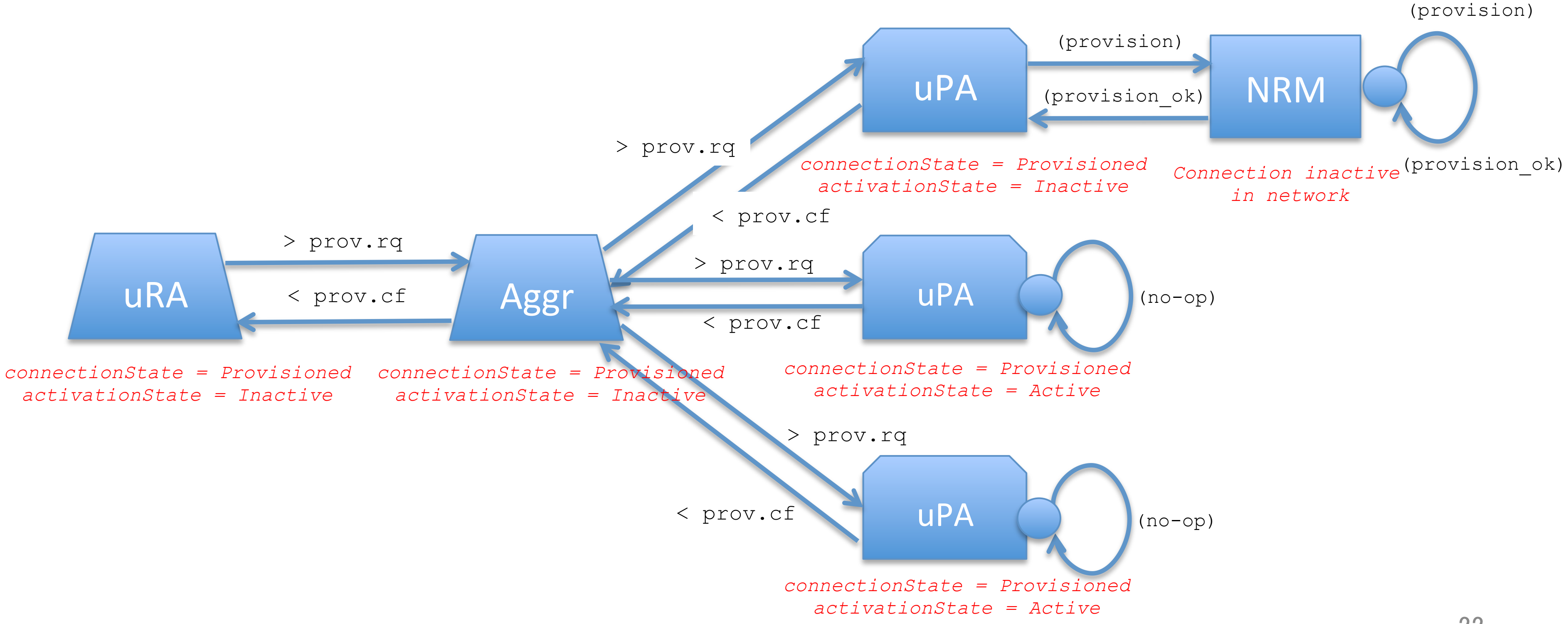

uPA
< activate_fl.nt
connectionState = Provisioned
activationState = Inactive
uRA
< activate_fl.nt
Aggr
< activate_ok.nt
uPA
connectionState = Provisioned
activationState = Inactive
connectionState = Provisioned
activationState = Inactive
connectionState = Provisioned
activationState = Active
< activate_ok.nt
uPA
connectionState = Provisioned
activationState = Active

Overloading the Provision command

- We have removed activation from the primary state machine, however, it is still used in combination with startTime to kick start the activation sequence.
- There are situations were activation of a reservation may fail, whether during initial activation, or at a later time in the schedule lifecycle.
- To handle these situations we permit the Provision command to be re-issued even when the state machine is already in the Provisioned state.
- This will allow a uRA to kick start a uPA to attempt another activation on an already provisioned reservation that is currently "Inactive".

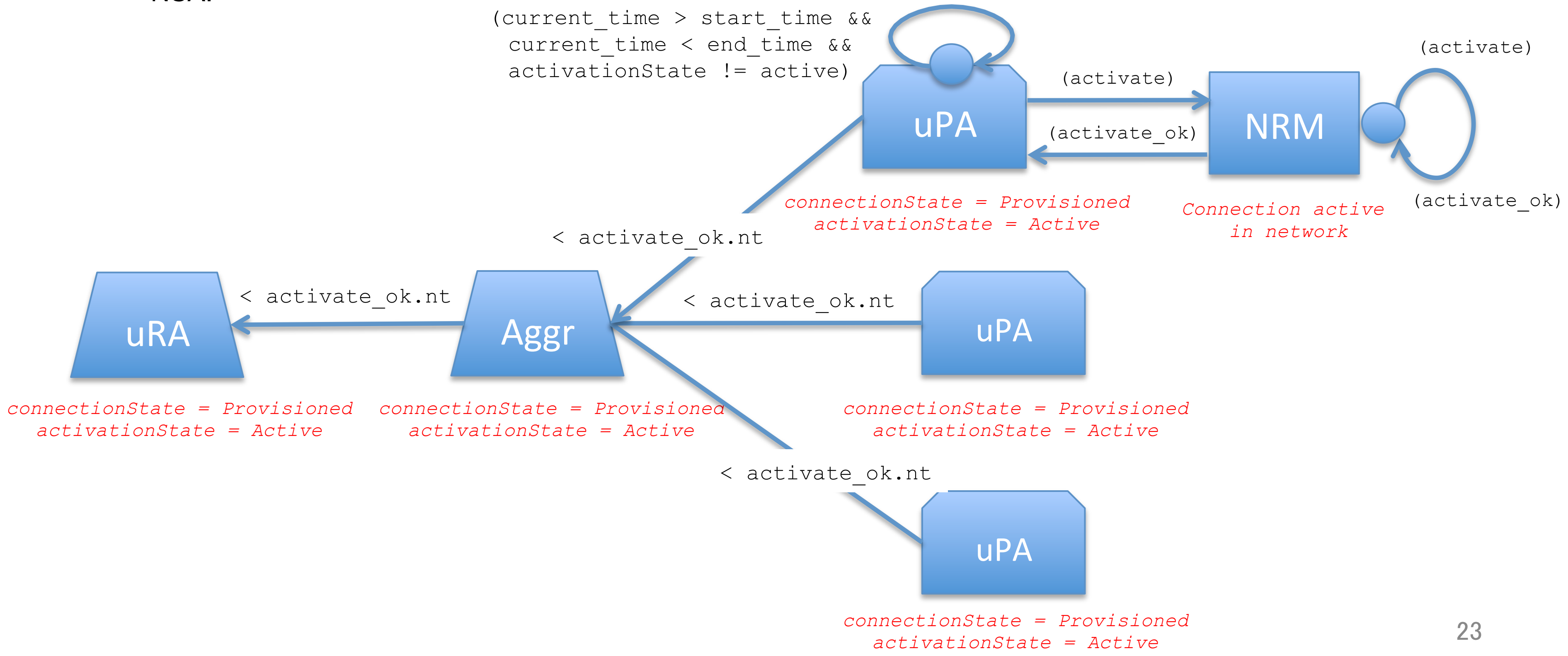
uRA Activation Failure Recovery Sequence

- To recover from a failed activation, the uRA can issue another Provision request (prov.rq) on the reservation.
- The Aggregator NSA need only propagate the Provision Request to all children NSA associated with the reservation.
- The uPA is responsible for message handling to the local NRM. If the reservation is already provisioned on the local NRM, then the uPA could skip provision and attempt to activate (next slide).
- Once the Aggregator NSA has received Provision confirmation (prov.cf) responses from all messaged children NSA, it must send a prov.cf to the parent NSA of the request.


uRA
> prov.rq
< prov.cf
Aggr
connectionState = Provisioned
activationState = Inactive
connectionState = Provisioned
activationState = Inactive
> prov.rq
< prov.cf
uPA
(provision)
(provision_ok)
NRM
(provision)
(provision_ok)
connectionState = Provisioned
activationState = Inactive
Connection inactive in network
> prov.rq
< prov.cf
uPA
(no-op)
connectionState = Provisioned
activationState = Active
> prov.rq
< prov.cf
uPA
(no-op)
connectionState = Provisioned
activationState = Active

uRA Activation Failure Recovery Sequence

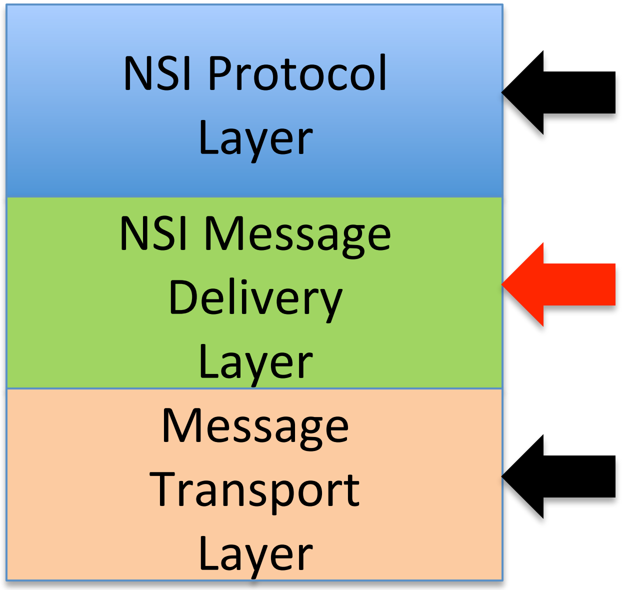
- The previous Provision request kicks the local uPA to start the NRM activation phase.
- In a successful activation of the NRM, the uPA transitions activationState to “Active” and generates an activate_ok.nt notification to the parent NSA.
- uPA that were previously “Active” and received the prov.rq must determine if this request is for a new reservation version and perform a activation, or a re-activation on an already active reservation an reply with the activate_ok.nt notification to the parent NSA immediately.
- The Aggregator NSA receives the activate_ok.nt and, now that it has received “Active” notifications from all children NSA, transitions activationState to “Active”, generating an activate_ok.nt notification to the parent NSA.



Reservation Version Identifier

- Performing a modification while simultaneously having a service activate can lead to confusion correlating the activation events to a specific version of the reservation.
- Introducing a sequentially increasing version Id into the reservation, and returning this Id in the activation notification, will allow proper processing of the activation sequence.
- If a provision activation event kicks off while a service modification is occurring, then the version Id for the provision activation would have an Id sequentially lower than the activation event resulting from the modify.
- Querying a reservation will return:
  - The reservation's version and the reservationState;
  - The provisioningState;
  - The activationState and corresponding reservation version.

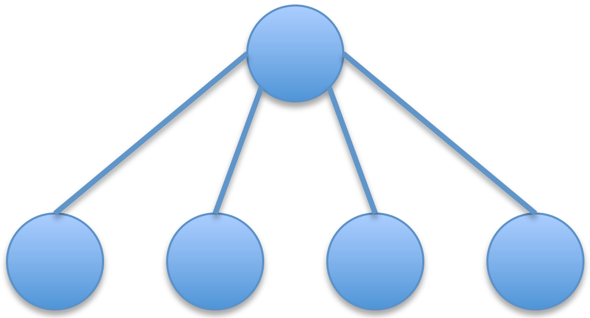
NSI message delivery layer (MDL)



State Machine works here

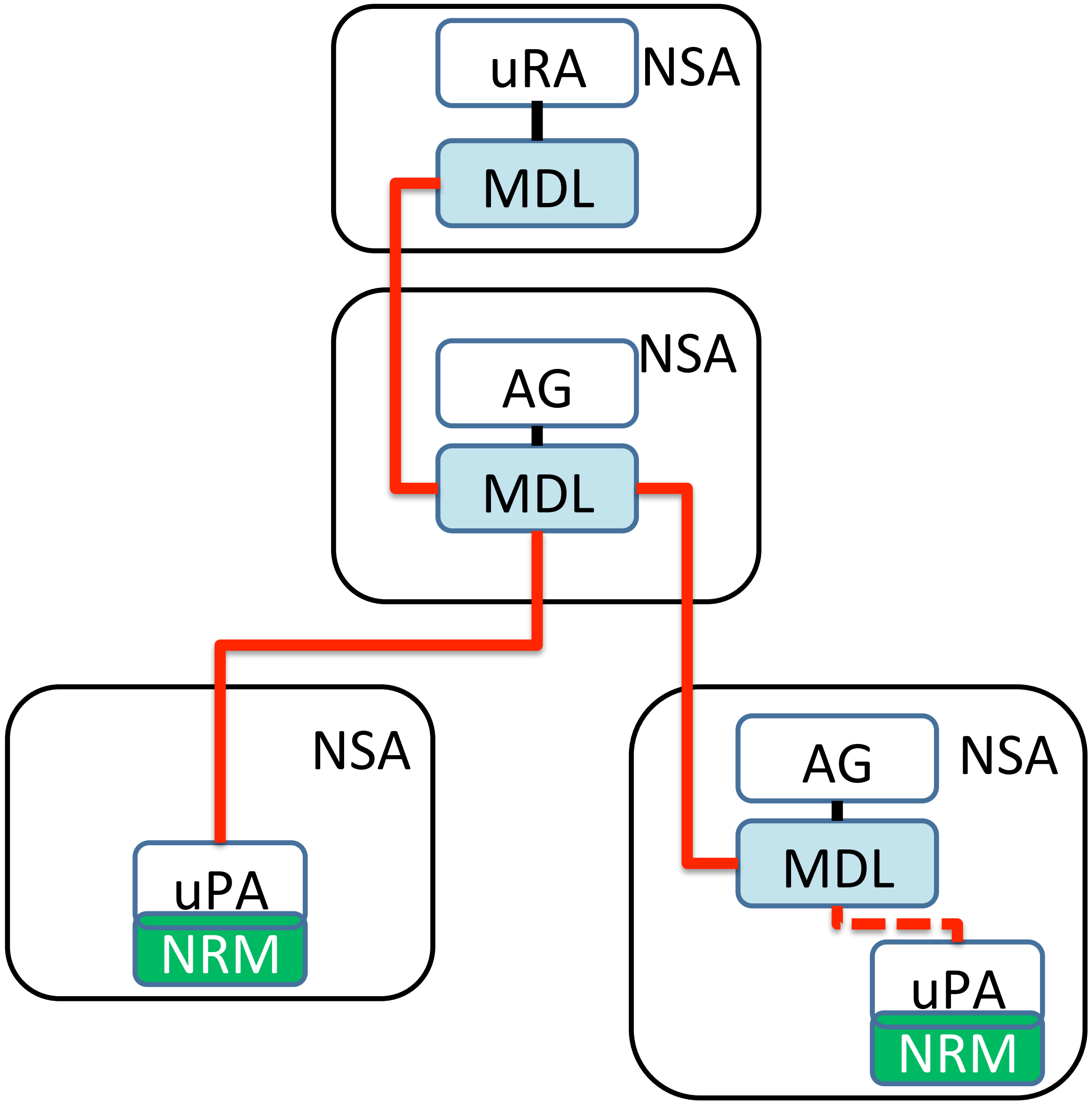
New layer which confirms delivery of message to all immediate children including uPA in the same NSA

Peer-to-peer message delivery



- MDL does
  - Aggregation of replies from children
    - all-ok/one-or-more-failed
  - Timeout/Re-try (as hard as possible)
- If MDL returns “fail”, NSA can retry by sending a request again

State machines and MDL, NRM



Terminology - Messages

Abbreviation	Description
rq (request)	The RA sends the request to the PA, for example reserveRequest.
cf (confirmed)	A PA sends this positive operation response message (such as reserveConfirmed) to the Requester NSA that issued the original request message (reserveRequest).
fl (failed)	A Provider NSA sends this negative operation response message (such as reserveFailed) to the Requester NSA that issued the original request message (reserveRequest).
nt (notification)	A Provider NSA can send an unsolicited messages to the RA (or notification) to communicate to the RA a local event in the PA that resulted in an autonomous state transition in the state machine. An example of this is the "activate_ok.nt" and "activate_ng.nt" notify messages sent from the PA to the RA to indicate a success or failure of the circuit setup in the PA.

Terminology – Reservation State Machine Operations

Abbreviation	Description
rsv (reserve)	The RA requests the PA to reserve network resources for a connection between two STP's constrained by certain service parameters.
term (terminate)	The RA request for the PA to release the provisioned resources and terminate the reservation.
mdfychk (modify check)	The modify check operation allows a connection reservation to be modified. If modification of current reservation is possible, the resources associated with the modification are held. The original reservation is not changed by this operation.
modify	The modify operation will change a reservation by the resources held by modify check operation. If the original reservation has been activated (i.e. modification sequence starts from "Activated" state), modification of activated resource must be done.
mdfycncl (cancel modify)	The modify cancel operation requests canceling of modification sequence. Held resources must be released. The original reservation must be preserved.
query	Mechanism for either RA or PA to query the other NSA for a set of connection service instances between the RA-PA pair. This operation can be used as a status polling mechanism.

Terminology – NRM operations/events for Reservation state machine

Abbreviation	Description
(reservation)	The local NRM must perform an internal reserve operation.
(reservation_ok)	The result of the local NRM reserve operation was successful.
(reservation_ng)	The result of the local NRM reserve operation was a failure.
(mdfychk)	The local NRM must check availability of resource requested, and if available hold them. The original reservation is not changed by this operation.
(mdfychk_ok)	The result of the local NRM mdfychk operation was successful.
(mdfychk_ng)	The result of the local NRM mdfychk operation was a failure.
(modify)	The local NRM must change its original reservation by the by the resources held by mdfychk operation. If the original reservation has been activated (i.e. modification sequence starts from “Activated” state), modification of activated resource must be done.
(modify_ok)	The modify operation was completed successfully by the NRM.
(modify_ng)	The modify operation sequence failed to complete.
(mdfycncl)	The modification sequence was cancelled and the local NRM must clean up any associated resources.
(mdfycncl_ok)	The modification cancel sequence was successfully completed by the NRM.
(mdfycncl_ng)	The modification cancel sequence failed to complete.

Terminology – Provisioning State Machine Operations

Abbreviation	Description
prov (provision)	The RA requests the PA to provision a previously committed reservation.
rel (release)	The RA request for the PA to de-provision resources without removing the reservation.
term (terminate)	The RA request for the PA to release the provisioned resources and terminate the reservation.

Terminology – NRM operations/events

Abbreviation	Description
(start_time)	The internal NRM event associated with the start time of a connection reservation.
(end_time)	The internal NRM event associated with the end time of a connection reservation.
(clean_up)	The reservation was terminated and the local NRM must clean up any associated resources.
(fatal_event)	The event says it all.

Terminology - Notifications

Abbreviation	Description
~~fcd_end (forcedEnd)~~	~~This notification is reported by the PA to the RA to notify that the PA has forced a termination of the reservation.~~
activate_ok	This notification is reported by the PA to the RA to notify that the PA has successfully activated the network resources associated with a reservation.
activate_ng	This notification is reported by the PA to the RA to notify that the PA has failed to activate the network resources associated with a reservation.