

Network Services Interface Connection Service v2.0

Tomohiro Kudoh (AIST)

(OGF NSI-WG)

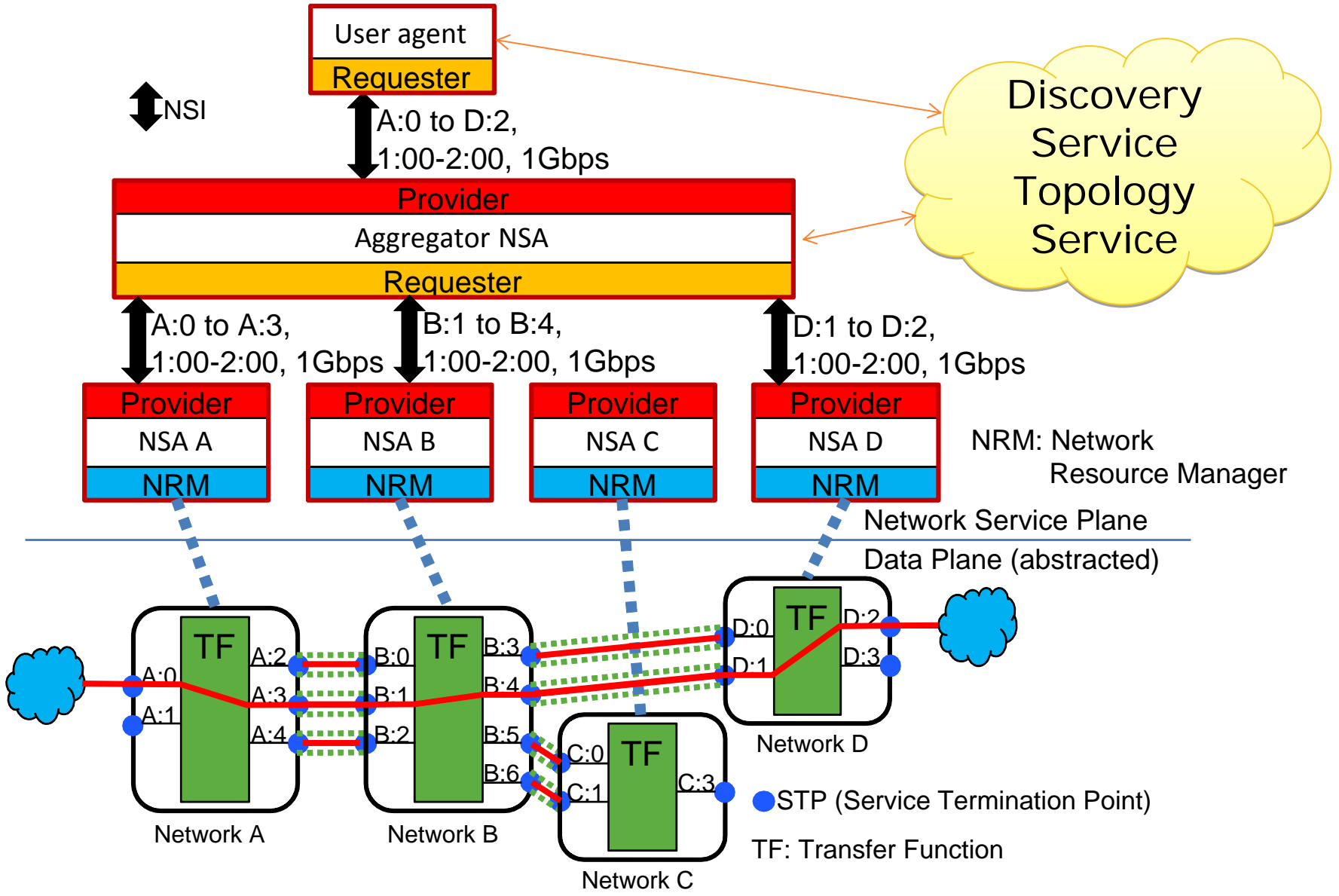
Network Services Interface (1)

- An open standard for dynamic circuit service interoperability
 - ▶ A simple API to support connectivity management
 - ▶ Dynamic assignment of bandwidth, VLAN ids etc.
 - ▶ Global reach: multi-provider enabled solution
- Being defined at OGF (Open Grid Forum) NSI-WG

Network Services Interface (2)

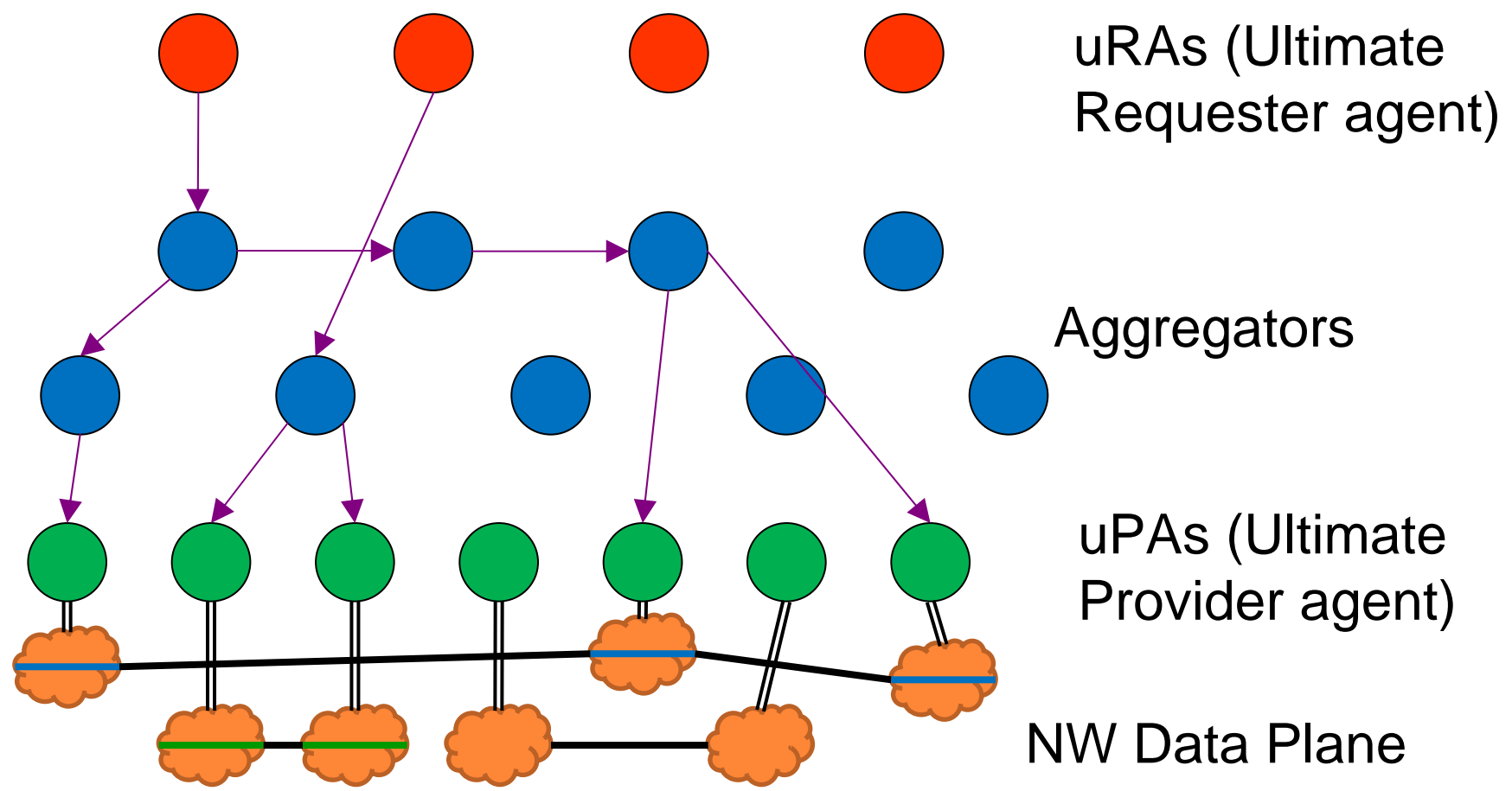
- Currently, NSI-WG is working to finalize version 2.0 of the **Connection Service** document, in which a service architecture and protocol for **end-to-end circuit provisioning interface** is described.
- Complementary services, such as a **Topology Service** and a **Discovery Service**, are being standardized to support the NSI Connection Service.
- To provide not just a connection but a dynamic network, **Switching Service** has been proposed and under discussion.
- NSI provides **abstracted view of networks**, hiding underlying physical infrastructure
 - ▶ Any control plane such as GMPLS or OpenFlow can be used under NSI

NSI architecture and connection service (CS)

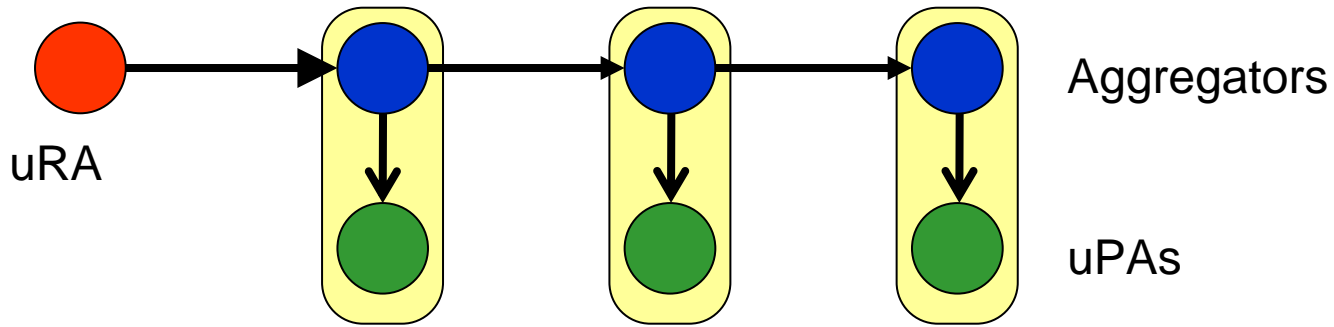


NSI tree model

There can be multiple ultimate requesters. Request propagation path is formed dynamically



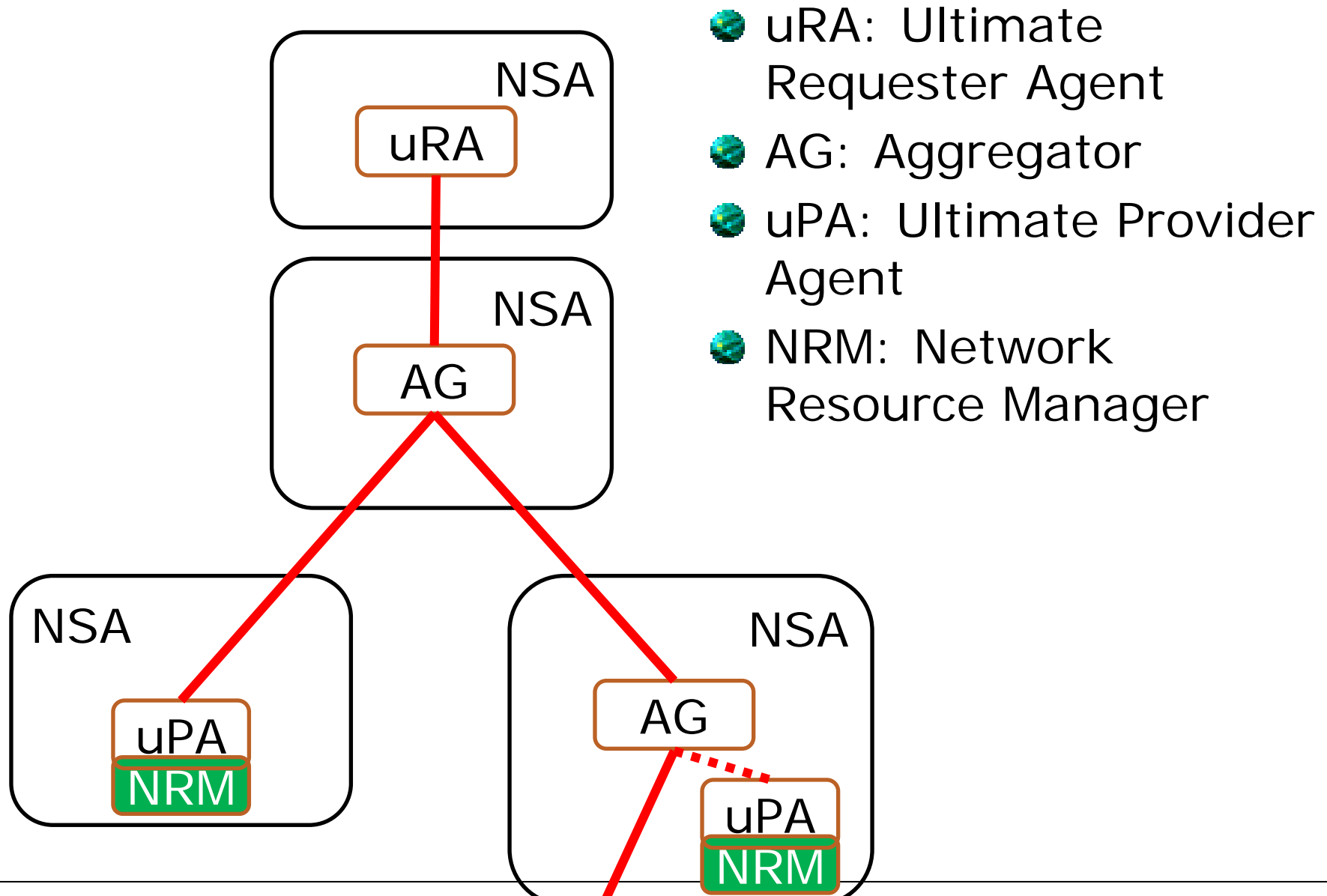
A chain on a Tree













STP and TF: topology abstraction

- Data plane topology is abstracted by STP (Service Termination Point) and TF (Transfer Function)
- STP is a logical label of a point at the edge of a network
- STPs are used in a connection request to designate termination points of intra-network connection.
- TF represents each network's capability of dynamically connecting two STPs of the network

uRA, uPA and Aggregator



Key extensions/changes of NSI CS 2.0

-  Separate state machines for reservation maintenance, provision control and terminate a connection
-  Support of modify and use of two-phase commit for reservation
-  Introduction of Coordinator
 -  Confirmation of delivery of request messages
 -  Supports aggregation of messages
 -  Notifications and error handling
 -  The majority of error states were removed from the state machine , and handled by Coordinator
 -  A new set of notifications were added to inform the uRA
-  Ability to use a query operation for polling
 -  If a requester (client) cannot receive reply messages or notifications (by message loss or NAT problem, etc.), the requester can get status by using query operations.

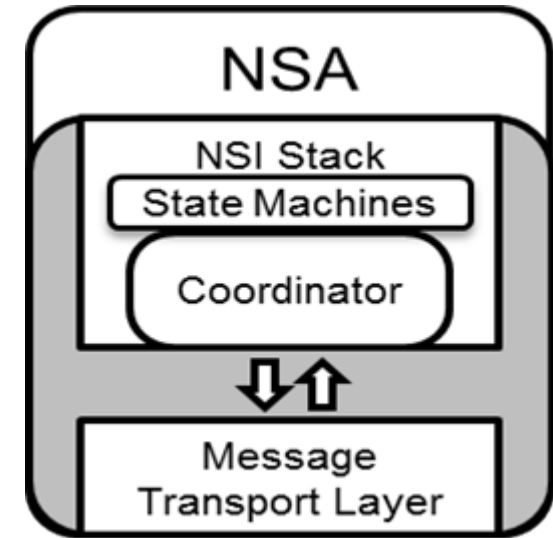
Coordinator and MTL

Coordinator's role:

- ▶ To coordinate, track, and aggregate (if necessary) message requests, replies, and notifications, and realize reliable requests/replies communication
- ▶ To process or forward notifications as necessary
- ▶ To service query requests

Coordinator maintains the following information on a per NSI request message basis:

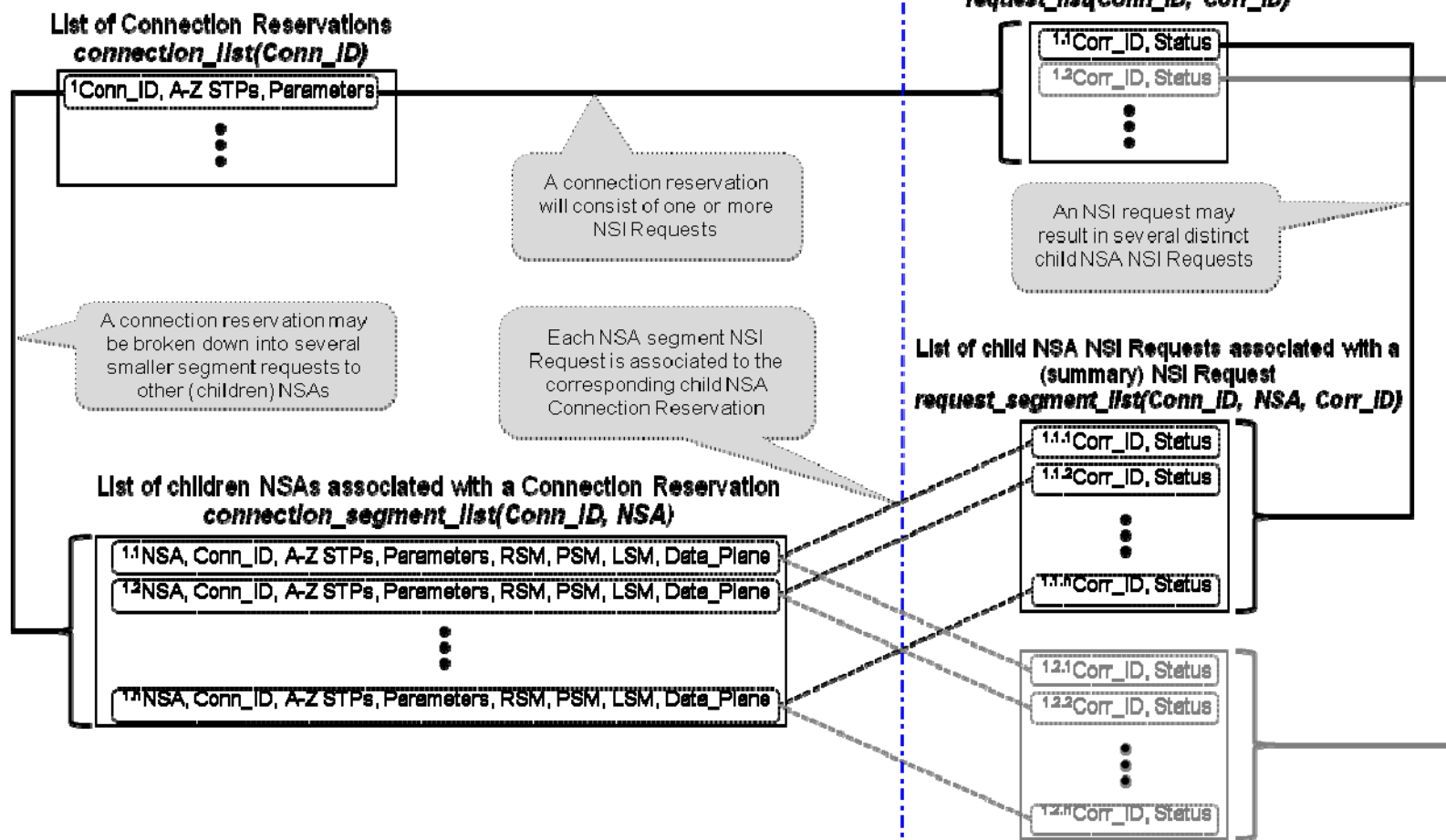
- ▶ Who the (NSI request) message was sent to
- ▶ Was the message received (i.e. ack'ed) or not (i.e. MTL timeout)
- ▶ Which NSA has sent back an NSI reply (e.g. *.confirm, *.fail, *.not_applicable) for the initial NSI request (e.g. *.request)






Information maintained by Coordinator

Information generated per Reservation (Connection)

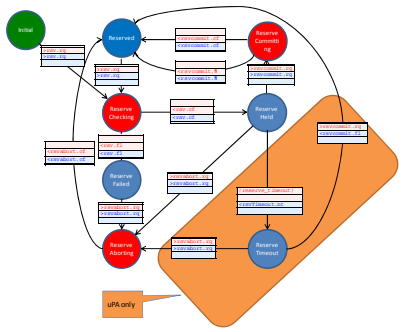
Information generated per NSI Request (Message)



Two-phase reserve/modify

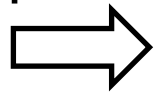
-  NSI is an **advance-reservation** based protocol
 - ▶ A reservation of a connection has properties such:
 - ⊗ A-point, Z-point (mandatory)
 - ⊗ Start-time, End-time (mandatory)
 - ⊗ Bandwidth, Labels (optional)
-  A reservation is made in **two-phase**
 - ▶ First phase: availability is checked, if available resources are held
 - ▶ Second phase: the requester either commit or abort a held reservation
 - ▶ **Two-phase is convenient when a requester requests resources from multiple providers, including other resources such as computers and storages**
 - ▶ Timeout: If a requester does not commit a held reservation for a certain period of time, a provider can timeout
-  **Modification** of a reservation is supported.
 - ▶ Currently, modification of start_time, end_time and bandwidth are supported

Reservation, Provisioning and Activation

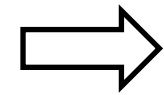


Reservation State Machine

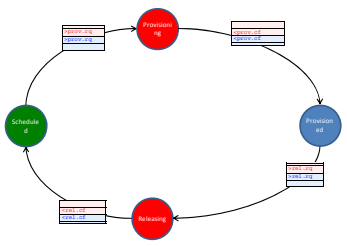
update



Committed Reservation

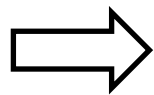


Data Plane is activated according to the latest committed reservation, when PSM is in "Provisioned" state AND during a reservation period

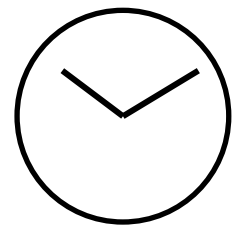
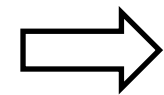


Provision State Machine

transition

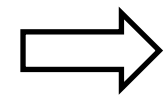


Provisioned / Scheduled



timer

$start_time < current_time < end_time$

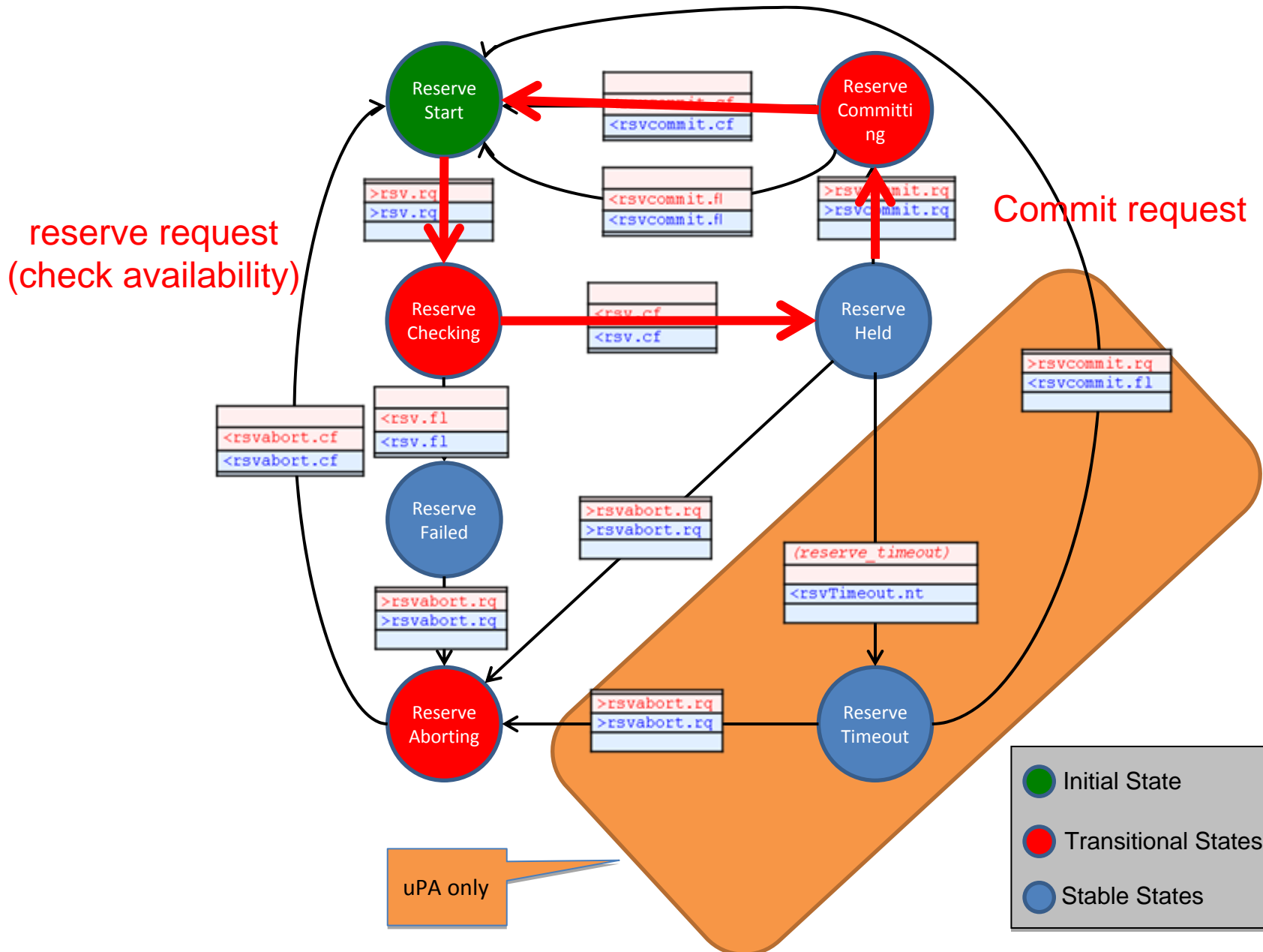


Reservation State Machine

- Reservation State Machine maintains reservation/modification message sequences
- Reservation data base is updated when a reservation/modification is committed

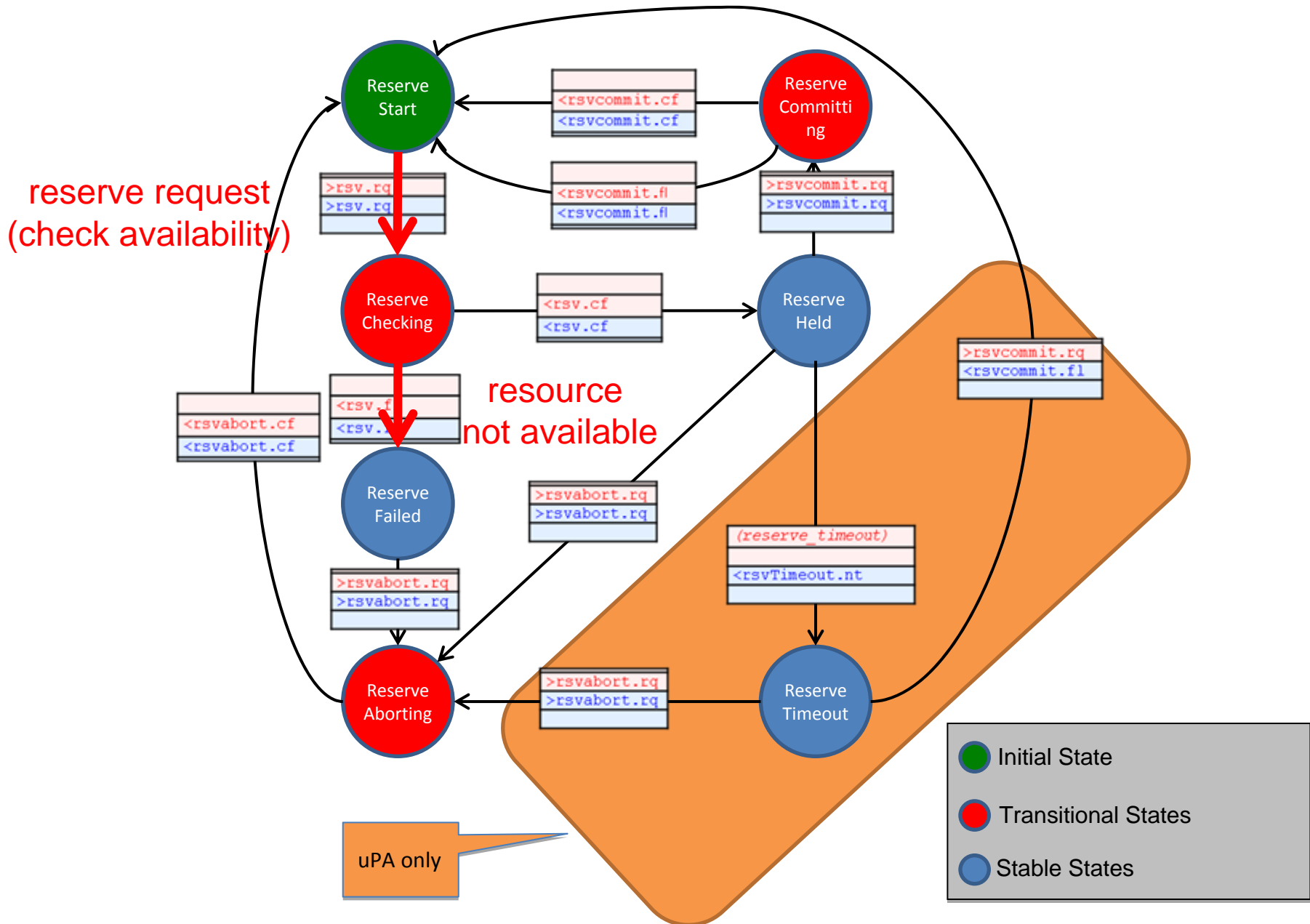
RSM: Reservation State Machine

reservation success case

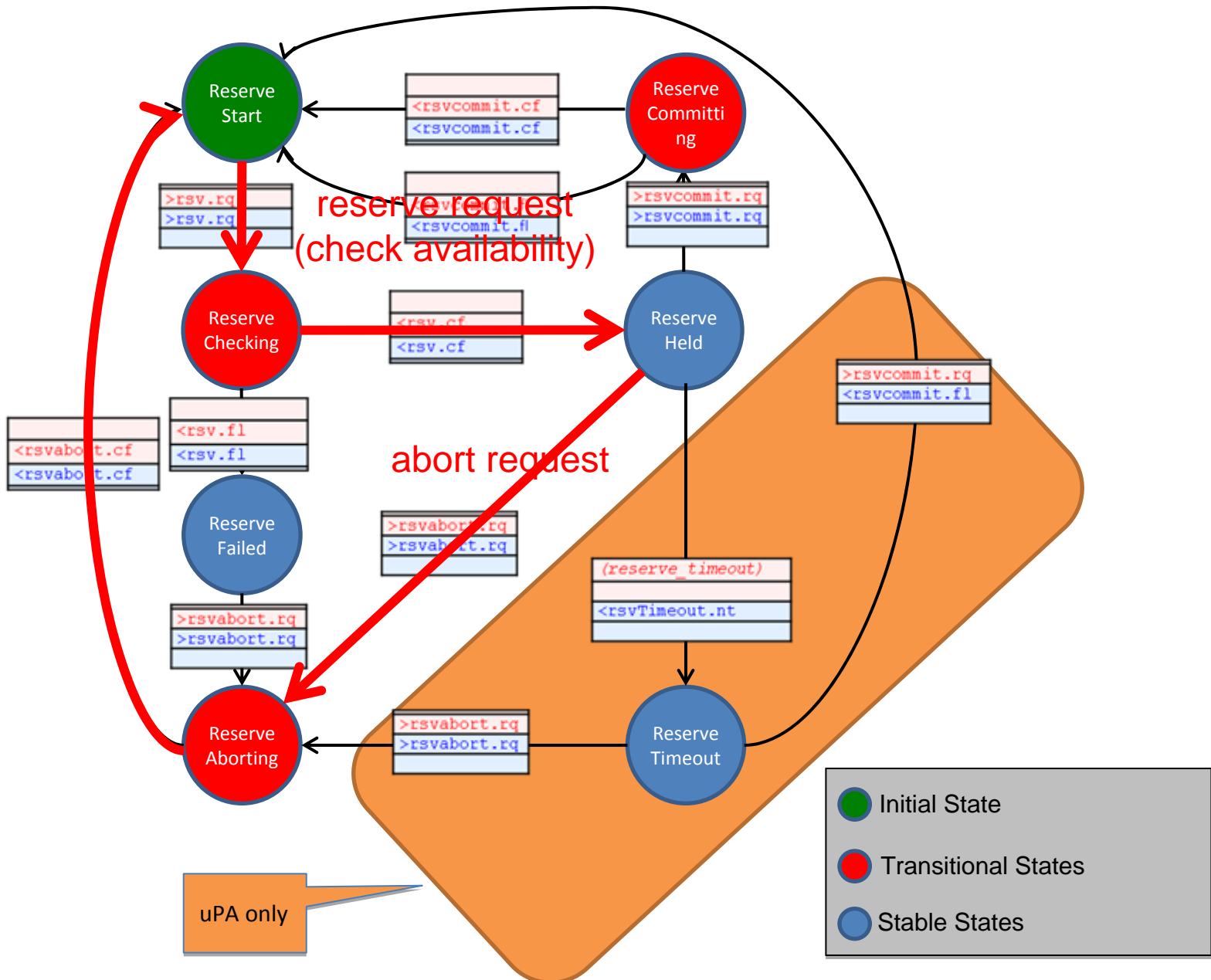


RSM: Reservation State Machine

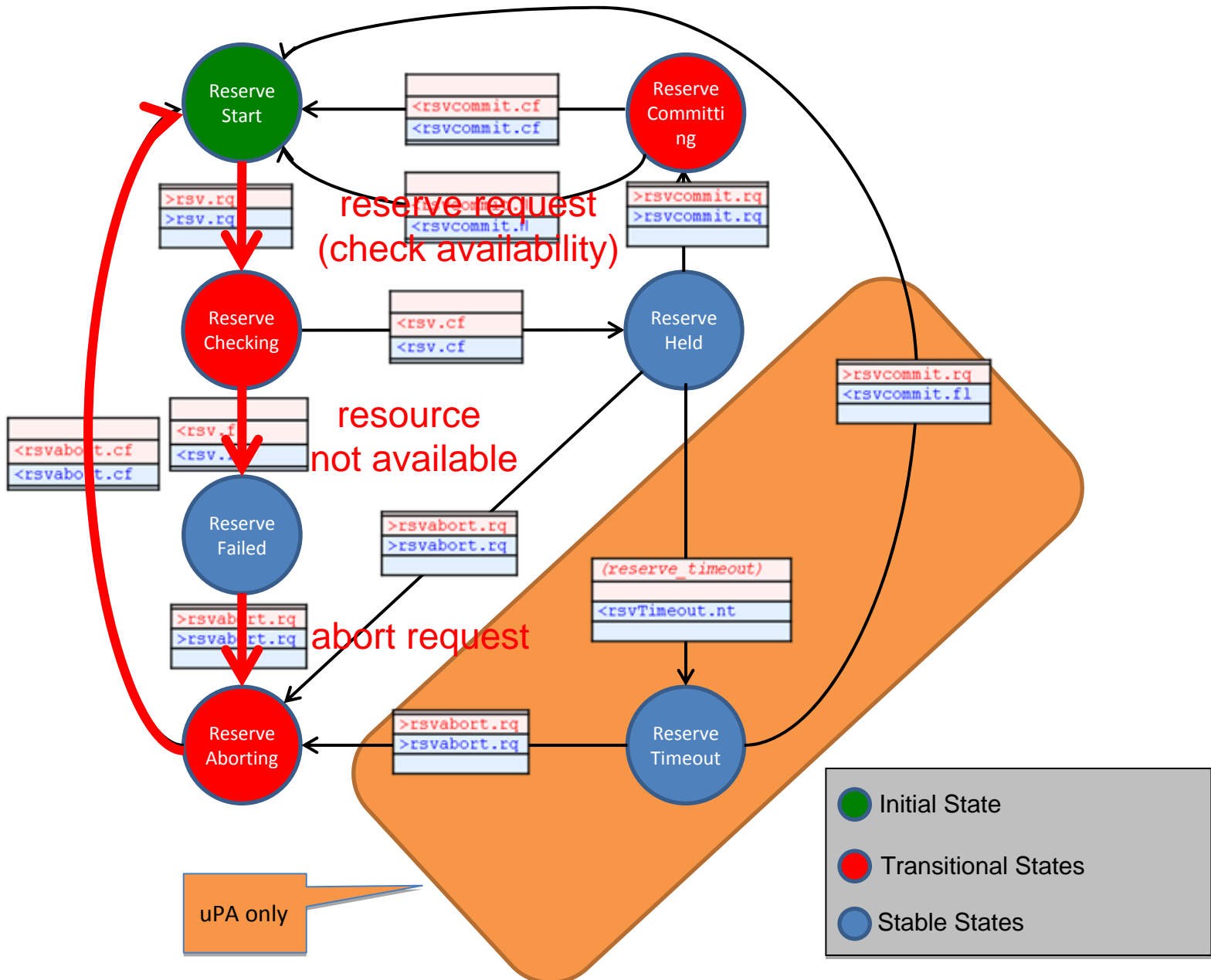
resource not available and failed



RSM: Reservation State Machine aborted after resource was held

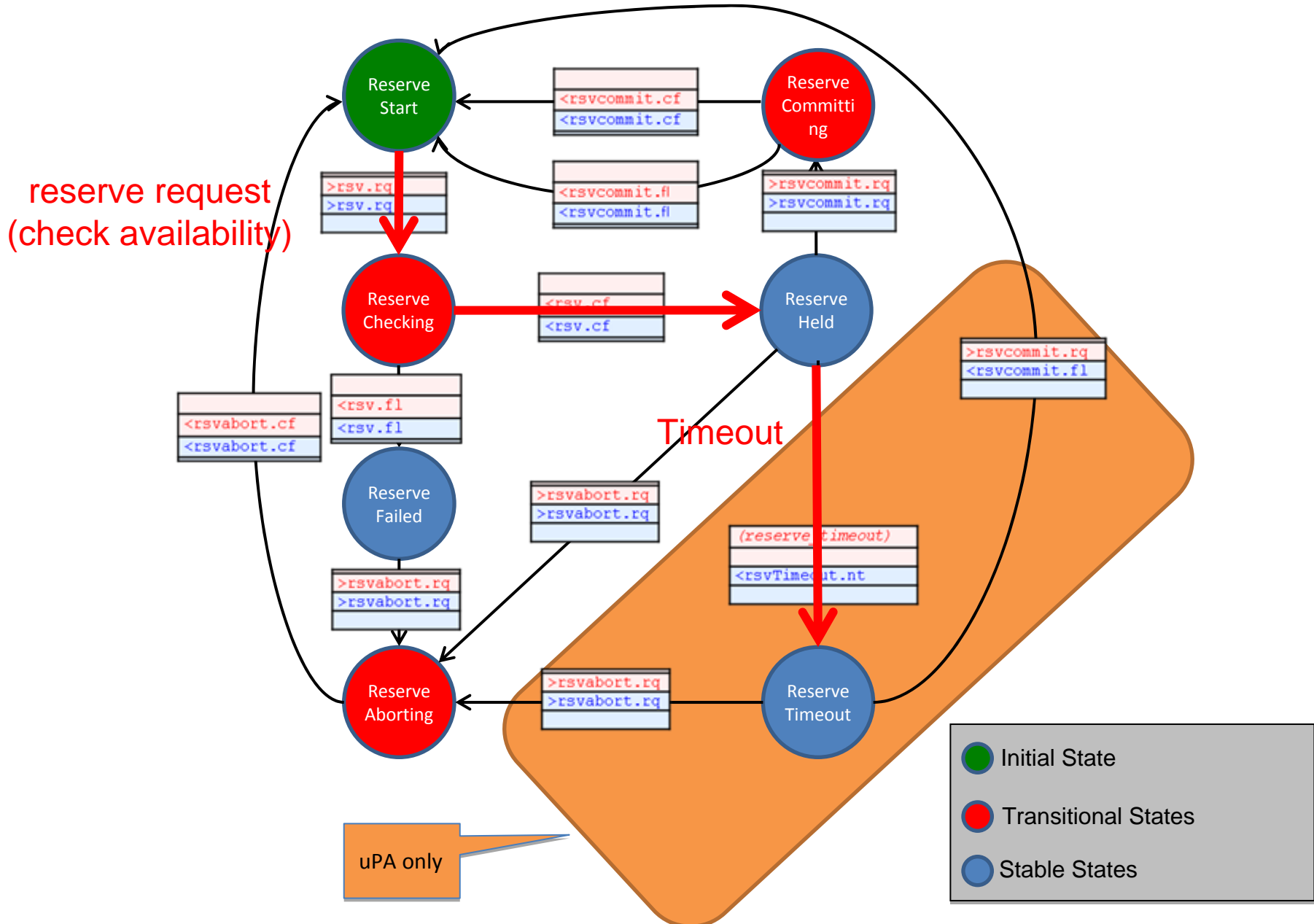


RSM: Reservation State Machine aborted after failed








RSM: Reservation State Machine

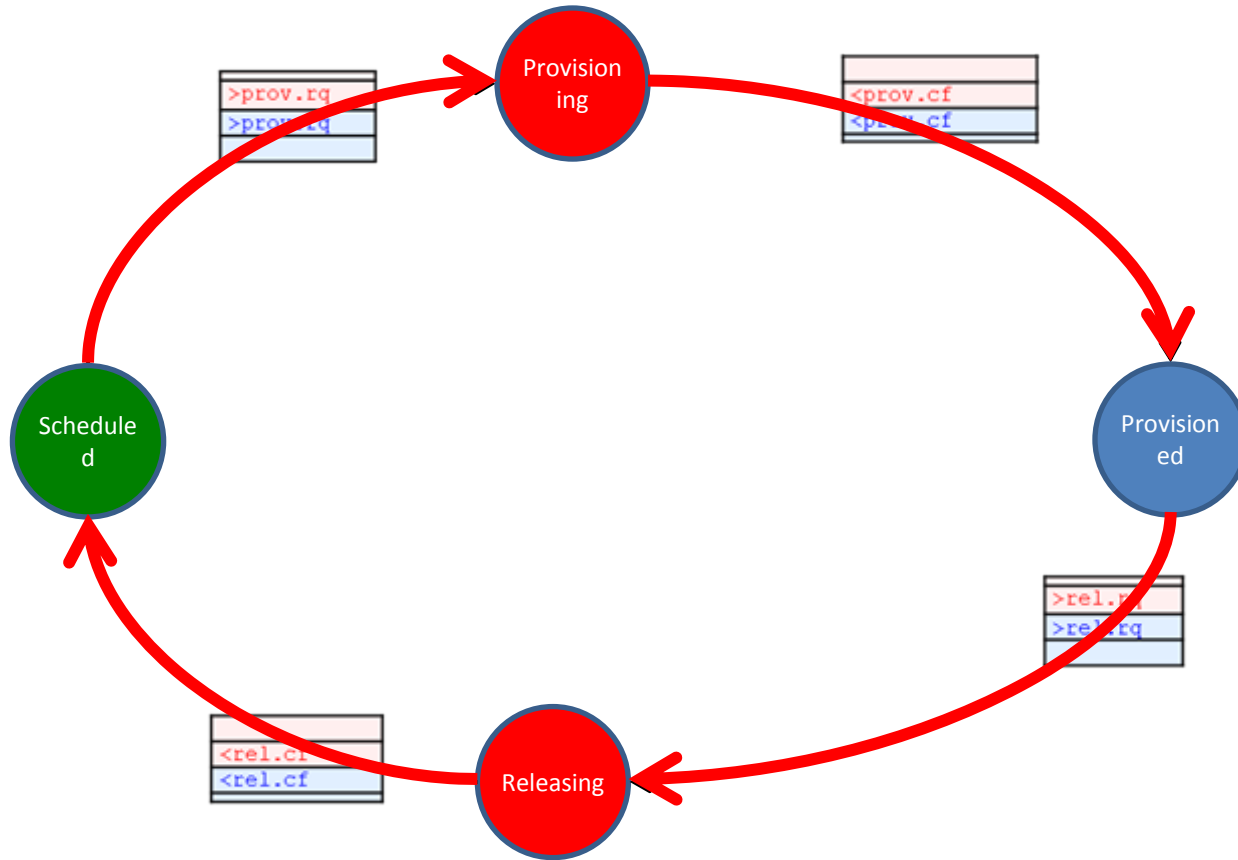
timeout (no commit or abort for held resource)



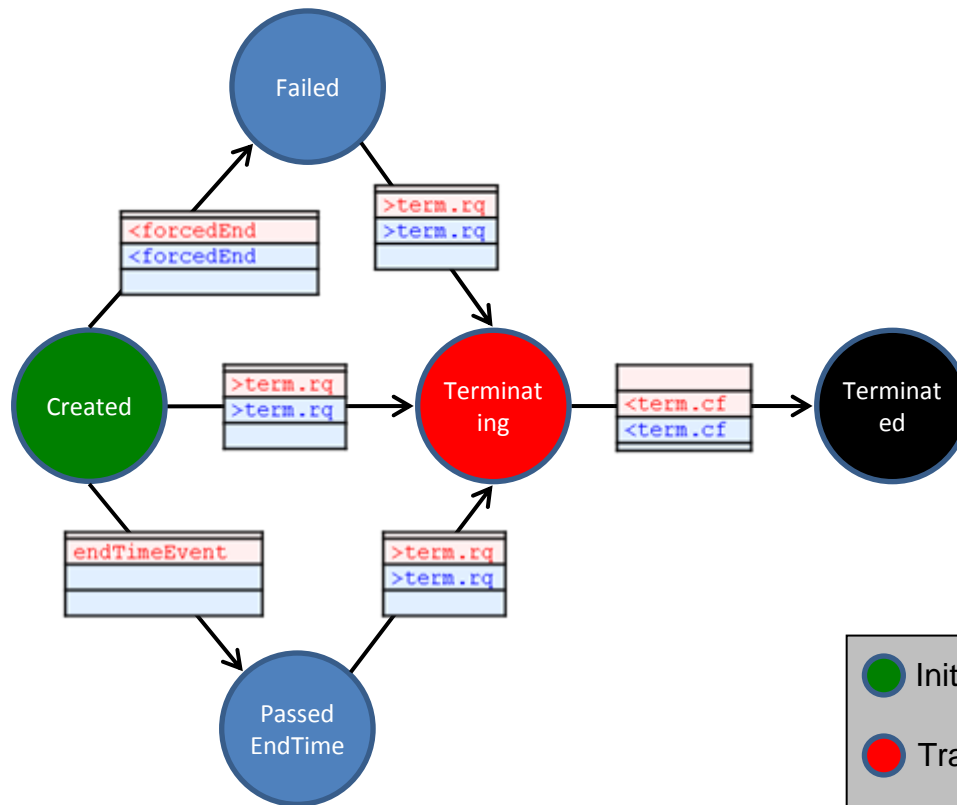
Provision State Machine

-  Simple state machine to explicitly provision / release a connection
-  Provision state machine transits independently from the reservation state machine
-  Provision state:
 -  Data plane is activated if the PSM is in "Provisioned" state AND $\text{start_time} < \text{current_time} < \text{end_time}$
-  A connection can be repeatedly provisioned and released

PSM : Provision State Machine



LSM : Lifecycle State Machine



● Initial State

● Transitional States

● Stable States

● Final State

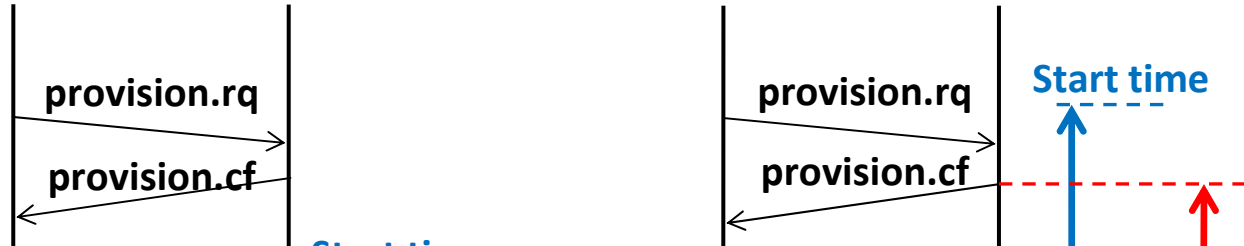
Data plane activation

- Activation may happen at the timing of following events (if the condition is met), using the latest committed reservation information
 - ▶ PSM transits to “Provisioned”
 - ▶ At the start_time
 - ▶ Reservation is updated (by commit of a reservation/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
 - ▶ 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

Provisioning Sequences

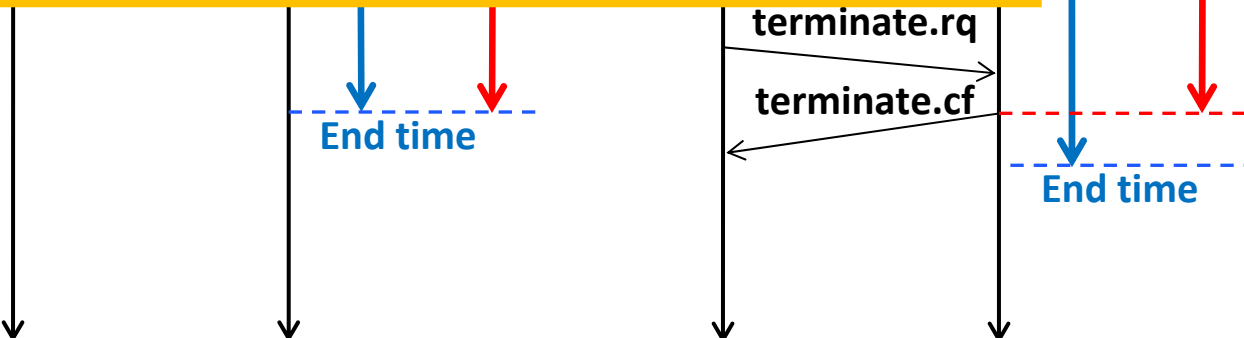
Automatic Provisioning
Requester Provider

Manual Provisioning
Requester Provider



On-demand reservation / provisioning

- start_time is same as or before the time a reservation is made
- provision.rq is issued immediately after a reservation is made

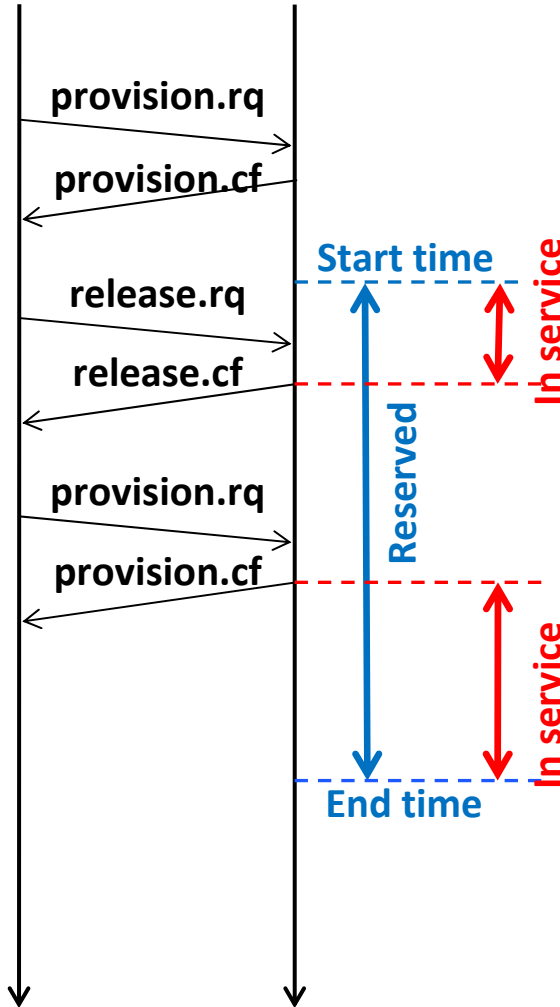


Provisioning Sequences (release and re-provision)

Automatic Provisioning

Requester

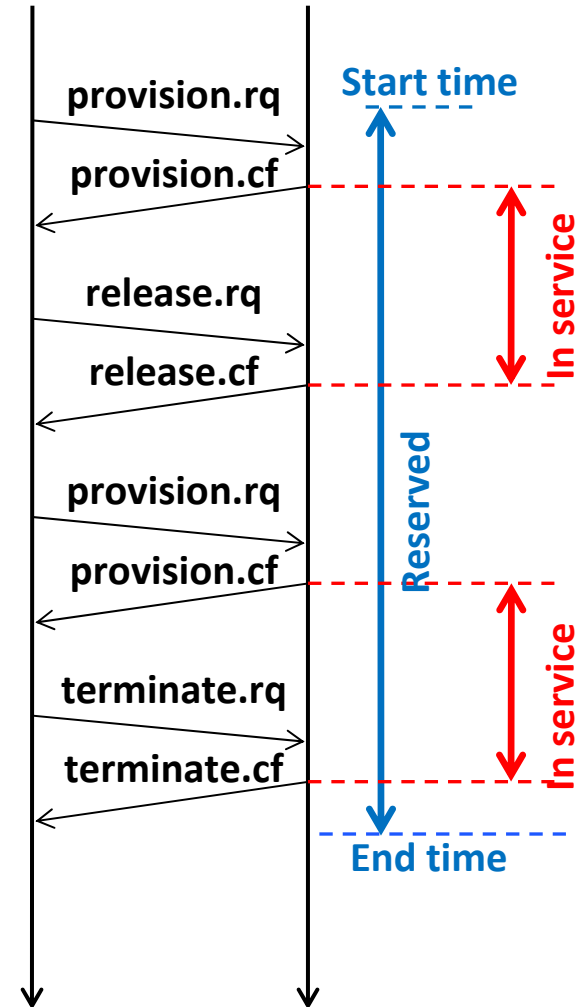
Provider



Manual Provisioning

Requester




Provider



Robustness of NSI CS2.0

- Error scenarios were carefully considered and supported
 - ▶ Supported scenarios include:
 - Ⓢ NSI message delivery failure (transport error of management plane)
 - Ⓢ uPA cannot provide data plane as reserved
 - ⊕ Temporal and permanent failure scenarios
 - ▶ Error notification messages are provided to notify errors from children to parents.
- uPA initiated commit timeout is supported
 - ▶ uPA does not have to hold resources forever when no commit or abort message is received.
- uRA can get statuses of children NSAs by query requests.
 - ▶ Simple (one level) and recursive (multi level) query operations are provided

NSI Development & Road Map

-  OGF NSI-CS version 2.0 is in draft now (May. 2013)
 - ▶ And will be finalized TODAY!
-  Version 2 implementations are under development
 - ▶ **OpenNSA** – NORDUnet (DK)
 - ▶ **OpenDRAC** – SURFnet (NL)
 - ▶ **AutoBAHN** – GEANT (PL)
 - ▶ **G-LAMBDA-A** - AIST (JP)
 - ▶ **G-LAMBDA-K** – KDDI R&D Labs (JP)
 - ▶ **DynamicKL** – KISTI (KR)
 - ▶ **OSCARS** – ESnet (US)
-  First production services planned for 2013:
 - ▶ NetherLight
 - ▶ StarLight
 - ▶ NORDUnet
 - ▶ GEANT
 - ▶ KRLight

Summary

- The NSI Connection Service protocol has been developed to allow dynamic service networks to inter-operate in a multi-provider environment.
- Version 2.0 of the NSI protocol is now being finalized.
- NSI v2.0 can be used as a key building block for the delivery of distributed virtual infrastructures

THANK YOU

Background

- With the trend towards Big Data, the **federation of data and services provided by different clouds** is becoming key to developing new services.
- To facilitate these new services, the efficient transfer of data between clouds is becoming more important, and **reliable network bandwidth between datacenters** is critical to achieving a stable high performance service.

Networks in Clouds

- Networks are often been taken for granted by Clouds.
- Activities to make network a manageable resource are on-going
- Longer distances may require using more than one provider
- Multi-provider network should be just another resource – easy integration with scheduling of computing/storage resources is important

Multi-domain cloud

