**Connection Service Architecture**

Status of This Document

This document provides information to the Grid community on the NSI Connection Service that operates on the interface between a requesting software agent and the provider software agent. It is intended to describe the protocol architecture and associated processes and environment in which software agents interact to deliver the Connection Service. Representing applications or other networks, these agents may request certain services of other network agents. Distribution is unlimited.

Copyright Notice

Copyright © Open Grid Forum (2008-2010). All Rights Reserved.

Trademark

OGSA is a registered trademark and service mark of the Open Grid Forum.

Abstract

This document describes the Connection Service for the Network Service Interface (NSI). The Connection Service is used to manage connection oriented circuits that transit network operators. The Network Service Interface (NSI) is defined to be the set of protocols and parameters that are used between a software agent requesting a network service and the software agent providing that Network Service. The Connection Service is intended to operate with the Network Service Framework described in (GWD-I-XX).

Contents

[1. Context and Overview 2](#_Toc273364720)

[2. Introduction to the Connection Service 2](#_Toc273364721)

[3. Connections 3](#_Toc273364722)

[4. Connection Service primitives and lifecycle 5](#_Toc273364723)

[5. QueryRequest and Notify Primitives. 6](#_Toc273364724)

[6. NSI Messages 8](#_Toc273364725)

[6.1 NSI message common attributes 8](#_Toc273364726)

[6.2 Connection Service common attributes 9](#_Toc273364727)

[6.3 Attributes of the Reserve primitive 10](#_Toc273364728)

[6.3.1 Reserve attributes usage in a request 10](#_Toc273364729)

[6.3.2 Reserve attributes usage in response 10](#_Toc273364730)

[7. Temporal aspects of the Connection Service 10](#_Toc273364731)

[8. Connection reservation and timing parameters 10](#_Toc273364732)

[9. Service Definitions for Connection Services 10](#_Toc273364733)

[10. Transport failure awareness 10](#_Toc273364734)

[11. The Path Object 10](#_Toc273364735)

[12. Tree and Chain Connection modes for inter-domain pathfinding 10](#_Toc273364736)

[13. Contributors 10](#_Toc273364737)

[14. Glossary 10](#_Toc273364738)

[15. Intellectual Property Statement 10](#_Toc273364739)

[16. Disclaimer 10](#_Toc273364740)

[17. Full Copyright Notice 10](#_Toc273364741)

[18. References 10](#_Toc273364742)

# Context and Overview

The NSI protocol is defined by a suite of documents. This informational (recommendation?) document describes the NSI Connection Service. The Network Services Framework document (GWD-I-XX) defines a platform for the provision of Connection Services.

The NSF defines several key architectural elements: a Network, a Network Service, a Network Service Agent (NSA), a Network Service Interface (NSI), and a NSI Protocol. These elements exist in a notional Network Service Plane. The framework describes an environment within which network objects are defined as manageable resources. Within the framework, these network resources can be selected, allocated, interrogated, and manipulated by software agents on behalf of requesting users.

Network resources and capabilities are presented to the consumer through a set of ‘Network Services’. The Network Services Framework presents a unified model for interacting with these services. Network Services include the ability to create connections, to share topology information, and to do other services needed by a set of federated NSAs.

Recently, some networking communities have developed tools and protocols to automate the process of network resource allocation, these tools allow the network user or application to participate directly in the path creation process. The proliferation of these new approaches to automating transport connection provisioning is driving the need for a new standardized Connection Service interface.

Further, there is a growing requirement to integrate customized networks resources into existing grid resources pools and applications. The ability to manage network connections effectively and easily by the grid community is perhaps an important driver for the OGF NSI specification effort.

Where Capitalized words are used in this document, these have a formal definition; see the glossary for details.

# Introduction to the Connection Service

This document defines a Connection Service to support the creation, management and removal of Connections. The Connection Service is built on the NSI Network Service Framework.

The Connection Service is message based command-response protocol that operates between a requester NSA and a provider NSA. The protocol includes a set of defined primitives that are intended to provide the control necessary to manage Connections. The command primitives are Reserve, Provision, Cancel, Query and Notify.

The Reserve primitive allows a connection to be requested of a network provider. The Provision primitive allows a requester NSA to request a scheduled connection be provisioned. The Cancel primitive allows a requester NSA to request that a connection be removed. The Query primitive allows a requester NSA to request the status of a service instance from the provider NSA. The Notify primitive allows the provider NSA to send spontaneous notifications to the requester NSA.

These command primitives are used to create, monitor and manage the connection state. In the NSI, a connection goes through five states: Reserving, Scheduled, Provisioning, In-Service, Releasing.

The use of NSI primitives to initiate, manage and remove a connection is shown in Figure 1.

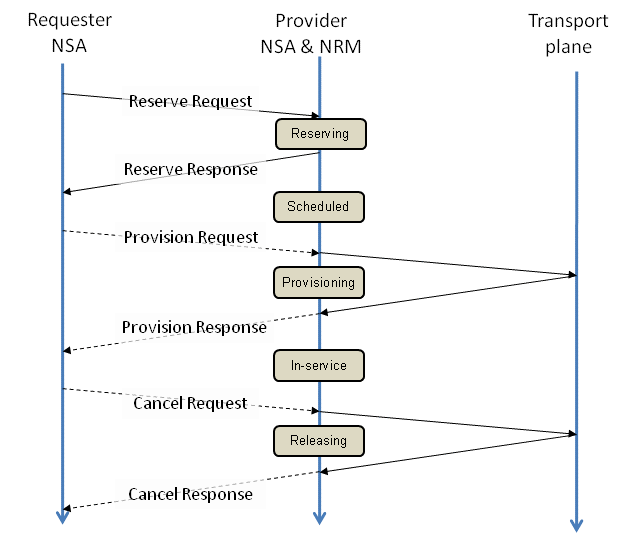


Figure : Connection Lifecycle

# Connections

The NSI Connection Service is used to reserve, schedule, and instantiate Connection instances. Single channel, point-to-point connections are supported in v1.0. These may be flagged as either uni-directional or bidirectional connections.

The NSI Connection Service is designed to create Connections that support a high capacity, highly asymmetric data flow such as occur in large file transfers or real-time streaming of digital media content.

As illustrated in Figure 2, the Connection consists of three basic components: an ingress point where user data enters the connection, a transport section that carries the data across the network, and an egress point where user data exits the connection.

Connections may be concatenated at ingress and egress points to create longer Connections. In the Inter-Network model, these points are modeled as Service Demarcation Points (SDPs) [1].

A service demarcation point can function as both an ingress point on one side and an egress point on the other. Two such connections that share a single service demarcation point in this way are said to be concatenated. These two concatenated connections then appear to the user payload as a single end-to-end transport plane data-path. In this way a service demarcation point becomes intermediate transit-point of a path or connection, i.e a routing point through which the connection must pass.

SDPs are constructed from pairs of SPS. Once instantiated, an STP may have properties such as a framing, bandwidth and a VLAN id. Some of these properties may reflect the requirements specified in the Service Definition. Labeling (cf. fiber id, wavelength, VLAN id) and aggregation (cf. combining multiple switch ports) can be modeled as a property of an STP.

Figure 2: Anatomy of a Connection

In the transport layer, the user data (the “payload data”) is carried across each section of the network inside a “framing protocol”. The framing protocol, provides the necessary timing, control, and data integrity functions required to move the payload from node to node through the network. It is important to distinguish between a) the access framing protocols, b) the transport framing protocols, and c) the user payload data carried inside each of these protocols.

* *It is the user payload data stream that must be preserved from ingress to egress in a Connection.*

The transport framing can be any framing protocol as long as the end-to-end preservation requirement is honored. In fact, the only constraints on the transport framing is that the transport section be able to adapt the ingress payload data to each of the successive transport protocols that may be used along the path and ultimately be able to adapt the user payload to the egress framing at the egress point. While specification of the connection end points, access framing, and other parameters associated with a connection are defined by the connection requester (or implicitly by the service definition), the choice of the transport protocol and associated transport path parameters are explicitly delegated to the network service provider in order to allow the provider the greatest latitude in finding a valid, available, and optimal path for the connection request. This is another example of how abstraction separates the user perception of a “connection” from that of the provider. While this abstraction simplifies the service concept, the NSI Architecture allows the omniscient requester to participate in these connection planning decisions.

# Connection Service primitives and lifecycle

Advance vs. immediate reservation

The connection request can have either a start-time and end-time, or a start-time and duration.

If the connection request includes a valid start-time and an end-time then the request is considered to be an advance reservation request. If the connection request has the start-time set to ‘asap’ and has a duration field rather than an end time field, the request is considered to be an immediate reservation request.

Automatic vs. explicit provisioning mode

Provisioning of a connection is achieved using a 1 phase commit mode. The transition from Scheduled state to Provisioning state can be either explicit (i.e. signaled by the requester NSA) or automatic (i.e initiated by the provider NSA). Each connection request will include a *flag* to indicate which of these two provisioning modes is to be used. It is not possible to mix these modes in a single connection.

When operating in explicit mode, it is the responsibility of the requestor NSA to signal the reservation to begin provisioning and to begin de-provisioning of the connection. These signals are known as the Provision and Cancel primitives.

Responses to connection requests

The provider NSA will send a response back to the Requester NSA once it has completed processing a connection request. This response will send back a pass/fail response and will confirm the state of the provider NSA.

When provisioning and de-provisioning have been completed it is necessary for the provider NSA to send a response back to the requester NSA to confirm completion.

As shown in Figure 4, a requester NSA first sends a Reserve command message to a provider NSA. The provider NSA schedules resources, and notifies the requester whether it can deliver the request. If explicit provisioning is used, the requester must send a Provison command message to the provider NSA, and the NSA then provisions a service instance. If the automatic provisioning is used, the resources are provisioned at the start time. The requester may send a Cancel message. A Cancel message received before the provisioning has begun removes the reservation from the reservation database. A Cancel message received after the provisioning is completed releases the provisioned resources.

First, a request is submitted to a provider thus beginning the life cycle of the connection within that provider Agent’s network. This first phase is called the “Reserving” phase. It includes path selection and resource reservation. In the NSI, path selection includes future scheduling as well as the performance and authorization checking. Once the scheduling phase is complete, the Requester NSA is notified and connection goes into a “Scheduled” state.

For automatically provisioned connections, when the service start time arrives, the Connection goes into a “Provisioning” phase. For signaled Connections, the “Provisioning” phase is initiated by a signal from the Requester NSA. Provisioning is the process where the connection is physically instantiated by configuring each device along the path to reflect the path plan developed and reserved in the Reservation phase.

Once provisioning is complete, the connection then moves into an “In-Service” state and the user are notified that the connection is ready for use. The In-Service phase is where user data is allowed to transit the connection.

When the connection is no longer needed (or the scheduled time expires) the connection is “Releasing”. The Releasing phase is where each network along the path is informed of the Release event and resources associated with the connection are released back to the available pool. Upon entering the releasing phase, the connection will no longer pass traffic. When the Release has completed, the connection object is deleted from the Service Plane.

# Query Primitive

The query primitive allows the Requester NSA to query the Provider NSA.

Figure : Attributes of Query primitive

Figure : Timing of Query primitive

# Notify Primitive

The notify primitive allows the Provider NSA to provide spontaneous notifications to the Requester NSA.

Figure : Attributes of notify primitive

Figure : Timing of notify primitive

# NSI Messages

Each NSI message includes a set of attributes. These are exchanged between NSAs to manage a connection. The NSI message attributes are divided into 3 groups:

* Message attributes
* Service attributes
* Primitive attributes

## NSI message common attributes

Message attributes includes the attributes that are common to all messages. This includes the NSI version, NSA addressing and security, the service that is being used and a transaction identifier. This is depicted as an UML entity relationship diagram below.

Figure : Attributes common to all messages

*NSIversion*

This message attribute identifies the NSI framework protocol version of the NSA that creates the message. This message attribute is included to allow new protocol versions to be released in future. In addition each service can have its own protocol version.

*RequesterNSA*

This message attribute identifies the requestor NSA sending a request or receiving a response to a request or receiving a notification. This message attribute also includes security parameters for the requester NSA.

*ProviderNSA*

This message attribute identifies the provider NSA receiving a request or sending a response to a request or sending a notification. This message attribute also includes security parameters for the provider NSA.

*ServiceType*

This message attribute identifies the service type, i.e. Connection Service, etc. being managed.

*messTransID*

This message attribute allows the requester NSA to match responses with requests where multiple responses are pending.

## Connection Service common attributes

Service attributes includes all attributes that are common to a particular service. In the case of the Connection Service, this includes the Connection Service version, the Connection identifier, the relevant service primitive and a transaction identifier. This is depicted as an UML entity relationship diagram below.

Figure : Attributes common to the connection service

*ConnServVersion*

This message attribute identifies the version of the connection service supported by the originating NSA. This is designed to allow for new version of the connection service in future.

*CID – Connection ID*

This message attribute identifies the connection associated with this message.

*Service primitive*

This message attribute identifies the connection service primitive. The allowed Connection Service Primitives are:

Reserve

Provision

Cancel

Query

Notify

*servTransID*

This message attribute allows the Connection Service to match responses with requests where multiple responses are pending.

## Attributes of the Reserve primitive

The Connection Service has a set of associated primitives. These are the ‘commands’ that are used to manage a connection. The connection is initiated with a Reserve primitive. This primitive has its own set of attributes: the start and end time of a connection, the service attributes derived from the Service Definition and the Path Object which contains routing information. This is depicted in the diagram below.

Figure : Attributes common to the Reserve primitive

The Connection Service has one or more associated Service Definitions (SDs). The Service Definition formally describes the service level that a user can request. This includes all of the attributes needed to define the performance of the circuit as experience by the user. This will typically include the client framing type, directionality, connection bandwidth and latency, jitter among others.

The Provider NSA must support the Service Definition nominated in the requested Connection Service instance for the connection request to be valid. The service definition includes a message attribute called ServiceCategory. The serviceCategory identifies the service type to be provided, the value of the ServiceCategory attribute is defined in the Service Definition.

An example of a ServiceCategory could be an Ethernet VLAN service. In this case the *ServiceParameters* attributes might be:

* ***Client framing***: client tagged VLAN as per 802.1Q
* ***Bandwidth***: 100Mb/s to 10Gb/s with a granularity of 100Mb/s
* ***Latency***: 1ms to 1000ms, granularity of 1 ms
* ***Directionality***: unidirectional/bidirectional

### Reserve attributes usage in a request

The attributes of the Reserve primitive are used to define the characteristics required of the connection service instance being requested.

*startEndTime*

In the request this message attribute includes the service start and end times (are these the in-service times or the provision times? This is still to be agreed). This also includes an attribute to indicate the service provisioning mechanism - autoSignalled.

*ServiceParameters*

In the request these set of message attributes are defined in the Service Definition. Each ServiceCategory will have its own set of ServiceParameters. This might include framing type, bandwidth, etc.

*PathObj*

In the request, these PathObj message attributes path describe a topological sequence of network objects that are included in a connection, typically these will be STPs, but could be other network objects such as networks. A request must include at least a pair of edge STPs. It may include additional network objects that can be either hints or requirements in the topological path of the requested connection.

### Reserve attributes usage in response

*startEndTime*

In the response these message attributes will return the start and end time attributes assigned by the provider NSA. If no available times are found, a ‘fail’ response will be returned.

*ServiceParameters*

In the response these message attributes will contain the service parameters provided by the path computation. If any service parameters cannot be provided, a ‘fail’ response will be returned.

*PathObj*

In the response these message attributes will optionally contain the completed object provided by the path computation. If no path can be computed, a ‘fail’ response will be returned.

# Temporal aspects of the Connection Service

Services, in which resources are dynamically requested, reserved and provisioned, require temporal aspects to be understood and deterministic. For the purposes of advance reservation Connection Service process must maintain its own real-time clock, and it is necessary for these clocks to be aligned.

When resources are sought by a requester NSA from a provider NSA, a service instance is created and an identifier is assigned to that service instance. Then, according to the parameters of the request (i.e. its Service Definition), the provider NSA identifies and reserves a set of available resources which satisfy the request and associates them to the instance. The resources are provisioned and released at some point on the temporal axis. The time information and signaling are used to specify the time boundary of the requested connection in-service period. It is the responsibility of the Provider NSA to attempt to deliver the connection as close to the start and end times as it is able. It should be noted that this may have some uncertainty as typically the duration of the provisioning phase cannot be precisely predicted.

In the case of the Connection Service, advance reservation requests will specify the required resources and the provisioning start and end time. The request is processed by a scheduler, and the scheduler finds a set of resources available for the requested duration and allocates them to the request to create a reservation. If the scheduler cannot find an available set of resources which satisfies the request, the request is denied. This scheduling process is part of path finding in the connection service. A reservation database (i.e. calendar) should be maintained by the scheduler or resource managers, and referred and updated by the scheduler. The detail of advance reservation is covered in detail in paragraph 4.

# Connection reservation and timing parameters

For advance reservation with *automatic* provisioning, the start-time refers to the time at which the connection moves from provisioning state to in-service state. It is the responsibility of the provider NSA to make sure that this in-service start time is met. This may require beginning the provisioning process in advance of the start-time and will require some knowledge of the expected provisioning time.

For advance reservation with *explicit* provisioning, the start-time refers to the time at which the provider is able to accept a provision signal. It is now the requestor’s responsibility to advance the explicit signal to ensure good in-service time. The reservation end-time refers to the time at which the reservation is removed. (If the user has not yet sent a CancelRequest signal the connection is de-provisioned first)

“Infinite” can be used as an end time. In this case, resources are reserved forever (i.e. until a release request is received or may be overwritten by policy limits). Note that the resource reserved forever cannot be used for other requests of later time.

It takes some time to process a request. Possible maximum time required to process a request and make resources ready for provisioning is called “guard time”. Each provider NSA must define its guard time and provide it to requester NSAs. A requester NSA should not request a reservation which start time is smaller (earlier) than (current time + guard time). Time required for message delivery should also be taken into account.

A flag is included to indicate whether a start-time later than specified guard is allowed. When allowed, these start times are treated as ‘now’ and warning may be sent back to the Requester NSA. Otherwise, if a provider NSA receives a reservation request which start time is before (current time + guard time), it simply denies the request (the start time is a constraint in the path finding process). Note also, that if explicit provisioning is used, the processing of a ProvisionRequest message will take some time to complete.

This system is designed to be compatible with systems based on 2PC. In a 2PC system, an additional phase exists between P2 and P3. This phase is the ‘commit’ phase. The commit phase allows the originating requestor NSA to collect reservation confirmations from child NSAs. The originating requester then sends out a commit request once all confirmations have been received. The purpose is to prevent provisioning beginning on any networks before all participating networks have confirmed their reservation. This prevents partially provisioned connections being created.

This operation can be replicated in the 1PC system defined here with the use of explicit provisioning mode. In this case the original requester may wait for all child NSAs to confirm their reservation before issuing a ProvisionRequest message. This in effect combines the commit and provision requests of the 2PC method into a single message.

# Service Definitions for Connection Services

The Service Definition formally describes each aspect of a service. Indeed, a ”service” only exists if it is formally defined in some manner. Within the NSI Architecture, each network presents one or more transport services at its inter-domain edge points. Each service is defined in a document called the Service Definition (SD). With respect to the NSI Architecture, this document is a machine readable format that allows the NSAs to access and validate service requests against the services offered by the associated network. In practice, the service definition should also be available in a human readable form so that users and applications developers have guidance as to what network transport capabilities are available.

The Service Definition has its roots and most immediate application in the definition of the NSI Connection Service offering(s), and for NSI v1.0 that is the sole purpose for which the Service Definition is adopted. (Note that the notion of formal service specifications is still a widely researched topic with new application to emerging network services from Connections to Topology to Monitoring. Further exploration and refinement of this helpful concept within the NSI Architecture will be a continuing effort in NSI futures.)

Each service offering has a service definition. The SD consists of a list of attributes or parameters that identify each characteristic of the service. For each service parameter the SD specifies the range of valid settings for that parameter. For instance, an “Ethernet Transport Service” might define a service parameter called “Capacity” that defines a range of allowable service capacities between 1 Mbps and 10 Gbps. Another Parameter, say “Access\_Framing”, may specify a set of framing protocols that the user may request for ingress or egress. In this case, the Access\_Framing might be “802.1”, “802.1Q”, and “802.1ad”, default = “802.1”, indicating that ethernet frames will be carried that conform to one of three IEEE standards. A default may also be specified in order to fully specify a service request where the user does not specify a value for a particular parameter, or where the requester may wish to allow greater degree of freedom to the NSA in selecting a path

If a service request describes a service instance that lies within the bounds of the set of defined service parameters, then it forms a “valid” request. Each provider NSA along a candidate path must compare the service request to the local Service Definition in order to insure each specified parameter lies within the range of valid settings that the service offering can support. If a service parameter is not present in the service request, then the provider NSA should “fill in the blanks” from default values in the Service Definition. As the request is processed down the NSA service tree, default values adopted in one transit network may implicitly constrain the request in downstream networks. Therefore, in general, each NSA should use default values that provide the greatest leeway to the pathfinder in satisfying the request both within the local network and in external downstream networks. Ultimately, the parameters explicitly specified by a requester agent must be honored. All other parameters must simply be compatible.

When the NSAs complete the reservation process for a service request, the reservation confirmation indicates that the network (*all* networks along the path) have agreed to provide the requested level of service. This constitutes a defacto service level agreement upon which the requesting agent should be able to depend.

A key architectural aspect of the Service Definition paradigm is that each service instance has a clearly defined profile that constitutes when that service instance is performing to specifications, and when it is not. This performance can be measured end to end by user agents. If a service instance does not provide the service level specified in the request, and confirmed by the network, then it is in violation of the service agreement. It is beyond the scope of NSA Architecture to discuss what the implications should be for a SLA violation. However, when a confirmed service instance does not meet its agreed upon performance levels, it should be detectable by both the Requesting Agent and the Provider Agent in some manner. The NSI architecture does not require such monitoring by either agent – the architecture will function without it. But Best Common Practice would council that a violation is indicative of a failure somewhere in the chain, and should raise flags to notify operations personnel or the end applications so that remedial action can be initiated.

The service definition provides a publicly available description of the service, and should be made available in a native language document that the users can reference in developing or configuring their applications. The users (application developers) should consult this service definition in order to understand what service capabilities are available to them within a given service offering.

The Service Definition also plays an important role long before a service request is received. The service definition can be used as design objectives during the engineering phase of deploying a new service. The network engineering team can look at the SD in order to select hardware and software than can meet the technical and administrative requirements of the service. Further, two networks with similar services may compare and negotiate a common, or at least interoperable, service definitions. For instance, if network A offers capacity in 50 mbps increments, and network B offers it in 1 mbps increments, these networks will still be compatible – though certain requests may require more resources than are actually necessary.

It is important to stress one more aspect of a service definition; if a parameter is not identified within the service definition document, then the user can make no inference about its presence, absence, or value in the service. For instance, if a service definition has no jitter specifications, the user can make no predictions or assumptions about the jitter characterisitcs. And the network has made no commitments regarding jitter. Indeed, a request satisfied on Monday might have excellent jitter characteristics, and the exact same request submitted and satisfied on Tuesday might have horrid jitter characteristics. As long as the service constraints presented on both requests were met, these are – from a formal service perspective – properly performing and identical service instances.

The converse is also true. The network should be very careful about how it defines service parameters. For instance, an Ethernet service may define connection capacities in “bits per second” (bps). On its face, one might construe that a 1 Gbps connection would accept bits at 1 billion bits each second measured over any one second period. However, if this 1 Gbps connection is provisioned over a 10 Gbps network link, this interpretation would allow a 100 millisecond burst at 10 Gbps followed by a 900 millisecond quiescent period. Such a burst of 125 megaBytes can easily induce buffer overruns and packet discards on interfaces along the connection path. …And yet the user would have been perfectly within their performance profile. The implication here is that simple fixed capacity connections in asynchronous packet transport networks requires sophisticated and detailed planning in order to guarantee service capabilities. The service definition allows the networks to specify burst characteristics they can support for connections reserved across their infrastructure. These burst charateristics can be pro-actively requested by the user, or implicitly applied during pathfinding.

When a service request is returned to the requesting agent, the full template of service parameters should be returned containing the values assigned during pathfinding and reservation. This allows the requesting agent to adapt local processes appropriately.

# Transport failure awareness

Move this section to the connection service part.

Failures in the transport plane can occur at anytime, however within the framework of the NSI architecture, there are two time windows in which a transport plane failure is significant:

1. The time between the service reservation phase and provisioning phase (i.e. TReservationCompleted to TProvisionStart), and
2. The time between the service provisioning phase and teardown phase (i.e TProvisionCompleted to TTeardownStart).

Of course, the errors only need to be handled by the NSA if the transport resource errors affect the user service.

Figure 10: Local/Remote Failures

A few illustrative examples will help describe the kind of failure and recovery scenarios that have to be considered when building the state machine for the NSI protocol.

Transport failure during service reservation phase and provisioning phase : An element in the transport plane becomes unavailable due to a soft or hard failure causing a provisioning failure of a confirmed reservation, the reservation manager can handle this by either reserving an alternate path as long as it meets the requested service characteristics or canceling the reservation with notification. Domain policy and availability of resources will determine what recovery action is taken by that domain.

Transport failure during provisioning phase and teardown phase: In case a failure in the transport plane affects an active connection requested in the service plane, the first recovery mechanisms will be triggered by the protection mechanisms provisioned with the service. If the connection service is unprotected, then the failure notification will be sent to the Domain’s NSA. At that point, NSA will take appropriate action based on service and user policies by either re-routing the connection within the domain or tearing down the service with notifications to other domains involved.

# The Path Object

The “Path Object” (or Path) describes a route through the topology. When present in a Connection Request, the Path specifies an ordered set of Service Termination Points (STPs) that the connection must transit, and in the order the connection must transit them. Within a Connection Request, the Path Object, at a minimum, must specify the ingress and egress STPs for the Connection. Additional intermediate transit points may be included in the Path, and when present, they are considered a required constraint on the Connection’s route and must be honored.

A Path Object associated with a confirmed Connection contains, or references, a significant amount of information regarding the user, the source or destination of flows, the global topology, and internal detail of specific networks, etc. This Path information may pose a security or privacy issue to the user or the involved networks, or may just be considered proprietary information. Within the NSI, access to such information is considered a policy decision of each agent involved. Therefore, Path information is available to external agents via an authorized Query() primitive to the Connection Service.

The provider NSA is responsible for maintaining, among other things, a Path describing the fully specified path for any Connection reserved across its network. In order to protect the PO, the provider NSA must store the Path locally and return a redacted Path containing a list of STPs, and/or Named Path, specified in order, according to its internal authorization policies.

Since Connection Requests submitted to other NSAs may return a Path identifier rather than a Path Object itself, there must be means for distinguishing the two and a clear understanding of how a path object fits into the path algebra. Since the Connection Request segmentation processing is tree-like, it follows that the reserved Path Objects will also be tree-like. So a Path Object must be able to contain not just directly referenced STP Names, but must be able to contained Named POs as well. A Path Object then consists of a list of objects that either directly or indirectly resolves to topological points. For named POs, the NSA that owns the Named PO must also maintain authorization association(s) for the PO.

# Tree and Chain Connection modes for inter-domain pathfinding

There are two levels of pathfinding related to the NSI architecture: the inter-domain pathfinding which is concerned with choosing a path across the global topology of Networks, and the intra-domain pathfinding concerned with the transport resources within the Network. NSI is concerned only with inter-Network pathfinding.

Inter-Network Connections extend across multiple networks; they are constructed by concatenating connections built across the individual networks. This is done by choosing appropriate STPs such that the egress STP of one connection corresponds directly with the ingress STP of the successive connection.

The choice of which sequence of networks a path should follow is a pathfinding function. Two modes are described, tree and chain.

In the tree mode of pathfinding, once a set of STPs is chosen, the connection requests corresponding to each segment can be issued simultaneously and directly to the NSAs responsible for each of the corresponding networks. The process can be recursively implemented in for creating multi-level trees in the Service Plane. That is, several NSAs without direct control over the NRM/networks can be deployed in a hierarchical architecture with one or more levels.

The Tree model processing computes a course grained inter-domain path first. It uses that network path vector to decompose the connection request into several concatenated connection segments. A benefit of the tree model is that it enables the NSA to reserve the segments in parallel via direct interaction with the respective networks. The disadvantage of this approach is that it is not known in advance whether each Network has the resources available internally to reach the next Network chosen by the inter-Network pathfinder.

Figure 11: Tree pathfinding mode.

Alternatively, if the local NSA does not have sufficient topology information or authorization credentials to identify and interact directly with all the downstream networks, the local NSA can simply choose a neighbor network as the next hop, and using the interconnect STP as the ingress point, forward a request to that next hop NSA for handling. This conventional hop-by-hop approach is called the Chain model.

Chain style processing reserves resources sequentially beginning at the source STP and working hop by hop successively through each downstream network to the destination. The path computation requires only a simple next hop reachability calculation (though more sophisticated path finders can be implemented), and no downstream resources are reserved until the upstream prefix path has been confirmed. It is highly distributed, scales well and is robust. But it does hide or delegates much network provisioning decision to [unknown] downstream agents.

Figure 12: Chain pathfinding mode.

In both the tree style processing and the chain style processing, the end-to-end connection cannot be confirmed until all of the constituent connection segments have been successfully reserved and confirmed. Which model will be more effective is unclear at this time and will likely be directly related to complexity of topology distribution and path analysis, robustness, authorization schemes, request volume, network diameter, utilization density, cost, ease of use, and reach (to name just a few aspects.)

Both the Tree and Chain model reduce pathfinding to a constraint-based search over a topology to build a k-preferred path tree. Both can accept requester guidance in path selection through the inclusion of intermediate transit points in the connection request (discussed further under Path Objects). The method, tree or chain, used to process a request is made exclusively in the requester NSA. The requesting agent implements a Tree model process by submitting individual requests for each connection segment. These individual segment requests are processed asynchronously and in parallel. The requesting agent implements a Chain model by allocating a path through the local network, and then forwarding the request to a neighbor domain to resolve the remaining downstream portion of the connection.

# Contributors

Joan A. García-Espín, I2CAT

Chin Guok, ESNET

Radek Krzywania, PSNC

Tomohiro Kudoh, AIST

John MacAuley, Surfnet

Takahiro Miyamoto, KDDI R&D Laboratories

Inder Monga, ESnet

Guy Roberts, DANTE

Jerry Sobieski, NORDUNET

Sebastien Soudan, Laboratoire de l'Informatique du Parallèlisme

John Vollbrecht, Internet2

# Glossary

Connection

A Connection is a conduit that transparently moves user information between STPs across a Network. A Connection has a set of properties (for instance, capacity, or authorization, or start time).  These properties, and their allowed range of values, are defined by a Service Definition. A Connection instance on the Transport Plane is identified by a Connection Identifier exchanged on the Service Plane. Connections are unidirectional.

Connection Service

A Connection Service is a service that allows a Requester NSA to request and manage a Connection from a Provider NSA

Control and Management Planes

The Control Plane and/or Management Plane are not defined in this document, but follow common usage.

Edge Point

A network resource that resides at the boundary of an intra-network topology, this may include for example a connector on a distribution frame, a port on an Ethernet switch, or a connector at the end of a fibre.

Inter-Network Topology

This is a topological description of a set of Networks and their transfer functions, and the connectivity between Networks.

Network  
A Network is an Inter-Network topology object that describes a set of STPs with a Transfer Function between STPs.

Network Resource Manager (NRM)

The Network Resource Manager owns a set of transport resources and has ultimate responsibility for authorizing and managing the use of these resources. Each NRM is always associated with a single NSA.

Network Services

Network Services are the services offered by an NSA. Each NSA will support one or more Network Services.

Network Service Agent (NSA)

The Network Service Agent is a concrete piece of software that sends and receives NSI Messages.  The NSA includes a set of capabilities that allow Network Services to be delivered.

Network Service Interface (NSI)

The NSI is the interface between Requester NSAs and Provider NSAs.  The NSI defines a set of interactions or transactions between these NSAs to realize a Network Service.

Network Services Framework

The Network Services framework describes a NSI message based platform capable of supporting a range of Network Services.

NSI Message

A NSI Message is a structured unit of data sent between a Requester NSA and a Provider NSA.

Requester/ Provider NSA

An NSA acts in one of two possible roles relative to a particular instance of an NSI.  When an NSA requests a service, it is called a Requester NSA. When an NSA realizes a service, it is called a Provider NSA.  A particular NSA may act in different roles at different interfaces.

Service Definition

The Service Definition consists of a set of attributes that formally and explicitly define the complete scope of a service offering. Each provider defines its service with an SD, each request identifies requirements in terms of SD attributes, and each Connection has an associated Service Definition instance.

Service Termination Point (STP)

Service Termination Points (STPs) identify the Edge Points in the intra-network topology.

Service Plane

The Service Plane is a plane in which services are requested and managed; these services include the Network Service. The Service Plane contains a set of Network Service Agents communicating using Network Service Interfaces.

Transfer Function

The Transfer Function is a matrix that describes the transport capabilities between STPs.

Transport Plane

The Transport Plane contains is the set of transport equipment and associated resources that carry user data through the network.

# Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

# Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

# Full Copyright Notice

Copyright (C) Open Grid Forum (2008-2010). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

# References

1. Network Service Framework GWD-I-XX