GFD-R-233
NSI-WG
nsi-wg@ogf.org

John MacAuley, ESnet
Tomohiro Kudoh, AIST
Chin Guok, ESnet
Guy Roberts, GÉANT
August 18, 2017

Applying Policy in the NSI Environment

Status of This Document
Grid Forum Document (GFD), Proposed Recommendation (R).

Copyright Notice
Copyright © Open Grid Forum (2008-2017). All Rights Reserved.

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

Abstract
This document describes an approach to applying policy to the Network Services Interface (NSI) Connection Service protocol.

Contents

1. Context and Overview

1.1 Network Services Interface

The Network Services Interface is a web service protocol that allows applications to monitor, control, interrogate, and support network resources that are made available by the provider of the network. The NSI Connection Service deals specifically with the request and management of network Connections on transport networks. NSI is inherently agnostic to the technology used in the transport plane. This technology agnostic approach is built into the NSI topology representation and is supported through the use of Service Definitions.

A Connection Service can be requested by any application that has implemented an NSI CS Requester Agent. Similarly, any network provider who has implemented an NSI Provider Agent can service the request.

Each service is managed by an exchange of NSI messages between agents. These messages operate using a set of service primitives. Service primitives are the set of instructions that allow the requester to set up and manage a service. Each service request will result in the allocation of a service id for the new service instance.

This document describes how policy is applied to the NSI Connection Service. This recommendation should be read in conjunction with GFD-R-212 Network Service Interface Connection Service version 2.0 [2], Open Grid forum GFD-I-213, Network Services Framework v2.0 [4] and NSI Authentication and Authorization [5].

2. Notational Conventions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in [RFC 2119]. Words defined in the glossary are capitalized (e.g. Connection). NSI protocol messages and their attributes are written in camel case and italics (e.g. *reserveConfirmed*)

3. Summary

Policy, in the context of this document, refers to the ability of a network provider to enforce acceptable usage policy in their local network. Network providers may apply policy rules that determine which users can request connectivity, how much bandwidth they can request or transit policies. Network providers all implement some level of authorization policy in their networks. This document describes an NSI CS policy framework that defines how the NSI CS protocol is used to provide sufficient information to the provider to allow them to understand the originator of the request and the routing details. The provider is then able to apply local policy in their network based on this information.

This document is organized into three main sections as follows.

- Section 4 ‘Policy Enforcement’ describes how policy can be enforced in the NSI CS. This section is normative.
- Section 5 ‘pathTrace ’ describes the *pathTrace* extensions to the NSI CS v2.0 schema and how it should be used. This section is normative.

Appendix A describes the research and education community use-cases for policy. A set of requirements are derived from these use-cases. This section is information and is included to help the reader understand how the NSI policy requirements were derived.

Appendix B contains the *pathTrace* extensions to the NSI CS v2.0 schema. Appendix B forms a normative part of this recommendation.

4. Policy Enforcement

This section describes how policy is enforced in the NSI Connection Service. This mechanism makes use of the inherent Service Plane trust between NSA along with proper operation of the NSI protocol to allow for uPA enforcement of Connection path related policies. The basis of the mechanism is the definition of a new NSI message header element which contains the detailed path proposed for the Connection. This new end-to-end path trace information is built by collecting proposed path segments from uPA involved in the reservation during the *reserve* message flow. The proposed end-to-end path is then propagated to each uPA involved in the connection as part of the NSI *reserveCommit* message flow, allowing the uPA to evaluate the full reservation path against local routing policy as part of the resource committing phase. A policy failure at the uPA is communicated using the standard *reserveCommitFailed* message, resulting in no changes to the core NSI CS protocol. During the *reserve* phase, the uPA will use local policy to accept or reject a reservation request based on information about its local Connection segment.

Note that NSI does not define a policy distribution mechanism, the method used to distribute policy is up to the implementer.

4.1 TREE Solution

This section describes how policy is implemented in the case of a TREE signaled Connection reservation. The NSI models for signaling and path finding are documented in detail in the NSI informational document GFD.217 Network Service Interface Signaling and Path Finding [6].

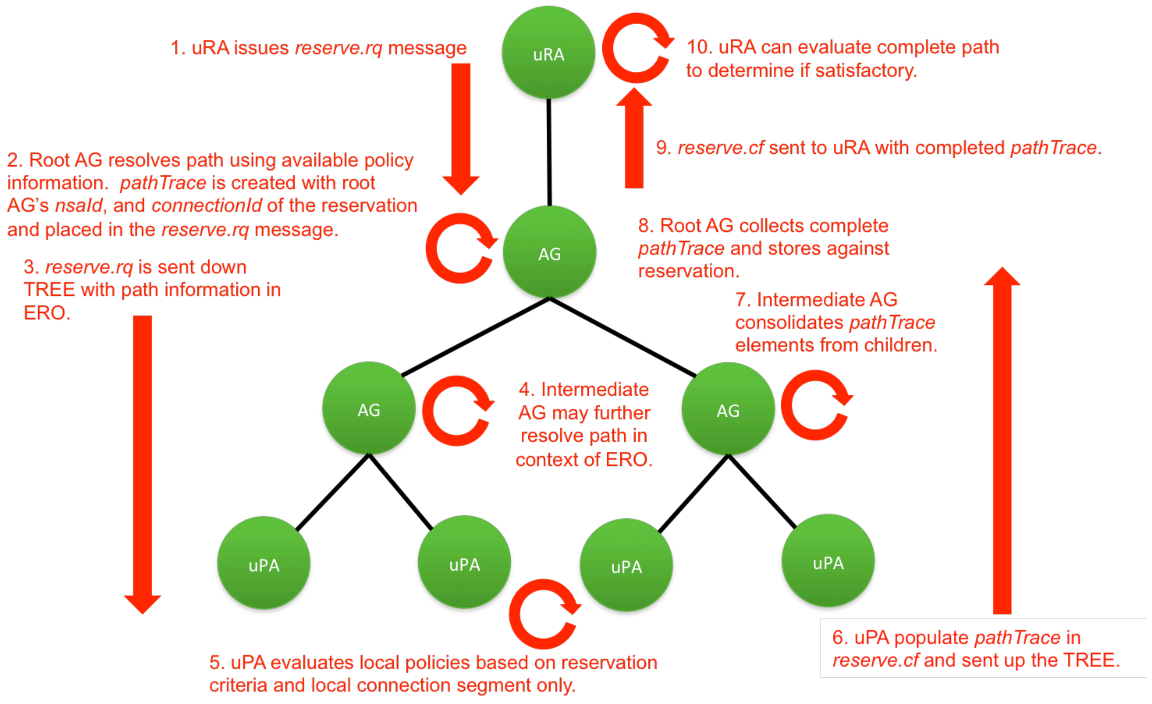
1. uRA issues *reserve.rq* message
uRA
10. uRA can evaluate complete path to determine if satisfactory.
9. *reserve.cf* sent to uRA with completed *pathTrace*.
2. Root AG resolves path using available policy information. *pathTrace* is created with root AG's *nsaId*, and *connectionId* of the reservation and placed in the *reserve.rq* message.
AG
8. Root AG collects complete *pathTrace* and stores against reservation.
3. *reserve.rq* is sent down TREE with path information in ERO.
7. Intermediate AG consolidates *pathTrace* elements from children.
AG
4. Intermediate AG may further resolve path in context of ERO.
AG
uPA
uPA
uPA
uPA
6. uPA populate *pathTrace* in *reserve.cf* and sent up the TREE.
5. uPA evaluates local policies based on reservation criteria and local connection segment only.

Figure 1 - TREE Mode – Reserve Phase.

In the reserve phase as shown in Figure 1, a uRA creates the initial *reserve* message and sends it to a (root) AG NSA that will act as the top-level root for this reservation. The root AG resolves a path using available policy information. The root AG then populates the *pathTrace* with its *nsaId* and the *connectionId* for the reservation but leaves the detailed path to be populated by the uPA(s). See Section 5 for the definition of the *pathTrace.*

The root AG then sends individual *reserve.rq* to each of the children NSA involved in the reservation, with the *pathTrace* element contained in the NSI header.

All other AG(s) along the reservation path from the root AG to the leaf uPA propagate the *pathTrace* untouched to children NSA.

When the *reserve.rq* reaches the uPA, the uPA will determine if resources are available and can be used based on local policy and:

a. If valid, the uPA will determine if there is a *pathTrace* in the request header. If there is, then it will populate the uPA's local path segment into the *pathTrace* element and put it in the NSI header of the *reserve.cf*. It will then return the *reserve.cf* back to the requesting AG.
b. If not valid the uPA will return a *reserve.fl* message.

AG(s) along the reservation path from leaf uPA to root AG aggregate the path segments within *pathTrace* elements from child NSA into more complete lists as the *reserve.cf* moves up the tree.

Once the root AG consolidates all the child *reserve.cf* messages it will end up with a complete end-to-end path which it stores against the reservation before sending the *reserve.cf* to the uRA with the *pathTrace* element in header for "information only." The uRA can use this *pathTrace* information to determine if the chosen path is satisfactory.

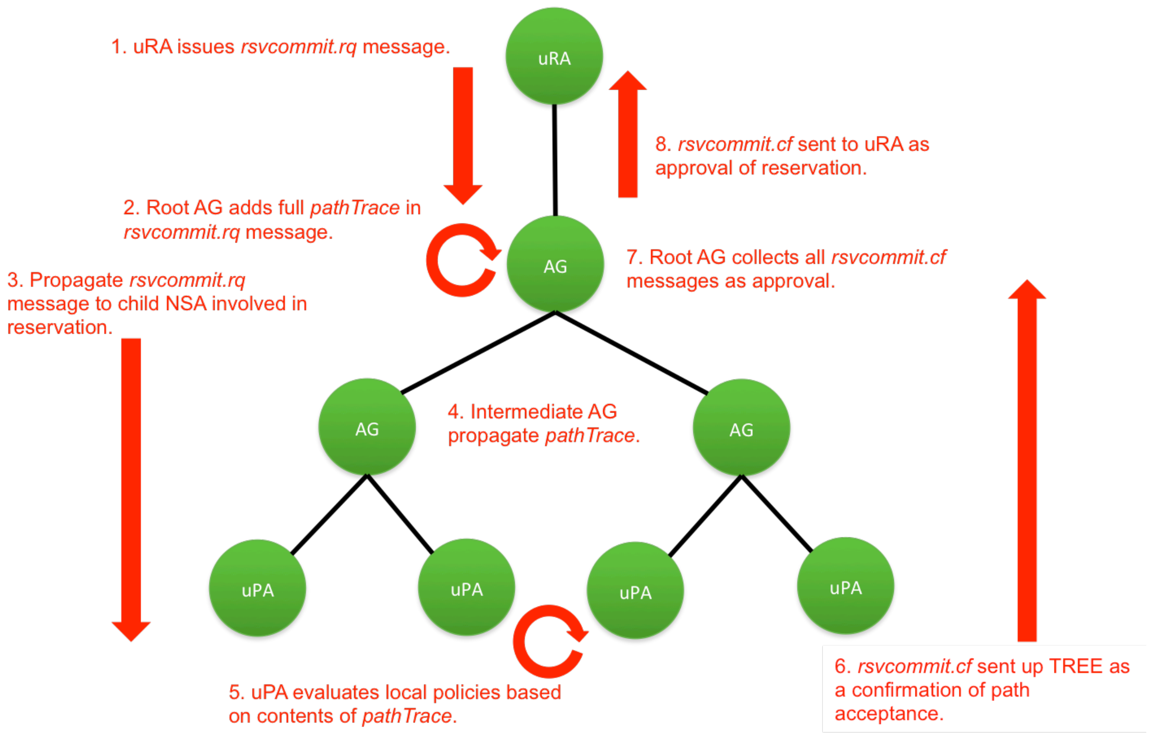
1. uRA issues *rsvcommit.rq* message.
uRA
8. *rsvcommit.cf* sent to uRA as approval of reservation.
2. Root AG adds full *pathTrace* in *rsvcommit.rq* message.
AG
7. Root AG collects all *rsvcommit.cf* messages as approval.
3. Propagate *rsvcommit.rq* message to child NSA involved in reservation.
AG
4. Intermediate AG propagate *pathTrace*.
AG
uPA
uPA
uPA
uPA
5. uPA evaluates local policies based on contents of *pathTrace*.
6. *rsvcommit.cf* sent up TREE as a confirmation of path acceptance.

Figure 2 - TREE Mode – Reserve Confirm Phase.

In the reserve confirm phase as shown in Figure 2, when the uRA issues the *reserveCommit.rq*, the root AG will add the completed *pathTrace* into the NSI header which then gets propagated to each uPA involved in the connection.

AG(s) along the path propagate the *pathTrace* unmodified, and may store the trace against the connection for additional debug information if so desired.

When the *reserveCommit.rq* arrives at a uPA, it can evaluate local routing policies against the full path within the *pathTrace* (if desired) and take the following actions:

- If the proposed path violates any local policies, the uPA issues a *reserveCommit.fl* with an appropriately populated *serviceException*.
- If the proposed path does not violate any of the uPAs policies, the reservation is committed and a *reserveCommit.cf* is issued.

The *reserveCommit.cf* and/or *reserveCommit.fl* messages traverse from the uPA(s) to the root AG. If all the uPAs approve their local paths, it will result in an end-to-end committed reservation. Otherwise there will be one or more failed segments which will result in a termination of the reservation (i.e. *term.rq* being sent) by either the uRA or root AG:

- If a policy was violated, the root AG will inform the uRA of the reservation failure due to the policy error, with the uRA having the option to abort the reservation as per the standard and try again at a later time.

OR

- The root AG can take corrective action itself, aborting existing path segments and using the learned policy information to compute an alternative path.

4.2 CHAIN Solution

This section describes how policy is implemented in the case of a CHAIN signaled Connection reservation. The NSI models for signaling and path finding are documented in detail in the NSI informational document GFD.217 Network Service Interface Signaling and Path Finding [6]. In the CHAIN solution each NSA associated with network resources is both an AG and uPA, with the AG capable of performing path finding and message forwarding, and the uPA managing associated network resources. For the purpose of this description we use the term NSA as the combined AG/uPA capabilities, and uRA for the standard initiator of the reservation request.

In the context of NSI, the CHAIN mode can be treated as a subset of the TREE mode, and as such the policy enforcement for the two modes are similar. However it should be noted that typically in TREE mode, an AG will first ensure its “local” uPA approves of the request before propagating it to the next-hop AG. Figure 3 - and Figure 4 illustrate the *reserve* and *reserveCommit* phases in the CHAIN mode respectively.

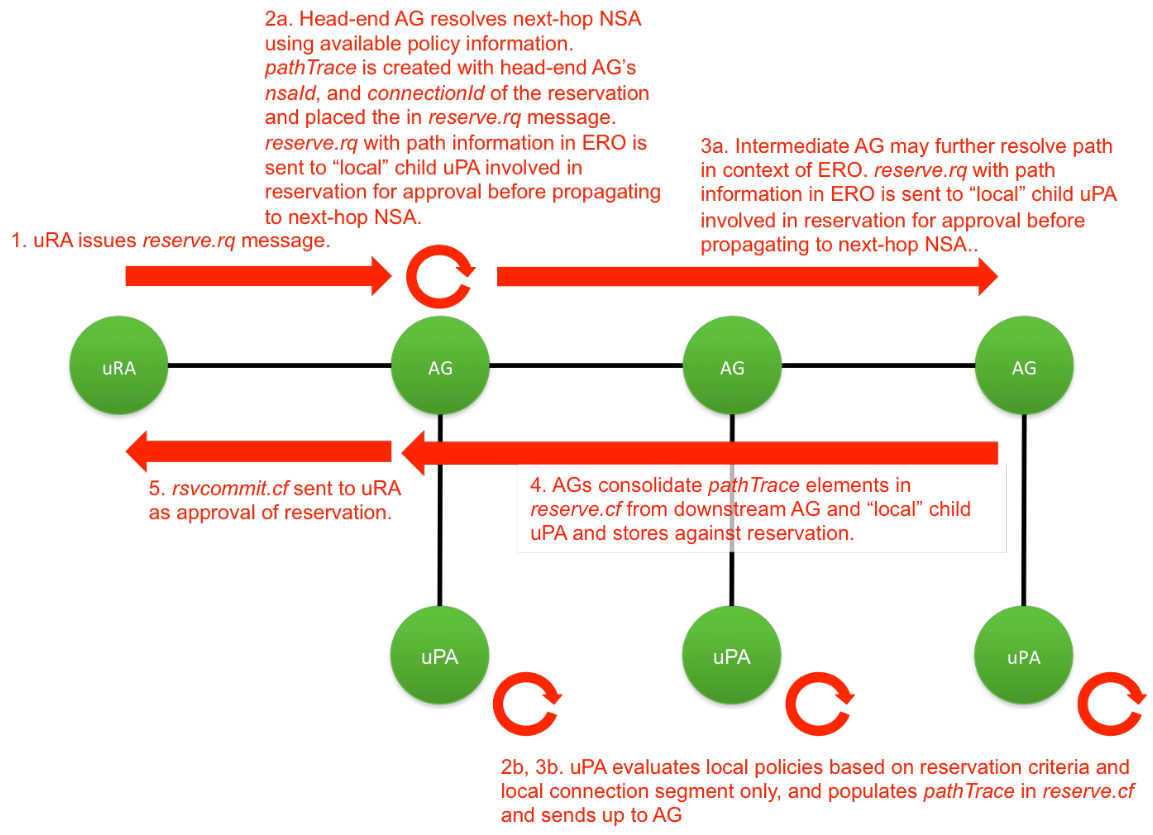
2a. Head-end AG resolves next-hop NSA using available policy information. *pathTrace* is created with head-end AG's *nsaId*, and *connectionId* of the reservation and placed the in *reserve.rq* message. *reserve.rq* with path information in ERO is sent to "local" child uPA involved in reservation for approval before propagating to next-hop NSA.
3a. Intermediate AG may further resolve path in context of ERO. *reserve.rq* with path information in ERO is sent to "local" child uPA involved in reservation for approval before propagating to next-hop NSA..
1. uRA issues *reserve.rq* message.
uRA
AG
AG
AG
5. *rsvcommit.cf* sent to uRA as approval of reservation.
4. AGs consolidate *pathTrace* elements in *reserve.cf* from downstream AG and "local" child uPA and stores against reservation.
uPA
uPA
uPA
2b, 3b. uPA evaluates local policies based on reservation criteria and local connection segment only, and populates *pathTrace* in *reserve.cf* and sends up to AG

Figure 3 - CHAIN Mode – Reserve Phase

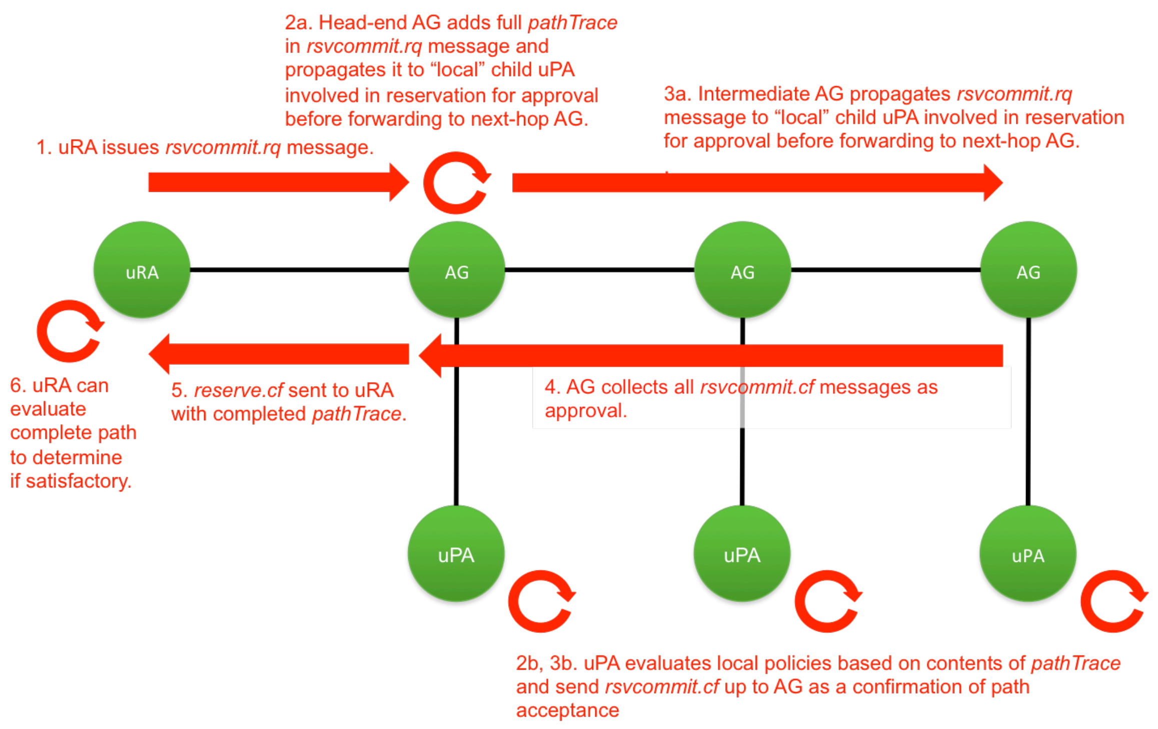
2a. Head-end AG adds full *pathTrace* in *rsvcommit.rq* message and propagates it to "local" child uPA involved in reservation for approval before forwarding to next-hop AG.
3a. Intermediate AG propagates *rsvcommit.rq* message to "local" child uPA involved in reservation for approval before forwarding to next-hop AG.
1. uRA issues *rsvcommit.rq* message.
uRA
AG
AG
AG
6. uRA can evaluate complete path to determine if satisfactory.
5. *reserve.cf* sent to uRA with completed *pathTrace*.
4. AG collects all *rsvcommit.cf* messages as approval.
uPA
uPA
uPA
2b, 3b. uPA evaluates local policies based on contents of *pathTrace* and send *rsvcommit.cf* up to AG as a confirmation of path acceptance

Figure 4 - CHAIN Mode – Reserve Commit Phase

5. pathTrace Definition

To help facilitate the communication of path trace information within the NSI reserve message exchange we introduce a new schema element called "*pathTrace*" into the NSI Header element.

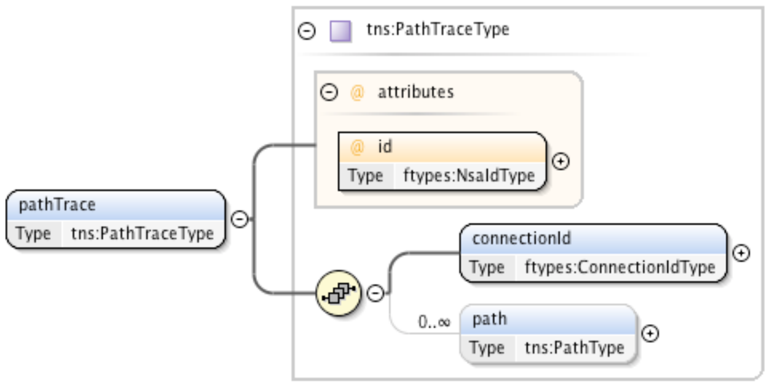

Figure 5 - The *pathTrace* element.

The *pathTrace* element contains the following parameters:

Parameter	Type	Mandatory	Description
id	Attribute: NsaIdType	True	This attribute contains the NSA identifier of the root or head-end aggregator NSA. This identifier is added for traceability back to the NSA performing initial path resolution.
connectionId	Element:	True	The connection identifier for the reservation

	ConnectionIdType		in the context of the NSA identified by the id attribute. This identifier is added for traceability to the originating reservation associated with this path.
path	Element: PathType	False [0..unbounded]	Contains individual path segments composing the overall detailed transport plane path for the reservation associated with *connectionId*. This path element is used by uPA to perform routing policy enforcement.

5.1 PathType definition

PathType is a type definition for path information within the path trace element. A *path* consists of a sequence of individual path segments.

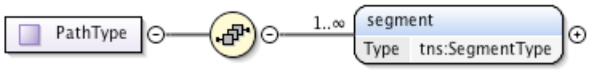

Figure 6 - The PathType definition.

The *PathType* structure contains the following parameters:

Parameter	Type	Mandatory	Description
segment	Element: SegmentType	True [1..unbounded]	A segment in the overall connection path.

5.2 SegmentType definition

SegmentType is a type definition for an individual path segment within the overall connection path. This represents the connection segment for a single domain.

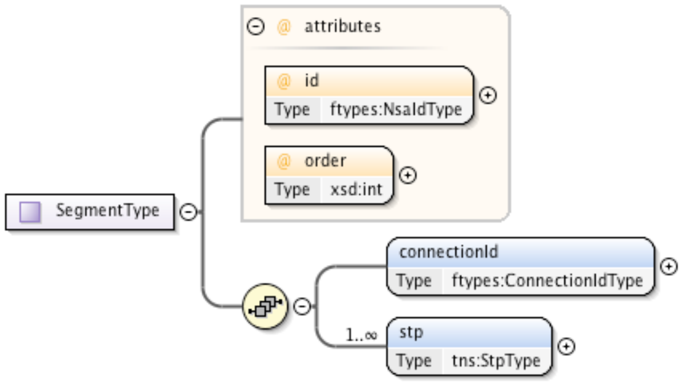

Figure 7 - The SegmentType definition.

The *SegmentType* structure contains the following parameters:

Parameter	Type	Mandatory	Description
Id	Attribute: *NsaIdType*	True	The NSA identifier for the uPA associated with this path segment.
order	Attribute: int	True	The sequence order of the connection in the overall path. As new segments are added in order to the path trace this value is increased

			by one over the last added segment.
connectionId	Element: *ConnectionIdType*	True	Connection identifier for the reservation in the context of the NSA identified by the "id" attribute. This identifier is added for traceability to the uPA reservation segment associated with this path.
stp	Element: *StpType*	True [1..unbounded]	An ordered list of STP identifier within this domain that are part of the overall path the connection segment. This list will usually only contain ingress and egress STP, however, more flexibility is provided to model internal STP as well.

5.3 *StpType* definition

An extended type definition for an STP identifier that includes ordering capabilities. This type extends from the based point-to-point service type's *StpIdType* definition (a simple string valued type) to include an *order* attribute.

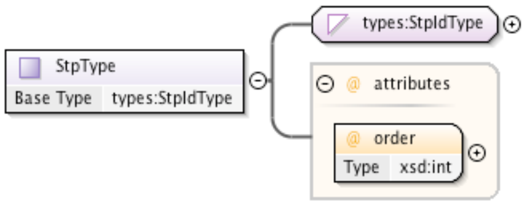

Figure 8 - The *StpType* definition.

The *StpType* structure contains the following parameters:

Parameter	Type	Mandatory	Description
Order	Attr	True	The order of this STP within a sequence of STP.

5.4 Example

The following is path trace example for a connection with three segments running through networks "Aruba", "Curacao", and "Bonaire". In this example, the user requests a reservation from "*stpA*" in network Aruba through to "*stpZ*" in network Bonaire using the Grenada uRA "*urn:ogf:network:grenada.net:2013:nsa-requester*".

reserveRequest()
The Grenada uRA NSA creates an NSI reserve.rq message with the requested reservation criteria and passes it on to the associated root aggregator NSA "*urn:ogf:network:grenada.net:2013:nsa-aggr*" for processing.

The root AG receives the NSI reserve.rq message and performs path finding. The root aggregator sees there is no *pathTrace* element in the NSI header so adds the following to all child reservation requests:

```
<tns:pathTrace xmlns:tns="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace"
        id="urn:ogf:network:grenada.net:2013:nsa-aggr">
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
</tns:pathTrace>
```

The root AG populates *pathTrace.id* attribute is populated with its own NSA identifier as the NSA performing rooting the reservation. The *pathTrace.connectionId* element is also populated with the connection identifier associated with this reservation for the purpose of traceability and troubleshooting (all NSA involved in the reservation will know the root AG and the root *connectionId*). The *reserve.rq* is propagated to all child NSA involved in the reservation. Intermediate NSA store the received *pathTrace* and propagate untouched to child NSA involved in the reservation.

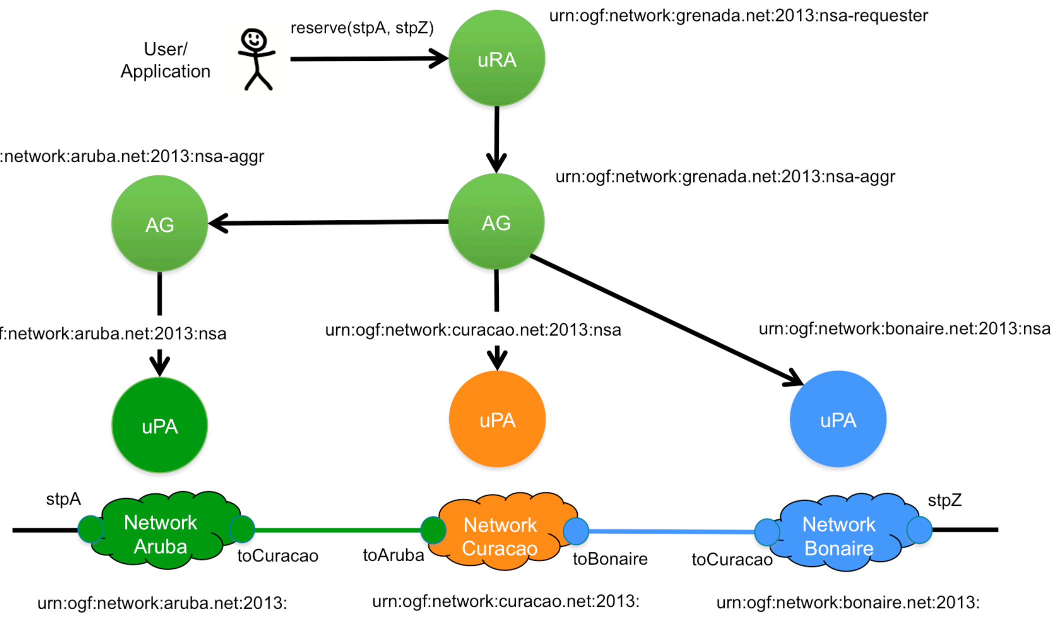
User/ Application
reserve(stpA, stpZ)
uRA
urn:ogf:network:grenada.net:2013:nsa-requester
urn:ogf:network:aruba.net:2013:nsa-aggr
AG
AG
urn:ogf:network:grenada.net:2013:nsa-aggr
urn:ogf:network:aruba.net:2013:nsa
urn:ogf:network:curacao.net:2013:nsa
urn:ogf:network:bonaire.net:2013:nsa
uPA
uPA
uPA
stpA
Network Aruba
toCuracao
toAruba
Network Curacao
toBonaire
toCuracao
Network Bonaire
stpZ
urn:ogf:network:aruba.net:2013:
urn:ogf:network:curacao.net:2013:
urn:ogf:network:bonaire.net:2013:

Figure 9 - Example network for *pathTrace*.

reserveConfirmed()

Each uPA will add its local path segment to the *pathTrace* in the NSI header of the *reserve.cf* message. As an example, uPA "*urn:ogf:network:aruba.net:2013:nsa*" will generate the following *pathTrace* element. It should be noted that each uPA will set the *order* to zero (0) for its local *segment* element.

```
<tns:pathTrace xmlns:tns="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace"
       id="urn:ogf:network:grenada.net:2013:nsa-aggr">
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <path>
        <segment id="urn:ogf:network:aruba.net:2013:nsa" order="0">
            <connectionId>urn:uuid:71d30ed8-f130-4522-a5c5-3d6d2e96fb1e</connectionId>
            <stp order="0">urn:ogf:network:aruba.net:2013::stpA?vlan=1790</stp>
            <stp order="1">urn:ogf:network:aruba.net:2013::curacao?vlan=1790</stp>
        </segment>
    </path>
</tns:pathTrace>
```

The uPA populates its NSA identifier in the "*id*" attribute of the <*segment*> element and the local connection identifier in the <*connectionId*> element. Continuing with the previous example, the uPA *"urn:ogf:network:curacao.net:2013:nsa"* will generate the following *pathTrace* element:

```
<tns:pathTrace xmlns:tns="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace"
       id="urn:ogf:network:grenada.net:2013:nsa-aggr">
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <path>
        <segment id="urn:ogf:network:curacao.net:2013:nsa" order="0">
            <connectionId>urn:uuid:aa4e5d68-4004-48c3-a99e-07bc7ae911fc</connectionId>
            <stp order="0">urn:ogf:network:curacao.net:2013::aruba?vlan=1790</stp>
            <stp order="1">urn:ogf:network:curacao.net:2013::internal1?vlan=200</stp>
            <stp order="2">urn:ogf:network:curacao.net:2013::internal2?vlan=560</stp>
            <stp order="3">urn:ogf:network:curacao.net:2013::bonaire?vlan=1790</stp>
        </segment>
    </path>
</tns:pathTrace>
```

Notice that internal STP are allowed in the *pathTrace* but they must be bound by external STP. Lastly, uPA "*urn:ogf:network:bonaire.net:2013:nsa*" will generate the following *pathTrace* element:

```
<tns:pathTrace xmlns:tns="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace"
       id="urn:ogf:network:grenada.net:2013:nsa-aggr">
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <path>
        <segment id="urn:ogf:network:bonaire.net:2013:nsa" order="0">
            <connectionId>urn:uuid:b1f0277d-63d5-4e5e-85e8-acd11a5baab9</connectionId>
            <stp order="0">urn:ogf:network:bonaire.net:2013::curacao?vlan=1790</stp>
            <stp order="1">urn:ogf:network:bonaire.net:2013::stpZ?vlan=1790</stp>
        </segment>
    </path>
</tns:pathTrace>
```

Each AG along the path from uPA to root AG will aggregate the *pathTrace* information from related child connection segments into a single *pathTrace* element returned in the *reserve.rq* message passed to its parent. If an AG has done additional path finding on the *reserve.rq* it MUST assemble the child path within the *pathTrace* element in topological order. The *order* number in the *segment* MUST start at zero (0) and MUST be incremented by one (1) for each new segment. The root AG of the reservation will complete the aggregation of child *pathTrace* segments and store the completed *pathTrace* element against the reservation for use in the *reserveCommit.rq*. The complete *pathTrace* element would appear as follows:

```
<tns:pathTrace xmlns:tns="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace"
        id="urn:ogf:network:grenada.net:2013:nsa-aggr">
    <connectionId>urn:uuid:59d6c0b2-a8e0-4583-ae8a-0fc84eb89f07</connectionId>
    <path>
        <segment id="urn:ogf:network:aruba.net:2013:nsa" order="0">
            <connectionId>urn:uuid:71d30ed8-f130-4522-a5c5-3d6d2e96fb1e</connectionId>
            <stp order="0">urn:ogf:network:aruba.net:2013::stpA?vlan=1790</stp>
            <stp order="1">urn:ogf:network:aruba.net:2013::curacao?vlan=1790</stp>
        </segment>
        <segment id="urn:ogf:network:curacao.net:2013:nsa" order="1">
            <connectionId>urn:uuid:aa4e5d68-4004-48c3-a99e-07bc7ae911fc</connectionId>
            <stp order="0">urn:ogf:network:curacao.net:2013::aruba?vlan=1790</stp>
            <stp order="1">urn:ogf:network:curacao.net:2013::internal1?vlan=200</stp>
            <stp order="2">urn:ogf:network:curacao.net:2013::internal2?vlan=560</stp>
            <stp order="3">urn:ogf:network:curacao.net:2013::bonaire?vlan=1790</stp>
        </segment>
        <segment id="urn:ogf:network:bonaire.net:2013:nsa" order="2">
            <connectionId>urn:uuid:b1f0277d-63d5-4e5e-85e8-acd11a5baab9</connectionId>
            <stp order="0">urn:ogf:network:bonaire.net:2013::curacao?vlan=1790</stp>
            <stp order="1">urn:ogf:network:bonaire.net:2013::stpZ?vlan=1790</stp>
        </segment>
    </path>
</tns:pathTrace>
```

The *reserve.cf* is passed to the uRA initiating the reservation with the completed *pathTrace* element in the NSI header.

6. Contributors

Chin Guok, ESnet
Tomohiro Kudoh, AIST
John MacAuley, ESnet
Guy Roberts, GEANT
Henrik Thostrup Jensen, NORDUnet

Hans Trompert, SURFnet

7. Glossary

Aggregator (AG)	The Aggregator is an NSA that has more than one child NSA, and has the responsibility of aggregating the responses from each child NSA.
Connection	A Connection is an NSI construct that identifies the physical instance of a circuit in the Transport Plane. A Connection has a set of properties (for instance, Connection identifier, ingress and egress STPs, capacity, or start time). Connections can be either unidirectional or bidirectional.
Connection Service (CS)	The NSI Connection Service is a service that allows an RA to request and manage a Connection from a PA.
Control and Management Planes	The Control Plane and/or Management Plane are not defined in this document, but follow common usage.
Discovery Service	The NSI discovery service is a web service that allows an RA to discover information about the services available in a PA and the versions of these services.
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 fiber
ero	An Explicit Routing Object (ero) is a parameter in a Connection request. It is an ordered list of STP constraints to be used by the inter-Network pathfinder.
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 Markup Language (NML)	The Network Markup Language is an XML based network resource description language developed in the OGF.
Network Resource Manager (NRM)	The Network Resource Manager is the entity that manages a network, typically this will be the equipment vendor’s network management system.
Network Services	Network Services are the full set of 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 Services Framework (NSF)	The Network Services framework describes an NSI message-based platform capable of supporting a suite of Network Services such as the Connection Service and the Topology Service.
Network Service Interface (NSI)	The NSI is the interface between RAs and PAs. The NSI defines a set of interactions or transactions between these NSAs to realize a Network Service.
NSI Message	An NSI Message is a structured unit of data sent between an RA and a PA.
NSI Topology	The NSI Topology defines a standard ontology and a schema to describe network resources that are managed to create the NSI service. The NSI Topology as used by the NSI CS (and in future other NSI services) is described in: GWD-R-P: Network Service Interface Topology Representation [3].
Open Grid Forum (OGF)	The OGF is the Standards Developing Organization (SDO) that is home to the NSI Standards.
Provision	Provisioning is the process of requesting the creation of the physical instance of a Connection in the data plane.
Requester/Provider Agent (RA/PA)	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 Agent (RA). When an NSA realizes a service, it is called a Provider Agent (PA). A particular NSA may act in different roles at different interfaces.
Service Demarcation Point (SDP)	Service Demarcation Points (SDPs) are NSI topology objects that identify a grouping of two Edge Points at the boundary between two Networks.
Service Termination Point	Service Termination Points (STPs) are NSI topology objects that identify the Edge

(STP)	Points of a Network 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.
Reserve	When a Provider Agent receives (and then confirms) a Connection Reservation request the Provider Agent then holds the resources needed by the Connection.
Terminate	Terminating is the process which will completely remove a Reservation and Release any associated Connections. This term has a formal definition in the CS state-machine.
Topology Distribution Service	The NSI Topology distribution Service is a service that allows the NSI topology to be exchanged between NSAs.
Transport Plane	The Transport Plane refers to the infrastructure that carries the physical instance of the Connection, e.g. the Ethernet switches that deliver the circuit.
Ultimate PA (uPA)	The ultimate PA is a Provider Agent that has an associated NRM.
Ultimate RA (uRA)	The ultimate RA is a Requester Agent is the originator of a service request.
XML Schema Definition (XSD)	XSD is a schema language for XML.
eXtensible Markup Language (XML)	XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

8. Security Considerations

Security considerations are dealt with in Open Grid forum GWD-R draft-trompert-gwdi-nsi-aa-v04, NSI Authentication and Authorization [5].

No additional security issues have been raised.

9. 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.

10. 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.

11. Full Copyright Notice

Copyright (C) Open Grid Forum (2008–2017). 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.

12. References

1. ITU-T, G.805: Generic functional architecture of transport networks
2. Open Grid forum GFD.212 Network Service Interface Connection Service, v2.0
3. IETF RFC 4655, "A Path Computation Element (PCE)-Based Architecture", http://www.rfc-editor.org/rfc/rfc4655.txt
4. Open Grid forum GFD.213, Network Services Framework v2.0, http://www.ogf.org/documents/GFD.213.pdf
5. Open Grid forum GWD-R draft-trompert-gwdi-nsi-aa-v04, NSI Authentication and Authorization, https://redmine.ogf.org/dmsf_files/13424
6. Open Grid forum GFD.217 Network Service Interface Signaling and Path Finding https://www.ogf.org/documents/GFD.217.pdf

13. Appendix A: Policy use cases

This section lists policy requirements that have been identified by network providers that wish to implement the NSI CS protocol in their network.

13.1 Link ownership

In some situations a network's demarcation point extends beyond the edge of their network to include the link that connects to the far end network. In Figure 10 below we see Link A is owned by Network A and Link B is owned by Network B.

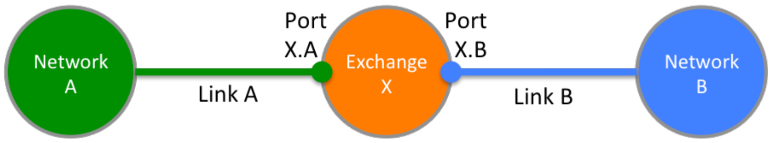

Figure 10 Link ownership

In this case, the Exchange X should not be able to initiate a Connection on port X.A without the approval of Network A, and similarly, it should not initiate a Connection on port X.B without the approval of Network B. If Exchange X was to make a Connection between port X.A to X.B without the explicit consent from either Networks A and B, this could block bandwidth in Networks A and B, constituting a denial of service attack. The NSI policy framework should include a mechanism that would allow policy to be instituted in Exchange X to prevent this.

13.2 Transit policies

NSI CS transit policies may be applied by the Network to determine what traffic may transit based on source and/or destination Network of a Connection. For example in Figure 11, Network A may apply a local policy allowing Network networks A1-A3 to connect to each other and to the networks B and C without restriction while preventing transit connections between Network B and C.

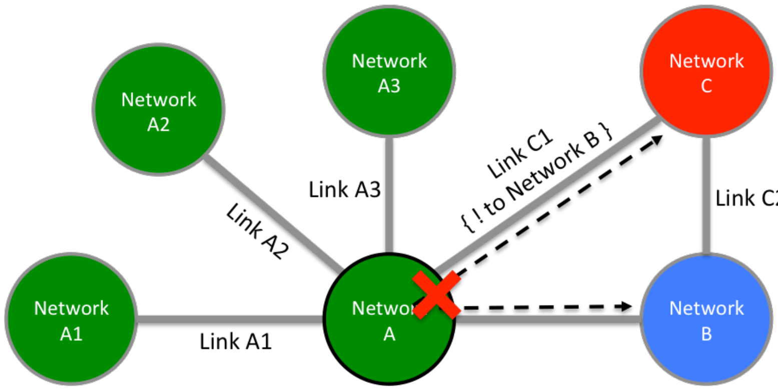

Figure 11 Transit policies

13.3 Link restrictive transit policies

NSI CS policy may be applied on a link restriction basis. In Figure 12 below, Exchange X has a local policy that allows traffic between Network A and Network C via Links A and C1, but not using Links A, B, and C2.

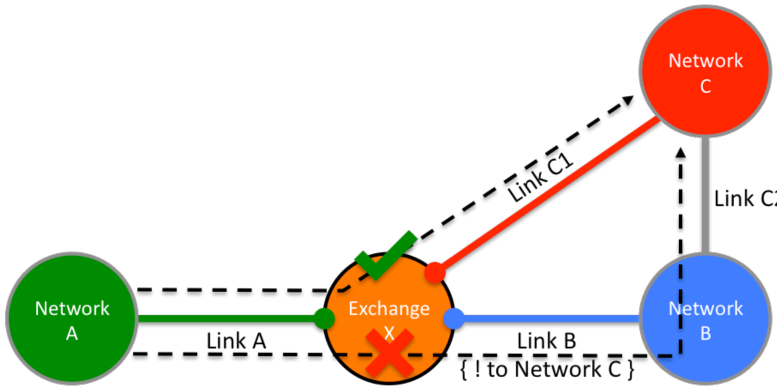

Figure 12 Link restrictive transit policies

13.4 Resource restrictive transit policies

An example of a resource based transit policy is shown in Figure 13 below. In this case Exchange X allows a maximum bandwidth between Network A and Network C independent of the path.

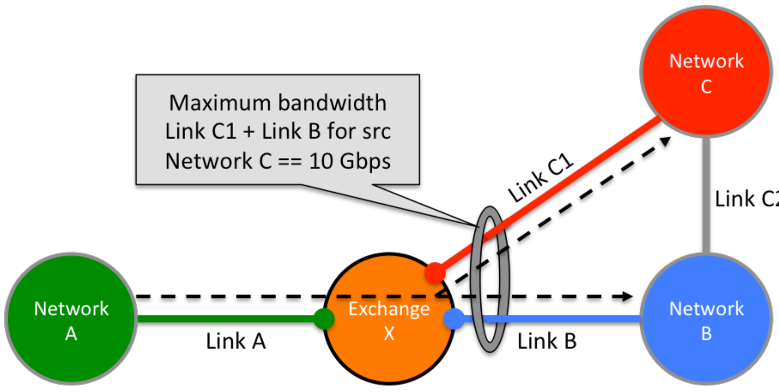

Figure 13 Resource restrictive transit policies

13.5 Resource allocation policies

NSI CS policy can be used to restrict the path a Connection may take through the network based on the segmentation of allocated resources within the network to specific groups of users. In the example Figure 14 below, Link C1 is tagged for use by user group 1 only, while all other links are tagged for cooperative sharing. Only users that are members of group 1 may use link C1 in reservation requests.

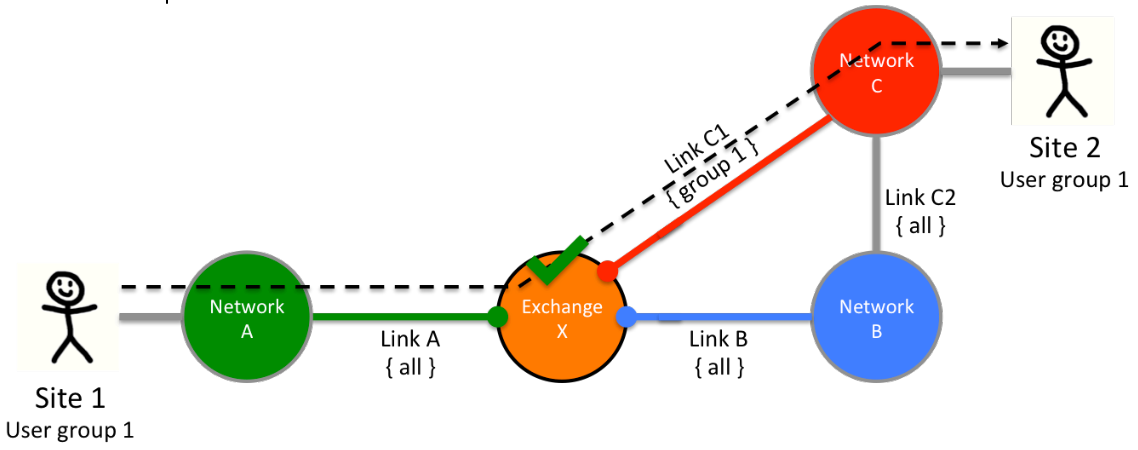

Figure 14 Resource allocation policies

13.6 Preferred link policies

NSI CS policy can be used to prefer one link to another. In Figure 15 below, due to transit policies within Network C, transit traffic from Network A to Network D should use preferred Link D1 until capacity is consumed, while transit traffic between Network B and Network D should use preferred Link D2.

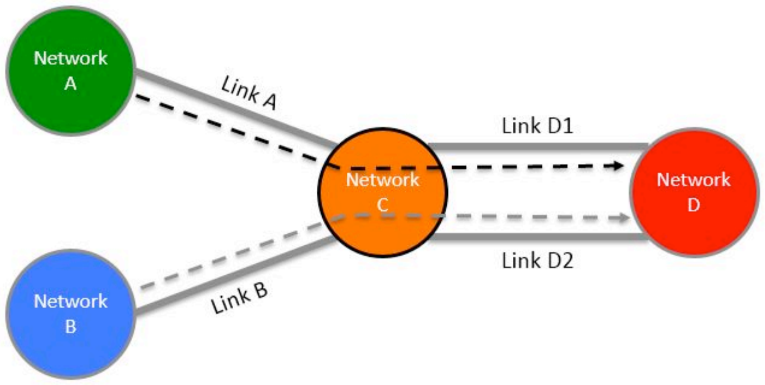

Figure 15 Preferred link policies

In the situation depicted below in Figure 16, there are two equal distant paths between Network A and Network D; however, Network A has a preferred link policy such that any traffic between Network A and D must transit Network C via Link AC, the preferred route.

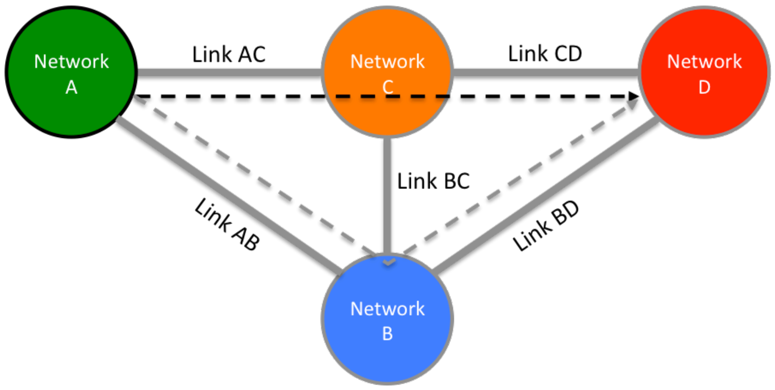

Figure 16 Preferred path policies

13.7 Selective reachability

This type of policy restricts the traffic that can terminate in a network, along with transit rules for associated traffic

- A should be able to reach B+D
- B should be able reach all.
- C should be able to reach B+D
- D should be able to reach A+B+C.

That is, A and C are not exchanging traffic, but C provides transit to D, and D has transit to A through B and C

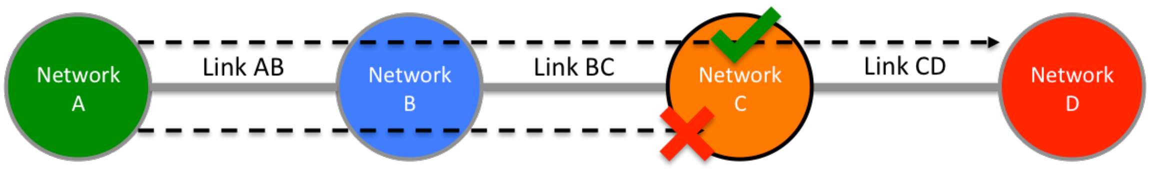

Figure 17 Selective reachability

13.8 Policy Requirements

The examples of policy-based routing decisions described in the previous section introduce new requirements for path finding and the process by which local policy is enforced:

> *A mechanism MUST be supported in the NSI CS protocol so that a uPA associated with any Network understands the proposed end-to-end path so that it can make a decision on whether to approve the local reservation segment.*
>
> *Where a reservation is rejected for policy reasons, there MUST be a mechanism to report this back to the uRA so that it can terminate the full end-to end reservation.*
>
> *The NSI CS policy framework MUST assume that the responsibility for terminating Connections is held with the uRA, this allows the uRA to make decisions on what to do under the failure scenario.*
>
> *Where a reservation is rejected for policy reasons, there MUST be a mechanism to provide information back to the requester about the reason for the rejection so that the requester can try alternative Connection requests.*
>
> *The NSI CS policy framework MUST use the existing NSI protocol and behaviors where possible to reduce protocol churn and will incorporate the solution in the existing NSI protocol framework.*
>
> *The NSI CS policy solution MUST support both source and hop-by-hop path computation models in TREE and CHAIN mode.*
>
> *Network routing policy information MUST be made available to pathfinders for more effective route resolution.*
>
> *The NSI CS policy solution MUST assume that the act of holding resources associated with a reservation can be done without approval, but the committing of a reservation will not occur without all uPA path segments confirmed.*
>
> *The NSI CS policy framework MUST assume that the service plane is trusted and that the NSA will behave correctly in the context of the NSI protocol.*

14. Appendix B: Schema extensions for pathTrace

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    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.

    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.

    Copyright (C) Open Grid Forum (2009-2016). 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.

    Open Grid Forum NSI Connection Services Protocol v2.0 - Path trace extensions.
-->
<xsd:schema targetNamespace="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:types="http://schemas.ogf.org/nsi/2013/12/services/types"
    xmlns:ftypes="http://schemas.ogf.org/nsi/2013/12/framework/types"
    xmlns:tns="http://schemas.ogf.org/nsi/2015/04/connection/pathtrace">

    <xsd:annotation>
        <xsd:appinfo>OGF NSI CS Path Trace 2015-04-30</xsd:appinfo>
        <xsd:documentation xml:lang="en">
            This is an XML schema document describing the path trace extension
            to the OGF NSI Connection Services protocol version 2.0.  This
            defines a new header element used to model the full path of a
            reservation.
        </xsd:documentation>
    </xsd:annotation>

    <!-- Import the common NSI framework types. -->
    <xsd:import namespace="http://schemas.ogf.org/nsi/2013/12/framework/types"
        schemaLocation="ogf_nsi_framework_types_v2_0.xsd"/>

    <!-- Import the common NSI service types. -->
    <xsd:import namespace="http://schemas.ogf.org/nsi/2013/12/services/types"
        schemaLocation="ogf_nsi_services_types_v2_0.xsd"/>

    <xsd:element name="pathTrace" type="tns:PathTraceType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The header element modeling a connection path through the network.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:element>

    <xsd:complexType name="PathTraceType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Type definition for the path trace information.

                Attributes:

                id - In the case of TREE, this attribute contains the NSA
                identifier of the aggregator that performed path resolution,
                while for CHAIN, it is the identifier of the first (head-end)
                NSA allocating transport plane resources.  This identifier is
                added for traceability back to the NSA performing initial
                path resolution.
                Elements:

                connectionId - Connection identifier for the reservation in the
                context of the NSA identified by the "id" attribute.  This
                identifier is added for traceability to the originating reservation
                associated with this path.

                path - In the case of a "source" routing the NSA doing path
                resolution includes the complete path, while for “hop-by-hop”
                each NSA allocating data plane resources adds its path
                segment on the outbound reserve request until the full path is
                completed by the tail end NSA.  This path element is used by
                uPA to perform routing policy enforcement.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="connectionId" type="ftypes:ConnectionIdType" />
            <xsd:element name="path" type="tns:PathType" minOccurs="0"
                     maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute   name="id" type="ftypes:NsaIdType" use="required" />
    </xsd:complexType>

    <xsd:complexType name="PathType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Type definition for path information within the trace. A path
                consists of a sequence of path segments.

                Elements:

                segment - A single segment in the overall connection path.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="segment" type="tns:SegmentType" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="SegmentType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Type definition for a path segment within the overall
                connection path.  This represents the connection segment for a
                single domain.

                Attributes:

                id - The NSA identifier associated with this path segment.

                order - The sequence order of the connection in the overall
                path.  As new segments are added to the path trace this value
                is increased by one over the last added segment.

                Elements:

                connectionId - Connection identifier for the reservation in the
                context of the NSA identified by the "id" attribute.  This
                identifier is added for traceability to the uPA reservation
                associated with this path.

                stp - A list of STP identifier within this domain that are part
                of the overall path ordered by their sequence within the
                connection.  This list will usually only contain ingress and
                egress STP, however, more flexibility is provided to model
                internal STP as well.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="connectionId"  type="ftypes:ConnectionIdType" />
            <xsd:element name="stp"  type="tns:StpType" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute   name="id"    type="ftypes:NsaIdType" use="required" />
        <xsd:attribute   name="order" type="xsd:int" use="required" />
    </xsd:complexType>

    <xsd:complexType name="StpType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                An extended type definition for an STP identifier that includes
                ordering capabilities.

                Attributes:

                order - The order of this STP within a sequence of STP.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension  base="types:StpIdType">
                <xsd:attribute  name="order"  type="xsd:int"  use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:schema>
```