

NSI Reachability-based Topology

May, 2014


Henrik Thostrup Jensen, NORDUnet, htj at nordu dot net

This document outlines a solution for NSI topology based on reachability information, as opposed to full description for each network. This has the advantage of minimizing the information that has to be distributed. The overall approach is similar to that of RIP used in ARPAnet, i.e., based on the Bellman-Ford algorithm. This approach is well studied, and the advantages and disadvantages are well known.






NML

The network modelling language (NML) introduces several concepts which are problematic:

- Decouples ports and networks (topologies), such that an NSA must explicitly know relations in order to do path finding.

This means that all changes must distributed, even if irrelevant for path-finding which is arguably the majority of such updates. This in turn means complex distribution, potential scaling problems, and more complexity in distribution and path-finding. It also prevents dynamical STPs, where resources are created on the fly from the STP definition. 

Further:

- Uses identifiers that cannot practically be enforced causing potential security problems 
- Bidirectionality is glued on, despite bidirectional ethernet being the de-facto technology. 
- Leaves bandwidth and policies as an exercise for the user.
- Ignores hard-learned lessons from RIP & BGP 
- No aggregation of network information 
- Makes it difficult to announce transit policies
- Makes it difficult to announce different reachability to different peers 
- Introduces new lingo, i.e., topology vs network



Note that NML, as such does not prevent different topologies per network; that is largely an NSIism).


Note that the solution presented in this document is not exempt from some of the above issues. Trade-offs must be made.

Topology Reachability Design

The system was originally designed by NORDUnet & SURFnet, with input from the AutoBAHN team. It is currently implemented in AutoBAHN, OpenNSA, and SURFnet BoD. The current system is not dependant on NML, but designed in such a way that it can be compatible with NML.

The system introduces two basic mechanisms:

- A structure in the STP identifiers, such that the network and port names be inferred from the identifier. This is done by splitting the urn base part at the last “:”. For NML, this means that the full STP (without label part) is the port id, and the prefix is the topology id. 
- Announcing reachable networks with a cost factor. Currently done with NML topology ids in the NSA description document. The current cost factor is network hops. NSAs learn the reachability of peers, which in turn allows them to announce that reachability - with an added cost factor - to their peers. 

Further the system assumes that control and data plane peerings go together. 
This becomes key in the path-finding process, which is explained later.

STP Example

The STP:

```
urn:ogf:network:bonaire.net:topology:arb-in?vlan=1780
```

is split into a URN and label, i.e.,

```
urn:ogf:network:bonaire.net:topology:arb-in  
vlan=1780
```

Hence the network part is (topology id in NML lingo):

```
urn:ogf:network:bonaire.net:topology
```

The NML port id is the full STP URN.

This allows an NSA to infer the network id immediately from an STP without looking up the port-topology relation in a big NML structure.

Reachability Example

```
<nsi:nsa id="urn:ogf:network:aruba.net:nsa" ...>
  <networkId>urn:ogf:network:aruba.net:topology</networkId>
  <interface>
    <type>application/vnd.ogf.nsi.cs.v2.provider+soap</type>
    <href>http://scandium:4080/NSI/services/CS2</href>
  </interface>
  <other>
    <gns:TopologyReachability>
      <Topology cost="1" id="urn:ogf:network:bonaire.net:topology" />
      <Topology cost="2" id="urn:ogf:network:curacao.net:topology" />
    </gns:TopologyReachability>
  </other>
</nsi:nsa>
```

The other element in the NSI discovery section it used to announce which networks / topologies is reachable from that NSA.

Construction of Reachability Information

Each NSA is bootstrapped manually to a number of peers, i.e., the URL of the discovery file is provided to it. We assume that control and data plane peering go together.

Initially, an NSA will announce its topology ids under the network id elements in the discovery file, e.g.:


```
<networkId>urn:ogf:network:aruba.net:2013:topology</networkId>
```

With regular intervals, the NSA will poll the discovery file of its peering NSAs to discover which networks the NSA is managing and which can be reached. The NSA will then update its reachability information with that. The NSA has a cost for each peering NSA, which is added to the costs of the network listed under the reachability. This list is then compared with the existing reachability information, and if any lower costs are discovered for network, the reachability information is updated accordingly.

Path Finding

The process of path finding is fairly straightforward. A request will fall into one of the following three categories:

- Having neither source or destination in the local network. In this case the NSA find the topology of the source / destination with the lowest cost, and forwards the request as-is towards that NSA (this may require multiple hops). Note: It may be a good idea to always forward to the source, as to minimize the chances for a loop to occur.
- The source and destination are both within the local network. Here the NSA simply sets up the local connection
- The source or (exclusive) destination is in the local network. Here the NSA reserves the local link towards the topology with the lowest reachability to the destination topology and issues a reserve request to the NSA responsible for the demarcating network.

This is the typical chaining process, and assumes that control and data plane follow each other. 

Note that the NSA should still prune the path to account for underspecified STPs and vlan/swapping, which may in some cases make it easier to setup the remote link first.

Connection Traces

To facilitate provenance of connections (where does it come from) and to provide a mechanism for loop detection, a list of the parent connections is added to the NSI header under the any element.

The connection trace provides the information needed to identify the parent connection(s), a feature needed in order to escalate issues to between domains. Something that is otherwise quite difficult in NSI. When combined with the chaining model, this provides a domain the capability to list all parent and children (via query), greatly aiding multi-domain debugging and escalation procedures.

The connection trace is required only for the reserve request.

To enable this trace, a scheme for addressing connections is introduced: NSA URN + : + connection id. Example:

NSA: `urn:ogf:network:aruba.net:nsa`

Connection Id: `AR-Tfe07c58e3fff`

Connection URN: `urn:ogf:network:aruba.net:nsa:AR-Tfe07c58e3fff`

XML snippet:

```
<gns:ConnectionTrace xmlns:gns="http://nordu.net/namespaces/2013/12/gnsbod">
  <gns:Connection index="0">urn:ogf:network:aruba:1:nsa:AR-Tfe07c58e3</gns:Connection>
```

```
<gns:Connection index="1">urn:ogf:network:bonaire:1:nsa:B0-s7780</gns:Connection>
<gns:Connection index="2">urn:ogf:network:curacao:1:nsa:CU-1234</gns:Connection>
</gns:ConnectionTrace>
```

gns namespace uri:

<http://nordu.net/namespaces/2013/12/gnsbod>

Limitations

- Announces reachability for NSA and not per-topology. This would be an obvious thing to change in the future.

However per topology/network still does not account for different ports having different reachability. However full port-to-port reachability is perhaps a tad to complex.

- Still not possible to announce different reachability to different peers. This would require some form of control-plane link state, which NSI is not well geared towards.
- Relies on NML to figure out if VLAN/Label swapping is possible. This would be fairly easy to add to a minimal topology description.
- The count-to-infinity problem (well-studied in text-books)

The Bellman-Ford algorithm propagates new information quite well through the network, but has a problem with stale information not leaving the network. We have run into this issue with three networks. It is easily overcome by discarding entries over a certain cost or by manually blacklisting stale information. The count-to-infinity problem allows control-plane loops to occur, hence it is strongly recommend to use something similar to connection traces to limit loops.

- Explicit traffic engineering, such as directing flows over certain networks is still complicated (but only as much as it was with NML - or any other solution). A potential improvement to this situation is to allow topology ids in the ERO, allowing for loose hop specification over certain networks.

