# NSI @ TNC17

***Attendees:***

Jerry Sobieski ([jerry@sobieski.net](mailto:jerry@sobieski.net))

Richard Hughes-Jones [Richard.Hughes-Jones@geant.org](mailto:Richard.Hughes-Jones@geant.org)

Naegele-Jackson, Susanne (RRZE) [susanne.naegele-jackson@fau.de](mailto:susanne.naegele-jackson@fau.de)

Migiel de Vos [migiel.devos@surfnet.nl](mailto:migiel.devos@surfnet.nl)

Hoeft, Bruno (SCC) ([bruno.hoeft@kit.edu](mailto:bruno.hoeft@kit.edu))

Atsuko Takefusa ([takefusa@acm.org](mailto:takefusa@acm.org))

Guy Roberts [Guy.Roberts@geant.org](mailto:Guy.Roberts@geant.org)

Gerben van Malenstein ([Gerben.vanMalenstein@SURFnet.nl](mailto:Gerben.vanMalenstein@SURFnet.nl))

Hans Trompert ([Hans.trompert@surfnet.nl](mailto:Hans.trompert@surfnet.nl))

Michal Hažlinský [hazlinsky@cesnet.cz](mailto:hazlinsky@cesnet.cz)

Tomohiro Kudoh ([kudoh@nc.u-tokyo.ac.jp](mailto:kudoh@nc.u-tokyo.ac.jp))

Chin Guok [chin@es.net](mailto:chin@es.net)

David Schmitz LRZ

Antonio Jauo Francisco ANSP

Jeronimo Benzema

Michael Stanton RNP

Jose Rotenone RNP

Kiachi Ikeda KDDI

Zhonghni Li CERNET

Joe Mambretti StarLight

***Mintues:***

* Decision: we will continue the NSI working group going with at least 4 voice calls per year, two face-to-face meetings

To do list:

* Next meetings:
  1. NSI WG call in June to review the NSI topology
  2. Request a BoF at SuperComputing 2017? If not we can have a side meeting at SC17 in November to discuss NSI multi-point services. Note: the exhibitors can attend BoF. BoF deadline is end of July. Possible options for meeting room at SC: can book rooms for up to 2 hours, also we may have a meeting space on our stand.
  3. Guy to update NSI Redmine page and ogf.org page text and provide steering committee with a working group update.
  4. Design NSI logo.
  5. Henrik to send out link to document on REST implementation
  6. Tomohiro Kudoh has sent an update of state machine and tables.. should be incorporated into CS v2.1
  7. Guy to bring together all CS updates
  8. Michal to work on multipoint proposal.

## On-demand model

- Determine how we can support an on-demand model

  - initiated by Hans

  - consider if the ServiceDefinition can support an on-demand model service description

  - Alternatively it might be an error message if start time is not "now", etc

Next steps:

* Hans to work on a service definition for this
* Add a new error message parameter to allow interoperability with other service definitions
* Extent of work: start with a draft work plan. Should include manpower, timescales, objectives
* Priority?: can be done relatively easily, have a use case for Surfnet, could make implementation easier and increase adoption.

## REST interface

- There is interest in defining a REST interface

  - initiated by John

  - next step is to come up with a proposal, e.g. REST w JSON/XML

  - also gauge who would be willing to put effort into this

Next steps

* Extent of work: start with a draft work plan. Should include manpower, timescales, objectives
* A binding document will be needed to define how a REST version of NSI is different from the SOAP version.
* How can the optional parts of the SOAP header be converted to REST?
* Priority? significant manpower required to both define the REST version and code an implementation. Generally seen as desirable

## Multi-point service

- There is interest in defining a multi-point service

  - initiated by Chin

  - identify interested parties who can assign resources to come up with a proposal

Next steps

* Priority? GTS have a use case. Difficult part is making multi-domain support.
* Extent of work: The GTS team need to formally define their requirements. Broadcast or multicast?
* Extent of work: GTS team should review the multi-point service proposal submitted by Miyamoto. This has been distributed on the NSI mailing list.
* Extent of work: GTS team to carry out an evaluation of options for the service
* Then need to allocate manpower to prepare a work plan.

## Unidirectional circuits

- Unidirectional and asymmetric (bandwidth) bidirectional circuits

  - initiated by Jerry

  - not urgent and can delay working on this

Next steps

* Put on ice for the time being

## NML topology extensions to show resource utilization

- NML topology to reflect resource usage (including specifics of each reservation)

  - initiated by John

  - need to understand the use case and privacy concerns

Next Steps

* John and Hans should define a use case.
* Use case: what are the objectives here? Do you want find out who is using what capacity? Or just find out how much capacity is reserved and how much is free?

# Henrik’s comments

## Provision/Release/Terminate

At a point in time we removed the provisionFailed, releaseFailed, and terminateFailed messages. I think this was with the reason that these actions weren't really allowed to fail. This is mostly true, with the exception that with security checks the requests can fail. There is no way to signal this back to the client AFAICT.

Action: changed provisioning state machine and terminating state machine, for failure in ‘ing states.

## State machine

Due to the above issue, aggregators can be stuck in "Provisioning" and "Releasing", with agents further down not wanting to move into provisioning due to missing credentials. I have solved this by adding loop transitions, so that the following are possible:

Provisioning -> Provisioning, on provision request

Releasing -> Releasing, on release request

A second issue is that on abort/timeout it is possible for the state machine to enter a state where it is not possible to see if resources are allocated or not (rather interesting, as this should really be the main purpose of the state machine - I think we focussed too much in messages, and not resource lifecycle).

I solved this by adding an internal flag to indicate if resources are allocated or not.

## Security/Identity flaws

The NSA identities are not tied to subject names in X.509 certificates, and there is no way to authenticate them. Inventing new ways of identifying services/hosts that isn't strictly tied into a CA or DNS is a bad idea.

The whole user/group/organization SAML thing we have is pretty much useless as there is not good way to authenticate them. The only proper security mechanisms I can see is X.509 certificates and tokens ala OAuth2.

## No notification mechanism

Notification are only provided the requester. All other have to issue query calls. This makes it impossible to get continuous updates (long poll / callbacks). This is very relevant for administration tools / portals.

## Security headers

Currently an NSI agent currently has to keep the body of the messages for future inspection (query). Messages (mainly the http header or soap header), can contain credentials such as tokens which should not be kept after they have been verified. We need a statement saying that the HTTP and SOAP header should be discarded. Ideally we just have a log of events, and when they occurred and who did it, instead of keeping full message bodies.

## Single-label source/destination (and STPS)

The idea that source and destination should be modelled as STPs is probably wrong. There is some impedance mismatch at least. If a flow comes in with ethernet+vlan+mpls it should be modelled as such. Not just as having a VLAN xor MPLS. This also makes adaptation quite tricky.

## Static Port List

The idea that we can list all ports in a network (NML or not) isn't realistic. In GTS/GVS, VMs are spawned dynamically, creating a new interface on the host machine. This is a new STP. Sure it will probably get a VLAN on the circuit of the VM host, but the endpoint is a logical interface, and can have labels on it, e.g. Q-in-Q where vlans of the VM are connected to difference places. This means that topology should not try and model all possible endpoints (because it can't), but only nodes/domains and the connectivity between these.

## Unidirectional Circuits

I've helped a lot of sites getting OpenNSA up and running, and have typically helped out with configuration, to get them up and running. The most recurring issue is the topology configuration. In particular getting the unidirectional ports right is the thing that almost always eats the most time. This combined with the fact that no one uses unidirectional circuits means that we have a lot of complexity with zero benefits.

## Callbacks

This isn't really a design issue, but having SOAP and callbacks is just a huge amount complexity for protocol and state keeping. It is immensely error prone, and makes client implementations quite complicated which is hinder usage tremendously. I've implemented a REST interface in OpenNSA. It doesn't have 100% feature parity, but it implements additional functionality like long-polling for notifications and auto provision/commit. The protocol stack is roughly 1/10 lines of code compared to the SOAP interface.