NSI Overview

The "Network Service Interface" (NSI) is an initiative of the R&E community intended to define a single standardized means for applications to request specific network services from a network service provider, and for those network services to interoperate with each other on a global basis.

There are five key elements of the NSI architecture:

1. The NSI Network service domain,
2. Service Termination Points (STPs),
3. Service Demarcation Points (SDPs),
4. Network Service Agent (NSA), and
5. Network Resource Manager (NRM).

A very brief description of each follows in order to understand how NSI and the LHCONE project might interact.

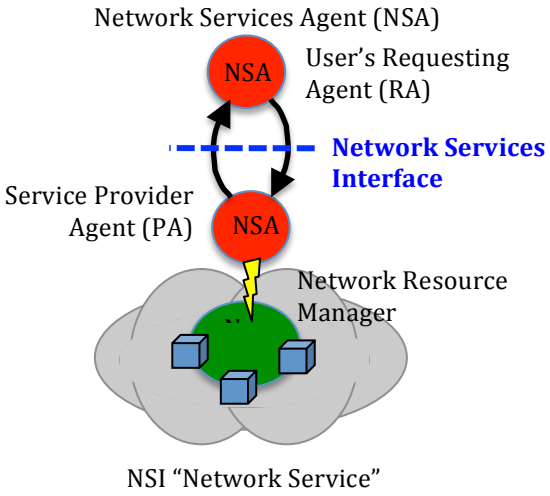
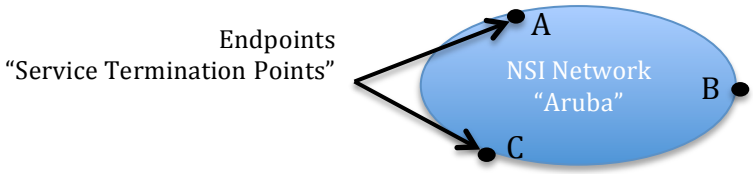


Figure 1 The Network Services Interface

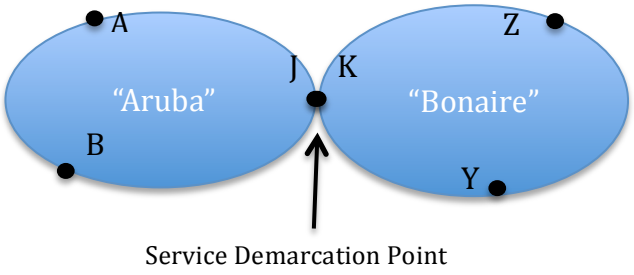
Within the NSI architecture, a NSI Network is defined to be a "service domain" – the logical network region connecting a set of end points via a common service. A software agent called the Network Services Agent (NSA) represents each NSI Network. The NSI protocols are implemented within the NSA and NSAs speak to one another using the NSI protocol(s).

A NSI service domain is – at its basics – a black box; i.e. NSI is an inter-domain protocol and therefore intra-domain functions are allocated to a separate [internal] agent called the Network Resource Manager (NRM). NSI differentiates the inter-domain NSA from the intra-domain NRM in order to allow each network to use whatever internal provisioning system they wish.



NSI Service Termination Points (STPs) are the endpoints that exist as part of an NSI service domain. In essence, the STPs define locations where data transits a NSI network boundary, and where connections may originate or terminate. STPs are specified as endpoints for connection requests.

NSI Service Demarcation Points (SDPs) define network-to-network interconnection points. SDPs are simply pairs of STPs that correspond to one another in two different networks. SDPs define the inter-domain adjacencies.



The NSI Connection Service protocol ("NSI-CS") is the first service protocol defined within the NSI Framework. This reflects NSI's assertion that the atomic form of a network service is a generic "*connection*" – a logical conduit that transports data unmodified from one point in the global network to another. In NSI, the connection is simply a transparent data transport mechanism. The "service" that provides the connection defines the parameters that can be used to describe the connection characteristics, and the topology associated with a NSI Network service domain enumerates the endpoint STPs that are reachable by that service.

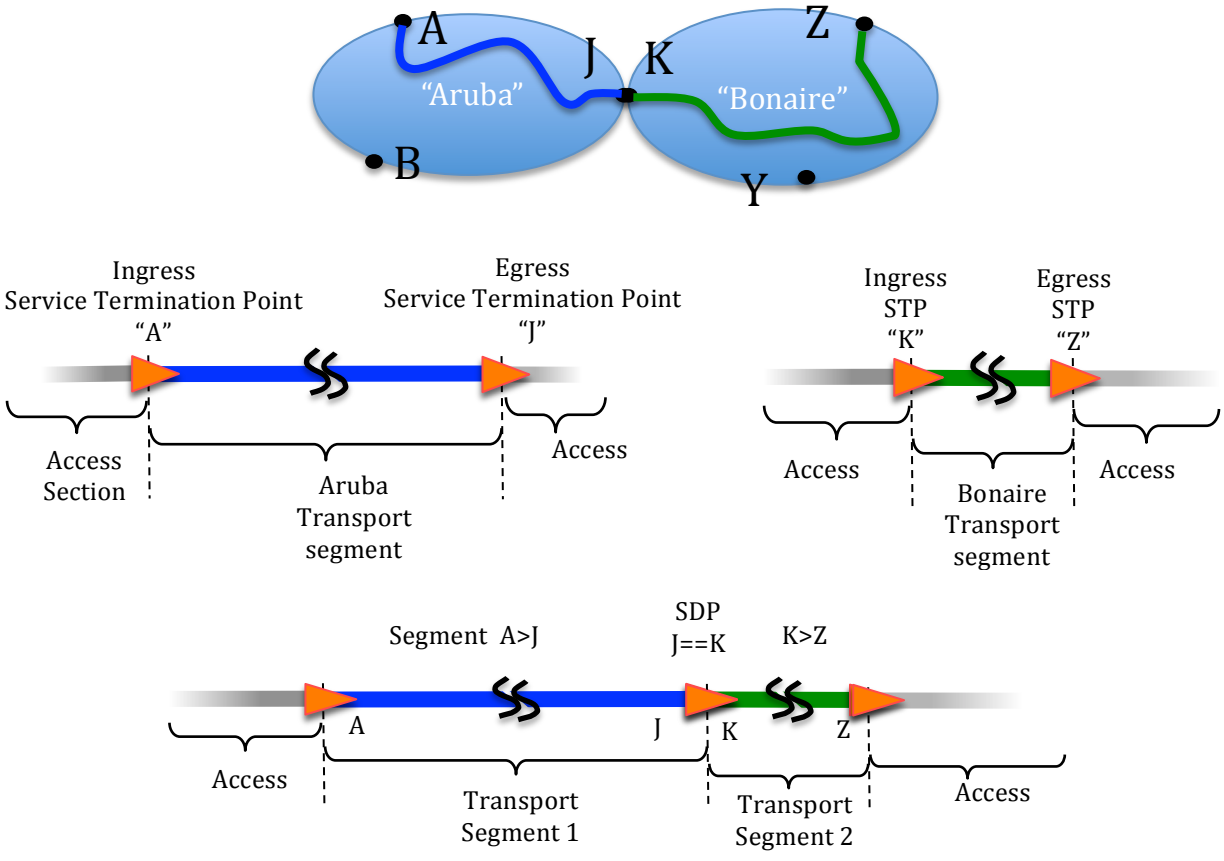
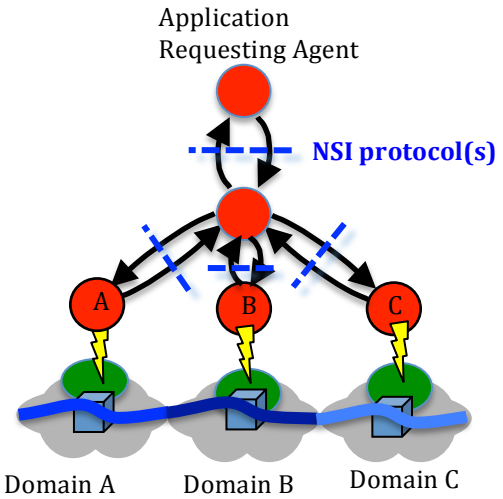


Figure 2 Concatenating connection segments to construct an End-to-End multi-domain NSI Connection instance " A > Z "

The NSI-CS protocol consists of a set of primitive functions that allow requesting NSAs to communicate their needs to provider NSAs. This same NSI-CS protocol is also used by the provider NSAs in turn to communicate service requests to other provider NSAs. Within the NSI framework and local policy, any NSA can talk to any other NSA in order to acquire the necessary end to end resources. This allows a great deal of flexibility – particularly where authorization policy cannot be delegated or an agent wishes to be in full control of the provisioning process. Thus, the NSI Framework, and the NSI-CS protocol in particular, allows users and networks to interoperate and to scale up to a global real-world multi-domain end-to-end service. No other provisioning framework to date offers these features.



NSI Connections, as a rule, do not modify the payload data - they simply transport it. Any payload content processing is performed by other agents/functions at the ends of the connection. At their very basic, Connections segregate traffic - data inserted at one end of a connection is not seen again until it pops out at the other end. Without any other criteria, this traffic segregation alone can be a useful feature for applications at either end of the connection (think of a VLAN connection that simply carries data encrypted with a specific key.) However, in some situations, there are performance criteria that are/can be associated with a connection as well. These performance criteria are used by NSI agents to select and reserve network resources along a path between the end points. Such performance criteria generally consist of some form of capacity, jitter, or reliability specification, but may include other criteria such as latency, book-ahead scheduling requirements (i.e. start or end times), cost, or other constraints.

From a user's technical perspective, the NSI CS protocol consists of a set of web service messages, or *primitives*, passed between a "Requesting Agent" (RA-NSA) and a "Provider Agent" (PA-NSA) that describe a connection and manage that connection though its life cycle.

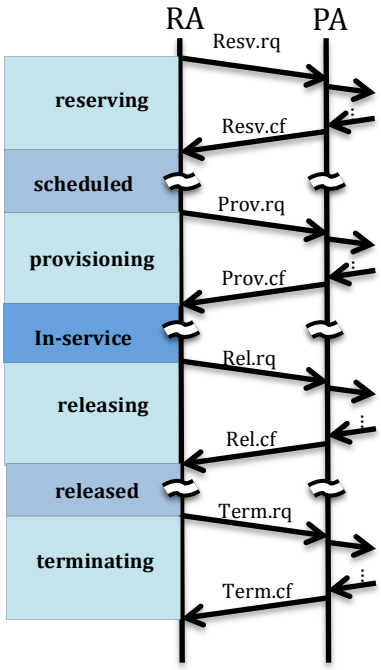
NSI CS offers seven basic primitives:

1. Reserve() – a message from a Requesting Agent (RA) to a Provider Agent (PA) requesting that a connection "reservation" be created. The Reserve() request specifies a service identifier, end points for the connection, and a set of service specific constraints such as capacity requirements, the connection's start and end time (possibly "now"), authorization credentials, and possibly some path hints if the requester desires. A ReserveConfirm response from the provider means the resources have been successfully reserved end to end and are guaranteed to be available as requested.
2. Provision() – a message from user to provider to tell the provider to reconfigure the hardware along the chosen path to place a specific connection into service. If the start time has passed, the connection is immediately provisioned and placed in service. If the start time has not yet arrived, the provisioning will occur automatically when the start time arrives.
3. Modify() – a primitive used to adjust the resources reserved for a Connection. Modify() is typically used to extend a reservation's end time or to adjust capacity. The primitive may be

issued at any time, and if resources are available and can be reserved within the policy constraints of each NSA along the path, then the Connection characteristics are modified and a confirmation returned.

4. Release() – a message from user to provider to indicate that the connection can be torn down, *but leaves reservation in place.* This is useful for long term reservations that may need to be taken out of service intermittently for maintenance or upstream/downstream activities – but these interruptions are known in advance and should not be considered outages or faults.
5. Terminate() – terminates a reservation. All resources are released back to the available pool.
6. Query() – This message requests information about a connection. A simple Query() provides the connection summary status. A detailed Query() can reveal complete end to end details about a global connection.
7. Notify() – the Notify() primitive is sent from a provider to the requester. It is used to communicate unexpected error conditions arising from data plane faults.

Each of these primitives constitutes an inter-domain service request, thus each primitive includes authorization and authentication information to be used by the receiving agent to allow or dis-allow the primitive request. This ensures security and privacy of NSI service information and control across domain boundaries and prevents unauthorized information leakage.



These seven "primitives" make up the basic set of functionality provided by NSI. In many cases, middleware tools or GUIs may hide the complexity introduced by distributed complex work flows. The GUIs and the automated agents behind them will intelligently use the NSI primitives to dynamically reserve network resources when and where they are needed within the work flow. Further, these NSI primitives can be incorporated into command line tools for use in scripts or other macro functions, making it easier to incorporate ancillary configuration processes (e.g. IP address configuration) at the end points.

With the NSI notion of an atomic "connection" coupled with these basic primitives to manage the connection life cycle, and with other appropriately engineered network resources, a whole range of new and more sophisticated services can be constructed: point to multipoint services, broadcast services, multi-path high capacity services, upper layer services (e.g. IPv4/6 VPNs), encryption, protected connections with customized protection schemes, content distribution networks, integrated workflows, etc.

The NSI framework offers other novel capabilities:

- Tree/Chain style reservations: NSI allows requesters to issue NSI primitives to any NSI network (tree style reservations). This enables the user to establish AAI relationships directly with remote domains and to then to use direct authorization credentials rather than relying upon intermediate agents to select an appropriately authorized path. Tree style provisioning is in addition to conventional hop-by-hop path selection with delegated authorization (chain style reservations).
- NSI allows the user to see and process global topology if they wish – the same topology traditionally only available to network agents. To the degree that topology is announced, this allows advanced users to make their own path preference decisions. (A Topology Service that will provide distributed topo management/query primitives is planned as part of NSI in 2013.)
- NSI is secure by design – it implements and enforces authorization policy at every inter-domain boundary.

- NSI is technology agnostic – since it relies on the Service Definition to specify service specific characteristics - not the protocol specification itself - the same framework and generic primitive constructs can be used to manage a wide range of connection services across a wide range of technologies.
- NSI networks are compatible at the **service** layer without imposing constraints on how the service is implemented. Thus an Ethernet transport service may be implemented internally as Ethernet/MPLS, Ethernet /GFP/SDH, native Ethernet, PBB, etc… internal service engineering is invisible to external service presentation.

Probably the most important aspect of NSI is that it is an open consensus standard – i.e. it comes from our community and is driven by the needs of that global R&E community. It is open- i.e. it can be integrated into any application, or a network can implement their own NSAs, and anyone can participate in the standards working group.

How do we get there?

We must recognize that for all NSI's demonstrated capabilities and progress, it is still a very young technology. As of Dec 2012, the NSI-CS version 2 standard is being edited, and beta versions are being tested within the GLIF Automated GOLE fabric. The testing is necessary as it is a primary means to identify and resolve small technical issues regarding the WSDL for the primitives, processing inconsistencies, etc.

The following are a list of key aspects that must be addressed to implement NSI within LHCONE:

1. Service Definition.

Several R&E networks have announced their intent to deploy production service(s) based upon NSI-CS. In order to do so, these networks will need to develop a Common Service Definition (CSD).

a. The CSD is joint agreement among a self-selected set of networks that describes the technical aspects of a [connection] service they intend to offer across each of their respective infrastructures. In the case of NSI, the CSD specifies the name of the service (e.g. "P2PCS") and explicitly defines the service specific parameters that the NSAs will recognize (e.g. "capacity", "averageFrameLossRate", etc). This does not dictate how each network engineers the service, only what the service delivers to the user.
b. Further, the CSD will specify a common security profile – i.e. the roles and priviledges that will be recognized and the means of authentication that will be used to determine authorization for CS primitives.

This process of defining the service in explicit terms has already begun with the Stockholm "P2PCS" draft document. This CSD is still in a malleable state, but it defines a service that is consistent with NSI and very similar to existing P2P services.

2. LHCONE Infrastructure Engineering

Within the context of LHCONE, NSI's role may be different depending on how LHCONE views itself:

If LHCONE is to be a monolithic core international network connecting the participating campuses and labs, NSI can be deployed to provide the allocation and provisioning of the LHCONE resources across that core to the LHCONE user community, and as a means of reaching beyond the specific LHCONE boundaries to extend connections into other networks. This approach will allow the LHCONE community to share costs and assert their own policy within the LHCONE core domain, and would afford the LHCONE user community a greater flexibility to

make comprehensive path selection decisions – inclusive of LHCONE or in conjunction with other transport resources.

An alternate model – or a complementary model - would be to set up virtual connections on a case by case basis to construct higher layer LHCONE services – for example, the LHCONE community could use NSI services from multiple traditional service providers to construct an IP network service, or construct an emulated VLAN service spanning multiple continents, or a yet more sophisticated content distribution network service for the dissemination of LHC data sets. In this overall model, the LHCONE effort is more of a community initiative that only acquires transport capability when and as needed for specific functional capabilities. It is the sum of these individual service constructs that make up the LHCONE network environment – yet the LHCONE services are not constructed from a single underlying fixed infrastructure. This model is highly adaptive – able to reconfigure itself rapidly from the global infrastructure without impacting the services presented to the LHCONE user community. However, it relies on the existing R&E networks to deploy full scale NSI services – a prospect that will likely lag LHCONE's requirements.

Indeed, these two approaches are compatible – the LHCONE community can acquire conventional static circuits and dynamically provisioned virtual links to create a common service network that could address perhaps certain policy constraints of the community, and in parallel have other services such as a global virtual ELAN (emulated LAN service) capability constructed from a separate set of infrastructure. It is important to understand that "dynamically" provisioned connections do not imply transient or short lived connections…automated multi-domain processes are necessary for scaling – to perform path finding, scheduling, monitoring, fault mitigation, etc for even long term persistent circuits end-to-end in an increasingly complex multi-layer multi-domain infrastructure environment. NSI is the foundation for doing this end-to-end in a global environment.

3. Topology.

The topology for the LHCONE and its participants and service providers must be resolved. NSI Version 2 has adopted the Open Grid Forum "Network Markup Language" (NML) draft standard for topology descriptions. LHCONE will need to develop a NML topology description for the core domain, and initial topology descriptions for participating lab and campus networks, or peering networks that will be part of the contiguous LHCONE / NSI service region. These initial topology descriptions can be skeletons, defining a very minimal service domain initially, but which can be easily expanded to serve a much larger set of endpoints and uses over time.

4. NSA Software Selection.

Each Participating network, whether it's the LHCONE core, or campus/lab networks will need to select a NSI implementation they plan to use as their NSA. This software may be an existing package such as OpenNSA, OpenDRAC, NSI/OSCARS, G-LAMBDA, AutoBAHN, etc but must be integrated with the local provisioning interface for local intra-domain hardware. OpenNSA, for example, has several NRM interfaces such as JunOS (Juniper Networks hdw), FTOS (Force10), Argia (a UCLP tool), and the DUD (virtual data plane) NRMs. Other packages including OSCARS and AutoBAHN have similar modules, so it is recommended that each network investigate which packages may be best suited to their infrastructure or preferences.

Alternatively, the NSI protocol can be integrated into an existing local NRM tool – which may be an easier initial approach. It should be noted however that some NRM adaptations only act as local providers to their own network and do not implement [yet] the NML topology processing and path finding necessary to act as a full service front end for global end-to-end NSI Connection requests. In the NSI framework, these local only NSAs are referred to as leaf nodes, or "provider only" NSAs since they only accept reservations that are local to their domain – i.e. they do not/cannot process multi-domain requests. This is another consideration for the local network service architects in NSA selection. As an interim, since [user] NSI agents can send primitives to

any NSA for service processing, arrangements can be made for the global end-to-end path selection and reservation to be performed by an NSA running elsewhere, and these requests are sent to this "aggregator" NSA first. Indeed, LHCONE may wish to have a single NSA that performs end-to-end segmentation and issues local only segment requests to the participating domains. This would potentially simplify the NSI deployment initially for these organizations in that they could deploy a provider only NSI/OSCARS or AutoBAHN agent posing minimal impact to existing processes, and use another trusted NSA such as G-LAMBDA or OpenNSA to perform the path segmentation. It should be noted that in general these packages all plan to implement the NML topology handling and more sophisticated global pathfinding, but this may not be available in initial v2 codes.

5. Application Integration.

With respect to end user applications and work flow tools, NSI is not so different from existing APIs. Indeed, NSI evolved from the IDC protocol. So applications and workflows that currently utilize dynamically provisioned circuits should be able to migrate to NSI api with minimal initial effort. These applications can be tested in the near term using the GLIF Automated GOLE testbed, and as LHCONE architecture and the NSI service planning are resolved, testing can be performed against mockups of the LHCONE facilities and services.

Technical specifics for the NSI-CS v1 interface can be found in the Open Grid Forum Network Services Interface Framework and NSI-CS v1 documents. NSI Version 2 will be there soon as well pending completion of the editing process. A beta version of the WSDL for the v2 available now for developers wishing to begin the migration process.

6. Operational Control, Monitoring, and Performance Verification

The NSI development has focused primarily on protocol development to date. A full featured network service will include a number of tools that allow operations and/or applications personnel to monitor and control connections and to ensure the service instances are operating to spec. These tools need development. Indeed, the architecture for scalable global performance verification, monitoring, and automated fault processing is an open issue- not just in NSI but for multi-domain connection services in general. This said, the NSI framework and the CS protocol provide most of the functionality required to perform these functions, but they need integration into a robust and easy to use tools and GUI interface package. And there are some issues that must be addressed within NSI to simplify these processes (for instance notification flooding of faults).

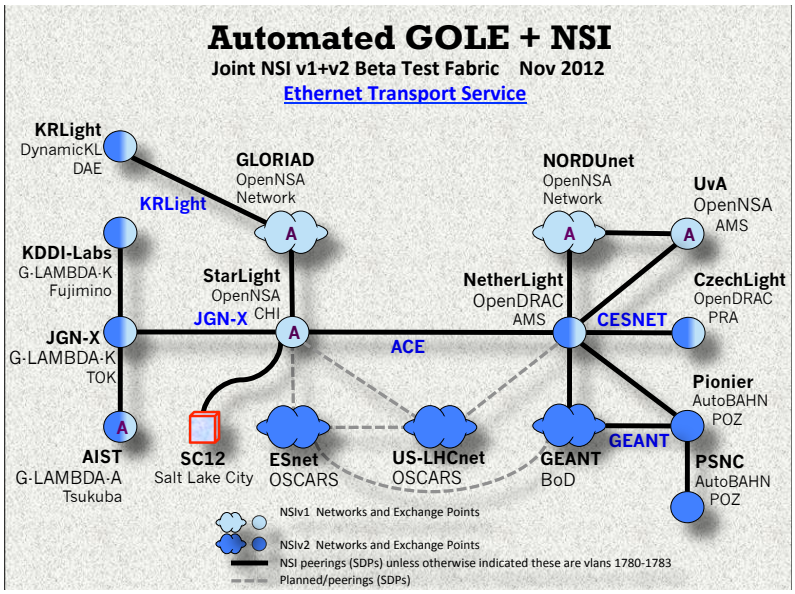
Packages such as perfSONAR and its variants have been used quite successfully to date for many of these functions, and are used currently in the Automated GOLE for basic testing. Nevertheless, emerging services are challenging conventional monitoring/measurement practices. A clean slate approach will be required to address a number of emerging new technologies (including NSI and connection services) to develop a scalable automated and deterministic instrumentation architecture for monitoring and verifying these new services. However, in the near term, perfSONAR tools can still be used to provide monitoring of various aspects of the data plane. This is essential to move the ball forward and get NSI deployed. While there are important issues that must be addressed in the future such as scalability, security, and accuracy of both the tools and the information required to perform end-to-end detailed service analysis and forensics, the NSI framework for inter-domain service provisioning represents a giant stride forward in promoting a global service architecture that lends itself to deterministic network service monitoring and measurement – particularly where it enables sophisticated performance verification processes and automated fault handling capabilities.

So perfSONAR can (and should!) continue to be used as we migrate to NSI. There will be modifications required to integrate it into the NSI framework and to adapt it to utilize NSI-CS primitives to expose provisioning details of NSI created Connections. And the migration towards the NML standard for topology descriptions will pose similar updating of perfSONAR

tools. While this integration forward may be substantial, this will retain the existing substantial pS archiving capabilities and access to those archives by a large and growing set of operational monitoring and forensics tools.

7. Training and support.

Finally, if the LHCONE community is interested in moving aggressively towards NSI based services, the various most active members of the NSI Working Group and the AutomatedGOLE Pilot Project can provide workshops and tutorials for applications developers and/or for network engineers planning NSI deployment. These are available today, and will be continuously updated to reflect NSI evolution.



The NSI Framework and standards continue to progress and evolve, but the basic Connection Service semantics have become stable. It is anticipated that only relatively minor syntactic changes will occur in the future as technical issues are refined thus making it safe for applications developers to begin wading into the water and migrating codes to NSI. The Framework will introduce additional new features and services as it continues to develop, as well as refinements under the covers that will be less evident in the service model but will simplify internal processes.

NSI continues to evolve – to offer a full range of capabilities that can be deployed globally. LHCONE is in a position to lead in the effort to refine and mature these services. The HEP community can reprise their traditional role as the prow of the icebreaker to push these technologies forward and move the priority up in the roadmaps of the conventional R&E networks. NSI represents a substantial step forward in network technologies, we should seize the moment.