Title
Enabling intent to configure scientific networks for high performance demands

Authors
Kiran, Mariam
Pouyoul, Eric
Mercian, Anu
et al.

Publication Date
2018-02-01

DOI
10.1016/j.future.2017.04.020

Peer reviewed

Enabling Intent to Configure Scientific Networks for High Performance Demands

Mariam Kiran*, Eric Pouyoul, Anu Mercian, Brian Tierney, Chin Guok, Inder Monga
Energy Sciences Network (ESnet)
Lawrence Berkeley National Labs, CA, USA
*mkiran@es.net

***Abstract*— Globally distributed scientific experiments usually involve massive data volumes and distributed data analysis being done by many collaborators. With complex workloads and heterogeneous resources, each user may require certain characteristics for their network paths. In this paper, we present the iNDIRA tool, which interacts with SDN north-bound interfaces to enable intent-based networking. It provides reliable, simple, and technology-agnostic communication between users and networks. Focusing particularly on science applications, iNDIRA uses natural language processing to construct semantic RDF graphs to understand, interact, and create the required network services. The technical challenges addressed by iNDIRA are: (1) development of a high-level descriptive language to query for network-application requirements, (2) keyword identification and condition checking based on user profiles and topology details, (3) allow user negotiation based on the current network state, (4) provide network provisioning guidance, and finally, (5) automatically provision and monitor a layer-2 network path for use by the application. iNDIRA is implemented on the ESnet network, where it interacts with OpenNSA (aka the NSI client) and Globus data transfer tools, to build complex cross-domain network paths for heterogeneous science applications and perform secure data transfer. We describe our implementation of iNDIRA running on ESnet's production network. We present results of iNDIRA's query processing mechanism, evaluate iNDIRA's intent language evaluation with other approaches, and describe future enhancements for iNDIRA.**

Keywords—Intent-based networking; Natural language processing; Ontology engineering, SDN north-bound interface.

I. INTRODUCTION

Scientific analysis and simulations such as climate modeling, or experimental facilities like the Large Hadron Collider (LHC) at CERN, have demonstrated the need for research communities to collaborate world-wide. These collaborations usually involve complex data-intensive activities such as secure data delivery to local sites, management of data-transfer latencies, and a requirement that the network can consistantly and reliably meet performance expectations. To further these efforts, as dataset sizes continue to grow, it is imperative to have advanced network services and integrated software tools to help manage high bandwidth demands and traffic control. For instance, an application needing to send a large dataset by a specific time deadline, needs to know how to provision the network to achive this. In practice, users request network resources, technology capabilities and service offerings, which are then provisioned manually by network engineers, or using tools such as OSCARS [13] and OpenNSA/NSI [29]. To help accelerate this process through network automation, software defined networks (SDN) are being employed to centrally control network flows [3, 28]. The separation of data and control planes in SDN allows network virtualization to dynamically manage switches and flows. Open-source toolkits such as Ryu, ONOS and ODL [3], with OpenFlow controllers [4] can control network resources through south-bound interfaces to network devices. These south-bound interfaces (SBI) transfer instructions from north-bound interfaces (NBI) and SDN controllers, using 'how-to' commands to provision the underlying network. This paper focuses on *Intent-based networking,* which allows users to achieve high-level network control using intent-based queries defined as 'network service requests'. This enables better quality service and experience (QoS/QoE) for the requesting applications. High-level queries or service requests are defined in a descriptive language, such as "allow traffic between A and B", which is translated into a prescriptive language as "from A:10.0.0.1 to B:10.0.0.2 set rule=allow" and rendered as OpenFlow rules to produce desired network connectivity [5].

High performance science applications such as LHC data analysis require high-volume bulk data transfers with dedicated or preferred topologies. Scientists often wish to specify network requirements, depending on the needs of their application. In order to optimize this exchange, Neuman et al [6] argued that there is a need to co-design and co-develop raw network capacity, system I/O architecture and network performance with distributed scientific instruments and computing infrastructure. The authors argued that SDN and Open vSwitch (OVS) are a potential solution, but require intensive system re-engineering. These would require new network setups and have a steep learning curve for non-network engineers on how to use networks. Abstracting at a level higher, *intent* allows multiple users to automate their network workloads with a common language and remove dependency on underlying network architectures.

The aim of this work is to develop a system (iNDIRA, or Intelligent Network Deployment Intent Renderer Application) to act as an interface between users/applications to the SDN NBI, understanding network QoS demands and automating the translation for scientific applications. iNDIRA uses natural language and ontology engineering to define queries and translates them to network commands, reconfiguring resources, preferences and policies. It then interfaces with network provisioning and transfer tools to allow the user's intent to be rendered into specifications recognizable by these tools. The tools currently used are NSI, which allows multi-domain

network provisioning, and Globus, to allow secure file transfer. Therefore, intent, through iNDIRA, presents a number of advantages such as portability, easy language definitions, and the ability to optimize network workflows.

In industry, there are multiple efforts in developing intent-based networking, discussed in the next section. However, these projects lack a common language specification and are difficult to use, requiring a number of configuration steps to prescribe and setup network services. The challenge goes further with lack of bandwidth availability and authentication/authorization checks before the intent is deployed. Our system can check for conflicts, match possible network services to intents, provision dynamic network resources across multiple network architectures, and communicate to users and applications, as needed. Our specific contributions are:

- We designed and implemented a system called iNDIRA to provision network resources based on user and application demands. The system currently interfaces with NSI and Globus tools to dynamically control networks for file transfer, and can be extended to multiple network providers.
- iNDIRA is able to converse with users in a natural manner, understanding their needs to discover contexts and give provisioning responses. iNDIRA is able to dynamically setup paths for heterogeneous science applications depending on the application requirements.
- iNDIRA automatically reserves, provisions and activates network states based on intent. We evaluate our system based on query processing time, topology size, conflict resolution and service satisfaction. We investigate these in two scenarios – user-iNDIRA interaction for setting up network provisions, and user-iNDIRA interaction for scheduling bulk data transfers using Globus tools [36] and API. We evaluate iNDIRA with current intent projects for usability, query processing time and improvements in network QoS satisfaction.

The paper is organized as follows: Section 2 presents motivation and related work, presenting the need for intent and background to current efforts. This section discusses how science can benefit from tools such as iNDIRA by automating and optimizing network control. Section 3 presents the iNDIRA framework and its design using semantics and natural language. iNDIRA has a two-way communication process, with the ability to communicate in a human-like manner to human users, and using meaningful error messages to applications. These cases are discussed in Section 4 along with language and implementation details. Section 5 describes a demonstration of iNDIRA's capabilities in setting up RDF graphs and performing conflict checking. Section 6 presents discussions and evaluation of the tool with current intent-based efforts. Finally, Section 7 presents conclusions and future work for increasing iNDIRA's capabilities for science applications and networks.

II. Motivation

Whether produced via simulations or consumed by other software, handling large data files is a cumbersome task. Networks are essential to connect data sources for analysis, and for collaboration between groups. However, collaborators located in different networking domains presents new challenges in multi-domain path setups, configuration files and policy checking. Figure 1 shows a high-level intent to setup a dedicated path between a scientist and collaborator involving multiple sites. Although a simple requirement, this case study presents multiple network configuration challenges in terms of unknown site addresses, unknown computing capabilities, security and maximum data speeds allowable. Further challenges include,

- Network congestion: Bursts of network traffic can cause reduced performance. TCP/IP congestion is common across science demands, and requires active management of traffic.
- Mismatch in IO capacity: Data transfer nodes have varying IO capabilities. For example, Node A can be sending data to Node B at a fast rate, but the rate at which Node B can sink these datasets can be slow, leading to packet loss [7].
- Time scheduling awareness: In science applications, some files need to be delivered by a certain deadline to enable further experiments or processing to start, e.g. remote data visualization. Eventually, this can lead to the workflows getting blocked.
- Heterogeneous applications with multiple network needs: Science collaboration activities have led to a plethora of applications that include real-time data rendering and transfers, on-demand communication tools, central file sharing repositories or processing data at high bandwidths for image and visual analysis. These have multiple network requirements such as guaranteed bandwidth, traffic isoloation and time schedulers.

Currently, network service provisioning is done manually. This involves writing complex network reservation and provision steps, where the implementation is closely tied with underlying infrastructure. This leads to architecture dependency and sometimes unoptimized networks, accommodating complex network services and conditions. Users also have a steep learning curve to understand how to write these complex network commands and how multiple network tools interact to produce desired effects. A recent report [15] recognized a multitude of application categories such as bulk data transfer, remote analysis for visualization and dedicated collaboration tools (e.g. Skype) requiring dedicated high and sometimes low bandwidth paths.

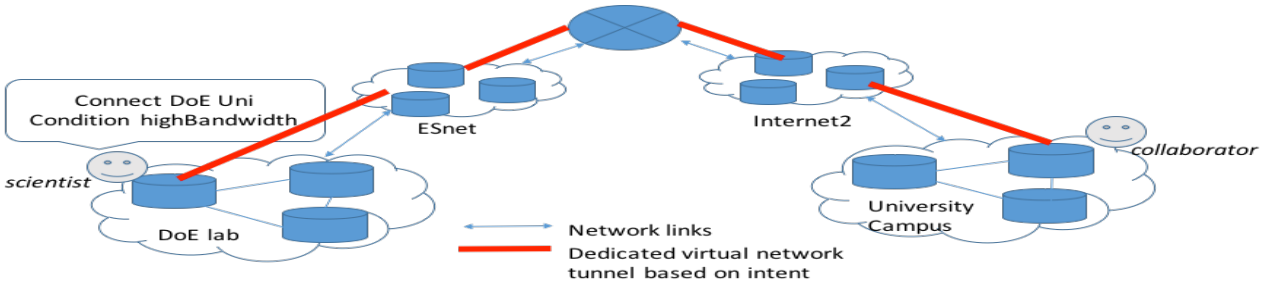

Fig. 1. Simple case of scientist requesting a high-bandwidth path between a DoE lab and University campus.

To automate provisioning and network control, iNDIRA is able to handle most network complexity through its parser and engine, communicating in easy English statements with users. Particularly in scenarios of science dataflows, we envision two main possible users,

- End-users running science applications, who have little knowledge of network engineering and underlying architectures but desire particular network performance.
- The scientific applications sending and ingesting data. These have a start, end, source, destination and transfer requirements all specified as parameters needed for applications to execute. This is similar to Globus jobs for transfering bulk data transfers.

iNDIRA's intent capability follows a four-step approach, the network service (1) is requested, (2) responded to, (3) selected for most appropriate configuration and finally (4) confirmed back to the user. It allows heterogeneous devices, discovered at run-time, to be automatically configured to satisfy user requirements. It also works across multiple domains catering to cross-domain interaction, where interface or port addresses are not known, including provisioning with complex uniform resource names (urn) which are difficult to remember. Certain performance characteristics have to be monitored and aligned with user profiles for security checks. iNDIRA allows automatic query parsing to configure networks suited to user needs.

III. Related Work

Descriptive languages can allow network providers with little programming experience to depart from traditional networking APIs [31]. Today, SDN research has led to a multitude of tools, multiple north and south bound interface controllers but lack implementation standards [17]. Most optimization algorithms are deployed using simulations and not implemented with OpenFlow protocols [18].

Networks are difficult to manage because of their constantly changing states and low-level network configuration knowledge which is usually unknown [34]. The policies do not allow network operators to specify high-level intent and is difficult to design languages to react to continuous network state changes. Gurbani et al [35] presented an Application Layer Traffic Optimization protocol, that provided an abstract view of the network to enable applications to leverage a network without exposing the network provider's internal details or policies. However, this work focused on content-level networks. Similarly, Salsano et al. [36] discussed north-bound APIs to optimize packet movement, but more user information was needed to make informed decisions to improve QoE [25]. Further, OpenStack [30] discussed how geographically distributed data centers, intent can help balance workload automatically, handle technical and even legislative procedures efficiently. However, they have not implemented a system to do so.

Cohen et al. [25] presented the DOVE architecture (Distributed Overlay Virtual Ethernet network) for network virtualization abstraction. Intent, in this case, uses VXLAN and virtualization management tools to perform intent as policy enforcement through virtual interfaces. Endpoints are created based on policy profiles to capture network functionality and demonstrated through simulation frameworks. In further work, Procera [26] discussed a policy engine and language as functional reactive programming in Haskell for implementing reactive network policies, built upon OpenFlow technology. Fujitsu [27] uses Virtuora management tools to collaborate with centralized SDN controller to enable distributed network control. The system allows intent to perform topology discovery, path calculation and network policy prioritization before pushing these to OpenFlow protocol. This work has been planned to be extended with ONOS. All of these, present implementation designs but have not produced deployable solutions yet.

Recent work by Boulder [23] describes intent as a 'principle of operation' which use intent 'expressions' and 'mapping' to separate implementation dependent parameters. Using information models to downstream network controller projects, Boulder proposes fragments to be part of an intent tree, with some initial work examples on usecases such as end-to-end service function chaining. Another approach, ONOS, uses policy-based derivatives for groups through application-policy rendering of the prescriptive module [32]. OpenStack, the cloud framework for networking across wide-area networks, has adopted group-based policy and focuses on application requirements rather than infrastructure requirements for satisfying application intents.

NEMO [20] uses a transaction-based Northbound API, that provides a network language that characterizes network with paths and flows, abstracting necessary transitions to make modifications to policies from network perspectives. NEMO includes details for enabling and disabling virtual nodes in virtual networks. It uses an engine to process a network language to render modules like OpenFlow. The language defines intents as communication paths, rather than endpoints,

which requires users to have some knowledge of the network before deploying intent.

Table 1 shows how intent definitions vary dramatically across different projects. In Boulder, intent follows the original ONF definition of having intent specify a 'need' rather than 'how'. But embedding policies or more network specific information, as in NEMO, can quickly allow intent to specify 'how' the network is to be configured, becoming more prescriptive than declarative.

The projects also use different ontologies to represent intent. In ONF and Boulder, intent is defined as subject, predicate and objects [20]. Boulder defines objects to contain constraints and conditions on the network state. Alternatively, NEMO defines intent as collection of objects, operation and results, where operation contains the constraints and conditions and result denote expected or avoidable network states. Both projects provide use case examples with their grammars. Both grammars are still being developed and are highly focused on network use cases. Table 1 compares these projects with iNDIRA's capabilities.

TABLE I. CURRENT INTENT-BASED NETWORKING EFFORTS

Open Source Projects (Associations)	Features
Boulder (ONF)	Controller agnostic, uses high-level language as expressions, maps to implementation-dependent parameters. Example "Bob connects to Alice with 10GB from 9 to 10pm"
NIC (ODL)	Define intent as set of commands with subjects, actions, and constraints. Uses Yang for abstraction. Defines intent to add, remove, and modify network.
GBP (ODL/ OpenStack/ Cisco)	Define intent as Policy-specific to certain controllers. Can configure ODL environment for each policy, involves several configuration steps.
Congress (OpenStack/ VMWare)	Implemented for Cloud use. Uses policy extension on applications.
ONOS Intent Framework (ONOS)	Interface with ONOS controller. Uses intent as policy derivatives. Uses intent commands as subjects and actions, translates them into installable actions on network.
NEMO (ODL/ Huawei)	Intent used as policy renderer, develop network modeling language to define intent deployment.
iNDIRA	High-level language to input intent similar to English commands Based on ontology engineering to identify services and arguments using RDF specifications. Can be extended further with other network semantic tools. Uses graph theory to identify conflicts, check rules and policies. Communicates network status back to users. Can be integrated with other tools to allow higher level of QoS translation.

A. Definition of Intent

Intent does not involve directly specifying routing or switching information, but rather a flexibility to allow easy portability to optimize networks for user needs. The statements can be either declarative, at a higher level stating the problem or need, or prescriptive, at lower level asking for solutions to solve the problem. In networking terms, a prescriptive example would be "Connect server A to B with a route across switch 22". Such cases do not serve as good intents, as users may receive a reply saying "switch 22 is not available". Rather a declarative approach, allows users to specify the problem context, any rules or constraints and the result wanted. These details allows the network to think for itself, the best possible solution for servicing the intent.

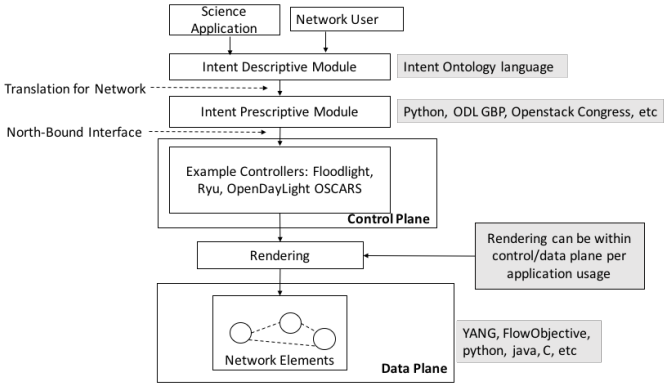

Fig. 2. Intent for users and applications filters through SDN layers.

Figure 2 shows how intent filters down to network elements, through various modules, languages and components. Users can specify intent, by either using templates or high-level statement queries. The queries are input through a CLI, which is then translated using an ontology into intent descriptions. The ontology allows these descriptions to be easily mapped into network needs or prescriptive definitions, such as in python or APIs like ODL GBP, Openstack Congress which are prescriptive in nature. These can then easily interact with multiple controllers, to provide the rendering of the intent onto networks. For instance, ODL NIC allows OpenFlow rendering in the control plane, where as some would render as part of the data plane, depending on application use. Once the intent is rendered successfully, it is transferred onto the data plane to automatically configure network elements. The three main modules used are,

- The Descriptive Module acts as an external API module where intent is input. The descriptive module can render without need for long configuration procedures, installing and maintaining SDN deployments. Example markup languages such as XML, UML or OWL extensions can help allow data abstraction and object mapping. Here ontology engineering can help define what information is defined. Most iNDIRA's research and innovation lies here.
- The Prescriptive Module uses endpoint and policy resolution, intent integration, conflict resolution and object mapping to discuss how network processes are executed (e.g. NIC, Congress).

- The Rendering Module is used as communication of Intent-NBI to South-bound nodes via the Controller. As SDN technology matures, more south-bound interfaces will be used to connect network edges. This would require rendering modules to include other prescriptive modules to accommodate these SBIs. For example, NIC is a prescriptive module that uses GBP renderer.

iNDIRA focuses on defining intent as a descriptive language. Using ontology engineering, it facilitates machine reasoning to understand user intents, through a common language. Web languages such as Web Ontology Language (OWL) and Resource Description Framework (RDF), can be employed for building constructs for describing performance in the past. However there is a lack of one ontology allowing researchers to write their own ontologies as needed [19].

Multiple network performance efforts have focused on optimizing QoS by using semantics to define service quality and conditions. Chui et al. [9, 10] presented a framework for integrating QoS support in service workflow composition system. The authors discussed a relationship between workflow execution time, cost and error propagation which could dynamically be adapted for user QoS preferences. These approaches have been successful in defining web services, SLA validation and QoS monitoring tools. For example, Network Description Language (NDL) and NOVI information models have been successful in combining internet services across multiple groups [12]. However, unlike service computing applications, networks have not seen many semantic and ontology-based approaches. Similar to solving multi-device communication, these approaches can be very useful to enable multiple providers, switches and routers to communicate toegther and remove inter domain dependency.

IV. iNDIRA Architecture

The iNDIRA framework interfaces between the SDN north-bound interface and users as shown in Figure 3. The intent language gets converted to RDF graphs, through parsing and rendering, to eventual network commands. The renderer takes inputs from multiple data files that contain user profiles and topology details.

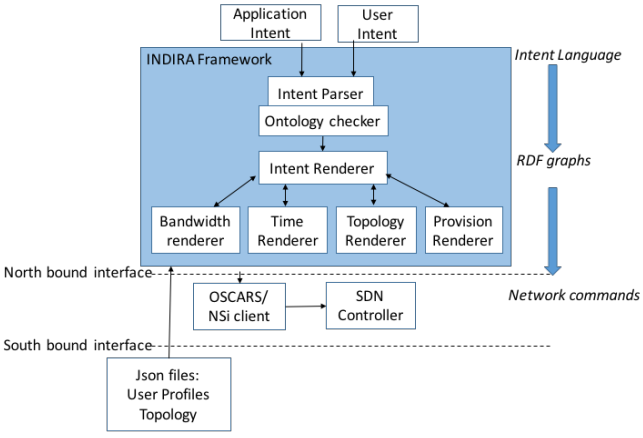

Fig. 3. iNDIRA architecture.

A. iNDIRA Application

Networks using iNDIRA can provide the following purposes,

- Move stored data files to allowable sites for a project.
- Stream data in real-time to remote locations.
- Fulfill user and application network requirements such as transfer by date and time, check for policies and bandwidth restrictions.
- Maintain a dashboard, to display user intent and current network states. Use this as a means of communication.
- Possibly use intent and network information to provide users with optimized network configuration choices.

The production networks must be able to support these functionalities to allow iNDIRA to fulfill its objectives. In order to do this, it needs to know the site topology details, user profiles, bandwidth restrictions and other security requirements. For example, a HD video-on-demand application can decide when to start and stop guaranteed bandwidth and setup low jitter paths between two or more sites.

Over ESnet, both OSCARS and NSI tools were explored for setting up reservation requests and topology on demand. iNDIRA is able to communicate with both tools using topology information and project profiles. However, to show a concrete example of its potential with multiple domain demonstrations, the example discussed in Section 4 focuses on network provisioning across multiple domains and transfer of files using the dedicated path (Figure 4). iNDIRA is able to interface with Globus via the Globus API [36] to schedule file transfers and add QoS details, without changing NSI and Globus code.

The iNDIRA user interface allows users to interact with iNDIRA modules. Requests are parsed with associated files to commands immediately deployable by the NSI client.

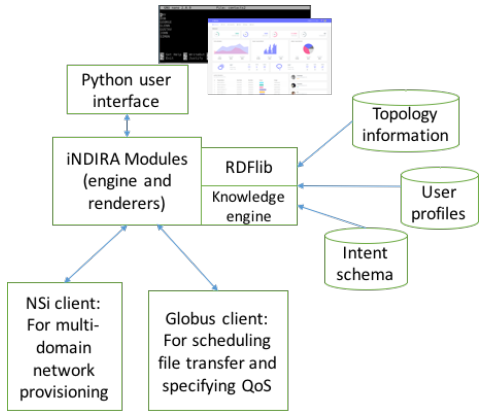

Fig. 4. iNDIRA's componets and interaction with other tools.

iNDIRA needs site topology information to:

- Produce an uptodate assessment on site, interfaces and available path information.
- Choose appropriate paths based on demands, for setting up the address details.

Specifying endpoints, such as site "LBL" (Lawrence Berkeley Lab) can be rendered into two possible interfaces with details on capacity, bandwidth and range availabilities. iNDIRA can use this information (returned as JSON) to choose one available interface.

iNDIRA also takes inputs on project details, users and security policies to check, before it sets up any network paths. This prevents users who do not have access to sites to create paths, or access data from there. For example, Project2 does not have access to certain sites. iNDIRA returns output error messages to users.

Profiles.json:


```
{
 "name":"admin",
 "description":"Administrators",
 "bandwidth":"unlimited",
 "topology":"*",
 "timezone":"UTC"
},{
 "name":"project2",
 "description":"Collaborative project 2",
 "bandwidth":"5096",
 "topology":["anl","lbl","oak","cern"],
 "timezone":"US/Pacific"
},
```


iNDIRA INTENT: For Project 2 connect BNL, LBL.

iNDIRA GUI OUTPUT:
> Sorry you dont have access to the following sites:
['BNL']

iNDIRA can capture a number of conditions, time constraints and resource compatibility to fulfill user intent. We define an ontology schema to help iNDIRA capture these demands and replace them with appropriate network service commands.

B. iNDIRA Language

Ontology can help standardize communication over multiple tools and services. Researchers have used it to document QoS needs, workflows and cost functions for web services [9, 10]. Ontology represents relationships between services and its arguments. iNDIRA uses RDF graphs to record intended services and arguments. This is similar to works in [8] using RDF to describe network actions, [11] creating an ontology to organize information and [12] defining a NOVI information model to express resources, services and policies for interoperability. The RDF graphs can document collection of network actions and conditions, arranging it as a triplet – Subject (Service or Condition), relationship (has Arguments) and objects (multiple parameters). For example, consider intent, "Project1 connect from LBL to ANL", the RDF triple represents two relationships:

- Project1 hasService Connect
- Connect hasArguments LBL, ANL

We have identified three services iNDIRA users may request – Connect, Disconnect and Tap. These can be extended with conditions such as following,

- No bandwidth limitations
- Isolate the traffic
- lowest priority traffic (if congested, drop this traffic first)
- Schedule start time and duration for service
- Schedule start and end time for service
- Schedule deadline for service to finish
- Define a duration for the service to run.

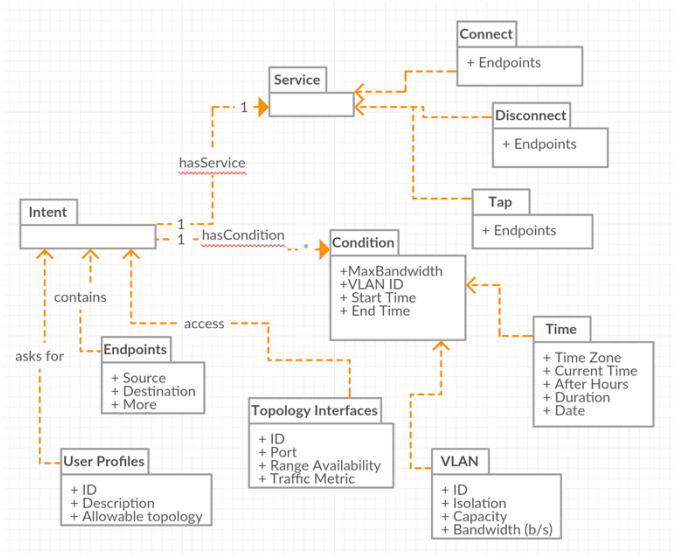

Fig. 5. UML diagram for iNDIRA's intent language

To allow iNDIRA to understand descriptive high-level language, it was equipped with a parser checker and an extendable ontology schema. iNDIRA is able to recognize full sentence queries and converse with users on what they expect from networks. For applications, it is able to take specific details as JSON header. It performs the following functions,

- Recognize and replace keywords such as 'move' to 'transfer' to identify network specific commands easily. A dictionary of similar words is able to parse through sentences to replace words into recognizable services.
- Remove words such as 'I want to' to only service commands, such as,
  (a)I want to (b) move datasource1 from LBL to ANL

  Here, only (b) is recognized as a 'transfer' service for datasource1 and 'from' and 'to' are recognized as endpoints.
- Ask users for conditions if not defined.
- For applications, define json headers to provide network connection specifications to automate setting up.

Figure 6 shows intent being parsed, "For project 1, connect sites ANL to LBL, with condition nobwlimit, nolimit and start time now". iNDIRA first looks for keywords and creates services and conditions. Each of these are constructed into an RDF graph with arguments. It goes through a series of RDF conversions for iNDIRA's rendering.

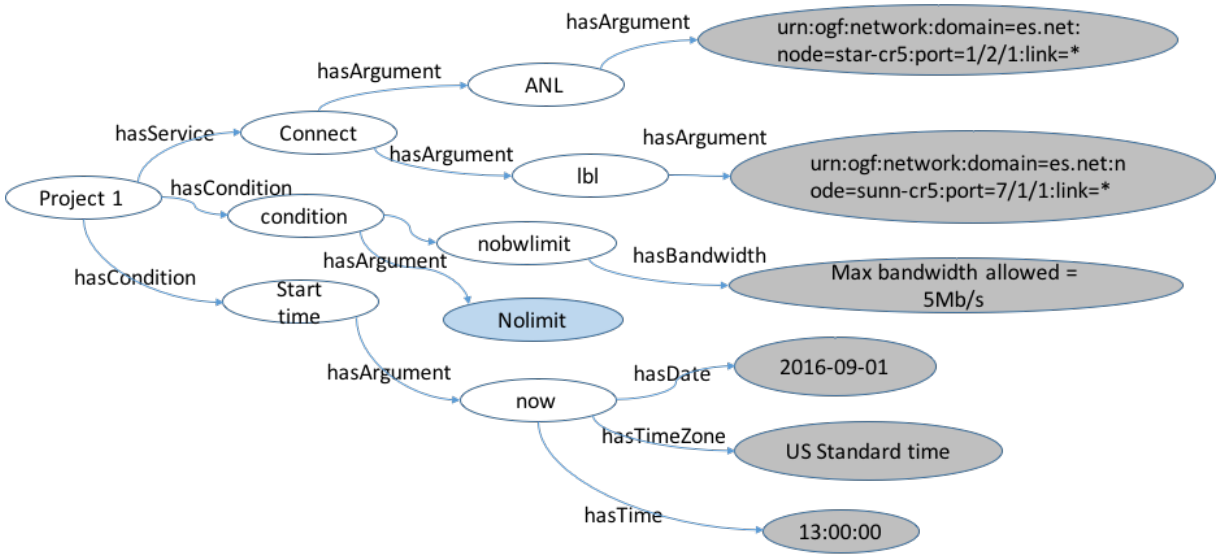

Fig. 6. iNDIRA finds relevant services and conditions, with their arguments. It then replaces all values with network commands.

- For sites ANL and LBL, these are replaced with their urn addresses.
- For 'nobwlimit' and 'nolimit' it recognizes that these are the same, so one is removed. Although the user asks for no bandwdith limit, the maximum allowed is 5 Mbps, which is then recorded.
- For time 'now', it returns current system time, the intent is recorded.
- All values are pushed on NSI client to call relevant network provisioning.

The RDF graphs can be extended for more complex relationships such as 'hasService', 'hasTime' or 'hasBandwidth'. These allow SPARQL queries to extract relevant information to reduce processing time.

C. NSI Deployment

Multi-domain deployments carry multiple provisioning issues, unique to every site involved. To tackle dynamic circuit services, the network service interface, protocols and network service agent have been developed using a standardized framework for abstracting inter-network topologies and provisions [13, 14]. Each connection is managed by an NSI module that communicates with one main provider. This provider then communicates across multiple providers and creates paths across multiple domains.

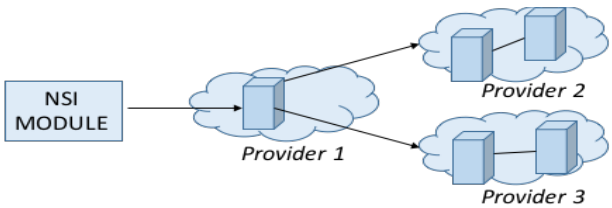

Fig. 7. NSI module communicates with a central provider server, which manages paths between multiple routers/switches/servers, distributed across multiple domains.

iNDIRA currently interfaces with NSI client for dynamic circuit service interoperability to support connectivity management. It can take inputs for bandwidth, VLAN ids, start time, end time and more; to provide multi-provider network solutions. It works with a two-way reservation system, by first asking if the resources are available and then reserving them. All of this information is communicated back to iNDIRA, where it is then translated into language to communicate with users and applications.

D. Globus File Transfer

Globus allows users to transfer bulk data securely and reliably across the science networks. Users can specify files to be moved between two Globus end-points, which have been previously verified as allowing access. However, Globus does not allow users to specify any provisioning details, schedule jobs or specify time restrictions when transferring files. iNDIRA is able to gather this information from users in advance and then communicate it with Globus. Not changing any of the underlying Globus code, iNDIRA is able to schedule future Globus jobs and specify QoS parameters such as time limits, bandwidth, and traffic isolation.

V. iNDIRA Demonstration

Using a natural conversation style [16] to intelligently reason with users and store responses into memory, iNDIRA can provide suggestions and ask for any missing input data. iNDIRA then merges this 'conversation' ability with ontology engineering to help identify and understand user needs.

The following is a negotiation example.

INDIRA> Hello and welcome! Tell me your project name?
Scientist> Bob
INDIRA> Is 'Bob' your project username? Y/N
Scientist> N
INDIRA> What is your project username

Scientist> Project_test
INDIRA> Tell me what want the network to do? Hint "I want to [connect] [disconnect] services
Scientist> I want to connect from LBL to ANL
INDIRA> Do you have any time constraints with the transfer from LBL to ANL?
Scientist> No
INDIRA> Ok. Give me the link to the file to be moved. Hint: use your globusid>foldername to destination.
Scientist> project_test@LBL/datasource1 to project2@ANL/dataset
INDIRA> Do you have any bandwidth requirements?
Scientist> Yes Maximum
INDIRA> Sorry, you have only 5Mb/s allowable to use. Do you want to use all of this?
Scientist>No
INDIRA> Thank you. Please contact the network administrator to allow you more bandwidth, as you don't have permission to go above 5Mbps.

Applications can also use iNDIRA's intent engine directly to handle network requests, by calling the engine APIs. By stating conditions, various isolated streams can be set up to prevent packet loss and schedule workflow tasks. Figure 8 shows the application defining data transfers, as future path provision and defining bandwidth restrictions for bulk transfers. These will be understood by iNDIRA and configured between the two data centers.

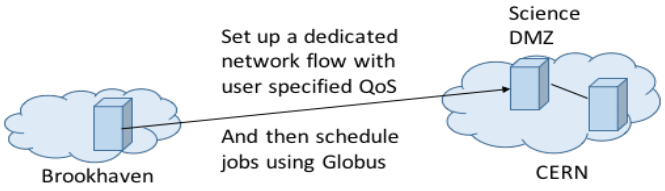

Fig. 8. Dynamically setting up paths with multiple characteristics taken as input parameters.

VI. Discussions and Evaluation

We developed the first version of the iNDIRA tool, which has its own language, parser and renderer to convert descriptive language into network commands. iNDIRA is able to perform complex rule checking, conflict resolution, provide suggestions, and has an extendable ontology schema to provide more network services in the future.

Initial experimental results of parsing intent queries are shown in Figure 9. Initially, intent processing time seems to stabilize as all possible conditions for the same topology are exhausted. However, as the topology becomes more complex, the intent processing time increases, due to more checks and topology data being processed. Although the processing time is still quite low (less than two seconds on a laptop VM), this shows that processing time will increase with bigger topologies and multi-domain setups. In future versions iNDIRA will be modified to take advantage of mutiple cores to help improve its scalability.

Currently iNDIRA provides the capability to add QoS on exisiting job schedules with Globus or NSI provisioning. Given that, this doesnot change any of the underlying tools, a delay of few seconds seems to be an overhead which can be beneficial to setting up networks paths where defined performance characteristics is difficult.

iNDIRA provides users with a higher level of control of the network, but has added software complexity. Extra time is needed for software to render requirements and translate them into network APIs. Additionally, efficient management of the semantic ontologies is needed to compensate for computational time to solve conflicts and condition checking. Similar processing can be made more efficient through use of OWL and other web ontology languages. Building such libraries, may result in more processing time and need additional checks.

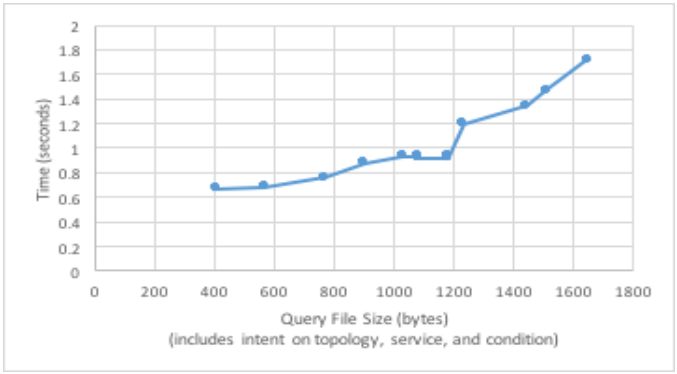

Fig. 9. Query processing time versus query file size on a single core laptop.

Working with multiple domains also provides new challenges in checking administration policies, authentication and topology information for underlying network. Currently we have all of this data for ESnet, but this may not be accessible in multi-domain situations. In order to fully optimize the network path, iNDIRA needs information on interfaces, their path capacities and bandwidth possibilities. As such iNDIRA, currently works very efficiently for science networks and research purposes where this information is available, but will need at least some configuration information between cross-domain network architectures. In future, this could be done by allowing multiple iNDIRA toolkits to communicate with each other and use local topology data available to make decisions for intent paths.

Compared to current intent projects discussed in Table 1, iNDIRA is able to show a complete toolkit, with language, checks and translation. By providing a capability to 'talk' to users and read application intents, iNDIRA is able remove network knowledge dependency for users. Unlike NIC and ONOS, it does not need multiple complex network configuration commands, it does not need users to remember complex addresses and can easily summarize multiple conditions into one intent. Through an easy to use interface, users can monitor their intent states, see how the network is responding and have more information on why their network requests 'failed'. These capabilities do not exist with any of the current intent projects [20-23].

iNDIRA's descriptive language specification is closely aligned with Boulder's. Project Boulder, has also explored

ontology developments for constructing intent for network use cases. However, Boulder, has focused on complex scenarios and not on specific applications and user intents, as iNDIRA does with science applications. In such a way, iNDIRA has conceptualized and also implemented, how science intents can be communicated to network, reducing the complexity for network engineers and scientists.

VII. Conclusions

An Intent-North Bound Interface (NBI) can allow automated, dynamic setup of paths between network providers. iNDIRA is able to recognize a scientist's needs and use an ontology-based approach to easily communicate with users, to optimally configure the network. iNDIRA provides a mechanism to improve scientific workflows, removing the manual dependency on how to setup a network circuit.

iNDIRA has great potential for future improvements. The RDF graphs allow intent to be saved dynamically and link multiple intents together. Even query processing intelligence can be added to optimize network flows after every new intent arrives. More services and complex conditions such as using AND or OR can help users create their own network scenarios to link multiple possibilities together.

iNDIRA demonstrates multiple portability advantages to the scientific community. The natural language parsing allows users to request a path using simple English commands, and questions the user for missing information. This allows network paths to be created easily with specific topologies and conditions.

We have demonstrated a working tool to orchestrate network intent into physical network provisioning commands and performing file transfers. This work provides for potential future enhancements by incorporating intelligence algorithms to predict user/application needs to configure networks for optimum use, and will be improved in future to work with commercial network providers as well.

Acknowledgment

This work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the US Department of Energy, under contract number DE-AC02-05CH11231.

References

[1] G. Malim, SDN's value is in operational efficiency, not capex control. Global Telecoms Business, June 2013.

[2] C. Wilson, NTT reaping opex rewards of SDN, August 2013.

[3] J. Wytrębowicz, T. Ries, K. Truong Dinh, S. Kukliński, SDN Controller Mechanisms for Flexible and Customized Networking, International Journal of Electronics and Telecommunications. Volume 60, Issue 4, Pages 299–307, ISSN (Online) 2300-1933, December 2014

[4] OpenFlow Switch Specification, Open Network Foundation, October 2013, version 1.4.0

[5] M. Weissenborn. ONF blogs on intent-based networking. 2016. https://www.opennetworking.org/.

[6] H. Newman, I. Legrand, A. Mughal, R. Voicu, D. Kcira, J. Bunn, High Speed Scientific Data Transfers using Software Defined Networking, Innovating the Network for Data Intensive Science (INDIS), SC 2015.

[7] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski. 2013. The Science DMZ: a network design pattern for data-intensive science. Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13).

[8] JJ Van der Ham, F Dijkstra, F Travostino, HMA Andree, Cees TAM de Laat, Using RDF to describe networks, Future Generation Computer Systems, 2006

[9] D. Chiu, S. Deshpande, G. Agrawal, and R. Li., Composing geoinformatics workflows with user preferences. ACM Int. Conf. Advances in Geographic Information Systems, 2008.

[10] D. Chiu, S. Deshpande, G. Agrawal, and R. Li. A dynamic approach toward qos-aware service workflow composition. IEEE Int. Conf. Web Services, 2009.

[11] R. Koning, P. Grosso, and C. deLaat. 2011. Using ontologies for resource description in the CineGrid Exchange. *Future Gener. Comput. Syst.* 27, 7 (July 2011), 960-965. DOI=http://dx.doi.org/10.1016/j.future.2010.11.027

[12] J. vander Ham, J.Stéger, S. Laki, Y. Kryftis, V. Maglaris, and C.deLaat. 2015. The NOVI information models. *Future Gener. Comput. Syst.* 42, 64-73. DOI=http://dx.doi.org/10.1016/j.future.2013.12.017

[13] R. Krzywania, J. Garcia-Espin, C. Guok, J.vander Ham, T. Kudoh, J. MacAuley, J. Mambretti, I. Monga, G. Roberts, J. Sobieski, A. Willner, Network Service Interface – gateway for future network services, Terena Networking Conference 2012, https://tnc2012.terena.org/core/presentation/30

[14] I. Monga, E. Pouyoul, B. Tierney, Dynamic creation of end-to-end virtual networks for science and cloud computing leveraging OpenFlow/Software Defined Networking, Terena Networking Conference 2012, https://tnc2012.terena.org/core/presentation/51

[15] E. Dart, M. Hester, J. Zurawski, Advanced Scientific Computing Research Network Requirements Review Final Report 2015, ESnet Network Requirements Review, April 22, 2015, LBNL 1005790

[16] J. Weizenbaum, ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine, Communications of the ACM, 9 (1): 36–45, 1966

[17] B. Nunes, M. Mendonca, N. Xuan-Nam, K. Obraczka, A Survey of Software-Defined Networking: Past, Present, and Future of Prog. Net, IEEE Comm. Surv & Tut, vol 16(3), Aug 2014

[18] V. Gurbani, M. Scharf, T. Lakshman, V. Hilt, and E. Marocco. Abstracting network state in software defined networks for rendezvous services. IEEE Int. Conf. Communications, pages 6627- 6632, 2012.

[19] Z. Gu and S. Zhang. Querying large and expressive biomedical ontologies. IEEE Conf. High Perf. Comp. and Comm., 2015.

[20] NEMO. Network model. http://www.nemo-project.net/

[21] ODL. Odl network intent composition. https://wiki.opendaylight.org/view/NetworkIntentComposition.

[22] ONOS. Onos intent framework. https://wiki.onosproject.org/display/ONOS/Intent+Framework.

[23] Boulder. https://github.com/ OpenNetworkingFoundation/boulder-Intent-NBI.

[24] A. Das, S. Gollapudi, E. Katchatman, O. Varol. Information dissemination in heterogeneous-intent networks. ACM Web Sc, 2016.

[25] R. Cohen, K. Barabash, B. Rochwerger, L. Schour, D. Crisan, R. Birke, C. Minkenberg, M. Gusat, R. Recio, and V. Jain. An intent-based approach for network virtualization. IFIP/IEEE Int. Sym. Integrated Network Management, pages 42- 50, 2013.

[26] H. Kim and N. Feamster, Improving Network Management with Software Defined Networking, IEEE Comm Mag Feb 2013

[27] T. Miyashita, T.Suzukim T. Soumiya and A. Yamada, SDN solution for Wide area networks, Fujtsu Science Technology Journal, Vol 52, no2, pp 28-34, Apr 2016.

[28] G. Malim. SDN's value is in operational efficiency, not capex control. Global Telecoms Business, June 2013.

[29] C. Guok, D. Robertson, E. Chaniotakis, M. Thompson, W. Johnston, and B. Tierney. A user driven dynamic circuit network implementation. Proc. Distrib. Autonomous Network Mgmt. Sys. Wksp., November 2008.

[30] Openstack. Prescriptive examples. 2016. http://docs.openstack.org/arch-design/multi-site-prescriptive-examples.html

[31] A. Das, S. Gollapudi, E. Katchatman, and O. Varol. Information dissemination in heterogeneous-intent networks. ACM Web Science, 2016.

[32] C. Rothenberg, R. Chua, J. Bailey, M. Winter, C. Correa, S. Lucena, M. Salvador, and D. Thomas. When open source meets network control planes. Computer Jour., 47(11):46–54, 2014.

[33] M. Bari, S. Chowdhury, R. Ahmed, and R. Boutaba. Policycop: An autonomic qos policy enforcement framework for sdn. IEEE SDN for Future Net. and Serv., 2013.

[34] H. Kim and N. Feamster. Improving network management with software defined networking. IEEE Comm. Mag., 51(2):114–119, 2013.

[35] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri. Information centric networking over sdn and openflow: Architectural aspects and experiments on the ofelia testbed. Int. Jour. Comp. and Tele. Net. archive, 57(16):3207–3221, 2013.

[36] I. Foster, Globus Toolkit Version 4: Software for Service-Oriented Systems, Network and Parallel Computing, vol 3779 LNCS, 2005