SDN for End-to-end Networked Science at the Exascale (SENSE)

Inder Monga
Energy Sciences Network
Lawrence Berkeley National Lab
Berkeley, CA, USA
imonga@es.net

Chin Guok
Energy Sciences Network
Lawrence Berkeley National Lab
Berkeley, CA, USA
chin@es.net

John MacAuley
Energy Sciences Network
Lawrence Berkeley National Lab
Berkeley, CA, USA
macauley@es.net

Alex Sim
Energy Sciences Network
Lawrence Berkeley National Lab
Berkeley, CA, USA
asim@lbl.gov

Harvey Newman
Division of Physics, Mathematics and Astronomy
California Institute of Technology
Pasadena, CA, USA
newman@hep.caltech.edu

Justas Balcas
Division of Physics, Mathematics and Astronomy
California Institute of Technology
Pasadena, CA, USA
jbalcas@caltech.edu

Phil DeMar
Computing Division
Fermi National Accelerator Laboratory
Batavia, Illinois, USA
demar@fnal.gov

Linda Winkler
Computing, Environment and Life Science Division
Argonne National Lab
Argonne, Illinois USA
winkler@mcs.anl.gov

Tom Lehman
Mid-Atlantic Crossroads
University of Maryland
College Park, MD USA
tlehman@umd.edu

Xi Yang
Mid-Atlantic Crossroads
University of Maryland
College Park, MD USA
maxyang@umd.edu

Abstract— **The Software-defined network for End-to-end Networked Science at Exascale (SENSE) research project is building smart network services to accelerate scientific discovery in the era of 'big data' driven by Exascale, cloud computing, machine learning and AI. The project's architecture, models, and demonstrated prototype define the mechanisms needed to dynamically build end-to-end virtual guaranteed networks across administrative domains, with no manual intervention. In addition, a highly intuitive 'intent' based interface, as defined by the project, allows applications to express their high-level service requirements, and an intelligent, scalable model-based software orchestrator converts that intent into appropriate network services, configured across multiple types of devices. The significance of these capabilities is the ability for science applications to manage the network as a first-class schedulable resource akin to instruments, compute, and storage, to enable well defined and highly tuned complex workflows that require close coupling of resources spread across a vast geographic footprint such as those used in science domains like high-energy physics and basic energy sciences.**

Keywords—Intent based networking, multi-resource orchestration, intelligent network services, distributed infrastructure, resource modeling

I. Introduction

Network designs are evolving at a rapid pace toward programmatic control, driven in large part by the application of software to networking concepts and technologies, and evolution of the network as a key subsystem in global scale systems, such as those serving major science collaborations that incorporate large scale distributed computing and storage subsystems. This software-network innovation cycle is important as it includes a vision and promise for greatly improved automated control, configuration and operation of such systems, in comparison to the labor-intensive network deployments of today. However, even the most optimistic projections of software adoption and deployment do not put networks on a path that would make them behave as a truly smart or intelligent system from the application or user perspective, nor one capable of interfacing effectively with facilities supporting highly automated data analysis workflows at sites located across the world.

Today, domain science applications and workflow processes are forced to view the network as an opaque infrastructure into which they inject data and hope that it emerges at the destination with an acceptable Quality of Experience. There is little ability for applications to interact with network to exchange information, negotiate performance parameters, discover expected performance metrics, or receive status/troubleshooting information in real time. As a result, domain science applications frequently suffer poor performance, especially so in highly distributed environments. Indeed, the ability for a science application to interact and negotiate with network infrastructure within a science ecosystem, should be a hallmark of truly smart networks and smart applications. It seems clear that current

978-1-7281-0194-1/18/$31.00 ©2018 IEEE
DOI 10.1109/INDIS.2018.00007

static, non-interactive network infrastructures currently do not have a path forward to assist or accelerate domain science application innovations.

We therefore envision a new smart network and smart application ecosystem that will solve these issues and enable future innovations across many Research and Education domain science communities. The SDN for End-to-end Networked Science at the Exascale (SENSE) [1] project has developed an architecture and implementation to address this vision. The high-level vision for this new application to network interaction paradigm includes the following:

- Intent Based: Abstract questions, requests and responses in the context of the application objectives
- Interactive: An ability to ask question and negotiate
- Real-time: Resource availability, provisioning options, service status, troubleshooting on a real-time scale of seconds to minutes.
- End-to-End: including multi-domain networks, end sites, and the network stack inside the end systems
- Full Service Lifecycle Interactions: continuous conversation between application and network for the service duration of service

We believe that these types of network services will be needed to support new workflows driven by Exascale resources. It is expected that existing workflows involving multiple network data transfers will be replaced by the establishment of deterministic network paths to support realtime data streaming directly to compute memory or burst buffers. This mode of operation will also support computational steering where instruments utilize data streaming and preliminary compute results to calibrate and guide experiments in realtime.

In addition to this new paradigm for application network interaction, the SENSE system also solves a variety of practical problems which are commonplace in Research and Education (R&E) cyberinfrastructure systems as well as experienced in deployment of multi-domain virtual circuits [2], as noted below.

Distributed scientific workflows need end-to-end automation so the focus can be on science, as opposed to infrastructure operations:

- Manual provisioning and infrastructure debugging require excessive time and is human capital intensive
- There is little to no service consistency across domains
- Service visibility and multi-domain automated troubleshooting is almost non-existent
- Lack of realtime information from domains impedes development of intelligent services

Science application workflows have not integrated with network service provisioning paradigms because:

- Network programming APIs are usually not intuitive and require detailed network knowledge, most not easily obtained
- Efficient resource utilization awareness and management is very difficult

Multi-domain orchestration and automation requires service visibility and troubleshooting:

- Data APIs across domains are needed for applications, users, and network administrators
- Mechanisms for authorized agents to obtain performance, service statistics, topology, capability, and other data is needed.
- This will require systems for the exchange of 'policy-scoped' and authorization information

The remainder of this paper will describe the SENSE Services (Section II), Architecture (Section III), Testbed Deployment (Section IV), Use Case Experimentation Results (Section V), and Summary with Future Plans (Section VI).

II. SENSE Services

The motivating focus for the SENSE project is the interaction between the application and the network. Therefore, network services are discussed here, prior to discussion of system architecture or implementation. As will be described in subsequent sections, new methods and techniques for network resource management and control had to be developed in order to realize the application facing functions. Starting the design process from a user services context provided the rationale for associated system designs. The term Application Workflow Agent (AWA) is often used to refer to the entity interacting with the SENSE system. It is expected that the particulars of where this AWA fits into the actual application architecture will vary by use case. Often it will be a middleware component which is providing services to the actual user and managing a diverse set of cyberinfrastructure resources. From a SENSE system perspective, the AWA is the entity requesting network services. The remainder of this paper utilizes the term application and AWA interchangeably.

The longer-term vision articulates a smart networked ecosystem where the network is an interactive component used by similarly featured smart applications, security systems, and other domain specific workflow agents. This intelligent network service plane forms the boundary layer between the smart network and the smart application. Application workflow agents can engage and obtain services from the smart network system, through interactions with this boundary layer. In this context, the following key features of intelligent network service plane are identified:

- Intent: The ability for an application to submit a service request in the form of a high-level statement of desired results or outcomes, as opposed to a specific set of network centric inputs [3].

- Interaction: The ability for an application workflow agent to engage in a bi-directional exchange aka "conversation" with the network as part of workflow planning. This conversation can include discovery of available services, asking "what is possible" or "what do you recommend" types of questions, engaging in iterative negotiations prior to actual service requests, or full-service life-cycle status and troubleshooting queries.

The expectation is that the intelligent network sservice plane will enable multiple new operational paradigms, including a fundamentally new concept of "consistent network experience". This is where stable load balanced high throughput workflows cross optimally chosen network paths, and only up to a preset high water marks to accommodate other traffic. This will be provided through automated interaction between the application workflow agents and the intelligent network service plane, responding to demands from the science programs' principal data distribution and management systems. The result will be a "consistent outcome" and "deterministic" (or more deterministic) end-to-end system performance.

The SENSE system has been developed to operate in "Development Operations (DevOps)" mode, where custom services can be rapidly developed in response to individual application requirements. The general system philosophy is that while not "every" service imaginable can be implemented, almost "any" service can be. That is, the system design is such that resource states and capabilities are sufficiently available to allow the construction of many different services. The user requirements will be utilized to form the basis of the actual services.

To add some more specificity to these ideas of smart network services, below is a description of some of the initial services which have been implemented by the SENSE project.

- Time-Block-Maximum Bandwidth: Application asks for a specific time block and would like to know (or provision) the maximum bandwidth available for a specific time period.
- Bandwidth-Sliding-Window: Application asks for a specific bandwidth and duration and provides an acceptable time window. For example, a request may be for 40 Gbps for a 10-hour time window, sometime in the next 3 days.
- Time-Bandwidth-Product (TBP): Application asks for "8 hours of transfer at 10Gbps" representing a TBP of 36 TBytes. The user also specifies an acceptable time window, and other options such as "prefer the highest bandwidth rate available", or the lowest.

For each of these services, the user can interact with SENSE in the following modes:

- Immediate Provision: If SENSE finds a resource path which satisfies the application request, provisioning starts immediately (after routine confirmations from both sides).
- What is Possible?: In this mode, SENSE simply conducts a "Resource Computation" and provides the results back to the requestor. No provisioning action is taken without further explicit requests from the user.
- Negotiation: One or more rounds of Resource Computation requests with subsequent provisioning request by the application user if desired.

In the context of SENSE services, the "network" includes the switching and routing elements AND the network stacks of the end systems, such as Data Transfer Nodes inside Science DMZ facilities. The data plane capabilities associated with these services are:

- Layer 2 point-to-point with QoS
- Layer 2 multi-point with QoS
- Layer 3 Flow QoS

Additional details regarding these (and other) services, the supporting system architecture, use case integration, and testing results are provided in the subsequent sections.

III. SENSE Architecture

The SENSE approach to end-to-end at-scale networking is based on software programmability and intelligent service orchestration. The SENSE orchestration architecture provides many intelligence, performance and assurance benefits through application oriented services. These are enabled by some novel technologies, including a) hierarchical service-resource architecture, b) unified network and end-site resource modeling and computation, c) model based realtime control, d) application driven orchestration workflow, and e) end-to-end network data collection and analytics integration.

A. Hierarchical Servcie-Resource Architecture

Within the SENSE orchestration architecture there are two distinct functional roles, Orchestrators and Resource Managers (RM). The interaction of Orchestrator(s) and RM(s) follows a hierarchical workflow structure whereby of the Orchestrator accepts requests from users or user applications, and determines the appropriate resource managers to contact and coordinate the end-to-end service request. The RMs are (administrative or technology) domain specific and are responsible for committing and managing local resources.

As illustrated in Fig. 1, this hierarchical structure of Orchestrator and RM components separates application facing service control functions from infrastructure facing resource control functions.

a) SENSE Orchestrator

The SENSE Orchestrator (SENSE-O) is expected to be closely associated with a domain science

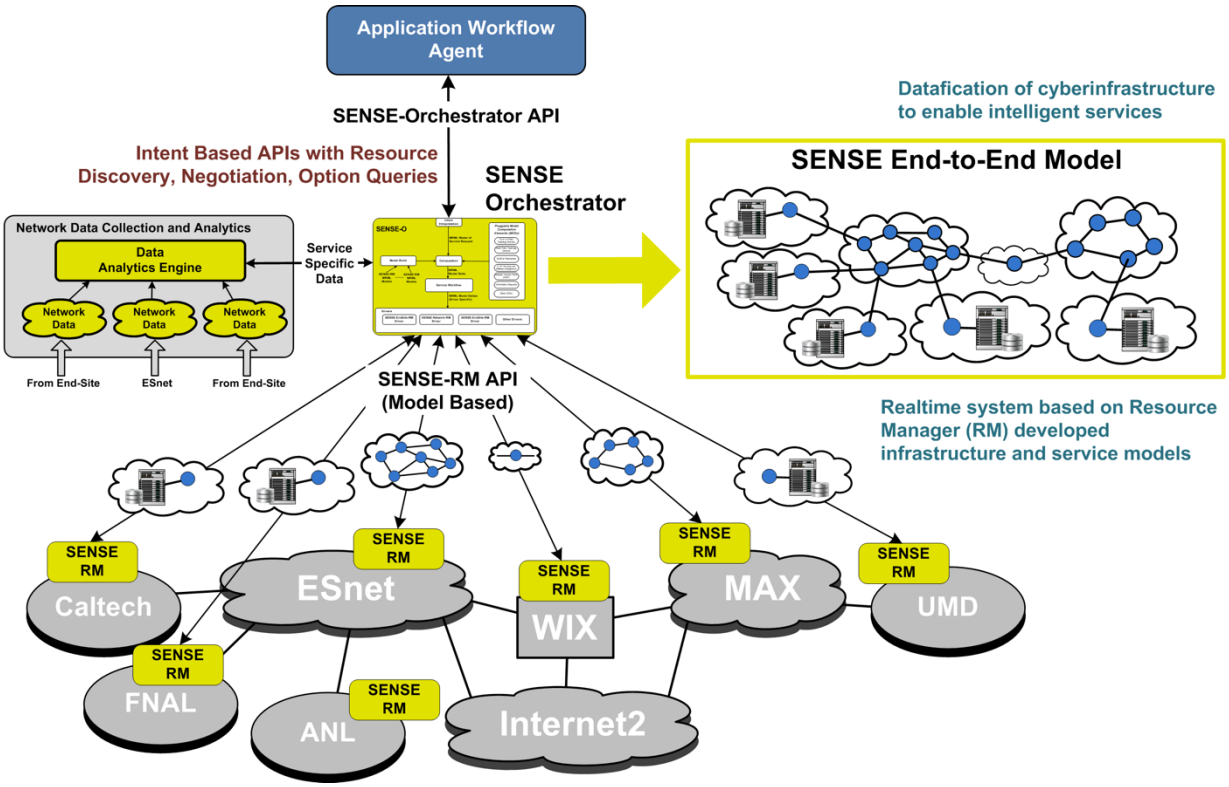

Figure 1 SENSE Architecture

collaboration/application (e.g. LHC/CMS [4], ExaFEL [5],etc) and processes "high-level" context sensitive intents to determine what resources are needed and coordinate the requests of "lower-level" (or sub) intents to the corresponding RMs. As such, the Orchestrator performs the following functions:

- Receives model-based resource descriptions from multiple RMs
- Receives and responds to the user's "high-level" intent requests (which is defined within the context of the user's domain science collaboration/application).
- Renders the user's "high-level" descriptive intent request into "low-level" prescriptive requests for required resources
- Performs multi-constraint resource computation (based on AuthN/AuthZ, resource availability, etc.) to determine the appropriate and necessary resources needed and corresponding RMs to contact.
- Coordinate requests and replies from RMs and feedback the results to the user accordingly.
- Support queries by the user for status and state.
- Provide resource notifications to users as necessary.

The SENSE-O can take on different functionality custom to the domain science needs and resources available to it (e.g. experiment, compute, storage, network, etc.). In the SENSE project, we are building a reference implementation that is specific to the big science models, controlling primarily data transfer and network resources.

b) SENSE Resource Manager

The SENSE Resource Manager (SENSE-RM) is tied to a domain with physical resources, e.g. a WAN (with network resources), a site (with Science DMZ resources), etc., and is responsible to facilitate the management of domain-specific resources. The SENSE-RM is responsible for the following functions:

- Provides (appropriately scope and abstracted) model-based resource descriptions
- Receives and responds to the "low-level" intent requests from the Orchestrator.
- Performs multi-constraint resource computation (based AuthN/AuthZ, resource availability, etc.) to determine the local resources appropriate and necessary to service the request.

- Coordinates resource allocations/commitments, provisioning and de-provisioning with local controllers (e.g. NMS, etc.) as necessary.
- Supports queries by the Orchestrator for status and state.
- Provides resource notifications to the Orchestrator as necessary.

The SENSE-RMs are specific to an administrative domain. However, within a single administrative domain, multiple instances of RMs may be deployed based on the distinct technology regions (e.g. DTNs, optical PKT/OTN, L2 OpenFlow, etc.). Conversely, a SENSE-RM may model multiple technology domains as a single resource description. For example, a network may have distinct switches and routers which provide layer 2 and layer 3 services correspondingly. However, the domain may instantiate a single RM which provides a unified resource description characterizing both sets of resources.

c) Many-to-Many Relationship between SENSE-O and SENSE-RM

We should not confuse SENSE-O with a central orchestration service for all applications. Instead, it is an architectural component that can have many instances independently serving different organizations, collaborations and application groups. In the SENSE orchestration model, it is expected that the many SENSE-O instances will communicate with multiple SENSE-RMs. The primary reason is that each scientific collaboration or workflow may have unique security, resource computation and specific resource allocations. For instance, one collaboration may use Shibboleth as its access and identity framework, whereas another may use Kerberos. Having distinct SENSE-O instances allows each collaboration to implement fine-grain AuthN and AuthZ functions in accordance with its security or resource allocation policies. Each SENSE-O instance in-turn has a unique trust relationship with the SENSE-RMs that it communicates with. This facilitates scalability in that a SENSE-RM does not need to manage all end-user authentication, authorization, and access to resources within its domain, but can enforce coarse-grain policies against the identity of the requesting SENSE-O instance and the negotiated Service Level Agreement (SLA).

In addition, different collaborations may have access to different resources within a SENSE-RM's domain. For instance, one collaboration may be restricted to a certain set of network links, whereas another collaboration may not have the same constraint. By having distinct SENSE-O instances per collaboration, a SENSE-RM may publish different resource descriptions based on SLAs that it has with the SENSE-O instance. The SENSE-O instance in turn may perform resource computation and allocation with priorities and constraints that are unique to the collaboration.

B. Orchestrator Northbound API and Services

From user services perspective the SENSE orchestrator provides application services via a programmable northbound interface, namely SENSE-O NBI. While the Orchestrator Core can support modular intelligence computation and almost arbitrary orchestrated services, exposed through the SENSE-O NBI is a select intent based API with emphasis on end-to-end network connection discovery and computation, featured with intelligent bandwidth and schedule negotiation and workflow assistance.

a) Examplary Connection Services for Quality of Experience

SENSE-O NBI service intent is defined in JSON format. The below example with service_alias "sc18-p2p-b1" is to request a 10G connection with hard capped bandwidth QoS between two DTN sites at NERSC and Caltech. An alternative service_type "*Multi-Point VLAN Bridge*" could be used to request for a VLAN connection of three and more terminals.

```
{"service_type": "Multi-Path P2P VLAN",
 "service_alias": "sc18-p2p-b1",
 "connections": [
  { "name": "connection 1",
    "terminals": [

{"uri":"urn:ogf:network:nersc.gov:2013:server+dtn11.nersc.gov",
       "label": "any"
      },
      {"uri":
     "urn:ogf:network:caltech.edu:2013:server+xfer-
2.ultralight.org",
       "label": "any"
      } ],
    "bandwidth": {
      "qos_class": "guaranteedCapped",
      "capacity": "10",
      "unit": "gbps"
    } } ] }
```

Our supported QoS classes include guaranteedCapped (no burst over capped limit), softCapped (allowing for burst over the cap when extra bandwidth is available) and bestEffort. For users who are not sure how much bandwidth to ask but want to firstly query for the maximum available, the intent can include a query statement as in the example "sc18-p2p-b2".

```
{"service_type": "Multi-Path P2P VLAN",
 "service_alias": "sc18-p2p-b2",
 "connections": [
  { "name": "connection 1",
            --- skipped content ---
    "bandwidth": {
      "qos_class": "guaranteedCapped"
    } ],
   "queries": [
   {"ask": "maximum-bandwidth",
```

```
  "options": [
   {  "name": "connection 1"
   } ] } ] }
```

Bandwidth QoS only represents one aspect of quality of experience for data transfer application users. Many users also want deterministic or predictable time schedule for data transfer. In the example with service_alias "sc18-p2p-bs", we introduce the schedule intent, which asks for the same 10G connection lasting for 4 hours that can be scheduled flexibly within next 2 days. This particular intent is called a "bandwidth-sliding-window". SENSE-O NBI also support the intent of "bandwidth-fixed-window" and "maximum-bandwidth-query-with-fixed-window" in its variation forms.

```
{"service_type": "Multi-Path P2P VLAN",
 "service_alias": "sc18-p2p-bs",
 "connections": [
  { "name": "connection 1",
            --- skipped content ---
    "bandwidth": {
      "qos_class": "guaranteedCapped",
      "capacity": "10",
      "unit": "gbps"
    },
    "schedule": {
      "start": "now",
      "end": "+2d",
      "duration": "+4h"
    } } ] }
```

Another interesting service intent is based on concept of Time-Bandwidth Product (TBP). For 8 hours of transfer at bandwidth of 10Gbps, the TBP represents 36000 Gbits or 36 TBtyes of data volume. Allowing users to query and negotiate with bandwidth and schedule based on a given TBP will provide better quality of experience as TBP is a good estimate of total amount of data to transfer. In the example intent "sc18-p2p-tbp1", user tries to find a schedule to transfer an estimated 10000 Mbytes (or 10 GB) data within a 2 day time window after October 1st 2018 8:00ET. The user wants to check for the fastest possible transfer speed using a " use-highest-bandwidth = true" option.

```
{"service_type": "Multi-Path P2P VLAN",
 "service_alias": "sc18-p2p-tbp1",
 "connections": [
  { "name": "connection 1",
            --- skipped content ---
    "bandwidth": {
      "qos_class": "guaranteedCapped",
    }
 } ],
 "queries": [
  {"ask": "time-bandwidth-product",
   "options": [ {
       "name": "connection 1",
       "tbp-mbytes": "10000",
       "start-after": "2018-10-01T08:00:00.000-0400",
       "end-before": "+2d",
       "use-highest-bandwidth": "true"}
 ] } ] }
```

Alternatively, user can ask for least bandwidth (or widest schedule) using a "use-lowest-bandwidth = true" option, or a bandwidth-bounded schedule using both "bandwidth-mbps >=" and "bandwidth-mbps <=" options. The latter will return a feasible schedule that satisfies both the time-bandwidth-product and the bandwidth upper and lower bounds.

b) Service Negotiation Workflow via Intent Based API

SENSE-O NBI offers a set of other intent API calls for service and resource discovery. But central to the end-to-end connection service orchestration are calls for intent based service negotiation and instantiation workflow. Herewith we only go through the service creation calls by their order in the workflow and skip those for service cancellation, modification and monitoring.

1. *Create Service Instance*

POST $*service_intent_v1* to **/sense/service**

This creates a service instance to persist session context. SENSE-O will compile and compute the initial service intent. When questions are asked in "queries" statement, it will provide the first answers to asked questions.

Use the above service intent "sc18-p2p-b2" as example. The request for "connection 1" has

```
 "bandwidth": {
     "qos_class": "guaranteedCapped"
   }
```

and

```
  "queries": [
  {"ask": "maximum-bandwidth",
   "options": [
    {  "name": "connection 1"
    } ] } ]
```

In the reply upon successful computation, we shall see something like

```
   "bandwidth": {
     "qos_class": "guaranteedCapped",
     "capacity": "10000",
     "unit": "mbps"
   }
```

and

```
  "queries": [
  {"asked": "maximum-bandwidth",
   "options": [
    {  "name": "connection 1",
     "bandwidth": "10000",
     "unit": "mbps"
    } ] } ]
```

Also included in the reply is the full text of computed service model in MRML language, which we will discuss in later sections.

2. Service Negotiation Rounds

POST $*service_intent_v2-N* to **/sense/service/$svc_uuid**

When a user has more questions to negotiate with SENSE-O, it will revise the intent and post a newer version to the same service session identified by the service instance ID ($svc_uuid) found in the reply from the initial service creation call. Following the above example, user knows the maximum bandwidth is 10Gbps for the requested end-to-end connection, however, this only applies to current time and gives the user rough idea of possible network capacity. Then it could negotiate for a feasible schedule in a sliding window for a time-bandwidth product that is bounded by maximum and minimum allowed bandwidth as follows.

```
"queries": [
 {"ask": "time-bandwidth-product",
  "options": [ {
      "name": "connection 1",
      "tbp-mbytes": "1000000",
      "start-after": "now",
      "end-before": "+2d",
      " bandwidth-mbps <=": "10000",
      " bandwidth-mbps >=": "2000"}
] } ] }
```

The reply could be

```
"queries": [
 {"ask": "time-bandwidth-product",
  "options": [ {
      "name": "connection 1",
      "bandwidth": "5000",
       "unit": "mbps",
       "start ": "2018-9-01T10:00:00.000-0400",
       "end ": "2018-9-01T10:26:40.000-0400" }
] } ] }
```

This means when asked for a TBP of 1 Terabytes to be transferred within next 2 days with acceptable bandwidth between 2 and 10Gpbs, SENSE-O provided a feasible solution for a transfer between 10:00:00 and 10:26:40 ET on September 1st 2018 at a speed of 5 Gbps. Step 2 can be performed for many rounds until user is satisfied with the reply or gives up.

3. Service Reservation

PUT to **/sense/service/$svc_uuid/reserve**

or

POST $*final_intent* to **/sense/service/$svc_uuid/reserve**

Once user is settled with the final intent, it could use PUT method to reserve the service, which refers to the reply of last round of negotiation as final intent. Or it could use a POST method to provide a last version of service intent with some final edits. The reserve call will propagate the service request through the SENSE-O core and to all involved SENSE-RMs. This is a transactional operation, meaning the SENSE-O and all involved SENSE-RMs must agree on the service and lock up the required resources. Otherwise, a complete rollback will be performed with none resource being held after. Such a transaction will be mostly data verification and database operation and can be done very quickly.

4. Service Commit

PUT to **/sense/service/$svc_uuid/commit**

In the commit step, the resources held by reserve call will be actually allocated. Compared to a "soft" reserve that is mostly a database operation, the commit call is "hard" operation, which can take quite long time for some resources. SENSE-O NBI offers both synchronous and asynchronous methods to execute the commit call.

5. Service Status Query

GET /sense/service/$svc_uuid

This call can tell user the current status of a service instance in progress. This is particularly useful for checking status of an asynchronous commit call.

The complete intent API document for SENSE-O NBI is published at [6].

C. Orchestrator To Resource Manager API

From resource providers perspective the SENSE RM API provides a means to integrate separated and diverse resource domains into the SENSE orchestration. It helps form the many-to-many relationships between SENSE-Os and SENSE-RMs.

The REST-based Orchestrator to RM API defined works on the fundamental principle of topology model manipulation. The Orchestrator queries the RM for a current view of topology available for use. The Orchestrator manipulates the provided topology to achieve its target goal, computes a delta between the original topology and the desired topology, and then proposes this resulting delta to the RM. The RM may accept or reject the proposed delta depending on a number of criteria including validity, local usage policies, resource availability, etc. If the RM accepts the delta the Orchestrator must then commit the change before the RM will apply the change to targeted resources.

a) MRML Resource Modeling

The SENSE-RM API is based on a resource model exchange and manipulation paradigm. The SENSE-O queries multiple RMs for a resource model which describes the infrastructure and services available for use. The resource model provided by each RM includes a description of its local resources and includes a definition for their interconnects to external resources. This external connection information allows the SENSE-O to build a model based connected graph which includes all of the RMs in its query space. This end-

to-end model based graph provides the basis for the SENSE-O to respond to user requests and construct workflows for service provisioning interactions with the proper RMs. In the orchestrator, Modular Computation Elements (MCEs) provide the mechanisms to translate high level intent based user requests into specific workflow orchestration steps and resource requests to individual SENSE-RMs.

The SENSE-O receives resource descriptions from SENSE-RMs and constructs a model based graph of the end-to-end SDN topology. The modeling framework is based on extensions to the Network Markup Language (NML) [7] standard developed by the Open Grid Forum (OGF) [8]. As part of a DOE ASCR research project, RAINS [9], extensions to NML were defined to allow other resource types in addition to network elements/topologies to be described and modeled. The base NML standard and these extensions define the Multi-Resource Markup Language (MRML) [10] which is utilized as the basis for resource modeling in the SENSE project. In this context, other resource types may include systems that are connected to the network such as Data Transfer Nodes (DTNs) [11], storage systems, instruments, and compute nodes.

b) Model Driven Realtime Resource Management

The topology of resources by each RM is an MRML document. As the topology and resource states change along time, SENSE-RM needs to manage a serial version of the MRML document. This versioned model document defines all the semantics for the SENSE-O API. Therefore, the API operations will be extremely reduced into two: model pull and delta push. The latter is divided into two methods, propagate and commit, to support a transactional two-phase push process. The model driven approach and simplicity of API methods helps SENSE to achieve better scalability. In the project, we also emphasize on another important performance metric: real-time. We will discuss what it means for end-to-end resource integration and service orchestration.

- Pull Model
  - The SENSE-O receives a model-based resource description from all of the RMs in the end-to-end SENSE ecosystem. The SENSE-O will integrate models from multiple SENSE-RMs to generate a multi-domain resource description model.
  - The individual SENSE-RMs will utilize local policy to determine what information is provided with regard to resources, abstraction degree, and any other factors based on use cases associated with an individual SENSE-O.
  - On the current SENSE Testbed, SENSE-O is customized to pull RM models every 30 seconds. The HTTP "If-Modified-Since" mechanism is used to reduce redundant data pull. SENSE-RMs will be responsible for tuning up abstraction degree and resource update frequency to satisfy the "real-time" requirement by SENSE-O, and also suppress excessive control traffic across the RM API.
- Propagate Delta
  - The SENSE-O will process intent based service requests from the SENSE-O API and generate a "model delta" which will be used to communicate a potential action/provision request to the SENSE-RM(s). The SENSE-RM is not expected to take any provisioning action based on the Propagate Delta method.
  - In response to the Propagate Delta method, the SENSE-RM should inspect, verify, and confirm the request of suggest revisions. For example, a specific VLAN may be requested in the Propagate Delta method, however, the SENSE-RM would prefer another VLAN. In this case the SENSE-RM should indicate the modified VLAN requests in the response via modifying the provided "model delta".
  - As the propagate call is purely data transactions, it could be executed very fast. A negotiation procedure has been built into this phase such that multiple rounds of fast propagate and feedback transactions can be performed to achieve an updated real-time result that may be different than the original "delta". This real-time negotiation and update is necessary as the SENSE-O and SENSE-RM are in many-to-many loosely coupled relationship and may not always have completely "real-time" synchronization on resource states.
- Commit Delta
  - The SENSE-O uses this method to ask the SENSE-RM to commit the changes negotiated as part of the Propagate Delta exchange(s).
  - This is where the SENSE-RM is expected to actually provision resources. As this procedure is normally time-consuming, it is separated from the transactional propagate method. The SENSE-RM API commit is always asynchronous so that none SENSE-O call to the SENSE-RM would be blocked for long. Real-time status query is supported to check result of the asynchronous commit.

D. Intelligent Orchestrator Core and Model Driven Computation

The core of SENSE-O is StackV[12], a general-purpose open-source orchestrator for networked multi-services. StackV is implemented based on the full-stack model driven intelligent orchestration approach. From very top of the stack, applications communicate to the orchestrator with abstract service intent. Intents including those specifically for SENSE take different forms for convenience of users. The SENSE-O NBI translates service intent into so called "Service Model Description and Abstraction", which is a formal MRML model that consists of abstract resources annotated with

service policy statements. The abstract model data are then fed to a dynamic compile procedure and compiled into a model-based computation workflow. A computation workflow consists of a variety of Model Computation Elements (MCE) as intelligence functions assembled into an execution tree. Each MCE uses system model data, service model data and policy data as input and accomplishes a specific function such as resource placement and connection computation. The output will be more detailed service model data, which could be used as input for another MCE. When the computation workflow finishes successfully, a System Model Delta will be created that provides detailed model statements about what need to change in the underlying infrastructures governed by RMs to satisfy the intent.

The benefits of model-based computation include eliminating conversion between interface, internal and persistence data structures, leveraging standard tools for data query, navigation, transformation and reasoning, and maintaining consistent data semantics through all the computation modules. MCE is the basic computation module. The input and output of an MCE are both model data based on RDF/OWL, MRML and policy ontologies. In the compiled computation execution workflow, each MCE instance computes for a specific purpose. For an example SENSE service, a Layer-2 VLAN Connection MCE takes in the initial service abstraction model that specifies connection terminals, bandwidth and schedule parameters. It creates model statements for end-to-end layer-2 connection across end sites and wide area networks, in an updated service abstraction model and exports it together with some intermediate policy data. Then a Layer-3 Address Assignment MCE uses this new service abstraction model and policy data (more detailed than the original one) as input to perform its own computation and add layer-3 modeling statements to the further updated service abstraction model. StackV has implemented sophisticated logic to concatenate MCEs and merge computation results. The basic idea of this technique is to use SPARQL [13] queries to "shape" the output of a upper stream MCE into custom JSON format and use JSONPath [14] queries to extract information and "fit" to the input of downstream MCEs. Success in finishing the computation workflow means StackV has resolved all model abstractions and policy annotations in the final product and has turned an application intent into a System Model Delta. This "delta" can be pushed down to the SENSE-RM API for instantiation.

E. Network Data Collection and Analytics Integration

Topological model and resource states are the basis for the SENSE-O intelligent computation for orchestration services. Further integration of real time and historical network data through analytics engine provides improved quality of experience for users through better understanding of end-to-end network states and more precise prediction of traffic trends. The analytics-based feedback also helps users better describe their service intents to the Orchestrator. An extended SENSE architecture includes a Data Analytics Engine that collects network data from end sites and transport networks, and provides analytics pre-processing and feedback to the orchestrator core. It collects extensive telemetry data from various monitoring and active measurement sources that reflect network resource utilization and real-time states. This data collection and analytics capability is not yet in place and is anticipated as part of future work. A description is included in order to fully explain the architectural vision.

The Data Analytics Engine is a component external to the SENSE-O. In the current SENSE Testbed, ESnet and many DTN end sites have deployed some sort of monitoring and data collection and archiving mechanisms. The planned SENSE analytics solution will consolidate these existing resources into a functional utility engine that has distributed data collection, archiving and access endpoints but common API and data schema definitions.

Following the suit of model driven API design, the interaction between the Data Analytics Engine and SENSE-O will be formally modeled Service Specific Data exchanged through a well-defined API. With per-user and per-service ownerships being annotated upon the collected data, data contents and formats will be customized on demand based on service orchestration needs. In addition, the analytics data will be integrated with the existing MRML model through abstraction, reference and annotation processing. Finally, modified and new MCEs will be able to leverage the custom, pre-processed, MRML friendly data from the analytics engine to compute improved results for existing service intents and provide answers to many new intent questions that we could not easily answer today.

The Service Specific Data bridge across the Analytics Engine and SENSE-O forms a closed control-feedback loop. The orchestration results will be monitored and measured and provided as feedback for fine tuning of future orchestration computation. On the other hand, the SENSE-O also provides information to the Analytics Engine to help verify and instrument the data collection and analysis more efficiently.

IV. TESTBED DEPLOYMENT

A SENSE testbed has been deployed which includes a mix development and production resources. A shown in Figure 2, this testbed is deployed at DOE Laboratories and Universities facilities. For the network resources the SENSE system interacts with production provisioning system ESnet and other networks. For the end-system resources, a mix of production and prototype DTNs are deployed. For the production DTNs limited access is provided resulting in tailoring the set of SENSE based dynamic configurations to match local site polices. This approach to use a mix of production and research resources enables experience with various real-world site deployments and considerations.

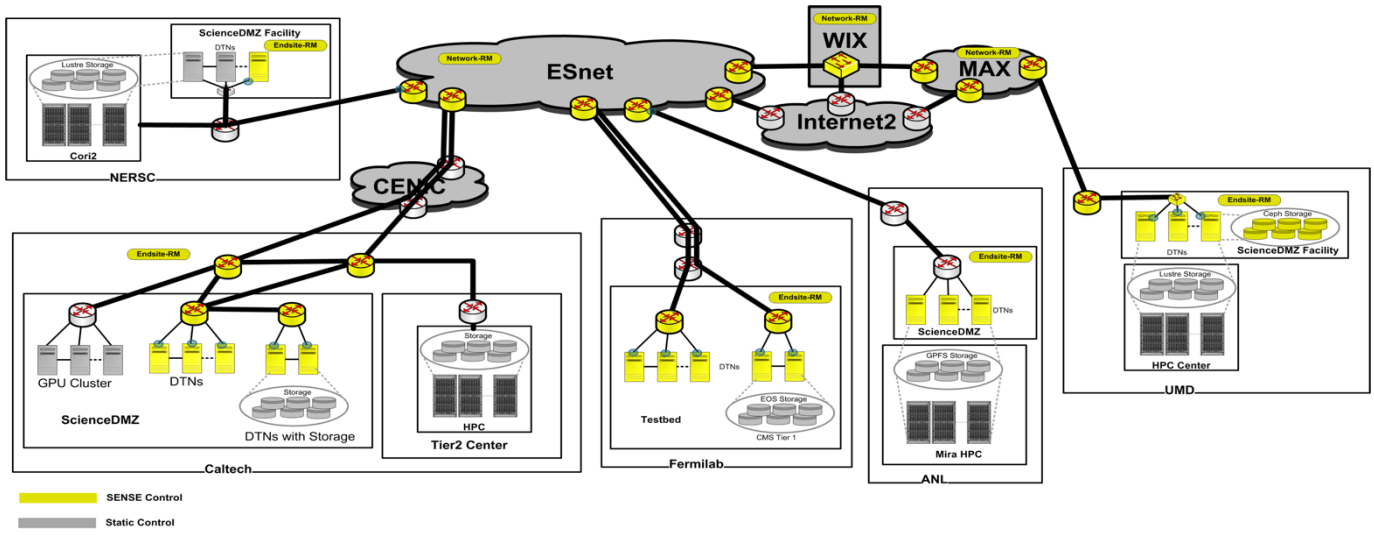

Figure 2 SENSE Testbed

This testbed is being utilized to develop and test the SENSE software, as well as test with domain science use cases.

V. Experimentation and Use Cases

The SENSE project is now in a phase where use case integration is a key focus area. The main use cases currently under test are as described below.

Data Transfer Node Priority Flow: Science DMZ located Data Transfer Nodes (DTNs) are a common method for moving data to/from compute facilities in the R&E community. For this use case, SENSE services are utilized enable a "DTN Priority Flow Service". Since SENSE services are provisioned across the switching and routing elements AND the network stacks of the end systems, this allows the creation of QoS enabled path that can be utilized for specific flows such that deterministic performance can be achieved regardless of the background traffic. The concept of operation is that these "SENSE enabled DTNs" can be place next to current production DTNs, or the SENSE software can be installed directly on the production DTNs. In either case, standard DTN operations and flows across the best effort routed IP paths continue as normal. When a SENSE flow is established between DTNs, this flow will receive priority access to network and host level resources. The best effort flows will continue, but maybe at a reduced rate. This SENSE capability is currently implemented as a Layer 2 point-to-point service. Work is underway to add Layer 2 multipoint and Layer 3 priority flow services. The workflow agent for this use case utilizes the "Time-Block-Maximum Bandwidth", the "Bandwidth-Sliding-Window", and the "Time-Bandwidth-Product (TBP)" SENSE services to instantiate Layer 2 paths with QoS. This workflow also utilizes the "What is Possible?" and "Negotiation" features to demonstrate those feature sets. The SENSE services and additional information regarding testing for this use case is available here [15].

Exascale for Free Electron Lasers (ExaFEL)[5]: The objective of this use case is to stream nano crystallography diffraction data from SLAC to NERSC in order to perform analysis on CORI PII and provide feedback to the beamlines in the form a 3D electron structure visualization. The workflow steps are as follows:

- Stream the data from the LCLS online cache (NVRAM) to the SLAC data transfer nodes
- Stream the data over an SDN path from the SLAC DTNs to the NERSC DTNs (the term DTN is used loosely here, could be a subset of the supercomputer nodes)
- Write the data to the burst buffers layer (NVRAM)
- Distribute the data from the burst buffers to the local memory on the HPC nodes
- Orchestrate the reduction, merging, phasing and visualization parts of the SFX analysis

The main components of the workflow are:

- Database (file catalog) to keep track of the status/location of the data
- state machine (set of Python scripts) to control and monitor the various steps
- Provision SENSE path via API
- Web interface to manage the components above

For this use case the ExaFEL application workflow agent utilizes the "Time-Block-Maximum Bandwidth" SENSE

Service to provision the network path. This includes establishment of Layer 2 paths with QoS with time domain scheduling. Additional information regarding testing for this use case is available here [16].

Big Data Express: BigData Express provides schedulable, predictable, and high-performance data transfer service for DOE's large-scale science computing facilities (LCF, NERSC, and US-LHC computing facilities, among others) and their collaborators. This project is focused on controlling resources at end-site locations. For wide area service the Big Data Express system utilize SENSE services to provision paths across ESnet. Big Data Express workflow agent utilizes the "Time-Block-Maximum Bandwidth" and the "Bandwidth-Sliding-Window" SENSE services to instantiate Layer 2 paths with QoS. This application also utilizes the "What is Possible?" and "Negotiation" features sets to co-schedule across multiple end-sites and network resources. Additional information regarding testing for this use case is available here [17].

The SENSE project is also working on use cases that integrate with Large Hadron Collider/ Compact Muon Solenoid (LHC/CMS) data movement and analysis workflows. A current theme is the use of a new compact event form called the "nanoAOD" [18] that enables the rapid widespread distribution, ingest and real-time processing through a set of "PhysicsTools" of entire datasets of one to a few terabytes, that can be subsequently further analyzed on users' desktops and laptops. The associated CMS analysis workflows, currently under development, are planned to be accelerated and scaled up in terms of the number of simultaneous workflows supported, through the use of SENSE's interactive bandwidth allocation and management services, together with the DTN-RM services at a number of CMS sites, and high throughput data transfer applications such as Caltech's open source Fast Data Transfer (FDT) [19].

Further related developments, underway through the NSF-funded SDN Assisted NDN for Data Intensive Experiments (SANDIE) project [20], include the use of Named Data Networking (NDN) and its caching and routing methods, to be supported in future by SENSE services to expand NDN's ability to deal with larger scale data intensive workflows.

VI. Summary and Future Plans

The SENSE system architecture and implementation presented utilizes model-driven datafication of cyberinfrastructure to enable intelligent network services. Science applications utilizing Intent-based APIs with automated resources discovery and negotiation enable a significantly different mode of operation as compared to current network usage modes. With dropping costs of 100GE capable devices, powerful end systems are increasingly being placed at edge locations where high-bandwidth connections directly to regional and national networks will be the norm. The Science DMZ based, National Research Platform Initiative [21] is an example of a high-performance end-system edge deployment. As a result, the expectation is that we are entering a cycle where network capacity will be easily overwhelmed by these advanced end-site and edge facilities. This indicates a need for methods to manage network resources and access in a more intelligent manner, which includes providing the application agents with sufficient information so that they can plan and optimize their operations. The SENSE project vision and implementation is focused on these issues to be prepared for the day where unmanaged network utilization and extreme over provisioning is no longer the preferred operational approach.

The SENSE architecture and service plan creates many avenues for investigation and provides a platform to address interesting research questions. These issues revolve around the focus on interaction, negotiation, the degree of realtime state management and consideration at many levels of the decision and control operation process. Future plans include exploring some of these issues as noted below as part of ongoing development and testing of the SENSE system:

- What are the tradeoffs between scaling and real-time state collection and performance?
- How to make the realtime vs scalability features dynamic and configurable so adjustments can be based on conditions and application needs?
- Which information/states should be routinely exchanged between Resource Manager(s) and Orchestrator? Which information should be accessible on demand? What are the best methods to make this dynamic/configurable to adjust based on different Resource Manager capabilities and policies?
- What is the right level of abstraction for Application Agent to SENSE system interactions? Is it necessary to provide variable levels from highly abstract to very detailed and resource specific?
- What is the best method for realizing multi-domain, multi-resource authentication and authorization? What is the proper granularity for this? User? Project? Domain? Individual network and end system resource elements? Flows?

As the SENSE architecture and implementation evolves through multi-institution testbed deployment, the focus of the project is to continue integration with domain science use cases and transition the SENSE services to production status for both the network and application operations.

Acknowledgment

We appreciate the contributions to Intent APIs and superfacility use case by Mariam Kiran and NERSC testbed deployment through Damian Hazen and Jason Lee. Work

discussed in this paper was supported through multiple projects from Department of Energy and National Science Foundation projects including the following:

Caltech
- OLiMPS, DOE/ASCR, DOE award #DE-SC0007346
- SDN-Next Generation Integrated Architecture (SDN-NGenIA), DOE/ASCR, DE-SC0015527
- SDN for End-to-end Networked Science at the Exascale (SENSE), DOE/ASCR, DE-SC0015528
- ANSE, NSF award # 1246133
- CHOPIN, NSF award # 1341024
- US CMS Tier2, NSF award # 1120138

University of Maryland
- SDN for End-to-end Networked Science at the Exascale (SENSE), DOE/ASCR, DE-SC0016585
- Resource Aware Intelligent Network Services (RAINS), DOE/ASCR, DE-SC0010716

Fermi National Accelerator
- SDN for End-to-end Networked Science at the Exascale (SENSE), DOE/ASCR

Argonne National Lab
- SDN for End-to-end Networked Science at the Exascale (SENSE), DOE/ASCR

Lawrence Berkeley National Lab/ESnet
- SDN for End-to-end Networked Science at the Exascale (SENSE), DOE/ASCR, FP00002494

REFERENCES

[1] SDN for End-to-end Networked Science at the Exascale (SENSE),http://sense.es.net

[2] I. Monga, C. Guok, W.E. Johnston, B.Tierney, "Hybrid Networks: Lessons Learned and Future Challenges Based on ESnet4 Experience", IEEE Communications Magazine, May 1, 2011

[3] M Kiran, E Pouyoul, A Mercian, B Tierney, C Guok, I Monga, "Enabling intent to configure scientific networks for high performance demands", Future Generation Computer Systems, August 2, 2017

[4] Large Hadron Collider/ Compact Muon Solenoid (LHC/CMS), https://home.cern/about/experiments/cms

[5] Exascale for Free Electron Lasers (ExaFEL), https://www.exascaleproject.org/project/exafel-data-analytics-exascale-free-electron-lasers/

[6] SENSE Orchestrator North Bound API, https://app.swaggerhub.com/apis/xi-yang/SENSE-O-Intent-API/0.9.0

[7] J. van der Ham, F. Dijkstra, R. Lapacz, J. Zurawski, "Network Markup Language Base Schema version 1", OGF GFD-R-P.206, May 2013.

[8] Open Grid Forum (OGF), http://www.gridforum.org

[9] Resource Aware Intelligent Network Services (RAINS), https://wiki.maxgigapop.net/twiki/bin/view/RAINS/WebHome

[10] Multi-Resource Markup Language (MRML), https://github.com/MAX-UMD/nml-mrml

[11] Data Transfer Node (DTN), https://fasterdata.es.net/science-dmz/DTN/

[12] StackV Open Source Orchestration Software Suite, http://github.com/MAX-UMD/StackV.community

[13] SPARQL Query Language for RDF, https://www.w3.org/TR/rdf-sparql-query/

[14] JSONPath - XPath for JSON, http://goessner.net/articles/JsonPath/

[15] Data Transfer Node Priority Flow testing, https://tinyurl.com/sense-demo

[16] ExaFEL use case testing, http://www.slac.stanford.edu/~wilko/psdm/netdemo/rates.html

[17] Big Data Express, http://bigdataexpress.fnal.gov

[18] nanoAOD, https://github.com/cms-nanoAOD

[19] Fast Data Transfer (FDT), https://github.com/fast-data-transfer/fdt

[20] SDN Assisted NDN for Data Intensive Experiments (SANDIE), https://www.nsf.gov/awardsearch/showAward?AWD_ID=1659403; https://datatracker.ietf.org/meeting/interim-2017-icnrg-02/materials/slides-interim-2017-icnrg-02-sessa-sandie-named-data-networking-for-data-intensive-science-edmund-yeh

[21] National Research Platform, http://prp.ucsd.edu