Journal of Network and Computer Applications 119 (2018) 70–82

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

AmoebaNet: An SDN-enabled network service for big data science

S.A.R. Shah[a,d], Wenji Wu[b], Qiming Lu[b], Liang Zhang[b], Sajith Sasidharan[b], Phil DeMar[b], Chin Guok[c], John Macauley[c], Eric Pouyoul[c], Jin Kim[d], Seo-Young Noh[a,d,*]

[a] *University of Science and Technology (UST), Daejeon, South Korea*
[b] *Fermi National Accelerator Laboratory (FNAL), USA*
[c] *Energy Science Network (ESNet), USA*
[d] *Korea Institute of Science and Technology Information (KISTI), Daejeon, South Korea*

ARTICLE INFO

Keywords:
Software-defined network
Network as a service
QoS
Data science
Big data
End-to-end path

ABSTRACT

Data transfer is now an essential function for science discoveries, particularly within big data environments. To support data transfer for big data science, there is a need for high performance, scalable, end-to-end, and programmable networks that enable science applications to use the network most efficiently. The existing network paradigm that support big data science consists of three major components: terabit networks that provide high network bandwidths, Data Transfer Nodes (DTNs) and Science DMZ architecture that bypasses the performance hotspots in typical campus networks, and on-demand secure circuits/paths reservation systems, such as ESNet OSCARS and Internet2 AL2S, which provides automated, guaranteed bandwidth service in WAN. This network paradigm has proven to be very successful. However, to reach its full potentials, we claim that existing network paradigm for big data science must address three major problems: the last mile problem, the scalability problem, and the programmability problem. To address these problems, we proposed a solution called AmoebaNet. AmoebaNet applies Software Defined Networking (SDN) technology to provide "QoS-guaranteed" network services in campus or local area networks. AmoebaNet complements existing network paradigm for big data science: it allows application to program networks at run-time for optimum performance; and, in conjunction with WAN circuits/paths reservation system such as ESNet OSCARS and Internet2 AL2S; it solves the last mile problem and the scalability problem.

1. Introduction

Big data has emerged as a driving force for scientific discoveries (Chen et al., 2013). Large scientific instruments (e.g., colliders, light sources, and telescopes) generate exponentially increasing volumes of data. To enable scientific discovery, science data must be collected, indexed, archived, shared, and analyzed, typically in a widely distributed, highly collaborative manner. Data transfer is now an essential function for science discoveries, particularly within big data environments.

The emergence of distributed, extreme-scale science applications is generating significant challenges regarding data transfer. We believe that two relevant dimensions characterize the data transfer challenges of the extreme-scale era:

(1) High-performance challenges. Because the sizes of scientific data sets are growing exponentially, it is important to transfer data in highest throughputs possible. Thus, scientists can quickly access and analyze data to facilitate scientific discovery.
(2) Time-constraint challenges. Scientific applications typically have explicit or implicit time constraints on data transfer. Providing real-time and deadline-bound data transfer is a challenging task in the extreme-scale era.

Networks are essential for data transfer. From an end user's perspective, we envision networks should provide the following capabilities and features in order to successfully address the high-performance and time-constraint challenges of data transfer for big data science:

- High capacity to provide ample network bandwidth.
- End-to-end network architecture. Many big data science applications require QoS-guaranteed end-to-end network paths to support their operations. Network QoS can include bandwidth, delay, loss, jitter, or their combination.
- Programmability. This feature enables science applications to program networks at run-time to suit their needs. A powerful and rich

* Corresponding author. Korea Institute of Science and Technology Information (KISTI), 245 Daehak-ro, Yuseong-gu, Daejeon, South Korea.
E-mail address: rsyoung@kisti.re.kr (S.-Y. Noh).

https://doi.org/10.1016/j.jnca.2018.06.015
Received 16 October 2017; Received in revised form 12 June 2018; Accepted 24 June 2018
Available online 28 June 2018
1084-8045/ © 2018 Elsevier Ltd. All rights reserved.

set of network programming primitives are necessary to support various science applications.
- Scalability. Networks must be scalable to concurrently support many users and a wide variety of science applications.
- Service availability. Networks must allow science applications to use network services at any time and from anywhere.

Over the years, the U.S. Research and Education(R&E) network communities have developed new network tools, services, and architecture in support of big data science. The existing network paradigm that supports data movement for big data science consists of three major components: terabit networks that provide high network bandwidth, Data Transfer Nodes (DTNs) and Science DMZ architecture (Dart et al., 2014; Science DMZ, 2017; Data Transfer Nodes, 2017) that bypasses the performance hotspots in typical campus network, and on-demand secure circuits/paths reservation systems, such as ESNet OSCARS (Guok et al., 2005) and Internet2 AL2S (Internet2's Advanced Layer 2 Service, 2017), which provide automated, guaranteed bandwidth service within the WAN. This network paradigm has been a big success. However, to reach its full potentials, we claim that the existing network paradigm for big data science must address three major problems:

- The last mile problem. An end-to-end network path typically consists of LAN segments (the last mile), and WAN segments. WAN QoS can be provisioned by utilizing ESNet OSCARS or Internet2 AL2S to reserve bandwidths between Service Termination Points (STPs) (Roberts et al., 2014). However, there lacks an automated network service in campus or local area networks to provision QoS for the last mile. As a result, end-to-end network paths with guaranteed QoS cannot be provisioned.
- The scalability problem. The computing models for large-scale science applications are becoming ever more globally distributed and grid-based, typically involving hundreds, even thousands, of computer systems. Often many of these science applications require QoS-guaranteed end-to-end paths to adequately support their operations. The scalability problem arises when establishing end-to-end paths between computer systems at numerous, widely dispersed sites.
- The programmability problem. Within the WAN, services such as ESNet OSCARS and Internet AL2Sprovide some degree of network programmability that allows science applications to reserve network bandwidths between STPs. However, there lacks a network service within campus and/or local area networks that allow science applications to program local networks at run-time to suit their needs. As a result, science applications cannot make the most efficient or optimum use of underlying network resources.

We are working on the AmoebaNet project to address these problems. AmoebaNet applies Software Defined Networking (SDN) technology (Shenker et al., 2011; McKeown, 2009) to provide "*application-aware*" network service in campus or local area networks. AmoebaNet supports a list of salient features:

- Providing an AMQP interface (Vinoski, 2006) to allow applications to program networks at run-time. The JSON-RPC style communication provides dynamic and flexible interaction with networks;
- QoS-based routing and path computation;
- Fast provisioning of end-to-end network paths with guaranteed QoS;
- Fine-grained control of network traffic;
- Network slicing;
- A rich set of network programming primitives to support a wide spectrum of use cases and application scenarios;
- A full-featured scheduler that allows network resources to be requested on-demand, or scheduled in advance to support complex network operations.
- Seamless integration of LAN and WAN;
- REST-based network initialization and configuration.

AmoebaNet complements existing network paradigms for big data science by: (a) allowing science applications to program networks at run-time to suit their needs, and (b) working in conjunction with WAN circuits/paths reservation system such as ESNet OSCARS and Internet2 AL2S to provide an end-to-end network service, which solves the last mile problem and addresses the scalability issue.

The rest of paper is organized as follows. Section 2 presents background and related works. Section 3 discusses the problems with existing network paradigms for big data science. Section 4 describes AmoebaNet design and implementation. In section 5 we discussed the network paradigm for big data science. Section 6 discusses our evaluation of AmoebaNet. And Section 7 concludes the paper.

2. Backgroudns and related work

2.1. DTNs and the science DMZ approach

The computer systems used for wide area data transfers perform far better if they are purpose-built and dedicated to the function of wide area data transfer. ESNet originated the DTN idea. A DTN is a computer system dedicated to the function of wide-area data transfer (Data Transfer Nodes, 2017), typically having the following features:

- Configured with one or multiple 10/25/40GE NICs. DTNs with 100GE NICs are in the very early stages of deployment.
- Having access to local storage, whether it is a local high-speed disk subsystem, a connection to a local storage infrastructure such as a SAN, or the direct mount of a high-speed parallel file system such as Lustre or GPFS, or a combination of these.
- Running the software tools designed for high-speed data transfer to remote systems – typical software packages include mdtmFTP (Zhang et al., 2018), BBCP (BBCP, 2017), and GridFTP (Allcock et al., 2005).
- No general-purpose computing tasks are allowed on the DTNs to mitigate security risks.

Science DMZ (Dart et al., 2014; Science DMZ, 2017) refers to a specialized DTN deployment that is typically local to a site's network perimeter. The hardware devices, software, configuration, and security policies of Science DMZ are structured and optimized for high-performance data transfer. In addition, Science DMZ has performance measurement and network testing systems that are regularly used to characterize the network and are available for troubleshooting. The Science DMZ model has been successful in many different science environments. DOE Leadership Computing Facilities and many university networks are now adopting the Science DMZ architecture to deploy DTNs.

2.2. Terabit networks

The U.S. Research and Education network communities are working toward deploying terabit networks in support of distributed extreme-scale data movement. ESnet and Internet2 pooled its stimulus resources together to deploy a massive 8.8 Terabit networks. Existing backbone networks have been upgraded with ultra-scalable 100-gigabit technologies (ESNet, 2017). In addition to its production network, ESNet has also created a national-scale network testbed, available to researchers and industry for experiments with new network technologies, protocols, and applications at 100 Gbps (ESNet testbed, 2017). Within the data centers, server performance growth drives system 40/100GE connection deployments, with n x 100 GE uplinks at the access layer.

The deployment of terabit networks has increased the physical network infrastructure capacities by orders of magnitudes, which is essential to meet the high-performance and time-constraint data transfer challenges for big data science. However, rapid growth in network capacities must be matched by corresponding improvements in network and transport layer protocols and higher layer tools.

2.3. ESNet OSCARS and Internet2 ION

Different from "commodity" networks, R&E networks must support data-intensive science applications and large scientific collaborations, which typically involve supercomputers, advanced instruments, and scientists distributed around the world. Typically, it can take weeks or months to manually configure a network to support large scientific collaborations.

To address this problem, ESNet has developed the OSCARS network reservation system (Guok et al., 2005), which can reserve and provision multi-domain, high-bandwidth virtual circuits among sites in a matter of minutes, instead of weeks. OSCARS gives users the ability to engineer, manage, and automate the network according to user-specified requirements for using scientific instruments, computation, and collaborations.

Similarly, Internet2 AL2S (Internet2's Advanced Layer 2 Service, 2017) is a network service that provides researches and network engineers the ability to automatically provision dedicated circuits across their networks in order to support bandwidth-intensive science applications. AL2S leverages ESnet's OSCARS technology.

2.4. SDN and its promise

SDN is a network architecture that disentangles the control plane from data forwarding, thus allowing the network control to be directly programmable (Shenker et al., 2011; McKeown, 2009; Kreutz et al., 2015; Nunes et al., 2014). The promise of SDN is that it allows network resources to be managed and re-configured automatically and dynamically, offering immense performance advantages for network operations.

Prior work Hedera (Al-Fares et al., 2010) designed a dynamic flow scheduling system that adaptively scheduled the switching fabric to reduce traffic collisions. On the PortLand testbed (Niranjan et al., 2009), Hedera can deliver up to 4× more bandwidth than state of the art ECMP technique with modest control and computation overhead. Hedera was designed for multi-root tree network topologies, which have a larger number of parallel paths between any given source and destination edge switches. However, Science DMZ networks typically do not adopt this network architecture. Therefore, Hedera cannot be readily applied in Science DMZ networks.

Wang et al. (2011) used OpenFlow (McKeown et al., 2008) to install wildcard packet-handling rules to balance the loads at each server replica while achieving considerably lower processing overhead. Efforts have also been made to reduce the costs associated with fine-grained control in OpenFlow (Curtis et al., 2011). This research works help to improve SDN scalability (Yeganeh et al., 2013).

SWAN (Hong et al., 2013) employed SDN to improve the link utilization of inter-data-center networks by orchestrating traffic and reconfiguring the data plane to match the current traffic demands. Google's B4 (Jain et al., 2013) – a private WAN connecting Google's data centers across the world – addressed a similar problem. By using SDN, B4's centralized traffic engineering service drives links to near 100% utilization, while splitting application flows among multiple paths to balance capacity against application priority/demands. Both B4 and SWAN adopt a single domain design; a logically centralized controller orchestrates all activities. They are not designed to address the multi-domain problems that are typically encountered in R&E networks. By contrast, AmoebaNet is designed to complement the existing network paradigm that supports data movement for big data science, which typically involves multiple domains.

SDN security or applying SDN to address security problems are discussed in(Zheng et al., 2016; Zhaogang et al., 2016; Scott-Hayward et al., 2013; Shin et al., 2013; Kreutz et al., 2013) and these are also hot research areas. However, our research does not address security issues.

Recently, a new high level programming language for routers and switches named P4 has been developed by some of the original founder of SDN (Bosshart et al., 2014). Different from OpenFlow which is focused on the control plane, P4 is designed to program and control silicon processor chips in the data plane. With the advent of P4, programmable networks can be controlled "top-down" to install any functionality the user wants (Kim et al., 2015; Popescu et al., 2017; Sivaraman et al., 2015; Laki et al., 2016).

2.5. End-to-end collaborative infrastructure efforts

The DTN and Science DMZ concepts, coupled with terabit WAN capabilities and emerging SDN technologies, all could be combined to provide end-to-end high-performance network services required by big-data science. The Pacific Research Platform (PRP) (PRP, 2017) represents just such an effort to create that type of network environment. Designed and implemented under the auspices of the Corporation for Education Network Initiatives in California (CENIC), PNP seeks to provide a multi-domain network infrastructure for large-scale science collaboration. The National Science Foundation (NSF) is currently in the process of expanding the PRP idea to national scale, with a National Research Platform (NRP) (PRP, 2017). While these initiatives establish physical infrastructures capable of supporting extreme performance data movement across multiple organizations, much of the middleware needed to optimize such large-scale data movement is still missing. AmoebaNet is proposed here to be one of these missing middleware elements. It is intended to provide a service capable of implementing SDN-based network services within the local component of the end-to-end network path. In addition, it is intended to provide science applications the capability to dynamically customize those local network services to meet the application's specific needs.

2.6. QoS-guaranteed end-to-end network path efforts

Many big data science applications require QoS-guaranteed end-to-end network paths to support their operations. The capability of fast provisioning of end-to-end paths with guaranteed QoS is a long sought goal in R&E network communities.

AutoBAHN (Geant AutoBAHN, 2015) is a GEANT-provided provisioning tool that facilitates the multi-domain dynamic circuit provisioning service. AutoBAHN is focusing on interoperability across domains. It easily negotiates the different networking technologies deployed by the different domains. However, AutoBAHN has a limited support for layer 3 and application layer operations. Therefore, it is better suited for WAN deployment.

The DYNES project (Dynamic Network Systems) (Zurawski et al., 2011) has attempted to deploy OSCARS to provision end-to-end layer 2 paths for large-scale data transfer. In the DYNES framework, OSCARS is not only deployed in WAN, but also in LAN and campus networks. However, we argue that the DYNES framework has deficiencies because OSCARS is not appropriate for campus or local area networks [see Section 3.1].

The DANCES (Developing Applications with Networking Capabilities via End-to-end SDN) (Hazlewood et al., 2016) is a NSF-funded SDN research project. It aims to enable network bandwidth scheduling via SDN technology and enhance the overall performance of scientific applications. DANCES develop a bandwidth management component, the Centralized OpenFlow and Network Governing Authority (CONGA) that schedules network resources. Because CONGA adopts a centralized architecture, the scalability problem would arise when it is deployed to support extreme-scale scientific applications.

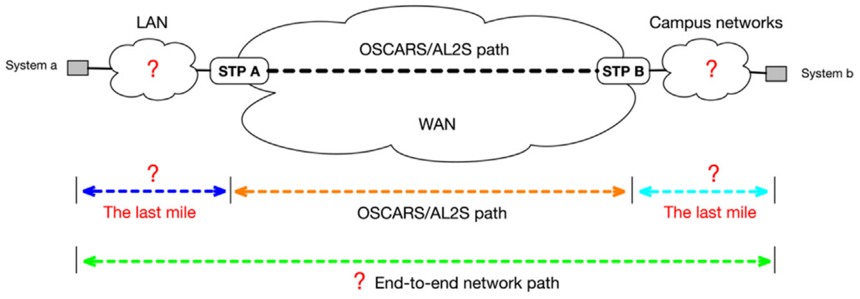

Fig. 1. The last mile problem.

3. Problems with existing network paradigm for big data science and out solution

To successfully meet the high-performance and time-constraint challenges of data transfer for big data science, we claim that existing network paradigm must address three major problems: the last mile problem, the scalability problem, and the programmability problem.

3.1. The problems

3.1.1. The last mile problem

As illustrated in Fig. 1, an end-to-end network path typically consists of LAN segments (the last mile), and WAN segments. WAN QoS can be provisioned by utilizing ESnet OSCARS or Internet2 AL2S to reserve bandwidth between STPs. However, there lacks an automated network service in campus and/or local area networks to provision QoS for the last mile. As a result, end-to-end network paths with guaranteed QoS cannot be provisioned.

The last mile problem has been a long-lasting issue that continues to puzzle the U.S. Research and Education network communities. DOE's Office of Science Advanced Scientific Computing Research (ASCR) office funded work on several research projects, such as Lambda station (Bobyshev et al., 2006) and TeraPaths (Gibbard et al., 2006), to address this problem. However, these projects only achieved limited success, and have not been put into practical use. One reason for this limited success is that the network resources were closed, proprietary, and difficult to program at the time. It was then difficult to provide a generic solution to address the problem.

Some researchers and network practitioners have also attempted to deploy OSCARS at campus and local area networks, in the hopes of addressing the last mile problem and the programmability problem (Zurawski et al., 2011, 2012). However, we argue that OSCARS is not appropriate for campus or local area networks due to several reasons. First, campus and local area networks need to support complex use cases and application scenarios. OSCARS does not provide a rich set of network programming primitives to support such operations. For example, OSCARS does not provide lock and/or atomic commit primitives to support multi-step operations, which are common for complex science workflows. Second, OSCARS has a limited support for layer 3 and application layer operations.

3.1.2. The scalability problem

Two factors contribute to the scalability problem. First, the computing models for large-scale science applications are becoming ever more globally distributed and grid-based, which typically involves hundreds, even thousands, of computer systems. Second, many science applications required QoS-guaranteed end-to-end paths to support their operations.

Assuming a science application involves m sites, with n_1 systems at site 1, n_2 systems at site 2, …, and n_m system at site m. In the extreme cases, $n_1 * n_2 * \ldots * n_m$ QoS-guaranteed end-to-end paths should be established between systems among these sites. Many network practitioners would establish VLAN-based point-to-point layer 2 circuits to implement end-to-end paths. Clearly, this model is simple and effective, but not scalable because it is not possible to have more than 4096 VLANs.

Spoke-hub distribution model (O'Kelly, 1998) is an effective mechanism to address the above scalability problem. In this model, point-to-point layer 2 circuits with guaranteed bandwidth are required only between STPs, instead of between computer systems (the tips of the spokes). At each site, traffic between systems in physically disperse sites are carefully multiplexed/de-multiplexed to/from the corresponding point-to-point layer 2 circuits between STPs. Multiple traffic flows can be multiplexed/de-multiplexed to/from a single point-to-point layer 2 circuit, which typically reserve an aggregated bandwidth of its component traffic flows. To meet the end-to-end QoS requirements for network traffic, a network service at each site is required to provide: (a) QoS-based routing and path selection; (b) fine-grained control of network traffic; and (c) seamless integration of LAN and WAN. Fig. 2 illustrates an example of using spoke-hub distribution model to solve the scalability problem.

3.1.3. The programmability problem

"Network as a service" is a long-chanted slogan in the U.S. Research and Education network communities. Network programmability is the prerequisite for this goal.

At present, the U.S. Research and Education network communities have developed some degree of network programmability within the WAN. Science applications can use ESnet OSCARS or Internet AL2S to reserve network bandwidths between STPs. However, a network service to support programmability within the campus or local area networks is lacking. As a result, science applications cannot make efficient and

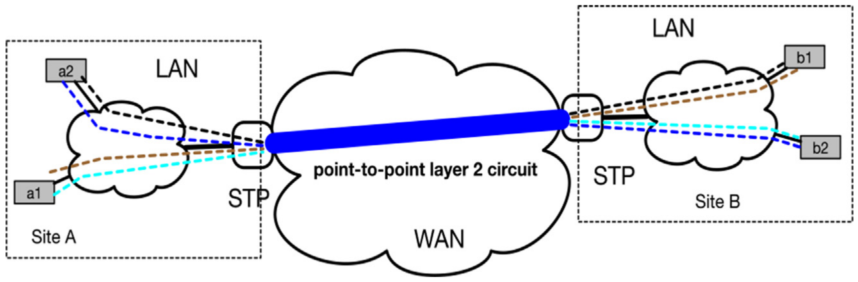

Fig. 2. Spoke-hub distribution model for the scalability problem.

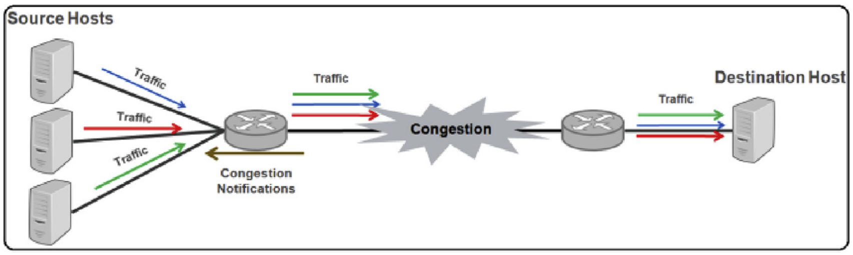

Fig. 3. Network congestion.

optimum use of underlying network resources.

3.2. Our solution

Our analysis indicates that there is the lack of an automated network service in campus and/or local area networks to address the previously mentioned problems. Our research will help fill this gap.

We believe that SDN is the right paradigm to implement such a network service. First, application-driven network programmability is a core component of SDN, facilitating the flexibility, automation, virtualization, and orchestration of network services. With the aid of SDN, developing an "application-aware" network service becomes feasible, with science applications directly scheduling network resources. Second, SDN can control and steer network traffic on a per-flow basis. Therefore, developing features that support fine-grained control of network traffic becomes simple and straightforward. Third, SDN provides centralized network provisioning that enable optimum use of network resources. Finally, SDN provides centralized network management that helps to reduce network operation costs.

We have designed and implemented AmoebaNet, an SDN-enabled network service for big data science. We present AmoebaNet's design and implementation in the next section.

4. AmoebaNet: the desing and implementation

4.1. Design goals

We have several design goals for AmoebaNet. First, "Network as a Service" is our primary goal. AmoebaNet must allow applications to program networks at run-time for optimum performance. Second, AmoebaNet must provide QoS guarantee for priority traffic. Third, AmoebaNet must support a wide spectrum of applications. Fourth, AmoebaNet must provide fine-grained control of network traffic. Finally, AmoebaNet must be easy to deploy and operate.

4.2. AmoebaNet design and implementation

We implemented AmoebaNet using Java upon the ONOS platform (ONOS Project, 2017). ONOS provides a number of high-level abstractions and models, through which AmoebaNet can learn about the state of the network and it can control the overall network dynamically. To achieve the target specific goal, we extended and reused the core northbound APIs of ONOS (such as Device, Host, Link, Topology, Flow, Meter, etc) in our proposed AmoebaNet solution, and we proposed a new set of purpose-build APIs (such as Path, Topology, Flow, Query, Reserve, etc). On top on of ONOS platform, we developed a set of new mechanisms for AmoebaNet, which includes: (a) an AMQP interface to allow applications to program networks at run-time. The JSON-RPC style communication provides dynamic and flexible interaction with networks; (b) a lock mechanism to avoid unsynchronized access to network resources; (c) a QoS-based routing and path computation mechanism; and (d) a full-featured scheduler to allow network resources to be requested on-demand, or scheduled in advance to support complex network operations.

To simplify the implementation of AmoebaNet, we logically splited the AmoebaNet's design into two different architectures: the data plane and the control plane. Both architectures are inter-dependent; however, the data plane architecture mainly considers the design and configuration of underlying SDN switches and the control plane represents the design and configuration of SDN controllers.

4.2.1. AmoebaNet data plane architecture

The key consideration of network operators is the efficient utilization of network resources (such as bandwidth). The existing data plane architectures are based on best-effort approches. However, the requirement of applications and services is not satisfying by this so called best-effort service approach. From applications or users perspective, the behaviour of best-effort approach is unpredictable and often in a poor Quality of Services (QoS). The traditional routing protocols calculates the optimal routing paths between two endpoints and continuously send the traffic on that path, even it becomes congested. Considering the congestion in networks is the key to maximize the utilization of bandwidth. As shown in Fig. 3; whenever congestion take place in network, the congestion control mechanisms just send the notification message to sender to reduce the data transmission rate; in result, the overall performance degradation in terms of required throughput.

AmoebaNet's data plane architecture is able to take into account of available bandwidth in network and enables a QoS-aware bandwidth reservation system for upper layer applications and services. To ensure the anti-congestion; the reservation system will not exceed more than the available bandwidth of network. Our proposed AmoebaNet solution supports two classes of services: priority and best-effort; we exploited the meters and queuing techniques at data plane level. We used meters to limits the rate of priority flows and we used queues to priortize the flows. The priority based flows requested always take place on the priority queues and the best-effort flows always placed on best-effort queues. AmoebaNet also provides a flexible mechanism to reallocation the flows on backup path in case of primary path fails. AmoebaNet applies the data plane as shown in Fig. 4. At each SDN switch, AmoebaNet uses QoS queues to differentiate priority and best-effort traffic; priority traffic is transmitted first. In addition, it meters which enables the priority traffic to enfore rate control. AeobaNet configure the both queues and meters dynamically on each underlaying devices.

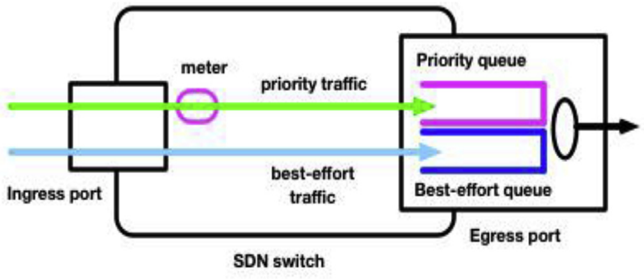

Fig. 4. AmoebaNet data plane model.

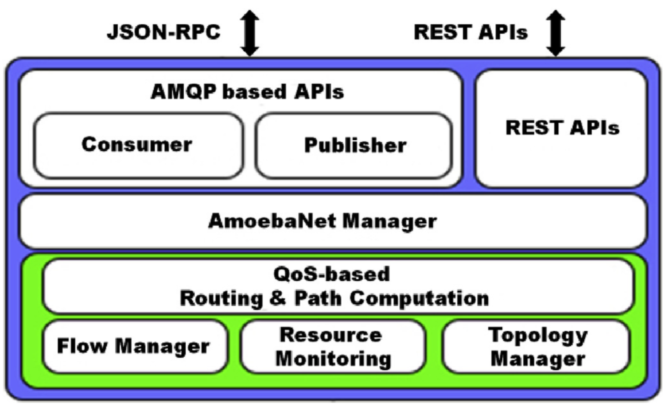

Fig. 5. AmoebaNet control plane architecture.

4.2.2. AmoebaNet control plane architecture

Fig. 5 illustrates AmoebanNet's control plane architecture. It consists of the following modules:

- *AMQP based APIs module.* An AMQP interface that allows applications to access AmoebaNet services. AMQP APIs are based on JSON-RPC style communication mechanism which allows dynamic and flexible interaction between application and AmoebaNet. The AMQP interface consists of consumer and publisher components, which enables two-way communication between AmoebaNet and application (e.g., BigData Express). The consumer component manages network service requests from application while the publisher component sends network status/statistics to application. AmoebaNet use this module to create or delete the publisher and consumer queues dynamically before starting the actual AmoebaNet application inside ONOS controller. Once the queues created, applications can start two-way communication with AmoebaNet. An example of AmoebaNet consumer and publisher is illustrated in Fig. 6.

In this example, when BDE application starts send JSON-RPC commnads to AmoebaNet's consumer by using rpc queue (aka: sdnagent_consumer_queue) and correlation id (aka: fnalserver); an anonymous exclusive callback queue (aka: amqp-xxx-yyy) created to send reply back to BDE application. The AmoebaNet's consumer always in waiting state for requests on queue. When a request appears, it perform the job and send a JSON reply with the results to the BDE application by

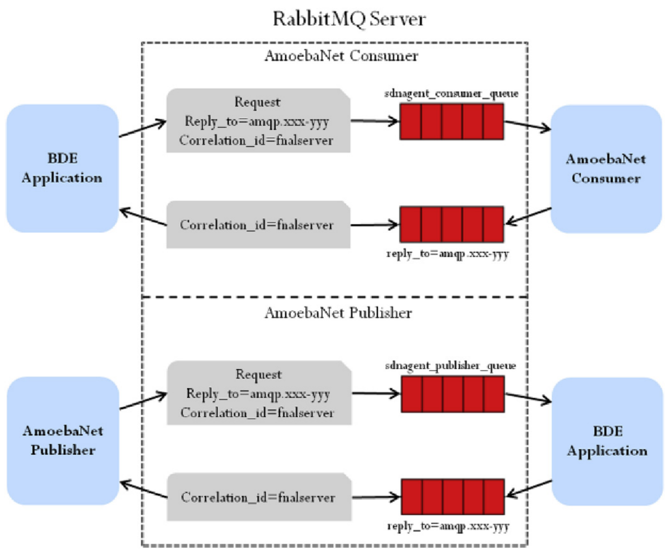

Fig. 6. AmoebaNet consumer & publisher examples.

using the callback queue. AmoebaNet publisher work similarly and send JSON-RPC messages to BDE application, but it uses new rpc queue name (aka: sdnagent_publisher_queue) and BDE application acts as a consumer. BDE application send reply back to AmoebaNet application by using its callback queue.

AMQP module also configured with the events delivery service to dispatch the events to notify its listeners about incoming messages/commands from applications. Event listeners are the components that implements the event listener interface. We implemented the event listener interface for AmoebaNet manager to handle the incoming messages/commands.

- *REST APIs.* Similar to AMQP module, REST APIs also provides an alternative interface for applications to interact with AmoebaNet services and program the network at runtime. By using RESTful APIs, any application can interact with AmeobaNet using simple JSON style format.
- *AmoebaNet Manager.* AmoebaNet manager is one of the core element of our proposed solution, it act as the coordinator. It is in charge of the communication among all modules of AmoebaNet, and it establish the workflow among each sub-module and internal services. AmoebaNet manager component handles the upper layer requests comes from above mentioned modules. This component keeps track of consumed resources and handling the service resiliency. When network service requests are received, it schedules and coordinates with other modules to perform related operations, including QoS-base path calculation, path reservation, path setting up/tearing down, and network status reporting. AmoebaNet manager also in charge of determining whether a newly requested service can be provide or not. During the entire lifetime of requested service, it takes into account the available resources (e.g. bandwidth) in the network.

As shown in Fig. 7, in this module we also implemented the event listener interfaces to handle the incoming services requests. In addition, it also listen to the asynchronous events sent by topology manager to inform applications about the changing conditions in the network.

- *QoS-based Routing & Path Computation.* This module of AmoebaNet is in charge of the computation of path. It calculates available paths between service termination points (STPs). As shown in algorithm 1, AmoebaNet utilizes the largest bottleneck bandwidth algorithm (a

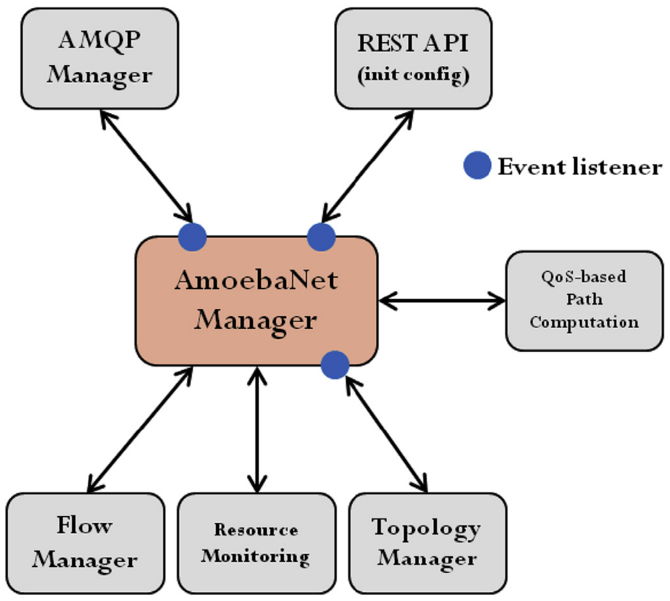

Fig. 7. AmoebaNet manager with event listener interfaces.

Dijkstra's shortest path variant) to calculate available paths between two end points in network, with bandwidth as constraints. AmoebaNet abstracts SDN network as a graph. Vertices in the graph represent SDN devices while edges in the graph represent network links. For each edge in the graph, attributes such as bandwidth is obtained through the SDN controller. A cost is assigned to each edge based on available bandwidth of the edge.

Algorithm 1
Largest bottleneck bandwidth.

```
1.  function Dijkstra_AmoebaNet(Graph, source):
2.    create vertex set Q
3.    for each vertex v in Graph:
4.      bandwidth[v] ← 0
5.      previous[v] ← UNDEFINED
6.      add v to Q
7.    bandwidth[source] ← INFINITY
8.    while Q is not empty:
9.      u ← vertex in Q
10.     remove u from Q
11.     for each neighbor v of u:
12.       alt ← min_bandwidth(u, v)
13.       if alt > bandwidth[v]:
14.         bandwidth[v] ← alt
15.         previous[v] ← u
16.   return bandwidth[], previous[]
```

By using our proposed largest botttleneck bandwidth algorithm, it is possible to determine the best available path (with maximum available bandwidth) between a start node and any other node in a graph. Our proposed algorithm consists of the following steps:

1. Initialization of all nodes with bandwidth "0"; and the starting node with “Infinity”
2. The bandwidth of all nodes sets up as temporarily and the starting node as permanent.
3. Starting node sets up as active.
4. To calculation the largest bottleneck bandwidth of all neighbour nodes of the active node, and checks the available bandwidth of the edges.
5. If the calculated bandwidth of a node is greater than the current one, then it updates the bandwidth and sets up the current node as antecessor.
6. The node with the maximum temporary bandwidth is sets up as active, and mark its bandwidth as permanent.
7. Repeating of steps 4 to 7 until there aren't any nodes left with a permanent bandwidth.

- *Flow Manager.* The module that installs, updates, modifies, and deletes the flow rules on underlying network devices. Whenever AmoebaNet receives a path reservation request from an application then the flow manager will receive flows related requests from AmoebaNet manager module, and it will perform flows related operations on each forwarding device independently. In addition, meter (for rate-control) and queue (for traffic priority) related operations are managed by this flow manager service to support QoS guarantee.
- *Resource Monitoring.* This module monitors and keeps track of network resources. The resource monitoring service is mainly in charge to monitor the resources (such as link cost, bandwidth, etc) related information and also provides that information to other services.
- *Topology Manager.* The module that provides the global network topology view to AmoebaNet manager. It coordinates with topology service of ONOS to updated information in AmoebaNet's local topology database. If any changes occur in underlying network topology, the Topology manager immediately triggers topology change events to AmoebaNet's manager by using the its event dispatcher service.

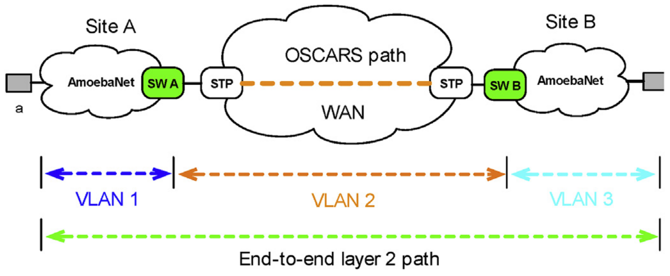

Fig. 8. VLAN operations at GWs.

4.2.3. AmoebaNet primitives

To target the programmability problem and realize Network as a Service, AmoebaNet provides a list of primitives to allow applications to program networks at run-time:

- initial_config(): configure the interfaces/ports of internal switches that are acting as gateways to external networks. At gateways, VLAN operations such as pop, push, and swap are typically needed to stitch path segments at different networks to provision end-to-end layer 2 paths. As shown in Fig. 8, switch SW A and SW B are configured as gateways for site A and B, respectively. An end-to-end layer 2 path has been established between DTN A and B, which actually consists of path segments at different VLANs. VLAN conversions of VLAN1 ←→ VLAN2 and VLAN2 ←→ VLAN3 are performed at SW A and SW B, respectively.
- Lock()/unlock(): lock/unlock AmoebaNet services to avoid unsynchronized access to network resources. It basically executes two key mechanisms: a lock mechanism to avoid unsynchronized access to network resources, and a full-featured scheduler to schedule network resources at user requests.

We give an example to illustrate why AmoebaNet needs a lock mechanism. Consider a network as shown in Fig. 9. Two scientific applications – app1 and app2 – need to reserve network resources to support operations. App1 requires two network paths – *path A-B* and *path B-C*, each with a bandwidth of 8Gbps; while App2 requires the same paths – *path A-B* and *path B-C*, each with a bandwidth of 6Gbps. If the two applications run simultaneously with no locks, it could lead to a deadlock situation if the operations proceed in the order shown below. As a result, neither application can proceed.

App1	App2
Reserve path A-B (succeed)	
	Reserve path B-C (succeed)
Reserve path B-C (fail)	
	Reserve path A-B (fail)

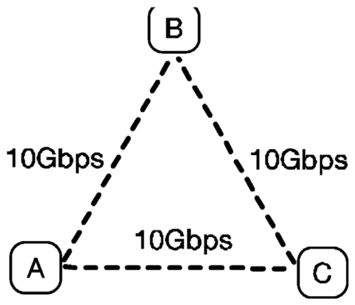

Fig. 9. Why does AmoebaNet need a lock mechanism?.

The lock ensures one application's network operations are complete before another application can access the shared network resources as shown below.

App1	App2
Acquire lock	
Reserve path A-B (succeed)	
Reserve path B-C (succeed)	
Release lock	
	Acquire lock
	Reserve path B-C (succeed)
	Reserve path A-B (succeed)
	Release lock

The full-featured scheduler allows network resourcesto be requested on-demand, or scheduled in advance to support complex network operations. Whether the request be immediate network provisioning, or network reservation in advance, the scheduler works as follows:

1) Call *QoS-based Routing & Path computation* to check whether requested network resoruces are available. If not, the request is rejected, otherwise, continue;
2) Allocate or reserve network resources in control plane;
3) Coordinate with *Flow Manager* to commit resources in data plane, immediately or when the reservation starts.
4) Coordinate with *Flow Manager* to release resources in data plane and recycle resources in control plane when the reservation ends, or when a release request is received.

- *query_virtual_netslice():* query the setting of a particular virtual network slice;
- *query_path():* query the setting of a particular path;
- *query_host_to_host():* query the maximum available bandwidth between two end hosts;
- *query_host_to_gateway():* query the maximum available bandwidth between a host or a list of hosts to gateway;
- *create_path():* create or reserve a layer 2/3 path between two end points with a specified bandwidth and at a specified time slot;
- *create_virtual_netslice():*given a list of IPs, create or reserves a virtual network slice among them with a specified bandwidth and at a specified time slot;
- *update_path():* update or modify a particular path's setting;
- *update_virtual_netslice():* update or modify a particular virtual network slice's setting;
- *release_path().* Tear down a particular path;
- *release_virtual_net_slice():* tear down a particular virtual network slice;
- *event_notification_registration():* register notification of particular network events.

5. A new networking paradigm for big data science

The development of AmoebaNet presents a new network paradigm for big data science (Fig. 10). In this paradigm, Science DMZ and Terabit Networks provide high-speed and high-capacity network infrastructures; AmoebaNet, in conjunction with OSCARS/ION, delivers “network as a service” to upper layer science applications; science applications intelligently program networks at run-time to suit their needs.

Fig. 11 illustrates a deployment example of the new paradigm. A big data science application runs at multiple sites. Each site is an independent administrative domain. To support the science application; each site features a dedicated cluster of high-performance computer systems, and a SDN-enabled LAN or campus network. The dedicated computer systems are deployed using the Science DMZ architecture. The SDN-enabled LAN or campus network consists of either physical or virtualized SDN-enabled switches or routers and connect the dedicated

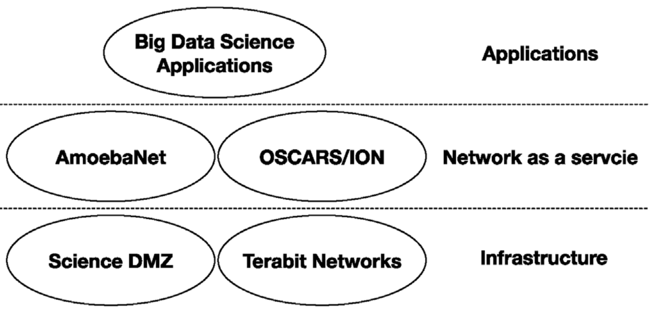

Fig. 10. A new network paradigm for big data science.

computer systems to a BR – a gateway router or switch that connects to WAN. AmoebaNet is deployed at each site to manage LAN or campus network, with OSCARS/AL2S running within the WAN to provides on-demand guaranteed. Although multiple SDN controllers can be deployed at a site, ONOS presents a global network view of the site to AmoebaNet.

To intelligently program networks at run-time to suit its needs, a science application typically performs the following operations:

1) Estimate and calculate the system-to-system traffic matrix for the application, and the related QoS requirements (e.g. throughput, delay).
2) Calculate the site-to-site traffic matrix of the application, and the related QoS requirements (e.g., throughput, delay).
3) Call ESnet OSCARS or Internet2 ION circuit service to set up or modify point-to-point layer 2 circuits among sites.
4) Call AmoebaNet at each site to program LAN or campus network. Traffic between systems in physically disperse sites are carefully multiplexed/de-multiplexed to/from the corresponding point-to-point layer 2 circuits between sites.

6. Evaluation and results

We evaluated AmoebaNet within Fermilab SDN testbed. Our goal is to verify and validate that AmoebaNet can meet our design objectives.

6.1. End-to-end network path provisioning within a domain

We run AmoebaNet in a single domain SDN testbed (Fig. 12). The testbed consists of four 40GE DTNs –DTN1, DTN2, DTN3, and DTN4, and two Pica8 SDN switches – P5101 and P3930. Each DTN is configured with one or multiple VLAN virtual interfaces and is connected to a Pica8 SDN switch. The two switches are connected back-to-back with a 40GE link.

We used iperf to run data transfer between DTNs with two classes of traffic. The data transfer includes three jobs, which last for 20 min.

- Priority traffic 1, which is sent from 192.3.3.2 (vlan3 nic@DTN2) to 192.3.3.4 (vlan3 nic@DTN4). The data transfer starts at 4th min, and ends at 15th min, wherein the maximum bandwidth is capped at 10Gbps.
- Priority traffic 2, which sent from 192.4.4.3 (vlan4 nic@DTN3) to 192.4.4.4 (vlan4 @DTN4). The data transfer starts at 7th min and ends at 18th min, wherein the maximum bandwidth is capped at 15Gbps.
- Best-effort traffic, which is sent from 192.2.2.1 (vlan2 nic@DTN1) to 192.2.2.4 (vlan2 nic@DTN4). The data transfer starts at 1st min and ends at 20th min.

An end-to-end path is established for each data transfer job. The path details are listed in Table 1. We developed a simple application to send AMQP JSON-RPC commands to AmoebaNet to program the

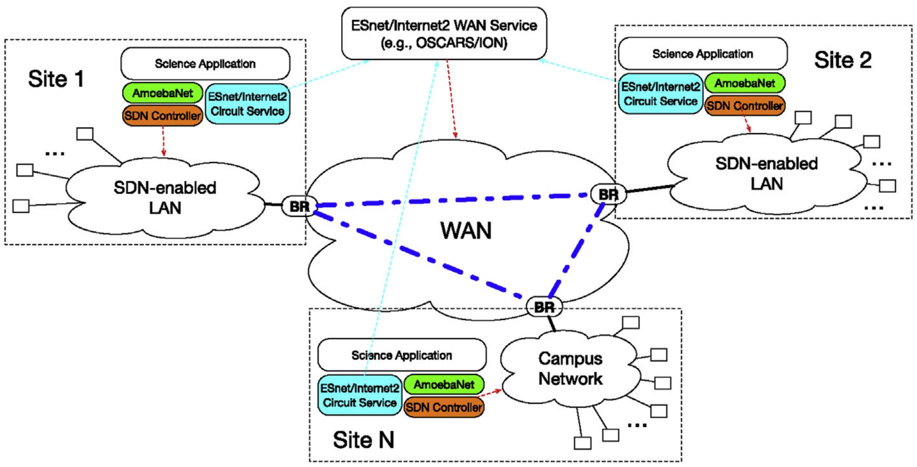

Fig. 11. A typical AmoebaNet use case for big data science.

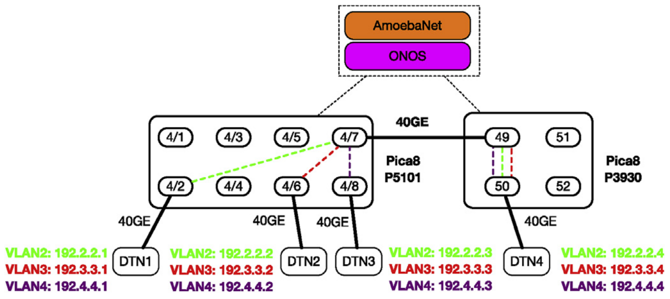

Fig. 12. End-to-end network path provisioning within a domain.

Table 1
End-to-end network paths.

	Srcip	Dstip	vlan	bandwidth	Class
Path 1	192.3.3.2	192.3.3.4	3	10Gbps	Priority
Path 2	192.4.4.3	192.4.4.4	4	15Gbps	Priority
Path 3	192.2.2.1	192.2.2.4	2	N/A	Best-effort

testbed at run-time to set up the paths. Essentially, AmoebaNet performs these tasks:

1) Calculate and route local LAN paths.
2) Install flow rules to set up local LAN paths.
3) Install meters and queues to enforce rate control and QoS guarantee for priority traffic.

For example, here is an AmoebaNet command for setting up a local LAN path for priority traffic 1.

```
{
   "cmd":"create_path",
   "hosts":
      [{
         "startTime": "immediate",
         "endTime": "2017-09-07 13:00",
         "srcIp":"192.3.3.2",
         "rate":"10000", // in Mbps
         "dstIp":"192.3.3.4",
         "trafficType":"priority traffic",
         "vlanId":"3",
         "routeType": "host-to-host"
      }]
}
```

We collected throughput measurements for the data transfer jobs. The evaluation results illustrated in Fig. 13. It can be seen that: (a) from 1 to 4min, best-effort traffic consumes the entire 40Gbps bandwidth of the 192.2.2.1–192.2.2.4 path. (b) from time 4–7 min, priority traffic 1 achieves a capped throughput of ~10Gbps, with the best effort traffic consuming the remaining ~30Gbps. (c) For time 7–15 min, priority traffic 2 is sent and attains a throughput of ~15Gbps, which reduces the best-effort traffic throughput to ~15Gbps, and priority traffic 1 remains unchanged at ~10Gbps. (d) between time 15 and 20 min, both priority traffic 1 and 2 transfers are completed, allowing the best-effort traffic's throughput to increase to ~40Gpbs. The best-effort traffic consumes the entire bandwidth of the end-to-end path.

This experiment clearly demonstrates that (1) AmoebaNet provides "Application-aware" network services. It allows application to program network at run time. (2) AmoebaNet supports differentiated services, and provides QoS guarantee for priority traffic. And (3) AmoebaNet provides fine-grained control of network traffic.

6.2. QoS-guaranteed network slicing within a domain

The goal of this experiment is to verify and validate that AmoebaNet is able to program the underlying network to create virtual network slices within an SDN-enabled network. The experiment was run in the same SDN testbed as shown in Fig. 7. We sent commands to AmoebaNet to create three virtual network slices:

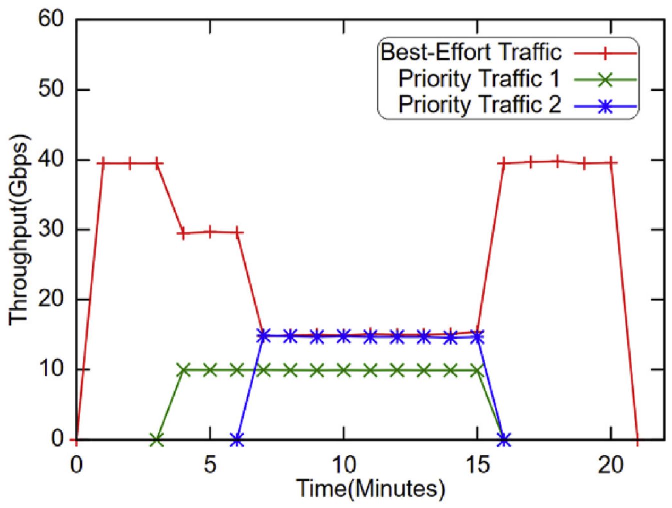

Fig. 13. Evaluation results.

- *Slice 1* includes three hosts, with a rate of 3Gbps
  - h1: 192.2.2.1 (vlan2 nic@DTN1)
  - h2: 192.2.2.2 (vlan2 nic@DTN2)
  - h3: 192.2.2.4 (vlan2 nic@DTN4)
- *Slice 2* includes two hosts, with a rate of 5Gbps
  - h1: 192.3.3.3 (vlan3 nic@DTN3)
  - h2: 192.3.3.4 (vlan3 nic@DTN4)
- *Slice 3* includes three hosts, with a rate of 10Gbps
  - h1: 192.4.4.1 (vlan4 nic@DTN1)
  - h2: 192.4.4.3 (vlan4 nic@DTN3)
  - h3: 192.4.4.4 (vlan4 nic@DTN4)

Here is an AmoebaNet command to create Slice 1 with a bandwidth of 3Gpbs:

```
{
  "cmd":"create_virtual_netslice",
  "hosts": [192.2.2.1, 192.2.2.2, 192.2.2.4],
  “startTime”: “immediate”,
  “endTime”: “2017-09-07 13:00”,
  "rate":"3000″, // in Mbps
  "vlanId":"2″
}
```

Once the virtual network slices were created, we used the iperf tool to run data transfer between any pair of hosts in the virtual network slices created above. The results are shown in Table 2.

The experiment shows that (1) the hosts within a virtual network slices can communicate with each other, while the hosts at different slices cannot communicate. (2) each virtual network slice can be created with a specified bandwidth. Therefore, the experiment verifies that AmoebaNet supports the feature of network slicing. This feature allows multiple science applications to share a common network infrastructure, with the advantages of performance isolation, security isolation, and easy management.

6.3. End-to-end network path provisioning cross domains

As illustrated in Fig. 14, we set up two logically independent sites – A and B. Site A is comprised of three 40GE DTNs – DTN1, DTN2, and DTN3, and a Pica8 P5101 SDN switch, with Port 4/1 configured as gateway connecting to the WAN. Site B consists of a 40GE DTN (DTN4), and a 1GE DTN (DTN5), and a Pica8 P3930 SDN switch, with Port 49 configured as gateway connecting to the WAN. For both sites, each DTN is configured with one or multiple VLAN virtual interfaces. AmoebaNet runs at each site.

Site A and B are connected through an OSCARS layer-2 loopback path, which can be dynamically set up and tear down using the ESnet OSCARS circuit service.

In this experiment, we used iperf to run data transfer from site A to B. The data transfer includes three jobs, which last for 20 min.

Table 2
Data transfer results.

Rates (Gbps)		Slice1			Slice 2		Slice 3		
		h1	h2	h3	h1	h2	h1	h2	h3
Slice1	*h1*	n/a	2.94	2.95	0	0	0	0	0
	h2	2.94	n/a	2.94	0	0	0	0	0
	h3	2.94	2.96	n/a	0	0	0	0	0
Slice2	*h1*	0	0	0	n/a	4.97	0	0	0
	h2	0	0	0	4.95	n/a	0	0	0
Slice3	*h1*	0	0	0	0	0	0	9.98	9.96
	h2	0	0	0	0	0	9.97	n/a	9.98
	h3	0	0	0	0	0	9.98	9.97	n/a

- Data transfer job 1, which sent from 192.2.2.1 (vlan2 nic@DTN1) to 192.2.2.4 (vlan2 nic@DTN4). The data transfer starts at 1st min, and ends at 20th min, wherein the maximum bandwidth is capped at 300Mbps.
- Data transfer job 2, which is sent from 192.3.3.2 (vlan3 nic@DTN2) to 192.3.3.4 (vlan3 @DTN4). The data transfer starts at 6th min and ends at 15th min, wherein the maximum bandwidth is capped at 200Mbps.
- Data transfer job 3, which is sent from 192.4.4.3 (vlan4 nic@DTN3) to 192.4.4.5 (vlan4 nic@DTN5). The data transfer starts at 11th min and ends at 15th min, wherein the maximum bandwidth is capped at 500Mbps.

An end-to-end network path is established for each data transfer job (Table 3). In this experiment, each end-to-end network path runs across three domains – Site A, ESnet, and Site B. Therefore, each end-to-end network path consists of a LAN segment @site A, a WAN segment @ ESnet, and a LAN segment @site B. We used the Spoke-hub distribution model to set up these paths. A single layer-2 point-to-point circuit with an aggregated bandwidth of 1Gbps is established between site A and B. AmoebaNet is responsible for setting up the local LAN path segments between DTNs and gateway at site A and B, respectively. All cross-site traffic is multiplexed/de-multiplexed to/from a single point-to-point layer 2 circuit. We developed a simple application to dynamically program the LANs (i.e. site A & B) and WAN by using AmoebaNet APIs and ESnet NSI service, respectively. Here is how it operates:

1) Call ESnet OSCARS service to establish a loopback WAN path to connect site A and B. The loopback path has a bandwidth of 1Gbps and a RTT of ~92 ms, with *vlan 3200* at one end and *vlan 3201* at the other end. List 1 illustrates an ESnet NSI command for setting up the loopback WAN path.
2) Send AMQP JSON-RPC commands to AmoebaNet at site A and B, respectively, to set up LAN segments for the paths. Here is the sequence of AmoebaNet commands that are needed to set up a LAN segment:
   a. lock();
   b. query_host_to_gateway();
   c. create_path();
   d. unlock();

For example, here is the create_path() command for setting up the LAN segment (i.e., the last mile) for data transfer job 1 at site A.

```
{
  "cmd":"create_path",
  "hosts":
    [{
      “startTime”: “immediate”,
      “endTime”: “2017-09-07 13:00”,
      "srcIp":"192.2.2.1″,
      "rate":"500″, // in Mbps
      "dstIp":"192.2.2.4″,
      "trafficType":"priority traffic",
      "vlanId":"2″,
      “routeType”: “host-to-gateway”
    }]
}
```

3) Send PING packets to verify that end-to-end paths are successfully established.
4) Launch data transfer jobs.

The experiment results are illustrated in Fig. 15. It can be seen in results that, from 1 to 20 min, data transfer job 1 has a throughput of ~300Mpbs; from time 6–15 min, data transfer job 2 has a throughput

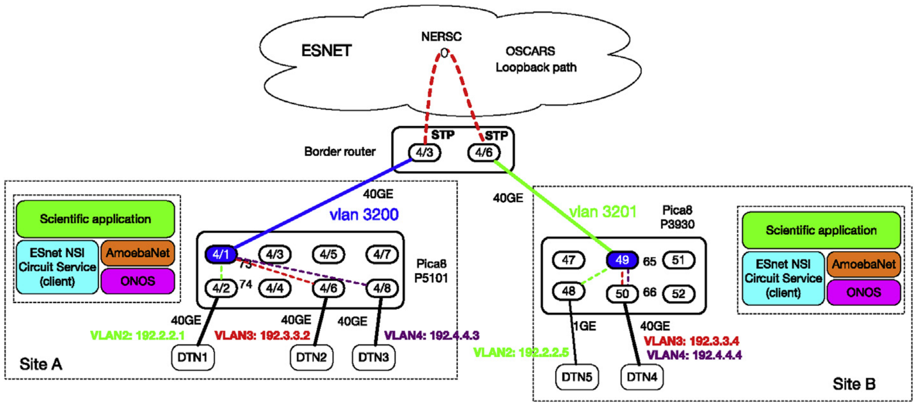

Fig. 14. Cross-domain end-to-end network path provisioning.

Table 3
End-to-end network paths.

	Srcip	Dstip	vlan	bandwidth
Path 1	192.2.2.1	192.2.2.4	2	300Mbps
Path 2	192.3.3.2	192.3.3.4	3	200Mbps
Path 3	192.4.4.3	192.4.4.5	4	500Mbps

List 1
ESnet NIS command to set up the loopback path.

```
onsareserveprovision\
-g 4b2beff5-11d8-4453-b255-b8a277e4e351
-d es.net:2013::star-cr5:6_1_1:+?vlan = 3200 \
-s es.net:2013::chic-cr5:3_2_1:+?vlan = 3200 \
-b 1000 \
-a 2017-08-9T20:00:00
-e 2018-07-30T20:20:20 \
-u https://nsi-aggr-west.es.net:443/nsi-v2/ConnectionServiceProvider\
-p es.net:2013:nsa:nsi-aggr-west \
-r es.net:2013:nsa:nsi-requester \
-h 131.225.2.17
-o 8443 \
-l./wenji.crt \
-k./wenji.key \
-i./etc/ssl/certs/ \
-x
```

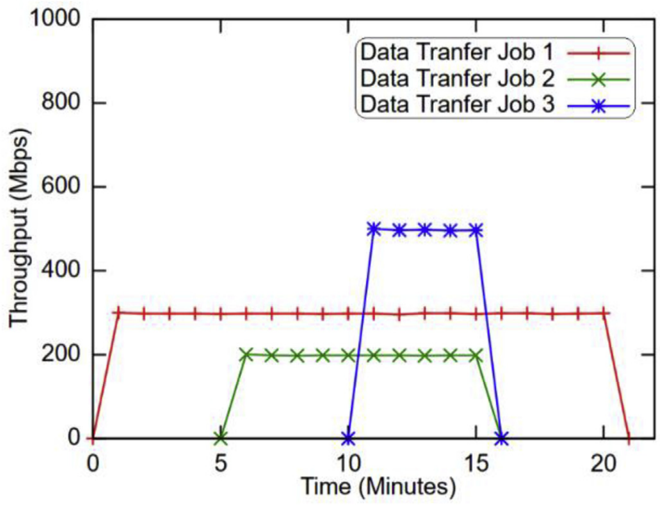

Fig. 15. End-to-end path provisioning.

of ~200Mbps; and from 7 to 15 min, data transfer job 3 has a throughput of ~500Mbps.

This experiment clearly demonstrates that, in addition to the programmability and fine-grain control already stated in the “End-to-end network path provisioning within a domain” test, AmoebaNet can address the last mile issue by acting as a campus network or local area network controller in conjunction with WAN connection services, such as ESnet OSCARS, to provision end-to-end network paths with guaranteed QoS across domains. In addition, AmoebaNet can multiplex/demultiplex multiple traffic flows to/from a single point-to-point layer 2 WAN circuit, with end-to-end QoS guaranteed, thereby addressing the scalability problem.

7. Conclusions

In the US, existing network paradigm that support big data science consists of three major components: terabit networks that provide high network bandwidths, Data Transfer Nodes (DTNs) and Science DMZ architecture that bypasses the performance hotspots in typical campus networks, and on-demand secure circuits/paths reservation systems, such as ESnet OSCARS and Internet2 AL2S, which provides automated, guaranteed bandwidth service in WAN. This network paradigm has proven to be very successful. However, to reach its full potentials, we claim that existing network paradigm for big data science must address three major problems: the last mile problem, the scalability problem, and the programmability problem.

We proposed the AmoebaNet solution to address these problems. AmoebaNet applies Software Defined Networking (SDN) technology to provide “QoS-guaranteed” network service in campus or local area networks. An initial version of AmoebaNet has been tested in Fermilab SDN testbed. The evaluation results show that AmoebaNet complements existing network paradigm for big data science: it allows applications to program networks at run-time to suit their needs; and, in conjunction with WAN circuits/paths reservation system such as ESnet OSCARS and Internet2 AL2S, it solves the last mile problem and the scalability problem.

Ideally, AmoebaNet should be compared with alternative solutions in terms of performance and capabilities. However, it is difficult for us to find alternative solutions to compare with AmoebaNet, due to several reasons. First, there are few alternative solutions that provide similar features as AmoebaNet in support of big data science. Second, although there many SDN research projects and efforts, few projects have developed software packages or systems that can be physically deployed.

Acknowledgments

This work was supported by the U.S. DOE ASCR network research program, by the program of the Construction and Operation for Large-scale Science Data Center (K-17-L01C06), and by National Research Foundation (NRF) of Korea (N-17-NM-CR01).

References

Al-Fares, Mohammad, et al., 2010. Hedera: Dynamic Flow Scheduling for Data Center Networks, vol. 10 NSDI.
Allcock, William, Bresnahan, John, RajkumarKettimuthu, Link, Michael, CatalinDumitrescu, IoanRaicu, and Ian Foster, 2005. The Globus striped GridFTP framework and server. In: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. IEEE Computer Society, pp. 54.
BBCP],2017 (accessed 02 April 2017), https://github.com/eeertekin/bbcp.
Bobyshev, A., Crawford, Matt, Phil, DeMar, Grigaliunas, V., Grigoriev, M., Alexander, Moibenko, Donald, Petravick, Rechenmacher, R., 2006. Lambda station: on-demand flow based routing for data intensive grid applications over multitopology networks. In: Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on. IEEE, pp. 1–9.
Bosshart, Pat, Daly, Dan, Gibb, Glen, Izzard, Martin, McKeown, Nick, Rexford, Jennifer, Schlesinger, Cole, et al., 2014. P4: programming protocol-independent packet processors. Comput. Commun. Rev. 44 (3), 87–95.
Chen, J., Alok, Choudhary, Feldman, S., Hendrickson, B., Johnson, C.R., Mount, R., Sarkar, V., White, V., Williams, D., 2013. Synergistic Challenges in Data-intensive Science and Exascale Computing: DOE ASCAC Data Subcommittee Report.
Curtis, Andrew R., et al., 2011. DevoFlow: scaling flow management for high-performance networks. Comput. Commun. Rev. 41 (4) ACM.
Dart, Eli, Rotman, Lauren, Tierney, Brian, Hester, Mary, Zurawski, Jason, 2014. The science dmz: a network design pattern for data-intensive science. Sci. Program. 22 (2), 173–185.
Data Transfer Nodes], 2017 (Accessed 10 April 2017), https://fasterdata.es.net/science-dmz/DTN/.
ESNet], (accessed 12May 2017), https://www.es.net/engineering-services/the-network/.
ESNet testbed], 2017(accessed 20 May 2017), https://www.es.net/network-r-and-d/experimental-network-testbeds/.
Geant AutoBAHN, 2015], (accessed 10 July 2017), https://forge.geant.net/forge/display/autobahn/Home.
Gibbard, Bruce, Dimitrios, Katramatos, Dantong, Yu, 2006. Terapaths: end-to-end network path qos configuration using cross-domain reservation negotiation. In: Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on. IEEE, pp. 1–9.
Guok, Chin, ESnet Network Engineer,David, Robertson, 2005. ESnet on-demand secure circuits and advance reservation system (OSCARS). In: Internet2 Joint Techs Workshop, Salt Lake City, Utah.
Hazlewood, Victor, Benninger, Kathy, Peterson, Greg, Charcalla, Jason, Sparks, Benny, Hanley, Jesse, Adams, Andrew, et al., 2016. Developing applications with networking capabilities via end-to-end SDN (DANCES). In: Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale. ACM, pp. 29.
Hong, Chi-Yao, et al., 2013. Achieving high utilization with software-driven WAN. Comput. Commun. Rev. 43 (4) ACM.
Internet2's Advanced Layer 2 Service],2017 (accessed 15 June 2017),https://www.internet2.edu/products-services/advanced-networking/layer-2-services/.
Jain, Sushant, et al., 2013. B4: experience with a globally-deployed software defined WAN. Comput. Commun. Rev. 43 (4) ACM.
Kim, Changhoon, Anirudh, Sivaraman, Naga, Katta, Bas, Antonin, Dixit, Advait, Wobker, Lawrence J., 2015. In-band network telemetry via programmable dataplanes. In: ACM SIGCOMM Symposium on SDN Research (SOSR).
Kreutz, Diego, Ramos, Fernando, Verissimo, Paulo, 2013. Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM.
Kreutz, Diego, Ramos, Fernando MV., Verissimo, Paulo Esteves, Esteve Rothenberg, Christian, Azodolmolky, Siamak, Uhlig, Steve, 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103 (1), 14–76.
Laki, Sándor, Horpácsi, Dániel, Vörös, Péter, Kitlei, Róbert, Leskó, Dániel, Tejfel, Máté, 2016. "High speed packet forwarding compiled from protocol independent data plane specifications. In: Proceedings of the 2016 ACM SIGCOMM Conference. ACM, pp. 629–630.
McKeown, Nick, 2009. Software-defined networking. In: INFOCOM Keynote Talk 17.2, pp. 30–32.
McKeown, Nick, et al., 2008. OpenFlow: enabling innovation in campus networks. Comput. Commun. Rev. 38 (2), 69–74.
Niranjan, Mysore, Radhika, Pamboris, Andreas, Farrington, Nathan, Huang, Nelson, Miri, Pardis, Sivasankar, Radhakrishnan, Vikram, Subramanya, Amin, Vahdat, 2009. Portland: a scalable fault-tolerant layer 2 data center network fabric. In: ACM SIGCOMM Computer Communication Review, vol. 39. ACM, pp. 39–50 no. 4.
Nunes, Bruno Astuto A., Mendonca, Marc, Nguyen, Xuan-Nam, Obraczka, Katia, Turletti, Thierry, 2014. A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutorials 16 (3), 1617–1634.
O'Kelly, Morton E., 1998. A geographer's analysis of hub-and-spoke networks. J. Transport Geogr. 6 (3), 171–186.
ONOS Project],2017 (accessed 05 Feb 2017), http://onosproject.org/.
Popescu, Diana Andreea, Gianni, Antichi, Moore, Andrew W., 2017. Enabling fast hierarchical heavy hitter detection using programmable data planes. In: Proceedings of the Symposium on SDN Research. ACM.
PRP], (accessed 12April 2017), http://prp.ucsd.edu/.
Roberts, Guy, Kudoh, Tomohiro, InderMonga, Sobieski, Jerry, MacAuley, John, Guok, Chin, 2014. Nsi Connection Service v2. 0, vol. 212. GFD, pp. 1–119.
Science DMZ],2017 (Accessed 10 April 2017), https://fasterdata.es.net/science-dmz/.
Scott-Hayward, Sandra, Gemma, O'Callaghan, Sakir, Sezer, 2013. SDN security: a survey. In: Future Networks and Services (SDN4FNS), 2013 IEEE SDN for. IEEE, pp. 1–7.
Shenker, Scott, et al., 2011. The Future of Networking, and the Past of Protocols. Open Networking Summit. .
Shin, Seungwon, Porras, Phillip A., Yegneswaran, Vinod, Fong, Martin W., Gu, Guofei, Tyson, Mabry, 2013. FRESCO: modular composable security services for software-defined networks. In: NDSS.
Sivaraman, Anirudh, Kim, Changhoon, Krishnamoorthy, Ramkumar, Dixit, Advait, Budiu, Mihai, 2015. Dc. p4: programming the forwarding plane of a data-center switch. In: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. ACM, pp. 2.
Vinoski, Steve, 2006. Advanced message queuing protocol. IEEE Internet Comput. 10 (6).
Wang, Richard, Butnariu, Dana, Rexford, Jennifer, 2011. OpenFlow-based Server Load Balancing Gone Wild.
Yeganeh, SoheilHassas, Tootoonchian, Amin, Yashar, Ganjali, 2013. On scalability of software-defined networking. IEEE Commun. Mag. 51 (2), 136–141.
Zhang, Liang, Wu, Wenji, DeMar, Phil, Pouyoul, Eric, 2018. mdtmFTP and its evaluation on ESNET SDN testbed. Future Generat. Comput. Syst. 79, 199–204.
Zhaogang, Shu, Wan, Jiafu, Di, Li, Jiaxiang, Lin, Vasilakos, Athanasios V., Imran, Muhammad, 2016. Security in software-defined networking: threats and countermeasures. Mobile Network. Appl. 21 (5), 764–776.
Zheng, Yan, Zhang, Peng, Vasilakos, Athanasios V., 2016. A security and trust framework for virtualized networks and software-defined networking. Secur. Commun. Network. 9 (16), 3059–3069.
Zurawski, Jason, Boyd, Eric, Lehman, Tom, McKee, Shawn, Mughal, Azher, Newman, Harvey, Sheldon, Paul, Wolff, Steve, Yang, Xi, 2011. Scientific data movement enabled by the DYNES instrument. In: Proceedings of the First International Workshop on Network-aware Data Management. ACM, pp. 41–48.
Zurawski, Jason, Ball, Robert, ArturBarczyk, Binkley, Mathew, Boote, Jeff, Boyd, Eric, Brown, Aaron, et al., 2012. The dynes instrument: a description and overview. In: Journal of Physics: Conference Series, vol. 396. IOP Publishing, pp. 042065 no. 4.

Syed Asif Raza Shah From fall 2014, S.A.R Shah joined KISTI (Korea Institute of Science and Technology Information) as a PhD research student of UST (University of Science and Technology), South Korea. His research interests include Software-Defined Network (SDN), Network Function Virtualization (NFV), cloud computing, virtualization, data science. Before joining the KISTI, he worked as an assistant manager (network & system security) at NESCOM (National Engineering and Scientific Commission) Islamabad. On behalf of KISTI, currently he is engaged in different international collaborative projects (e.g. BigData Express). Mr. Asif received his BS in Information Technology degree from Preston Institute of Science and Technology (PIMSAT) Karachi in 2006. He also completed his MS(CS) degree from Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST) Karachi in 2012. He has more than 9 years of industry experience in the fields of network and system administration. Contact him at asif@kisti.re.kr

Dr. Wenji Wu is a Principal Network Research Investigator at Core Computing Division, Fermilab, where he has worked on high-speed networking, distributed systems, and bulk data transfer. His research focus is to utilize multicore and many core to address performance challenges in high-speed networks. Dr. Wu is now responsible for two DOE network research projects, the MDTM project (http://mdtm.fnal.gov) and the BigData Express project (http://bigdataexpress.fnal.gov). He is also working on the WireCAP project (http://wirecap.fnal.gov). Dr. Wu earned his PhD in computer engineering from University of Arizona. Contact him at wenji@fnal.gov

Chin Guok joined ESnet in 1997 as a network engineer, focusing primarily on network statistics. He was a core engineer in the testing and production deployment of MPLS and QoS (Scavenger Service) within ESnet. He is the technical lead of the ESnet On-demand Secure Circuits and Advanced Reservation System (OSCARS) project which enables end-users to provision guaranteed bandwidth virtual circuits within ESnet. He also serves as a co-chair of the Open Grid Forum On-Demand Infrastructure Service Provisioning Working Group. His research interests include high-performance networking and network protocols; dynamic network resource provisioning; network tuning issues; hybrid network traffic engineering. Guok received a M.S. in Computer Science, from the University of Arizona in 1997 and a B.S. Computer Science, University of Pacific in 1991. Contact him at chin@es.net

Eric Pouyoul is a Senior System Engineer in ESnet's Advanced Network Technologies Group at Lawrence Berkeley National Laboratory (LBNL). His interests include all aspects of high performance big data movement, networking, hardware, software and distributed systems. He has been ESnet lead for designing Data Transfer Nodes (DTN) as defined in the Science DMZ architecture as well as ESnet's work in Software Defined Networking (OpenFlow). Mr Pouyoul joined ESnet in 2009 and his 25 years' prior experience includes realtime operating system, supercomputing and distributed systems. Contact him at lomax@es.net