Applying OSiRIS NMAL to Network Slices on SLATE

Jeremy Musser[1]*, *Ezra* Kissel[1], *Martin* Swany[1], *Joe* Breen[2], *Jason* Stidd[2], *Shawn* McKee[3], and *Benjeman* Meekhof[4]

[1]Department of Intelligent Systems Engineering, Indiana University, Bloomington, USA
[2]University of Utah, Salt Lake City, USA
[3]Physics Department, University of Michigan, Ann Arbor, MI, USA
[4]Advanced Research Computing - TS, University of Michigan, Ann Arbor, USA

Abstract. The Network Management Abstraction Layer (NMAL) extends perfSONAR capabilities to include automated network topology discovery and tracking in the Unified Network Information Service (UNIS), and incorporate Software Defined Networking (SDN) into overall operations of the OSiRIS distributed Ceph infrastructure. We deploy perfSONAR components both within OSiRIS and at our "client" locations to allow monitoring and measuring the networks interconnecting science domain users and OSiRIS components. Topology discovery (using an SDN controller application) and Flange Network Orchestration (NOS) rules are used to dynamically manage network pathing in our testbed environments. NMAL components have been containerized to operate within the Services Layer at the Edge (SLATE) infrastructure, and we describe our experiences in packaging and deploying our services.

1 Introduction

The Open Storage Research Infrastructure (OSiRIS) Network Management Abstraction Layer (NMAL) provides services for curating a near real-time model of the network as well as applying rules for the management and orchestration of the discovered network. SLATE provides a platform for Services Layer At The Edge, enabling containerized centrally defined and managed applications deployed from a curated application catalog. In this paper we will discuss a joint collaboration between OSiRIS and SLATE, incorporating NMAL capabilities to better support SLATE applications. Our recent efforts have allowed NMAL services to be integrated into the SLATE service deployment and orchestration platform and we will discuss that work and our results in this paper.

2 Background

2.1 OSiRIS

OSiRIS[1] is a collaboration of scientists, computer engineers and technicians, network and storage researchers and information science professionals from University of Michigan/ARC-TS (UM), Michigan State University/iCER (MSU), Wayne State University (WSU), and Indiana University (IU). Recently we have also collaborated with the Van Andel Institute (VAI) in Grand Rapids, MI to extend our Ceph[2, 3] cluster with a small, fast cache.

*e-mail: jemusser@iu.edu

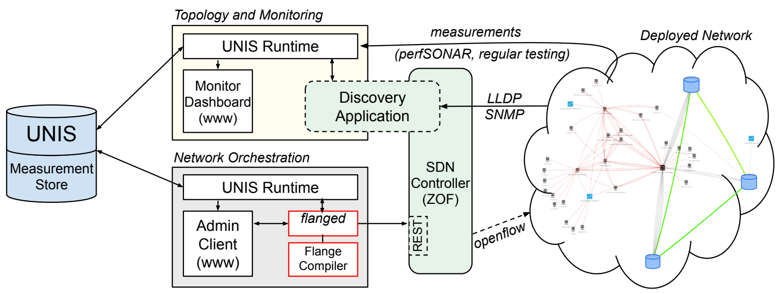


Figure 1. High-level diagram of key NMAL components.

OSiRIS is one of four NSF "Campus Computing: Data, Networking, Innovation: Data Infrastructure Building Blocks" (CC*DNI DIBBs) projects funded in 2015. The project focus is on prototyping and evaluating a software-defined storage infrastructure for three primary Michigan research universities, and is designed to support multiple science domains with one system. The project's goal is to provide transparent, high-performance access to a single distributed storage infrastructure from well-connected locations on any of the campuses hosting OSiRIS. By providing a single data infrastructure that supports computational access "in-place," it can meet many of the data-intensive and collaboration challenges faced by its supported research communities and enable them to more easily undertake research collaborations beyond the border of their own universities.

A single scalable infrastructure is easier to build and maintain than isolated campus data silos. Data sharing, archiving, security, and life-cycle management can all be implemented under one infrastructure. At the same time, the OSiRIS architecture allows customized configuration for each research domain to optimize the system for their needs.

2.2 NMAL

NMAL provides active network monitoring, management, and network orchestration for OSiRIS. It incorporates Periscope[4] monitoring components to extend perfSONAR[5] test points. This approach allows us to maintain a richer network model. With this model, our distributed system can optimize the network for performance and resiliency through SDN (Software Defined Networking). As shown in Figure 1, some key components include the Unified Network Information Service (UNIS) [6] that combines the capabilities of resource lookup and network topology services within a general data model and reference implementation, the Flange network orchestrator and domain specific language (DSL) [7] that drives network configuration and management, and a topology discovery and SDN component. The controller is informed by information collected in UNIS and managed by Flange, which allows for dynamically modified network behavior.

2.3 SLATE

The SLATE[8, 9] platform is being created to enable the rapid deployment of centrally defined and managed applications. Built to assist both small scientific collaborations as well as large NSF Facilities, SLATE provides a curated catalog of applications and an API server which harness Kubernetes[10, 11], Docker[12], and Helm[13]. SLATE uses these tools to facilitate deployment and operation of edge services such as Globus[14–16], perfSONAR, HTCondor CE[17–19], StashCache[20], Jupyter[21, 22] and other applications appropriate to meet the needs of SLATE users. This approach allows groups to reduce the complexity of bringing campus compute and storage resources to shared national cyberinfrastructure, such as the Open Science Grid[23, 24]. SLATE enables distributed automation, centralized delivery and operation of software, gateway and workflow infrastructure.

Beyond hosting the distributed applications necessary to analyze and process the large amount of data coming from the experiments and facilities, the SLATE platform is able to also host additional network applications in order to optimize the transfer of the data between sites. For the SLATE-NMAL collaboration, we packaged the UNIS, Periscope and Flange network applications and deployed them onto SLATE at various sites. SLATE already had perfSONAR in its catalog for deployment so we were able to deploy instances quickly to any point on the distributed topology. We used SLATE to deploy the Network Functions rapidly throughout the core of the network and to the remote sites. Using the NMAL tools on top of SLATE, we were able to manipulate the network flows at each point in the network to optimize data movement.

3 Methodology

In our approach, we deployed perfSONAR components across a national Internet2 SDN Testbed on top of the SLATE platform with an SDN controller, the UNIS data store, and the Flange NOS. By monitoring and measuring the networks interconnecting each site, we mimicked what science domain users might see between various classes of endpoints. Through these measurements, UNIS provided a topology database which the Flange NOS used to dynamically manage network flow. The SDN controller provided the network control interface for the Flange NOS.

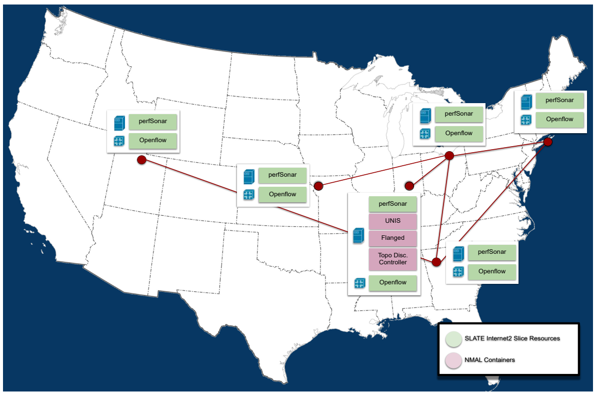


Figure 2. Internet2 SDN Testbed with NMAL and SLATE instances

3.1 Topology and Network Orchestration

Realizing the network abstraction and management goals of NMAL requires the ability to keep an accurate and up-to-date representation of resources. NMAL's discovery tools are built around managing metadata in UNIS for a standardized mechanism for reasoning about the network. NMAL incorporates tools built around SNMP, Traceroute, LLDP, perfSONAR, and additional Layer 3 metrics to supply other types of data to UNIS such as the service capabilities.

The Internet2[25] national backbone provides Advanced Layer2 Services[26] which supports an overlay SDN Testbed. On the Internet2 SDN Testbed, we can deploy different topologies across Points of Presence (PoPs). Figure 2 shows the full topology of the SLATE-NMAL collaboration across the national infrastructure. By provisioning SLATE at each PoP with perfSONAR and NMAL network applications, we were able test orchestration across disparate topologies.

To that end, the Flange DSL is used manage the selection of traffic flow between SDN-enabled switches. Each expression submitted may be static or predicated on conditional network sate. Figure 3 shows examples of the Flange rules with a predetermined threshold and generated candidate circuits.

The Flange NOS and DSL utilize topology discovery and measurement tools to provide automatic provisioning services to network administrators. OSiRIS administrators provide Flange with a program describing the parameters to be enforced on the network. Each instruction defines, either permissively or prohibitively, how data may flow between endpoints.

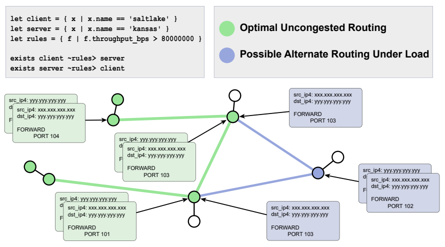
Flange compiles against the current network state to solve for conditionally optimal solutions for each asserted data flow. These solutions are abstract: Given a network state $\mathcal{S}$ on topology $\mathcal{G}$, Flange yields $\mathcal{S}'$. $\mathcal{S}'$ is interpreted through optional back-end modules. In detail, the compilation stage generates $\mathcal{S}'$ and registers a requested network state as a *netpath*. Netpath re-evaluation may be triggered by new programs, topological changes, or invalidating conditions on the network. NMAL flows can interact with multiple back-end agents independently. For example, a single path representation may be interpreted by both an OpenFlow switch and a P4 switch independently. This provides NMAL with the flexibility to adapt to new technologies and configuration mechanisms.

3.2 Controllers and Measurements

The ZoF[27] OpenFlow[28] controller is a python OpenFlow controller for managing traffic on OpenFlow capable switches. We have taken the ZoF framework and written two applications for NMAL that work side-by-side as the backbone of the network configuration system. The topology discovery application captures LLDP packets and supplements the existing set of devices detected by ZoF on startup to generate a network model. The second application serves as a configuration endpoint for Flange. Our variant of the ZoF REST application implements arbitrary OpenFlow 1.3 rules on top of the default behavior. Though we have implementations in both the Ryu[29] and ZoF frameworks, for large scale deployment, we have opted to use Ryu for its stability and complete documentation, and the ZoF controller for experimentation with its relative ease of modification and introspection.

While the SDN controller provides link-layer topology information, to maintain an accurate model of the behavior of the network, we also track the status of network resources with time series measurements within NMAL. Data is stored as a sanitized time series of key/value pairs, while descriptive information is stored as a separate metadata record. This approach allows the Flange NOS to treat measurements anonymously; the source of the measurement is abstract and largely irrelevant for the operation of the Flange NOS.

NMAL uses multiple separate measurement agents depending on context, measurement type, and network resource capabilities. The primary measurement comes from perfSONAR and tracks active latency and throughput across the topology, and host resources may also report path information from tools such as *traceoute*. Further measurements can be gathered through agents supporting host sflow or other node metric exporters. These measurements are aggregated as needed by the Flange NOS when resolving program solutions.



```
let client = { x | x.name == 'saltlake' }
let server = { x | x.name == 'kansas' }
let rules = { f | f.throughput_bps > 80000000 }

exists client ~rules> server
exists server ~rules> client
```

Figure 3. Programs written in the Flange DSL and compiled solutions.

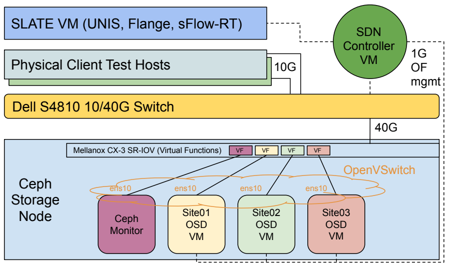


Figure 4. Testbed used to evaluate QoS tuning of Ceph cluster network traffic.

4 Evaluation

4.1 Wide Area Evaluation

We have shown the use of NMAL orchestration capabilities in collaboration with SLATE, controlling an Internet2 SDN testbed using SLATE-hosted resources to run the Flange NOS. The Flange NOS software was used to control traffic on a slice of the Internet2 SDN testbed with resources provided by the SLATE platform. We were able to demonstrate traffic flows being diverted dynamically between two different paths based on active measurement feedback. A Flange program was successfully used to control the selection of traffic flow between SDN-enabled switches based on a predetermined measurement threshold applied on either a throughput or latency dimension (Fig. 3).

4.2 QoS Evaluation

With a mature implementation of UNIS and Flange, one of the overarching goals of NMAL is to better manage the flow of traffic between OSiRIS sites. Such flow management is of particular importance when there exists asymmetric bandwidth between one or more deployments which presents a bottleneck condition during normal Ceph operations. This fact has necessitated throttling of Ceph operations to avoid creating denial-of-service behavior at sites with less bandwidth than other deployments.

One approach to resolving the capacity imbalance is to enable quality-of-service (QoS) configuration at each site to manage traffic originating from Ceph operations at the network layer. We are evaluating two general mechanisms to address the issues described above 1) applying priority queues to ensure that adequate bandwidth exists for Ceph operations to prevent timeouts, and 2) applying traffic shaping to provide better transport protocol performance between sites with asymmetric link capacities.

We have deployed a testbed to evaluate traffic shaping techniques using an SDN controller called OpenVSwitch (OVS) (Fig. 4). Three hosts in the testbed topology represents OSiRIS sites to which we can apply various latency, bottleneck, and QoS paramaters using a combination of OVS and Linux traffic classification. This testbed provides an opportunity to stress-test the mechanisms being evaluated, observe the behavior on a controlled Ceph installation (emulating the software running on OSiRIS), and develop a set of best practices for eventual deployment at the production sites.

The testbed was configured to simulate the behavior of OSiRIS; each host, *OSD1* through *OSD3*, contains an OVS bridge with a pre-configured queue with a max-rate of 900 Mb/s.

A single host (*OSD3*) is configured to induce a twenty millisecond latency and constrict bandwidth to 1Gbps.

In order to induce congestion on the network we used the RADOS benchmark tool[30] with 120 second runs. During this time, we observed the expected performance impacts from the induced latency and bandwidth constraints on *OSD3*.

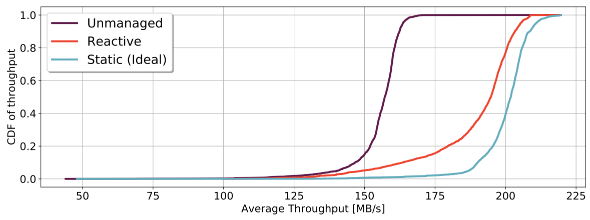


Figure 5. Distribution of average throughput per sample

4.3 Traffic shaping

To examine the effects of traffic shaping on the performance of the RADOS benchmark, we performed two distinct test runs using Flange to assert configurations onto the OVS bridges: a static flow set, and a reactive flow set. The Flange program for the first test explicitly defined the congested link and set the queue for egress traffic of the neighboring OSDs. This test represents a best case scenario, but also indiscriminate. The second Flange program is given an abstract condition: traffic on two links is above 200MB/s. When this condition holds, the program invokes the Flange NOS to add the queue rules to the uncongested links. This test represents a middle ground case, Flange detects and configures the network based entirely on network conditions. This implementation suffers from a lag time between the congesting event and the Flange NOS asserting new rules. In exchange, burst traffic can exceed the threshold.

Figure 5 shows the comparative performance of the three tests. With our configuration and 2000 sample sets we observed an average throughput of 155MB/s when fully congested and an average throughput of 200MB/s under ideal conditions for an improvement of 28.8% performance. With Flange managing traffic shaping actively we observed an average throughput of 188MB/s for an improvement of 21.6% performance.

5 Conclusions

Using the SLATE platform, we have deployed an operational implementation of the OSiRIS NMAL service running the Flange NOS both in the wide area case on the Internet2 SDN Testbed as well as on a simulated SDN testbed. We have also demonstrated the ability to generate responsive network configurations with minimal administrative overhead. Further, we have shown that we can mitigate the effects of asymmetric site capacities by carefully shaping network traffic. With the Flange NOS, this traffic shaping may be performed both proactively or reactively.

6 Acknowledgements

We gratefully acknowledge the National Science Foundation (grant 1541335, grant 1724821) which supported the work in this paper.

References

[1] S. McKee, E. Kissel, B. Meekhof, M. Swany, C. Miller, M. Gregorowicz, *OSiRIS: a distributed Ceph deployment using software defined networking for multi-institutional research* (2017), Vol. 898, p. 062045, `http://stacks.iop.org/1742-6596/898/i=6/a=062045`

[2] S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, C. Maltzahn, *Ceph: A Scalable, High-performance Distributed File System*, in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation* (USENIX Association, Berkeley, CA, USA, 2006), OSDI '06, pp. 307–320, ISBN 1-931971-47-1, `http://dl.acm.org/citation.cfm?id=1298455.1298485`

[3] The Ceph Foundation, (2012, August 11), *CEPH: An Open-Source, Unified Distributed Storage System*, Retrieved from `http://www.ceph.com`

[4] E. Kissel, D. Gunter, T. Samak, A. El-Hassany, G. Fernandes, M. Swany, Tech. rep., Technical Report 2011/04, University of Delaware (2011)

[5] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, M. Swany, S. Trocha, J. Zurawski, *PerfSONAR: A Service Oriented Architecture for Multi-Domain Network Monitoring*, in *In Proceedings of the Third International Conference on Service Oriented Computing (ICSOC 2005)* (2005), ACM Sigsoft and Sigweb, pp. 241–254

[6] A. El-Hassany, E. Kissel, D. Gunter, M. Swany, *Design and Implementation of a Unified Network Information Service*, in *10th IEEE International Conference on Services Computing (SCC 2013)* (2013)

[7] J. Musser, E. Kissel, G. Skipper, M. Swany, *Multi-layer Stream Orchestration with Flange*, in *IEEE International Conference on Fog Computing (ICFC 2019)* (2019)

[8] SLATE CI, (2019, April 3), *Services Layer at the Edge*, Retrieved from `https://slateci.io/`

[9] J. Breen, L. Bryant, G. Carcassi, J. Chen, R.W. Gardner, R. Harden, M. Izdimirski, R. Killen, B. Kulbertis, S. McKee et al., *Building the SLATE Platform*, in *Proceedings of the Practice and Experience on Advanced Research Computing* (Association for Computing Machinery, New York, NY, USA, 2018), PEARC '18, ISBN 9781450364461, `https://doi.org/10.1145/3219104.3219144`

[10] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes, Queue **14**, 70 (2016)

[11] Linux Foundation, (2020, May 25), *Kubernetes, Production-grade Container Orchestration*, Retrieved from `https://kubernetes.io/`

[12] C. Negus, *Docker Containers*, 2nd edn. (Addison-Wesley Professional, 2015), ISBN 9780134397511

[13] Helm Authors, (2020, March 5), *Helm, The Package Manager for Kubernetes*, Retrieved from `https://helm.sh/`

[14] I. Foster, C. Kesselman, The International Journal of Supercomputer Applications and High Performance Computing **11**, 115 (1997)

[15] I. Foster, C. Kesselman, The grid: blueprint for a new computing infrastructure pp. 259–278 (1999)

[16] Globus, (2020, May 18), *Globus*, Retrieved from `https://globus.org`

[17] M.J. Litzkow, M. Livny, M.W. Mutka, Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (1987)

[18] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Cluster Computing **5**, 237 (2002)

[19] D. Thain, T. Tannenbaum, M. Livny, Concurrency and computation: practice and experience **17**, 323 (2005)

[20] D. Weitzel, M. Zvada, I. Vukotic, R. Gardner, B. Bockelman, M. Rynge, E.F. Hernandez, B. Lin, M. Selmeci, *StashCache: A Distributed Caching Federation for the Open Science Grid*, in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)* (Association for Computing Machinery, New York, NY, USA, 2019), PEARC '19, ISBN 9781450372275, `https://doi.org/10.1145/3332186.3332212`

[21] M. Ragan-Kelley, F. Perez, B. Granger, T. Kluyver, P. Ivanov, J. Frederic, M. Bussonnier, *The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication.*, in *AGU Fall Meeting Abstracts* (2014)

[22] T. Kluyver, B. Ragan-Kelley, F. Pérez, B.E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J.B. Hamrick, J. Grout, S. Corlay et al., *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, in *ELPUB* (2016), pp. 87–90

[23] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein et al., *The open science grid*, in *Journal of Physics: Conference Series* (IOP Publishing, 2007), Vol. 78, p. 012057

[24] M. Altunay, P. Avery, K. Blackburn, B. Bockelman, M. Ernst, D. Fraser, R. Quick, R. Gardner, S. Goasguen, T. Levshina et al., *A Science Driven Production Cyberinfrastructure – the Open Science Grid* (Springer Netherlands, 2011), Vol. 9, pp. 201–218, ISSN 1570-7873, `http://dx.doi.org/10.1007/s10723-010-9176-6`

[25] Internet2, (2020, June 2), *Internet2*, Retrieved from `https://www.internet2.edu`

[26] Internet2, (2013, October 30), *Internet2 advanced layer 2 services*, Retrieved from `https://www.internet2.edu/products-services/advanced-networking/layer-2-services/`

[27] ZoF project, "OpenFlow Python3 Microframework" [software], version 0.19.0, Available from https://github.com/byllyfish/zof [accessed 2019-02-04]

[28] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, ACM SIGCOMM Computer Communication Review **38**, 69 (2008)

[29] Ryu SDN Framework Community, (2013, June 3), *Ryu SDN Framework*, Retrieved from `https://ryu-sdn.org/`

[30] The Ceph Foundation, (2015, March 13), *Benchmark Ceph Cluster Performance*, Retrieved from `https://tracker.ceph.com/projects/ceph/wiki/Benchmark_Ceph_Cluster_Performance`