2013 IEEE 10th International Conference on Services Computing

Design and Implementation of a Unified Network Information Service

Ahmed El-Hassany*, Ezra Kissel*, Dan Gunter†, Martin Swany*,
* *School of Informatics and Computing, Indiana University, Bloomington, IN 47405*
Email: {ahassany, ezkissel, swany}@indiana.edu
† *Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720*
Email: dkgunter@lbl.gov

***Abstract*—A holistic view of the network is key to the successful operation of many distributed, cloud-based, and service-oriented computing architectures. Supporting network-aware applications and application-driven networks requires a detailed representation of network resources, including multi-layer topologies, associated measurement data, and in-the-network service location and availability information. The rapid development of increasingly configurable and dynamic networks has increased the demand for information services that can accurately and efficiently store and expose the state of the network. This work introduces our *Unified Network Information Service* (UNIS), designed to represent physical and virtual networks and services. We describe the UNIS network data model and its RESTful interface, which provide a common interface to topology, service, and measurement resources. In addition, we describe the security mechanisms built into the UNIS framework. Our analysis of the UNIS implementation shows significant performance and scalability gains over an existing and widely-deployed topology, service registration, and lookup information service architecture.**

I. Introduction

Networking is central to distributed, service-oriented computing. Indeed the model of cloud computing derives its name from computing resources available in the network "cloud." At the same time, a consequence of location independence is considering the network as opaque with respect to computing services. Within the network community, there are many approaches to monitor and optimize network performance. Despite many efforts to bridge the gap, networking and computing services are often considered separately, with network-aware applications or application-driven networks remaining an elusive goal. This work unifies network and compute service representations, exposing actionable network information to services, and representing services as part of the network graph.

To improve their effectiveness, services designed to provide advanced network capabilities may benefit from knowledge about the networks on which they operate. This information may include the availability and location of services in the network, details about the attributes of attached devices, the characteristics of network links, and additional context for measurements that provide a picture of the current state of the network. Services must also be able to register their own information about what features and capabilities they provide in order to be discovered and utilized.

A key goal of this work is the unification of service, network topology, and measurement information into a common framework with a well-defined data model and schema representation. Describing complete topology and measurement information in a general way is made difficult since an understanding of multi-layer connectivity is often desirable or even necessary to many applications that wish to make use of distributed or cloud-based services. The task is further complicated when dealing with lower-layer network features such as link aggregation and protocol encapsulation and translation between network domains (e.g., VLAN tags and nested VLAN encoding), which is critical for performing pathfinding operations or exposing accurate network infrastructure measurements. The recent advancements in Software Defined Networking (SDN) have only increased the need for an information service that provides more complete view of the network that reflects both physical and "virtual" topologies and provides both real-time and historical data about network resources.

The understanding of network performance from an end-to-end perspective is one area where a complete view of the network, from end sites and data centers to regional service providers and backbone networks, is a critical requirement. In many high-performance computing environments in particular, network links may be dynamically configured using technologies such as OpenFlow [1] and OSCARS [2] to support high-demand flows. The collection of end-to-end real-time network measurements, identification of bottlenecks using metrics that relate to the detailed topology representation, analysis of application behavior to predict future network usage patterns, and the reconfiguration of network paths or application behavior are all key to achieving optimal utilization and efficiency in advanced network environments. If multiple network models are used for this task, the problem is made more difficult; reaction to changes in the network will be impeded, and the presence of multiple incompatible implementations introduces an administrative and deployment burden.

Driven by these concerns, we have developed a network data model that extends existing services and draws upon a number of best current practices. Our goal is to describe

978-0-7695-5026-8/13 $26.00 © 2013 IEEE
DOI 10.1109/SCC.2013.81

network resources as completely as possible while remaining flexible enough to handle future technology developments. In this work, we present our unified network model as a service: the Unified Network Information Service (UNIS). UNIS brings together the notion of *lookup* and *topology* services within a general data model and implementation. Accessible through a common API, UNIS maintains descriptions of multi-layer topologies and associated measurement metadata along with the services running within the network, allowing UNIS to answer complex queries about the network with minimal overhead.

The remainder of this paper is organized as follows: Section II provides additional background on the origins and goals of UNIS. A number of requirements for the UNIS service are outlined in Section III. Section IV introduces the UNIS data model followed by a description of the API in Section V and a discussion of security considerations in Section VI. A performance analysis of UNIS is shown in Section VII. Related work is covered in Section VIII and Section IX concludes the paper.

II. Background

Collecting and publishing service, topology, and related measurement data on a large scale has driven major efforts in a number of network communities. While successful in many areas, the lack of a consistent and general data model with associated APIs and common implementations has left a fractured landscape of approaches often tailored to specific use cases. Networks are frequently viewed from different perspectives: design, configuration and management, monitoring, and analysis. As a result, numerous models have been developed to represent each of these different aspects of the network. For example, NETCONF [3] exposes network device configuration and management information using the YANG [4] modeling language to describe the configurable elements. The perfSONAR [5] network monitoring system uses the NMWG [6] schema for describing network characteristics and measurement metadata. Each model and associated implementation(s) is designed and optimized with unique properties and levels of abstraction to serve a particular and limited view of the network, the end result of which is that we are often left with inconsistent and incompatible representations of the same underlying resources.

UNIS is primarily intended to be deployed in the context of the existing service-oriented multi-domain measurement framework perfSONAR (Performance-focused Service Oriented Network monitoring ARchitecture) [5], [7]. perfSONAR is deployed internationally on major research networks, including the US Department of Energy's ESnet, linking all major DOE laboratories and facilities, US Internet2 linking college and university campuses, Brazil's research network Rede Nacional de Ensina e Pesquisa (RNP), and Europe's GÉANT2 serving over 34 million users in the European research and education community.

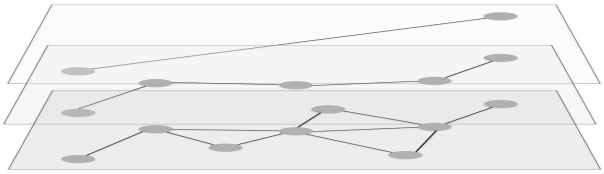

Figure 1. *Various Layers in Network Graphs, e.g. starting with a single TCP link (top)*

perfSONAR provides services for collecting and transforming measurements, with which UNIS will interoperate. The two services that UNIS would, in the perfSONAR framework, replace are the Lookup Service (LS) and Topology Service (TS). In the original perfSONAR design, measurement endpoints register in the LS with capabilities and user-defined attributes, and the TS collects topology information from the network. Thus, to perform the common query of finding relevant measurement endpoints for a given slice of the network, one must first locate a service using known attributes using the LS, then query the TS for "nearby" services, then return to the LS to find these services, and continue alternating between the LS and TS until all measurement endpoints are discovered. It was observed that unifying the LS and TS into a single topologically-aware service would improve both the efficiency and accuracy of the results.

This seemingly minor observation leads to a fundamental change in the model of the measurement framework. Rather than viewing lookup services as "support" for the primary entity of measurement points, UNIS models services, including measurement points, as *annotations on the topology graph of the network*. This in turn modifies the interaction model to one where, using hyperlinks, the existing services are accessed, on-demand, as a by-product of the navigation through the topology. This traversal of multi-layer network topologies is also facilitated through the UNIS data model.

Networks are obviously representable as graphs, a fact that underlies the understanding of topology in both the perfSONAR and UNIS implementations. There are many graph representations of networks and services, at many different granularities. The problem lies in how to represent these various aspects in a single model. The graph of processing elements is different than that of the graph of physical network paths. The network graph at "Layer 2" of the Internet model (e.g., Ethernet) is distinct from the graph at "Layer 3" (e.g., IP.) As shown in Figure 1, we can consider the "overlay network" of a TCP (Layer 4) connection between two nodes as an edge in the graph, and simultaneous consider the Layer 3 (L3, hereafter) topology that underlies this connection, and the L2 topology that underlies that. As described in Section IV, UNIS unifies

these layers through the concept of a vertical *relation*, allowing for easy traversal of complex topologies at different levels of detail and layers of the network stack.

III. Requirements

The requirements of any monitoring system stem from the goal of delivering useful views of the system's performance to the network operators, network managers, distributed project participants, and end users with minimal perturbation of that system. To this end, we highlight the following requirements:

R1. *Real-time discovery and navigation* of all relevant resources within a domain or network, or along a path. It should be not only possible, but easy, to build a client that can query and operate on network topology and measurements in order to monitor, configure, analyze or optimize the network.

R2. *Handle dynamically changing object state.* Dynamic Circuit Networks (DCNs) and SDNs, important components of modern networks, are both highly dynamic. In DCNs and SDNs circuits and flows can be created and terminated at any time. Creating, updating, and terminating circuits or flows must be reflected in UNIS in real time.

R3. *Provide security mechanisms that work in a multi-domain deployment.* Networks can be very large and in more than one administrative domain. Users, projects, etc. need to understand detailed performance across these domains. Thus, security credentials must be federated across multiple domains.

R4. *Provide extensibility to new topology elements.* There are many types of devices in a network, both virtual and physical. New types of devices occur, from a global perspective, continuously. Without any centralized reconfiguration, UNIS should be able to present new topology elements in a standard format, in relation to the overall topology.

R5. *Provide distributed configuration management.* Changing devices and services configurations in the network can potentially change the network topology. UNIS should be able to provide distributed configuration management for network services and devices.

R6. *Integrate with existing perfSONAR framework.* The UNIS should be able to, with protocol translators, replace the Lookup Service and Topology Service functions in existing perfSONAR deployments.

R7. *Leverage web architecture by following RESTful principles.* Efficiency and interoperability is enhanced by following the principles underlying the web and most large web-based services, in particular describing system objects as *resources* and using the HTTP verbs appropriately.

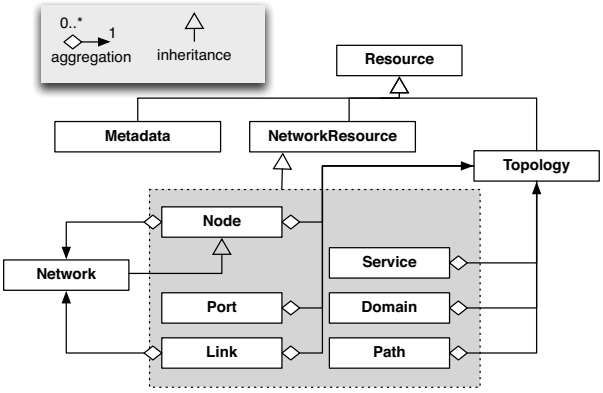

Figure 2. UNIS Object Model. All objects inherit from Resource, the objects in the grey box inherit from NetworkResource and are contained in a Topology.

R8. *Scale to extremely large systems.* We must consider both per-instance scalability and distribution across many instances, taking advantage of locality for efficiency.

R9. *Represent the multi-layer topology of the network with different levels of abstraction and granularity.* Networks can be represented in extreme detail down to the physical infrastructure, or simplified to a basic overlay view, depending on the context. UNIS must represent all views in a single framework. .

In the following sections, we highlight elements related to these requirements with the notation $\langle Rn \rangle$, where n is the requirement number.

IV. Data Model

In this section we present the UNIS data model. One of the key tenets of UNIS has been to use the same basic elements for network resources such as `Node`, `Port`, and `Link`, and to extend them as appropriate with layer specific attributes $\langle R4 \rangle$. These entities at different layers of the network stack are described along with their relationships both *horizontally* - L2 `Port` connected to L2 `Port` via L2 `Link`, and *vertically* - L3 `Port` atop L2 `Port`. This model then captures each of the logical components traversed as data flows through a network, and that ensemble can be referred to directly via another UNIS element, `Path`, abstracting the details when appropriate.

A UML diagram of the data model is shown in Figure 2. In UNIS, the physical and virtual devices, network services and the metadata about measurements are modeled as *resources* and connections between resources are modeled as *relations*. The remainder of this section describes these two object types.

A. Resources

A resource is an abstract type with time-varying state, and a location given as a URL. Each resource is uniquely and

globally identifiable and addressable by its URL. Network resources include: hosts, routers, network interfaces, disks, memory, and even on-chip networks for multi-core processors.

The abstract base class for all resources, `Resource`, contains a locally unique identifier, URL, schema (type) URL, and the timestamp of the last change in the resource. Thus, all resources must be typed and uniquely identified, and must be timestamped.

A `Resource` has only three direct subclasses:

- `NetworkResource` is the abstract base class of all other objects. It adds additional attributes for naming, describing, and locating an actual resource, that are generally applicable to any object on the network.
- `Topology` is logical collection of related network resources based on a criteria defined by the user.
- `Metadata` describes the type of measurement data (the event type), the resource(s) being measured (the subject of the measurement), and the particular parameters of the measurement. The subject is a hyperlink to the resource(s) being measured $\langle R7 \rangle$.

The leaves of the class hierarchy model the specific types of objects found in modern networks. A `Node` is generally a device connected to (or in) the network, that may be a physical machine or group of devices that connect at a single point. A `Port` connects a node to other network resources, via a `Link` (or `Path`.) Both `Ports` and `Nodes` may have forwarding rules. A `Link` is a unidirectional or bidirectional connection between two `Port` objects. A `Path` is an ordered list of connected `NetworkResources`, which can also be unidirectional or bidirectional. A `Domain` is a collection of `NetworkResources` that are part of an administrative domain. A `Service` describes a certain capability being offered by a `NetworkResource`. Finally, a `Network` is a collection of connected `NetworkResources`. To enable recursive composition, a `Network` can look like a `Node` with a list of `Ports` that connects it to other `Networks`.

B. Relations

The connectivity of the network of resources is expressed in a flexible and extensible form using *relations*. Every `Resource` has a list of relations. A single relation expresses a one-to-many relationship as a key/value pair, where the key is the type of the relation and the value is a list of hyperlinks to other network resources $\langle R7 \rangle$. This allows definition of one-to-one and one-to-many relations (many-to-many relations can be decomposed to two one-to-many relations).

For example, an IPv4 port has a relation "over" with the URL of the UNIS resource that represents the underlying Ethernet port $\langle R9 \rangle$.

The type of a relation can be any URN, allowing for extensibility $\langle R4 \rangle$. For interoperability it is recommended to use a set of relation types is defined by the Network Markup Language (NML) working group within the Open Grid Forum [8]. Some of the containment ("has a") relations from NML are included in the base UNIS model, as indicated by the aggregation arrows, such as between `Node` and `Topology`, in Figure 2.

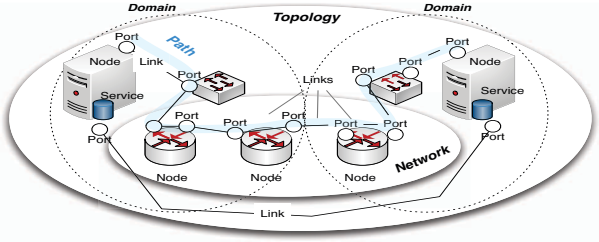

Figure 3. UNIS topology example

Relations can be used to tie together multiple representations for the same physical resources. For example, a set of load-balancing servers running a web service appear as one server to the clients. In UNIS, each server can be represented as a `Node`. An additional `Node` represents the clients' view of the load-balanced servers as a single server. The higher level `Node` links the other nodes by a specific relation, *implementedBy*, indicating that this `Node` is an abstract node for multiple servers $\langle R7 \rangle$ $\langle R9 \rangle$.

C. Example

An example of UNIS model applied to a simple network topology is shown in Figure 3. Here, two servers running a service (*e.g.*, a data transfer service), each in a separate domain. The servers are connected to their local switch, which are connected at layer 3 via routers; both the switches and routers are modeled as Nodes. The endpoints and routers are contained in a Path. Any subset of the Nodes, for example the routers, could be considered a Network. Notice that Port and Link objects can be used at any of the traditional network "layers", both to connect to peers at the same layer and to connect adjacent nodes on the stack.

V. UNIS API

In this section we present the design of the UNIS API and discus the implications of its RESTful implementation for our requirements.

A. Access API

The UNIS API is designed using the REST Architectural Style [9], thus leveraging widely deployed HTTP infrastructure and related standards $\langle R7 \rangle$.

Every `Resource` in the UNIS data model described above has a URL that identifies it and a uniform interface to manipulate it. The HTTP methods GET, POST, PUT and DELETE are used to manipulate resources. UNIS replaces purely RESTful dynamic discovery of a few of its core

Action	Verb	Noun	Description
Insert	POST	/{resource-type}	Creates new resource(s).
List	GET	/{resource-type}	Return all resources.
Get	GET	/{resource-type}/{id}	Return the resource representation
Update	PUT	/{resource-type}/{id}	Update the specified resource.
Delete	DELETE	/{resource-type}/{id}	Delete the specified resource.

Table I
UNIS REST API.

operations with pre-defined URL patterns, which are shown in Table I. In these patterns, the `resource-type` is any of the subclasses of `Resource` except `NetworkResource` (see Figure 2): node, port, link, path, service, domain, network, topology, or metadata. The `id` is a *locally* unique identifier assigned when the resource is first created.

B. Support for multiple representations

UNIS uses HTTP content negotiation [10] to support multiple representations. Each UNIS (HTTP) request and response is annotated with the `Content-type` header to indicate the data format of the carried message. Requests and responses use the `Accept` header to indicate the data formats supported by UNIS and the client. This gives the potential for UNIS to adapt more representations for its model $\langle R4 \rangle$.

To meet our scalability $\langle R8 \rangle$ and real-time $\langle R1 \rangle$ requirements, we needed to minimize the serialization, deserialization, and data transmission times. We chose JSON [11] as a native format for UNIS because it is universally supported and on par with other text formats in size and processing time. In general JSON is considerably more compact than XML, for example for a large UNIS topology representing much of ESnet the JSON representation is 811KB whereas a 1:1 equivalent XML representation is 1036KB, or almost 30% larger. An additional important factor was the availability of several highly scalable NoSQL "document-oriented" databases (*e.g.*, MongoDB, CouchDB, OrientDB, RavenDB) that use JSON to access and model data; see Section VII for a comparison with a far less performant XML database (the one currently used by perfSONAR). The other major advantage of using JSON is that it is the *de facto* standard web interchange format $\langle R7 \rangle$, so users can easily build web-based applications that consume UNIS data objects to visualize network topologies, graph performance measurement time-series, etc. For more efficient encoding and object traversal, UNIS supports the binary encoded JSON format, BSON [12].

C. Caching

UNIS is designed to take advantage of HTTP caching mechanisms to reduce the latency of frequently repeated requests $\langle R1 \rangle$, $\langle R8 \rangle$. In order to make UNIS HTTP cache-friendly, each response by UNIS is annotated with the last modification time, an entity tag (`ETag`) [10], and possibly an expiration time (if the network resource has a defined "lifetime" attribute). The client, or any proxy, can use the ETag in downstream queries to see whether its cached version of the resource can be used without fetching a new resource representation, thus reducing network bandwidth and load on the server.

D. HTTP Streaming

With the increasing use of SDN to dynamically control the network topology, changes in network state can be very frequent. Frequent polling of UNIS for changes in network resources can be expensive due to per-connection overhead; UNIS gives the clients the ability to avoid this overhead by re-using a single connection to subscribe to multiple updates for selected resource(s). This is done using standard HTTP mechanisms. When the UNIS client adds the HTTP `Keep-alive` header to a request, UNIS will automatically keep the TCP connection open for multiple updates $\langle R8 \rangle$.

E. Recording changes over time

Network configurations change over time and those changes might affect the behavior of the network. Keeping a historical record of network topology changes helps performance prediction, trend analysis, and anomaly detection. To enable these analyses, UNIS adds a timestamp to each change in the resources and allows users to retrieve old representations of the network resources $\langle R2 \rangle$. The total number of historical copies of the resources stored in the backend database is a run-time parameter. Timestamping resource states is also a form of versioning that can aid clients in achieving eventual consistency [13].

F. Example

An example of using UNIS is in the context of a distributed configuration management service $\langle R5 \rangle$. In this scenario, various network services are registered in UNIS as resources, and each service representation is annotated with its configuration. A network administrator or authorized configuration agent needs only to update the configuration in UNIS while each service can either subscribe or poll UNIS for configuration changes. If the service detects a change, it may then update or merge its running configuration with the current resource state in UNIS.

VI. Security Considerations

The dissemination of topology and network information in an open and unencumbered fashion can have a number of benefits. In R&E network environments, services such as OSCARS [2] take advantage of the perfSONAR LS and TS to enable the free exchange of topology, location, and control plane information for dynamic circuit provisioning between peering domains. While access to both topology and measurement data may be restricted to well-defined R&E network address space using firewalls or other external means, these widely used information services have typically not included robust authentication and authorization (AuthN/AuthZ) or accountability mechanisms. As networks continue to grow more dynamic and abstracted, and programmable network provide increased potential for disruption, the addition of security features for defining access control policy to known and trusted identities is a key challenge to meet in providing a next-generation information service. $\langle R3 \rangle$

To meet this challenge, the UNIS implementation makes use of the well-known and widely implemented public-key infrastructure (PKI) for the management of digital certificates using the X.509 standard. In this scheme, identity is determined via public keys, which are protected by an associated private key. Digital certificates map public keys with a unique entity and may be issued by a certificate authority (CA). These CAs act as trust anchors and can verify the validity of certificates. A typical UNIS deployment may store a number of CA certificates that correspond to trusted organizations or federated entities.

UNIS uses TLS/SSL to provide secure connections from peering UNIS instances or other clients. Connection requests over a secure port are required to present a client certificate to identify the remote peer. In this manner, basic authentication of requests may be achieved by verifying the client certificate has been issued by a trusted CA. The exchange of CA certificates between running UNIS instances allows for the authentication of users from peering domains.

Beyond certificate-based authentication, UNIS integrates the *ABAC* implementation and supporting libraries from ISI [14] to provide an authorization mechanism known as attribute-based access control. ABAC relies on attribute certificates (i.e., credentials) to map identities to specific access roles, and the ABAC decision engine generates a proof graph to determine if a given identity has rights to access or modify a given resource in UNIS. We have developed a modular authorization framework around ABAC in UNIS that allows for the expression of any number of access policies in an extensible manner. For example, network resources in UNIS may be restricted to a set of identities based on specific, well-defined properties of each resource, and the given authorization module would provide new REST endpoints for managing the addition and deletion of user credentials and roles. In effect, this allows UNIS to maintain a complete representation of the network while providing a number of customized *views* for privileged identities.

The authorization approach in UNIS also allows for the delegation of access privileges to particular entities through attribute certificate management. While the details are beyond the scope of this paper, the ability to grant a remote entity fine-grained access to existing roles is a powerful feature when considering in the scope of complex peering arrangements and multiple user scenarios. Delegated roles may also have lifetimes that limit access over predetermined windows. Such temporary credentials are useful when enabling short-lived services that require access to specific resources within UNIS.

Finally, we note that many existing distributed service-oriented frameworks, including perfSONAR and its authentication service, make use of AuthN/AuthZ mechanisms based on the exchange and management of X.509 certificates. This fact allows for the development of modules within the UNIS security framework to enable interoperability with existing architectures.

VII. Analysis

In this section we present the performance of UNIS for several basic operations and compare these results with the same operations using an instance of the perfSONAR TS. We limited our tests to perfSONAR TS because it shares most of the code with perfSONAR LS and both uses the same database backend. The results show that UNIS is consistently at least 10 times faster, and more scalable.

A. Test Environment

We used the same server of all our experiments. Our server has 8GB RAM, a quad-core Intel Core i7 processor, and a 32GB SSD hard drive. The clients are virtual machines (VMs) with close proximity to the server (round trip time (RTT) $<$ 4ms). Each client has one virtual CPU core and 2GB RAM.

UNIS is implemented using the non-blocking web server framework Tornado [15] and uses MongoDB [16] as a backend database. perfSONAR services are implemented using Perl and uses Berkeley DB XML [17] as its backend database.

B. Inserting a network resource

In the first experiment, we test the time to insert a single network resource to both TS and UNIS. We choose to insert a basic IPv4 port. Figure 4 shows the time per client for 1, 2, 4, 8, and 16 clients for UNIS and 1, 2, and 4 clients for TS. For up to 4 clients, UNIS ran at one to two orders of magnitude faster. Above 4 clients, a bug in the TS database implementation, related to DB locking, caused it to fail to respond with the error message ”Couldn’t open database”. Clients serialized on an exclusive write lock also explains the large variation in performance of TS for 4 clients.

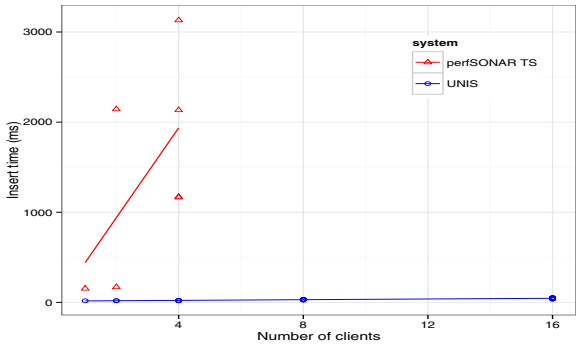

Figure 4. *Comparison of time for up to* 16 *concurrent clients to insert one network resource into a UNIS and perfSONAR topology.*

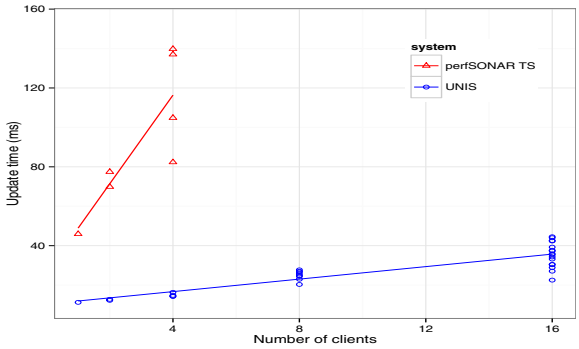

Figure 5. *Comparison of time for up to* 16 *concurrent clients to update one resource in a UNIS and perfSONAR topology.*

C. Updating a network resource

Our second experiment was to test updating the representation of a network resource. We chose to change the IP address of a port that was previously inserted by each client, so that each client updated a different network resource. Figure 5 shows that UNIS was roughly 10x faster than TS for up to 4 client and that, again, the TS implementation failed with more than 4 clients.

D. Querying network resources

Our third experiment was to test finding a network resource by its URN. We chose to query for a port that was previously inserted by each client, so that each client is a querying a different network resource. Figure 6 shows that UNIS was roughly 10x faster, and also far less variable, than TS. Unlike in the insert and update experiments, the TS service was able to handle queries from 16 concurrent clients.

In summary, we found that UNIS is a large improvement in terms of performance over the currently deployed perfSONAR implementation. We believe this is due mostly to the difference in performance between MongoDB and the Berkeley DB XML back-end; the limitations of the latter

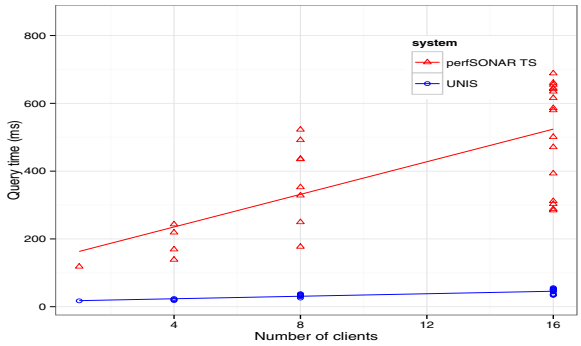

Figure 6. *Comparison of time for up to* 16 *concurrent clients to query one element in a UNIS and perfSONAR topology.*

are also reported in [18].

VIII. Related Work

There are a number of distributed and service-oriented architectures that share common goals with UNIS. In Section II we have described the evolution of UNIS as inspired by early work in the perfSONAR [5] efforts. Other measurement and network description frameworks include MonALISA [19], the Network Weather Service (NWS) [20], [21], and the Monitoring and Discovery Service (MDS) [22]. The focus of both MonALISA and the NWS is on system monitoring and the collection of measurement data while relying on a rudimentary topological view of the network that include basic reachability and connectivity information. Both MonALISA and the MDS include lookup service features for service discovery and registration, but are limited in scope with a bias towards grid environments. For example, the MDS is the Grid Information Service (GIS) in the Globus Toolkit [23], and while it accurately and completely describes grid compute nodes (CPU and number of cores, memory, disk, etc.), it lacks any features to describe complete networks or the interconnection between end hosts. In contrast, UNIS aims to provide a general data model that subsumes each of the capabilities just described and exposes them through common interfaces.

UNIS is also not alone among the many efforts to define network data models. NMWG [6] is used within perfSONAR, and UNIS has extended and re-engineered the schema into a more complete and efficient model that moves beyond classification of network characteristics and measurement methodologies. The Network Description Language [24] makes use of the Resource Description Framework (RDF), and while generic and expressible, there are efficiency impacts in parsing the verbose network representations using the RDF ontology. The Common Information Model (CIM) [25] models networks; but CIM is focused on the needs of enterprises to *manage* a large set of devices and therefore adds much complexity needed to exploit divergent

capabilities, which is irrelevant for understanding performance and in practice conflicts with UNIS requirements (see Section III) for independent extensibility $\langle R4 \rangle$, RESTful operation $\langle R7 \rangle$, and consistent multi-layer representation $\langle R9 \rangle$. Other related work in this area includes the Resource and Service Description (RSD) project [26] and Remos [27].

IX. Conclusion

In this paper, we presented our network information service as a unification for perfSONAR TS and LS services. UNIS integrates well with and takes advantage of the existing modern web architecture. UNIS offers an extensible model for network topology and measurements as a service, this enables designing smarter and possibly faster application-driven networks.

We showed how UNIS provides many advantages over the traditional perfSONAR services – such as, providing a security model, linking the measurement metadata to the network topology, keeping historical record of changes in the network, and leveraging web caching. Our experimental results show that UNIS is at least 10 times faster when compared head to head with perfSONAR services.

As future work, we would like to leverage UNIS's services for multi-layer multi-domain pathfinding in dynamic networks such as OSCARS and for complex service creation in GENI [28]. Also, we are interested in the topology-aware analysis of network measurements.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[2] Esnet oscars. [Online]. Available: http://www.es.net/services/virtual-circuits-oscars/

[3] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, June 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6241.txt

[4] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6020.txt

[5] A. Hanemann, J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, M. Swany, S. Trocha, and J. Zurawski, "PerfSONAR: A service oriented architecture for multi-domain network monitoring," in *In Proceedings of ICSOC 2005*, December 2005.

[6] J. Zurawski, M. Swany, and D. Gunter, "A scalable framework for representation and exchange of network measurements," in *TRIDENTCOM*, 2006.

[7] perfsonar. [Online]. Available: http://www.perfsonar.net/

[8] Network mark-up language working group (nml-wg). [Online]. Available: http://www.gridforum.org/gf/group_info/view.php?group=nml-wg

[9] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, 2000, aAI9980887.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, Internet Engineering Task Force, June 1999, updated by RFCs 2817, 5785, 6266, 6585. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

[11] "Json," http://json.org.

[12] "Bson," http://bsonspec.org.

[13] M. Shapiro and B. Kemme, "Eventual consistency," in *Encyclopedia of Database Systems*, 2009, pp. 1071–1072.

[14] Deterlab ABAC: Attribute-based access control. [Online]. Available: http://abac.deterlab.net/

[15] Tornado web server. [Online]. Available: http://www.tornadoweb.org/

[16] Mongodb. [Online]. Available: http://www.mongodb.org/

[17] Oracle berkeley db xml. [Online]. Available: http://www.oracle.com/technetwork/products/berkeleydb

[18] X. Xiang and B. Plale, "Performance evaluation of mysql 5.0 and berkeley db xml as a grid resource information manager (grim) with a benchmark/workload," Indiana University - Bloomington, School of Informatices and Computing, Tech. Rep. TR645, Feb 2007.

[19] I. Legrand, H. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, and C. Dobre, "Monalisa: An agent based," *Dynamic Service System to Monitor, Control and Optimize Grid based Applications, CHEP*, 2004.

[20] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," *Journal of Future Generation Computing Systems*, vol. 15, pp. 757–768, 1999.

[21] M. Swany and R. Wolski, "Representing dynamic performance information in grid environments with the network weather service," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, May, pp. 48–48.

[22] S. Fitzgerald, "Grid information services for distributed resource sharing," in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, ser. HPDC '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 181–. [Online]. Available: http://dl.acm.org/citation.cfm?id=874077.876489

[23] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications*, vol. 11, pp. 115–128, 1996.

[24] J. van der Ham, P. Grosso, R. van der Pol, A. Toonk, and C. de Laat, "Using the network description language in optical networks," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, 21 2007-yearly 25 2007, pp. 199 –205.

[25] I. Distributed Management Task Force. Common information model (cim). [Online]. Available: http://dmtf.org/standards/cim

[26] M. Brune, A. Reinefeld, and J. Varnholt, "A resource description environment for distributed computing systems," in *Proc. 8th Intern. Sympos. High-Performance Distributed Computing HPDC99*. IEEE Computer Society, 1999, pp. 279–286.

[27] B. Lowekamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok, "A resource query interface for network-aware applications," in *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, Jul, pp. 189–196.

[28] Global environment for network innovation. [Online]. Available: http://geni.net.