Requirements and Design Notes for the GENI Instrumentation and Measurement Architecture: perfSONAR and pGIMI

Guilherme Fernandes[1], Ezra Kissel[1], Martin Swany[1],
Matt Zekauskas[2], and Jason Zurawski[2]

[1] Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
{fernande, kissel, swany}@cis.udel.edu
[2] Internet2
1000 Oakbrook Drive, Suite 300
Ann Arbor MI 48104, USA
{matt, zurawski}@internet2.edu

Abstract. The Global Environment for Network Innovations (GENI) is a unique virtual laboratory for at-scale networking experimentation [8]. GENI will include a broad range of networking technologies running over facilities such as next-generation optical switches, novel high-speed routers, city-wide experimental urban radio networks, high-end computational clusters, and sensor grids. As a design goal, GENI will support extensive instrumentation that makes it easy to collect, analyze, and share real measurements.
Leveraging and Abstracting Measurements with *perfSONAR* (LAMP) is a project to create an instrumentation and measurement system, based on *perfSONAR*, for use by experimenters on ProtoGENI[29, 32, 20]. LAMP will collaborate, with other GENI projects and the Instrumentation and Measurement Working Group, on a plan to develop a common monitoring architecture and framework. This effort will include a representation of GENI topology to be used to describe measurements and experiment configuration, as well as an extensible format for data storage and exchange [9]. This document is an analysis of how *perfSONAR* can be leveraged as a basis for a common GENI monitoring infrastructure and explores the requirements for the I&M system and the integration of related measurement projects.

1 Introduction

The Global Environment for Network Innovations (GENI) is a unique virtual laboratory for at-scale networking experimentation [8]. The early focus of GENI is placed upon building a network infrastructure that can be used to setup and run experiments across slices of the GENI network. However, an equally important component of any experiment is measuring and observing the experiment. Creating *measurement infrastructure* is in many ways as challenging as creating the GENI infrastructure used by the experiments themselves. Creating such a measurement infrastructure involves first deploying measurement resources and then creating a service that can:

– Identify which measurement resources are available

– Select the resources to be used
– Specify what to measure (and at what level of detail)
– Record the measurement data in a common format
– Store/archive the data
– Filter/process the data
– Locate distributed data from around the framework
– Provide users with (secure/authenticated) access to the data, and support tools for viewing the data

In short, adding a "measurement plane" to a user's experiment/slice is critical functionality for GENI. Leveraging and Abstracting Measurements with *perfSONAR* (LAMP) is a project to create an instrumentation and measurement (I&M) system, based on *perfSONAR*, for use by experimenters on ProtoGENI [32, 20]. Together with our partners, we will create a prototype ProtoGENI Instrumentation and Measurement Infrastructure (pGIMI). Our goal is to develop a comprehensive and easy to use system for I&M.

The high-level goal of this effort is to simplify the task of instrumentation and measurement, making it easy to set up the measurement system so that users can focus operation of their experiment, not the measurement system. The LAMP I&M system aims to not only make it possible for users to instrument and obtain measurement data, but also actually simplify the task by helping them create and interact with their slice's measurement plane. This system will build on, and extend, the initial instrumentation and measurements capabilities being developed as part of the ProtoGENI cluster work and will ideally extend into the larger GENI infrastructure. The LAMP I&M system will be realized by adapting the burgeoning *perfSONAR* network measurement framework to provide performance measurements from the various components of the GENI environment.

Another important aim of the LAMP project is to collaborate with the GENI I&M working group [9] and other GENI I&M projects in designing a common monitoring architecture. Integrating different measurement systems requires, among other things, the use of a common but extensible format for data storage and exchange, the use of a common representation of GENI topology to describe measurements and experiment configuration, and, when possible, the use of similar and well-defined component APIs. In fact, these challenges have been key guiding principles behind the design and implementation of the *perfSONAR* measurement framework. Key features of *perfSONAR* that can be leveraged by GENI to expedite the definition of a common I&M architecture include:

– **Extensible, unified network and host metrics.** The basic *perfSONAR* measurement format and storage infrastructure can be extended to uniformly store all instrumentation in the GENI system. The different GENI I&M projects can easily define their own extensions if existing metrics do not apply. Furthermore, individual GENI slices can define fully compatible extensions for *experiment-specific* metrics. The use of such a format maximizes collection, storage and analysis code reuse and makes comparison possible.

– **Explicit representation of network topology, tied to measurements as metadata.** The Unified Network Information Service (UNIS) [38] representation outlined in this document allows for a uniform expression of all the GENI infrastructure. In addition, it implicitly provides a configuration interface for these dynamically-configurable resources.
– **Measurement sharing.** The *perfSONAR* system is easily extensible to represent any type of performance values and events. Widespread and easy measurement sharing requires a common format that is extensible enough to support new constructs of the sort envisioned and proposed to run on GENI.
– **Proven measurement architecture components and APIs.** The *perfSONAR* framework identifies several key components necessary in a wide-scale, federated, interoperable measurement infrastructure. Requirements and standardized interfaces for exchange and operation are defined for each component. The *perfSONAR* framework is also extensible enough to incorporate new types of components into the existing architecture.

This document presents an initial analysis of how the *perfSONAR* framework can be leveraged to integrate the different GENI I&M systems under a common instrumentation and measurement architecture. Specifically, Section 2 of this document presents broadly the general requirements and motivations for a common I&M architecture. Section 3 provides an overview of the *perfSONAR* framework, describing core components and the formats used for representing measurement data and topology information. Section 4 discusses the first steps needed to integrate the existing GENI I&M systems based on *perfSONAR*. Finally, we conclude and outline future work in Section 5.

2 GENI Instrumentation and Measurement

GENI is in the second phase of exploratory rapid-prototyping, called GENI Spiral 2 [11]. One focus of this new phase is the development of architecture, tools and services enabling experiment instrumentation. For this purpose, several projects have joined the GENI prototyping effort with the objectives of designing and prototyping instrumentation and measurement (I&M) systems for GENI. There are currently at least ten different projects working to provide I&M services in GENI [11], each following their own methodology and targeting specific, some times overlapping, use cases. While this parallel prototyping approach is a good way of identifying requirements, difficulties and solutions in providing I&M services across the highly heterogeneous GENI environment, it also creates a highly diverse, sometimes incompatible set of I&M systems. This in turn reduces the usability of GENI to experimenters that now need to understand how to operate services and how to retrieve and interpret measurement data from the different I&M systems.

The GENI Instrumentation and Measurement Working Group [9] has been created with the objective of designing an architectural framework for GENI's I&M infrastructure. The first draft of the GENI I&M Architecture Document [10] outlines requirements, use cases and components for a common GENI I&M framework. Specifically, this doc-

ument identifies:

- Several key Instrumentation and Measurement services, informed by the existing GENI I&M projects' proposals
- The need for standardized interfaces, protocols and schema for measurement data exchange and representation
- The need for standardized interfaces, protocols and APIs for using the I&M services
- The need to locate distributed measurements both from experiments and the physical support infrastructure
- Ownership and privacy concerns for experimenters data
- Experimenters' workflows for experiment and I&M configuration
- Use cases and other requirements

In this document, we concentrate on the first four items and describe how the *perfSONAR* framework can be used as a starting point to quickly achieve the desired goals for a common GENI I&M architecture. Several of the I&M services identified in [10] have direct counterparts on the *perfSONAR* architecture (i.e. Measurement Point, Measurement Analysis and Presentation, Measurement Data Archive). Other services (i.e. Measurement Orchestration and Measurement Collection) are welcome extensions to the current *perfSONAR* framework. The functionality of these services have been historically provided underneath the *perfSONAR* middleware layer, without explicit representation on the *perfSONAR* framework, but are still part of any *perfSONAR* measurement infrastructure. In Section 4, we elaborate on the commonalities and mappings between the components defined by the *perfSONAR* architecture and the services identified on the GENI I&M Architecture Document and on the design of different I&M projects.

The foundation of the *perfSONAR* approach to network monitoring is the definition of standard, fully featured ways to represent measurement data. All *perfSONAR* data formats are derived from the groundbreaking work of the the Open Grid Forum (OGF) working groups on network measurement [25]. Data representation relies on two simple principals:

- Identification of common "shared" components as well as the variable "dynamic" components of a measurement
- Incorporation of the above into scalable XML formats

Once defined, these data formats lend for easy storage, exchange, and search. By encouraging service developers to buy into the *perfSONAR* architecture, API development and re-use becomes commonplace. Construction of a data specific API to access a particular type of measurement, e.g. utilization of a network link, can be built using existing libraries and data abstractions for other services in the *perfSONAR* architecture. In addition to data storage, all *perfSONAR* services communicate using a common protocol. This message structure is a well scripted and documented interaction between services, clients, and other aspects of the framework. The protocol is based in part on the data representation: this allows for significant reuse of core libraries and APIs.

Finally, measurement systems are only useful if they are easily locatable and searchable for material of interest. Integration of measurement subsystems with a discovery

framework is key to the success of *perfSONAR*. The Unified Network Information Service (UNIS) is a solution to the problem of measurement discovery. UNIS combines a proven measurement location system, the *perfSONAR* Lookup Service (LS), with a developing solution for integrating knowledge of network topology, the *perfSONAR* Topology Service. When combined, these two components are able to foster a complete view of network performance and awareness, and prove to be vital to all components of GENI including the Control Frameworks (CFs).

The following section will describe the current state of the *perfSONAR* framework. Following this, we describe ways to integrate the existing *perfSONAR* solutions into a unified I&M system for GENI. We conclude with some final remarks about the nature of the solution and next steps for LAMP.

3 perfSONAR

perfSONAR is a framework that enables network performance information to be gathered and exchanged in a multi-domain, federated environment. The goal of *perfSONAR* is to enable ubiquitous gathering and sharing of this performance information to simplify management of advanced networks, facilitate cross-domain troubleshooting and to allow next-generation applications to tailor their execution to the state of the network. This system has been designed to accommodate easy extensibility for new network metrics and to facilitate the automatic processing of these metrics as much as possible. GENI is developing a unified monitoring infrastructure in which all layers of the system can be configured and customized.

perfSONAR is a collaborative project among several national R&E networks and partners. The complete set of participants is available from the *perfSONAR* web site [29] — the core network operator participants are: the US DOE's ESnet, the E.U.'s GÉANT, Internet2, and RNP in Brazil [6, 7, 17, 33]. While *perfSONAR* is currently focused on publication of network metrics (e.g. one- and two-way latency, achievable bandwidth, utilization), it is designed to be flexible enough to handle new metrics from technologies such as middleware or host monitoring.

A focus of the *perfSONAR* project has been to define standard schemata and data models for network performance information. As such, *perfSONAR* has emerged as a community-driven standard for unified network performance monitoring via forums such as the Open Grid Forum (OGF). Development of actual, interoperable implementations has followed the Internet Engineering Task Force (IETF) spirit of multiple working interoperable implementations. There are numerous organizations producing *perfSONAR*-compliant software implementations at this time. The core data models and protocols used in *perfSONAR* were designed to be extensible to new metrics. The key to the success of *perfSONAR* has been the unification of various network-related metrics in a single framework. Basic *perfSONAR* services have been implemented to report host metrics as well.

Previous work on *perfSONAR* have described the original overall architecture[13], the data model and schemata [41], network topology [3], the *perfSONAR* Lookup Service [40] and experiences in *perfSONAR* deployment [2, 12].

3.1 Architecture

perfSONAR is an example of a Service Oriented Architecture (SOA), which offers the ability for specialized, autonomous services to join under a common access scheme. Thus, it is possible to separate the roles of monitoring, storage, processing, and visualization of data into specialized service instances.

The different *perfSONAR* components are implemented using Web Services (WS) technology therefore the interaction between services and between clients and services are performed using well defined language independent interfaces. All WS interfaces are defined using eXtensible Markup Language (XML). *perfSONAR* web services furthermore extend an XML schema defined by the OGF's Network Measurement Working Group NM-WG [23]. These schemata are used to provide a uniform encoding and exchange mechanism for performance information within and across administrative domains. This vastly simplifies system component interactions as well as storage requirements. *perfSONAR* also utilizes the network characteristic taxonomy defined by the OGF NW-WG [21].

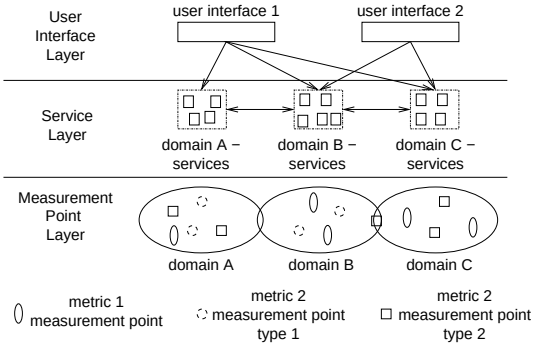

Fig. 1. perfSONAR Measurement Framework

3.2 Components

In this section, we briefly describe the applications and services that make up *perfSONAR*. The core components of the *perfSONAR* architecture used in this case are the data producers - Measurement Point (MP) and Measurement Archive (MA) services, data consumers (Analysis clients) and discovery - Information Services (IS). The MPs and MAs are responsible for exposing performance metrics, and, in the MA case, in potentially storing metrics for later retrieval. The IS is responsible for helping clients find available services and even finding relationships between specific network topology elements.

Measurement Archives The storage of archived measurement data can take on many forms (e.g. relational databases, RRD[34], flat files). Regardless of the proposed archive

solution, a general truth is that most storage schemes require a lightweight and capable mechanism for retrieving the data. A hallmark of the *perfSONAR* design is the design and functionality of the various services that implement a Web Services (WS) [39] interface around measurement storage schemes. The "Measurement Archive" (MA) is a common feature in *perfSONAR* deployments due to it's utility: the ability to store and provide data, potentially of diverse types, in a compact yet efficient manner. Current implementations of the MA have focused on the storage of "popular" network measurements including one way latency [28], two way latency [31], passive measurements [5], and bandwidth [4], [19], [24], [37].

Measurement Points The measurements that *perfSONAR* archives and exposes must come from lower level tools: either passive or active measurement systems are required to gather network metrics. The *perfSONAR* Framework has the notion of "Measurement Points" (MPs), services that offer an interface to low level measurement tools. The interface may function in an "on demand" fashion, e.g. allowing a user to request a live measurement, or in a "scheduled" fashion, e.g. where the MP may be configured to make regular measurements, and store the results in local storage or through an affiliated MA.

Information Service The *perfSONAR* Information Service (IS) is used for service registration, service discovery, data discovery, and network topology representation. These services were previously separated into a Lookup Service (LS) and a Topology Service (TS), but those systems overlap significantly in some cases. The query syntax of the two is essentially the same, and the infrastructure used to support local registration and global discovery is common as well, so these were merged into a single IS.

The discovery function of the IS involves accepting registration information from *perfSONAR* services. As each component updates its information, other components and clients may locate these deployed services via queries. All service descriptions and network metrics, (both actual data and descriptions of the types of data an MP may collect) are defined using XML schema and encoded in XML. Included in this function are mechanisms to allow for scalable wide-scale deployments. These mechanisms include a hierarchical structure with different levels of summarization of data, synchronization among top-level IS, and leader election algorithms.

The topology service functionality within the IS stores a representation of the elements of the network topology. This is used for pathfinding, representing relationships between elements about which performance data has been gathered, to make decisions about topologically-appropriate network services, and network visualization.

3.3 perfSONAR-PS

perfSONAR-PS (pS-PS) is a set of independent software services that implement the *perfSONAR* protocols for network performance monitoring. pS-PS services are designed to be compatible with all other *perfSONAR* software that implements the *perfSONAR* protocols. pS-PS is able to federate between deployments, particularly those

that span multiple domains, making the job of solving end-to-end performance problems on paths crossing several networks much easier to address. [30]

The pS-PS software suite is developed almost entirely in the Perl programming language, taking full advantage of numerous language features and benefits including the Comprehensive Perl Archive Network (CPAN) distribution system. This software manager makes pS-PS the ideal choice for integration into typical NOC environments. Appendix 6 summarizes currently available pS-PS components.

3.4 Deployment Footprint

perfSONAR data is currently available on major R&E backbone networks including ESnet, Geant, Interent2, and RNP. Regional and National Research and Educational Networks (**NREN**s) both in the US and Europe are adopting *perfSONAR* tools at a rapid pace. Campus networks, the "end of the line" in many cases, are also realizing the importance of these tools for their users. Virtual Organizations (**VO**s) from scientific disciplines, such as the LHC project, continue to experiment with the viability of on demand testing as well as stitching together measurements from the networks of the world.

3.5 Schemata

The different *perfSONAR* components are implemented using Web Services (WS) technology therefore the interaction between services and between clients and services are performed using well defined language independent interfaces. All WS interfaces are defined using eXtensible Markup Language (XML). *perfSONAR* web services furthermore extend an XML schema defined by the Open Grid Forum (OGF) [25].

These schemata are used to provide a uniform encoding and exchange mechanism for performance information within and across administrative domains. This vastly simplifies system component interactions as well as storage requirements. *perfSONAR* also utilizes the network characteristic taxonomy defined through the the OGF's Network Measurement Working Group (NW-WG) [21].

The schemata provide an abstract framework with an explicit separation of the data values, which are expected to be quite numerous, from the less dynamic metadata that describes that data and is, by comparison, rather static. This basic concept, coupled with the notion of a "base" format explicitly designed with extension in mind, delivers a simple yet powerful vehicle for the description of network measurements. This basic approach delivers efficiency wherein metadata can then be stored, searched, and transmitted separately from the data sections. It is also possible to use identifying keys to aid in the explicit linking of the metadata and data sections; even when they are not stored in the same location.

To clearly represent the diversity offered in both network performance data and network topology, it is necessary that the basic representation be as simple and extensible as possible. The "Base Schema" instances realize the goals set forth by the NM-WG to minimally describe information, and provide ample room to extend the approach to

more complex ideas and concepts. Section 3.6 describes the schemata used in the storage and exchange of network measurements. Section 3.7 describes the schema components that are used to define network topology — a concept that is important for network measurements as well as control frameworks that describe network components.

3.6 Measurement Description

A key motivating factor in the design of the NM-WG data representation format is the need to balance interoperability and flexibility. Agreeing on standard mechanisms for sharing data in a large and diverse group like the OGF has made it clear that defining an interface and storage format is difficult, and there are many different environmental issues to consider. Any solution which is so rigid as to preclude the inevitable advances in this area will not be successful. The goals of our measurement and monitoring framework must address:

- Normalized data encoding in canonical formats
- Extensibility to new data sources
- Flexible re-use of basic components
- Incorporation of existing solutions and technologies
- Language/Implementation independence

Keeping these in mind, the basic goal of the storage and exchange format is to allow the separation of rapidly changing information, henceforth the "*data*", from relatively constant information, or the "*metadata*". A simple example involves a **traceroute**, which would have as *data* the IP address, time and measured value of each network probe. This is completed with associated *metadata* that includes the source and destination host of the entire operation along with any specific parameters that were specified.

The care taken in the design of this separation leads to an obvious gain in efficiency when it comes to storage and delivery. *Metadata* descriptions may be re-used across multiple *data* sets leading to faster search procedures and a reduction of storage requirements. *Data* sets are minimal, containing only information they require to be useful and utilize encoded identification methods to identify any *metadata* instance they may be related to. Appendix 6 goes into great detail regarding the design of the perfSONAR schemata.

3.7 Topology Description

Network measurements need to refer to the elements of network infrastructure as the *subject* of their data. Initially, we made efforts to define canonical forms of recurring network elements. Subsequently, we observed that if we included the relationship between those various elements, we would arrive at a representation of the topology of the network. This topology representation is useful in its own right and can be applied in a variety of ways, including:

1. Determining relevance of network measurements

2. Representing which elements share infrastructure
3. Location of appropriate points from which to measure

Network topology is a rich subject, with many possible approaches available to solve complex modeling problems. Appendix 6 goes into great detail regarding the design of the perfSONAR topology schemata. This section is provided as a supplement, as the design of network topology representation does not impact the overall purpose of having it available through the network monitoring subsystem.

4 Integration

4.1 Topology Representation

Network measurement representation is intrinsically tied to network topology description. This need to represent network topology information culminated in the NM-WG topology schema used by *perfSONAR* as presented in Section 3.7. However, the topology schema was conceived to represent network topology in general (i.e. without being tied to the network measurement domain), and has in fact been adopted by other network based services, most notably the Inter-domain Controller Protocol [14] (ION [18], AutoBAHN [1], OSCARS [27]). This greater community has proved a valuable source of use cases and insights on the general representation of network topology, which have guided extensions and other modifications to the original NM-WG topology schema. This emerging network topology schema is evolving into what is being called the Unified Network Information Services (UNIS) schema.

The UNIS topology schema builds upon the same base elements as defined in the NM-WG topology schema (see Section .1), e.g. *domains*, *nodes*, *ports*, *links*, *networks*, *paths* and *services*. The UNIS schema also establishes the same guidelines for the identification scheme of the different topology elements, albeit now recognizing that specific domains may have greater flexibility provided that they are self-contained. The notion of relationships among topology elements, as represented by the relation schema element, continues to be a central part of the topology schema. The Network Markup Language Working Group (NML-WG) of the OGF is working to combine the efforts of multiple projects (including UNIS) in describing network topologies into one standardized network description ontology and schema. The UNIS schema closely follows the current (in-progress) NML schema.

A number of key factors make the UNIS effort relevant to the needs of GENI I&M. Besides the need to represent topological elements as subjects of network measurements and instrumentation, several components of the GENI architecture require a standardized way to represent the resources that compose GENI's substrate and relations between them. To date, this has been addressed by the GENI RSpec. RSpec defines a data structure describing the set of resources available to an experimenter. In order to obtain or configure these resources, a GENI user may invoke the specified RSpec via privileged operations (GMC calls). The RSpec describes the substrate resources through a core schema and a standardized extension mechanism.

While some GENI clusters have built upon existing multi-layer network topology schemata to define their RSpec (e.g. ORCA and the Network Description Language

(NDL) schema [3]), other have takes a more simplistic, ad-hoc approach to network topology representation. In many cases, these approaches provide only the basic topological elements suitable for their substrates, e.g. defining nodes, links and interfaces operating primarily on the Ethernet and IP layers. This method of representing ProtoGENI nodes has been expanded through the RSpec extension mechanism along with an identification scheme, highlighting the need for a flexible and unified approach to resource and topology representation.

The RSpec is at the core of the GENI workflow, and the latter ad-hoc approaches will prove hard to scale to the more complex environments that are targeted by GENI. The benefits of using a general network topology schema as the basis for the network elements representation in the RSpecs are compounded by the ability to integrate with other components of the GENI architecture that also require network topology representation (e.g. the Operations and Management plane for Fault and Performance Management, and, the focus of this document, the Instrumentation and Measurement plane). In this context, we find it appropriate to present an example of how the UNIS schema could be used as the basis for the ProtoGENI RSpec. (Note that the description presented in Section 3.5 for the topology representation in network measurements applies directly to the needs of the I&M architecture). The following XML document shows the ProtoGENI RSpec Tunnel Example based on the UNIS topology schema.

```
<pgeni:rspec xmlns:pgeni="http://www.protogeni.net/resources/rspec/2"
    xmlns:gretun="http://www.protogeni.net/resources/rspec/ext/gre-tunnel/1"
    xmlns:unis="http://ogf.org/schema/network/topology/unis/20100528/"
    type="request" >

    <unis:node id="node-0">
        <unis:nodePropertiesBag>
            <pgeni:nodeProperties exclusive="true"
                    component_manager_id="urn:publicid:IDN+emulab.net+authority+cm">
                <pgeni:sliver_type name="raw-pc" />
            </pgeni:nodeProperties>
        </unis:nodePropertiesBag>

        <unis:port id="node-0:if0">
            <unis:portPropertiesBag>
                <gretun:nodeProperties>
                    <gretun:interface_ip address="192.168.0.1" netmask="255.255.255.0" />
                </gretun:nodeProperties>
            </unis:portPropertiesBag>
        </unis:port>
    </unis:node>

    <unis:node id="node-1">
        <unis:nodePropertiesBag>
            <pgeni:nodeProperties exclusive="true"
                component_manager_id="urn:publicid:IDN+uky.emulab.net+authority+cm">
                <pgeni:sliver_type name="raw-pc" />
            </pgeni:nodeProperties>
        </unis:nodePropertiesBag>

        <unis:port id="node-1:if0">
            <unis:portPropertiesBag>
                <gretun:nodeProperties>
                    <gretun:interface_ip address="192.168.0.2" netmask="255.255.255.0" />
                </gretun:nodeProperties>
            </unis:portPropertiesBag>
        </unis:port>
```

[3] ORCA and NDL, along with UNIS, are all collaborating on the NML standardization effort.

```
    </unis:node>

    <unis:link id="link">
        <unis:type>gre-tunnel</unis:type>

        <unis:portIdRef>node-0:if0</unis:portIdRef>
        <unis:portIdRef>node-1:if0</unis:portIdRef>
    </unis:link>
</pgeni:rspec>
```

As shown in this example, the current approach for technology specific extensions in UNIS uses a technology specific properties element that annotates, perhaps along with other properties elements, the base topology element. All the GENI specific attributes and (complex) elements required to describe a node can be cleanly specified inside the *nodeProperties* element of the GENI XML namespace. This extension mechanism has several advantages over the initial type hierarchy extension approach of the NM-WG topology schema:

- All network topologies are described using only the base elements annotated with technology specific properties. This allows network services to understand the basic topology description without requiring the knowledge of technology specific namespaces.
- Clean validation of messages can be done against the base schema independently of technology specific extensions used.
- Avoids the proliferation of messy multiple inheritance representations and facilitates the reuse of properties defined in different namespaces (e.g. combining layer 3 and layer 2 properties).

We currently believe, as does the NML-WG, that these base elements are sufficient for representing any type of network topology given an expressive extension mechanism such as the one presented above. Given this extensible nature, UNIS is uniquely suited to adapt to new technologies and emerging networks, in particular those of large-scale, heterogeneous environments with bleeding edge technology as are being prototyped within GENI.

4.2 Architectural Components and APIs

GENI I&M currently features several diverse projects, each with strengths that will facilitate a diverse selection of measurement tools. Integrating each project within *perfSONAR* has a unique set of challenges:

- Construction of new data schemata for previously un-categorised measurement formats
- Normalizing data formats from existing metrics
- Integration of WS interfaces into previously un-accessible resources
- Integrating *perfSONAR* protocols into daily operations
- Construction of new APIs to access all information

The following sections will discuss some current GENI projects, their contributions to the measurement space, and potential integration strategies. This is not an exhaustive list, but does touch on efforts that will either benefit from *perfSONAR* integration or will require changes to the *perfSONAR* framework and protocols to support.

INSTOOLS Instrumentation Tools for a GENI Prototype (INSTOOLS) is a project to create a GENI-enabled testbed based on the existing University of Kentucky Edulab, and to implement and to deploy instrumentation capabilities that will enable GENI users to better understand the runtime behavior of their experiments. [16] Currently INSTOOLS provides data collection capability for passive measurements (e.g. SNMP). Metrics of interest are network based (e.g. traffic statistics, utilization, host statistics). There are several analogous components that map directly to *perfSONAR* components, a partial list of INSTOOLS components:

- **Measurement Controller** - Controls the measurement collection activities
- **Measurement Points** - Gathers measurements, stores the results in backend database technologies
- **Archive Server** - Allows access to measurements
- **Content Management System** - Manages components

Integration with INSTOOLS components would involve defining new metrics for data that *perfSONAR* does not currently deal with. INSTOOLS components would be required to expose archived information through Web Service interfaces, and could utilize existing *perfSONAR* protocols for communication.

OnTimeMeasure This effort provides GENI with an on-demand measurement service used in forecasting, anomaly detection, and fault-location diagnosis in GENI experiments and GENI operations. The project will deploy a prototype measurement service to support operations and early experimenters use in the first year, and will revise the service in each development spiral to improve services and integration, based on GENI community feedback. [26]. Current incarnations of OnTimeMeasure utilize existing stores of *perfSONAR* data to perform forecasting duties. There are several analogous components that map directly to *perfSONAR* components, a partial list of OnTimeMeasure components:

- **Node Beacon** - Active measurement component
- **Root Beacon** - Collection of all measurements, home of visualization components
- **Policy Authority** - Controls access to measurements
- **Publish Authority** - Provides information from the backend storage

OnTimeMeasure utilizes metrics from the *perfSONAR* framework, but is also interested in several new measurements (e.g. pathrate, pathload, pathcar). Extension to new schemas is a must to accomplish these goals. In addition to mapping new metrics, certain components of OnTimeMeasure must incorporate *perfSONAR* protocols into operational behaviors.

S3MONITOR Scalable, Extensible, and Safe Monitoring of GENI (S3MONITOR) is an effort to develop a prototype shared measurement service based on the existing S3 service, integrate it with ProtoGENI and deploy it for GENI experimenter's use. This shared measurement service will emphasize scalability and safety to best utilize network resources associated with measurements. The project will also analyze GENI

privacy and security requirements for measured data, and prototype the service to address appropriate requirements in each development spiral. [35] S3MONITOR collects or plans to collect several existing metrics that are mapped in the *perfSONAR* framework including latency and bandwidth. There are several analogous components that map to *perfSONAR* components, a partial list of S3MONITOR components:

- **Sensor Pods** - Manage data repositories, configurations
- **Sensing Manager** - Aggregates and disseminates data
- **Engines** - Inference service to increase scalability

S3MONITOR's achitecture is distributed and scalable. Integration into the *perfSONAR* framework should prove to be a powerful addition to the overall design of these components. The ability to communicate via the standard *perfSONAR* protocols as well as take advantage of planned enhancements to control the policy and security surrounding measurements via the *perfSONAR* Authentication mechanisms will empower S3MONITOR to complete goals centered on the "safety" of measuring in an active experimental framework.

The Integrated Measurement Framework and Tools for Cross Layer Experimentation will develop and integrate the GENI Integrated Measurement Framework (IMF) for optical communication substrates into the ORCA control framework prototype, and integrate the FIND SILO framework into the ORCA control framework prototype, and IMF and SILO with each other, to enable cross-layer experimentation involving the physical layer of an optical network. [15]

IMF relies in a "publication" and "subscription" model to data access, this is something that *perfSONAR* is working to integrate. Additionally, IMF will benefit greatly from the UNIS schema and service as a way to glean and contribute details about the topology of the network. There are several metrics that IMF requires that are not currently mapped in the *perfSONAR* data representation. It is expected that extension will be required to achieve access to these results.

5 Conclusion

This work has presented an detailed project, based on the *perfSONAR* framework, that addresses the Instrumentation and Measurement needs of the ProtoGENI project. In addition to describing the software components of *perfSONAR*, we have presented an open and extensible topology and measurement description language that may be implemented by other I&M themed projects to unify the measurement infrastructure across the ProtoGENI project.

6 Acknowledgements

This material is issued under NSF Cooperative Agreement CNS-0737890.

References

1. GEANT2 AutoBAHN. http://www.geant2.net/server/show/ConWebDoc.2544.
2. B. Tierney and J. Boote and E. Boyd and A. Brown and M. Grigoriev and Y. Li and J. Metzger and M. Swany and M. Zekauskas and J. Zurawski. Instantiating a Global Network Measurement Framework. Lbnl technical report, Lawrence Berkeley National Laboratory, January 2009. http://acs.lbl.gov/ tierney/papers/perfsonar-LBNL-report.pdf.
3. A. Brown, M. Swany, and J. Zurawski. A general encoding framework for representing network measurement and topology data. *Concurr. Comput. : Pract. Exper.*, 21(8):1069–1086, 2009.
4. Bandwidth Test Controller (BWCTL). http://www.internet2.edu/performance/bwctl/.
5. J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP). RFC 1157, May 1990.
6. Energy Sciences Network. http://www.es.net.
7. pan-European research and education network. http://www.geant.net.
8. GENI. http://www.geni.net/.
9. GENI Instrumentation and Measurement Working Group. http://groups.geni.net/geni/wiki/GeniInstMeas.
10. GENI Instrumentation and Measurements Architecture. http://groups.geni.net/geni/wiki/GeniInstrumentationandMeasurementsArchitecture.
11. GENI Spiral Two. http://groups.geni.net/geni/wiki/SpiralTwo.
12. M. Grigoriev, J. Boote, E. Boyd, A. Brown, J. Metzger, P. DeMar, M. Swany, B. Tierney, M. Zekauskas, and J. Zurawski. Deploying distributed network monitoring mesh for lhc tier-1 and tier-2 sites. In *17th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2009)*, March 2009.
13. 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 *Third International Conference on Service Oriented Computing - ICSOC 2005, LNCS 3826, Springer Verlag*, pages 241–254, Amsterdam, The Netherlands, December 2005.
14. InterDomain Controller Protocol. http://www.controlplane.net.
15. IMF: Integrated Measurement Framework and Tools for Cross Layer Experimentation. http://groups.geni.net/geni/wiki/IMF.
16. INSTrumentation TOOLS for a GENI Prototype (INSTOOLS). http://groups.geni.net/geni/wiki/InstrumentationTools.
17. Internet2. http://www.internet2.edu.
18. Internet2 ION. http://www.internet2.edu/ion/.
19. Iperf. http://dast.nlanr.net/Projects/Iperf/.
20. Leveraging and Abstracting Measurements with perfSONAR (LAMP). http://groups.geni.net/geni/wiki/LAMP.
21. B. Lowekamp, B. Tierney, L. Cottrell, R. Hughes-Jones, T. Kielmann, and M. Swany. Enabling Network Measurement Portability Through a Hierarchy of Characteristics. In *4th International Workshop on Grid Computing (Grid2003)*, 2003.
22. W. Matthews and L. Cottrell. The PingER Project: Active Internet Performance Monitoring for the HENP Community. IEEE Communications Magazine on Network Traffic Measurements and Experiments, May 2000.
23. Network Measurements Working Group (NM-WG). http://nmwg.internet2.edu.
24. NUTTCP. http://www.lcp.nrl.navy.mil/nuttcp/.
25. Open Gird Forum. http://www.ogf.org.

26. OnTimeMeasure: Centralized and Distributed Measurement Orchestration Software. http://groups.geni.net/geni/wiki/OnTimeMeasure.
27. ESnet On-demand Secure Circuits and Advance Reservation System (OSCARS). http://www.es.net/oscars/.
28. One-way Ping (OWAMP). http://www.internet2.edu/performance/owamp/.
29. Performance focused Service Oriented Network monitoring ARchitecture. http://www.perfsonar.net.
30. perfSONAR-PS. http://psps.perfsonar.net.
31. Ping. http://en.wikipedia.org/wiki/Ping.
32. ProtoGENI. http://www.protogeni.net/trac/protogeni.
33. Brasilian National Research and Education Network homepage. http://www.rnp.br/en/.
34. Round robin database. http://oss.oetiker.ch/rrdtool/.
35. Scalable, Extensible, and Safe Monitoring of GENI (S3MONITOR). http://groups.geni.net/geni/wiki/ScalableMonitoring.
36. NM-WG schema and prototype repository. http://stout.pc.cis.ude.edu/NWMG/.
37. Thrulay Network Capacity Tester. http://e2epi.internet2.edu/thrulay/.
38. UNIS. https://spaces.internet2.edu/display/ISWG/Home.
39. Web Services. http://www.w3.org/2002/ws/.
40. J. Zurawski, J. Boote, E. Boyd, M. Glowiak, A. Hanemann, M. Swany, and S. Trocha. Hierarchically federated registration and lookup within the perfsonar framework (short paper, poster session). In *Tenth IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, 2007.
41. J. Zurawski, M. Swany, and D. Gunter. A scalable framework for representation and exchange of network measurements. In *IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, Barcelona, Spain, March 2006.

Appendix A

Here we summarize a number of currently available *perfSONAR-PS* components.

SNMP MA The *perfSONAR-PS* SNMP based Measurement Archive (SNMP MA) is able to expose data collected via variables from the Simple Network Management Protocol (SNMP) protocol found on networked devices and stored in Round Robin Databases (RRD) archives. The SNMP MA provides a simple interface that is cable of exposing these files after basic configuration for consumption by *perfSONAR* client applications and services alike. The Web Service front end provides a uniform method of access using the *perfSONAR* XML protocols and delivers the data in an unambiguous manner, thus eliminating the mystery associated with the backend storage.

gLS/hLS The *perfSONAR-PS* Lookup Service (LS) addresses the always challenging problem of resource registration and discovery for the *perfSONAR* framework. Service instances that manage datasets are only useful when they can be contacted by consumers. Consumers can only function when there is data available. To manage these problems in a dynamic environment such as *perfSONAR*, it is necessary to register, maintain, and query for the services that may contain interesting data. The distinction between global and host Lookup Services (gLS/hLS) provides a hierarchical, scalable infrastucture for managing service queries across federated domains.

The *perfSONAR-PS* LS relies on an XML database, Oracle DB XML, to store service registration information in a native manner. Using the power of the XPath and XQuery standards it is then possible for client applications and services alike to query for information in a uniform and powerful manner. All *perfSONAR* services are capable of registering information with an LS instantiation, thus deploying the LS within an I&M framework is paramount.

Status MA The *perfSONAR-PS* Status Collector and Service allows networks to monitor network elements and make available its operational and administrative status information. The *perfSONAR-PS* Status Collector can collect status information via a number of methods. It can be configured to run a script allowing it to query devices that the service doesn't natively support, or to consult an existing database of status information. The collector can also be configured to obtain status information directly from the switches and routers.

PingER The *perfSONAR-PS* PingER service is an evolution of the PingER project [22] with more than 10 years experience in collecting and analysing network performance across the world. The *perfSONAR-PS* PingER service is composed of both a storage backend (MA) and measurement frontend (MP) to conduct and store ping measurements and to make available such data for consumption by interested parties. Network characteristics supported include availability, latency and jitter, which provide a broad spectrum of determining end-to-end network performance.

PSED The *perfSONAR* Event Daemon (PSED) is a lightweight monitoring component designed to collect a wide range of time series data and store it in a consistent, space-efficient format. PSED uses a client/server model where distributed monitoring clients communicate with a centralized (experiment-wide) collection daemon to report measurement events. With the ability to run on a number of platforms, PSED has been primarily deployed as an end-host monitoring tool where it can collect network and CPU utilization, running process load, I/O performance measurements and more.

PSED formats and stores time series data as a $< timestamp, event{-}type, value >$ 3-tuple along with identifying host information. The PSED component implements a lightweight binary protocol for exchanging this time series data leading to a very small measurement traffic footprint. For storage, a number of backends have been implemented within the collection daemon, including SQLite and PostgreSQL, and the underlying data format is easily exposed via Measurement Archives for consumption by monitoring frontends.

A PSED measurement client may be designed to collect data suitable for time series analysis from a number of resources on a deployed system. Currently available host monitoring clients for Linux-based systems report information read from the */proc* filesystem, allowing for a high degree of configurable. For monitoring device or driver-specific hardware components, PSED provides a common way to integrate end-host measurement collection within the full range of *perfSONAR* services.

perfSONAR-BUOY MA and MP The *perfSONAR-PS* perfSONAR-BUOY Measurement Archive (MA) functions as both a storage facility and a regular testing framework in conjunction with the BWCTL and OWAMP measurement tools. These archived measurements, stored in a MySQL database, are exposed through a web services interface.

The perfSONAR-BUOY MA provides a simple interface that is capable of exposing these files after basic configuration for consumption by *perfSONAR* client applications and services alike. The Web Service front end provides a uniform method of access using the *perfSONAR* XML protocols and delivers the data in an unambiguous manner, thus eliminating the mystery associated with the backend storage.

Appendix B

The major components of base schema are illustrated in Figure 2. In this figure, the major sections, data and metadata, are shown side-by-side with the subsections listed vertically within each section.

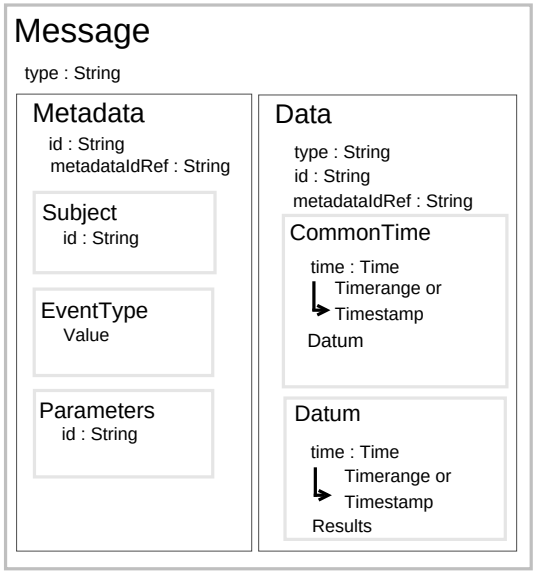

Fig. 2. NM-WG Base Schema

The schema for the top-level message envelope is shown below.[4] The message envelope may contain multiple metadata and data sections. The message "type" allows distinguishing between storage and query, for example, when the underlying communication system may not provide such information.

```
namespace nmwg =
     "http://ggf.org/ns/nmwg/2.0/"

element nmwg:message {
     attribute type { xsd:string } &
     ( Metadata | Data )+
}
```

The schema for the *metadata* element is shown first, along with several supporting elements. Every *metadata* must contain an "id" attribute and may contain an optional "metadataIdRef" attribute; this can refer to another *metadata* instance. Through this simple construct it is possible for *metadata* to be linked or "chained", further reducing the storage and exchange overhead.

[4] In all the schemas presented inline, some small details have been left out or modified to enhance readability. Full schemas are available at [36]

The *metadata* itself, when properly constructed, should be akin to a verbal description of a specific operation. In any well-formed sentence there will be a noun (e.g. the "subject"), a verb (e.g. "eventType") and potentially some modifiers to describe the subject or verb (e.g. "parameters"). A description of each element in the *metadata* section follows:

- Subject – The physical or logical entity being described. In most cases this corresponds directly to a topological element or group of elements, the structure of which will be explored in Section 3.7. Examples of a subject could be the interface of a network capable device, or two ends of a point to point measurement.
- EventType – The canonical name of the aspect of the subject being measured, or the actual event being sought. These take the form of hierarchical type based on URI instances such as "http://ogf.org/ns/nmwg/characteristics/latency/2.0/".
- Parameters – The way in which the description is being gathered or performed. The command line arguments of some tool are normally candidates for this role, although other informational items such as "units" that may be needed to describe any stored data can be stored in this way as well.
- Key – This can be substituted in place of the previous three items and should be used by implementations to save time on recovery of specific information. The key is a very malleable, and does not have a very specific structure leaving the implementations the ability to define it as they wish.

```
namespace nmwg = "http://ggf.org/ns/nmwg/2.0/"

Metadata =
  element nmwg:metadata {
    attribute id { xsd:string } &
    attribute metadataIdRef { xsd:string }? &
    Subject &
    EventType? &
    Parameters? &
    Key?
  }

Subject =
  element nmwg:subject {
    attribute id { xsd:string }
  }

EventType =
  element nmwg:eventType {
    text?
  }

Parameters =
  element nmwg:parameters {
    attribute id { xsd:string }
  }

Key =
  element nmwg:key {
    attribute id { xsd:string } &
    Parameters
  }
```

An example of a legitimate measurement, taking into account these three constructs would be: "*Host 140.232.101.101* performed a *TCP bandwidth measurement* to *Host*

131.243.2.17 for *10 seconds* with a *window size of 32 Kb*". Decoding our example leaves us with:

- Subject – Host 140.232.101.101, Host 131.243.2.17
- EventType – bandwidth
- Parameters – Length of 10 seconds, Window Size of 32 Kb

In the following example, we illustrate all of the aforementioned components of a *Metadata* element. As a matter of style we omit every possible combination of attributes and elements, as well as extraneous namespace declarations.

```
<nmwg:metadata id="1" xmlns:nmwg="http://ggf.org/ns/nmwg/2.0/">
  <nmwg:subject id="sub1">
   <nmwgt:endPointPair>
    <nmwg:src address="140.232.101.101" />
    <nmwg:dst address="132.243.2.17" />
   </nmwgt:endPointPair>
  </nmwg:subject>
  <nmwg:eventType>http://ogf.org/ns/nmwg/characteristics/bandwidth/tcp/2.0/</nmwg:eventType>
  <nmwg:parameters>
   <nmwg:parameter name="windowSize">32768</nmwg:parameter>
   <nmwg:parameter name="duration">10</nmwg:parameter>
  </nmwg:parameters>
</nmwg:metadata>
```

The schema for the *data* element is shown second, along with several supporting elements.

```
namespace nmwg = "http://ggf.org/ns/nmwg/2.0/"

Data =
  element nmwg:data {
    element id { xsd:string } &
    element metadataIdRef { xsd:string } &
    (
      CommonTime? &
      Datum*
    )
  }
CommonTime =
  element nmwg:commonTime {
    Time &
    Datum*
  }
Datum =
   Time
}
```

Every *data* element must contain an "id" and "metadataIdRef" attributes; these identifiers are used to track relationships to some specific *metadata*. The entire purpose of the *data* element is to serve as a container for measurements and time related to a specific *metadata*. There are three major parts of the *data* element:

- CommonTime – Can be used to factor out commonly seen time elements and save time in both encoding, decoding, and transmission.
- Datum – The actual result of measurement. Can contain time (e.g. a Time element or attribute) or may be enclosed by a CommonTime element.
- Time – Representation of a time stamp, or time range in a specified format.

Time is fundamental to network measurements, and is the only required part of each datum. The 'CommonTime' section allows the common case of factoring out a set of data that is associated with a single time range or timestamp. Note that by extending the EventType of the name into the namespace, effectively creating a unique name for each type of event, the timestamp may be all that is necessary.

Time-related elements reside in a sub-namespace from the base. This separation makes the time definition more portable, for re-use in extension namespaces. It also adds flexibility, allowing the time representation to change independently of the base namespace. The schema for the time namespace is shown below.

```
namespace nmtm =
  "http://ggf.org/ns/nmwg/time/2.0/"

Time =
  element nmtm:time {
    attribute type { xsd:string } &
    (
      TimeStamp |
      (
        StartTime &
        (
          EndTime |
          attribute duration { xsd:string }
        )
      )
    )
  }

StartTime =
  element nmtm:start {
    attribute type { token } &
    attribute inclusive { token }? &
    TimeStamp
  }

EndTime =
  element nmtm:end {
    attribute type { token } &
    attribute inclusive { token }? &
    TimeStamp
  }

TimeStamp =
  attribute value { xsd:string } |
  element nmtm:value { xsd:string }
```

We close with a proper example of *metadata* and *data* elements, using several of the above constructs.

```
<nmwg:metadata id="1" xmlns:nmwg="http://ggf.org/ns/nmwg/2.0/">
  <nmwg:subject id="2">
    <nmwgt:endPointPair>
    <nmwg:src address="140.232.101.101" />
    <nmwg:dst address="132.243.2.17" />
   </nmwgt:endPointPair>
  </nmwg:subject>
  <nmwg:eventType>http://ogf.org/ns/nmwg/characteristics/bandwidth/tcp/2.0/</nmwg:eventType>
</nmwg:metadata>
<nmwg:data id="d1" metadataIdRef="1">
  <nmwg:datum value="34343" time="123213213" type="unix" />
  <nmwg:datum value="35678">
    <nmtm:time xmlns:nmtm="http://ggf.org/ns/nmwg/time/2.0/" type="unix" value="123213214" />
```

```
  </nmwg:datum>
  <!-- more -->
</nmwg:data>
```

Appendix C

The core idea behind any schema is to define the base ontology that will be used in describing elements within the framework. This topology schema is meant to describe networks consisting mainly of devices connected through links. This structure lends itself to the use of graph concepts; namely by using nodes and links as base elements.

When placed in the context of a network graph, it is common to see each *node* directly connected to a *link*. Network topology adds an additional level of indirection by placing an *interface* between some network device (commonly a *node*) and *links*. Previous discussion had considered the notion of allowing this concept (now known as a *port*) to simply be a *node* within a *node*; while this would satisfy the ontological needs of the network, the semantic meaning would be more difficult to grasp.

Devices and links are not commonly thought of as existing solely in relative locations as they appear in a graph context; they are commonly thought of as being part of a domain. The concept of this administrative entity is required in all topological descriptions; it enforces that nodes can only exist in a single place at a point in time. This may be a limiting factor, as often devices and links exist in multiple networks simultaneously. A single node might be part of a "real" network along with a cross-domain overlay network or a virtual private network (VPN).

By expanding the notion of domains to include the many possible networks that may be constructed, it becomes harder to structure the overall global topology. To alleviate this, an element "network" has been introduced into the schema, and can be used to create arbitrary groupings of network elements into logical networks. The domain structure is retained as the higher order grouping structure for all nodes, enabling the creation of an unambiguous global view of the overall network while permitting flexible logical groupings to be created.

.1 Base Elements

The base set of node, port, link, domain and network can be used to describe network topologies. Compositions and specializations of these basic components can be used to represent a wide variety of situations. For instance, many networks have been provisioning Virtual Circuits [18]. At one level of abstraction, a circuit is a direct, point-to-point connection. At another, a circuit can be thought of as simply a path through a topology. Therefore a general "path" element could be used to describe circuits. This is general enough to be extended to any topological concept that can be described as an ordered grouping of network elements.

While not specifically being a topological entity, the concept of a software service plays a significant role in any network. These services can provide information, collect measurements, run applications, or even perform low-level tasks such as adapting a network flow between different network technologies or characteristics. Obtaining access to services that have certain network properties is a primary reason for obtaining network topology.

The base elements therefore consist of nodes, ports, links, domains, networks, paths and services. With this set of base elements, it is possible to describe a wide-variety of

topological concepts in a way that could be reasonably understood by users and easily mappable to measured topological elements.

.2 Technology Specific Elements

After defining the set of elements that the topology will contain, it is necessary to define the set of properties that each will require to effectively describe network elements. Including all properties of any given technology in the basic elements described above would limit future extensibility. A namespace-based hierarchy was defined that allows for the addition of new types while permitting existing items. This hierarchy also ensures that the technology-specific properties of an element are defined separately, to prevent collision with other existing definitions. To manage the hierarchy of namespaces, both existing and newly created, the hierarchical structure of the URI allows for a method of encoding inheritance that is friendly for client applications. If a new namespace was being created for TCP elements, and it inherited from the base "http://ogf.org/schema/network/topology/base/20070828/", the new namespace's URI might be defined as "http://ogf.org/schema/network/topology/base/20070828/tcp/20071029/".

.3 Identifiers

There is a key difference between the identifier attributes in the topology space and the identifiers used in the *metadata*/*data* in the measurement schema space. Because data exchange was a key aspect when designing the format of the measurement data, rules regarding scope are centered around the concept of *request* and *response* pairs. Simply stated, the identifying attributes are only valid in a series of related *request* and *response* messages between software implementing the protocols. While this remains a sensible and valid construct for measurement, topology elements must exist outside of this *request* and *response* paradigm.

If mandated to be globally unique, topology identifiers can be used as a general and technologically-independent way to uniquely identify specific network elements. This allows for network interfaces to be described by simply specifying a given identifier, independent of whether that interface was a Layer 2 Ethernet address, a Layer 3 IPv4 or IPv6 address or even a Layer 4 listening socket. The construction of these identifiers must be done to ensure they are globally unique while still retaining reasonable readability for administrators.

Construction of identifiers can be problematic; choosing the appropriate source to use as a base in the construction is an important consideration. A natural choice is to use network addresses as a starting point, as this is a common feature to network accessible devices. While descriptive, problems arise with regards to address formats (e.g. physical addresses used in **Ethernet** vs. **IP** addresses used in network layer communication), as well as accessibility (e.g. non-routable addresses are still unique, albeit unknown globally).

Unable to determine a sound naming structure using existing information, the next option is to impose no requirements on identifying names. This would allow each domain to create their own identifiers, the only requirement being global uniqueness. Although this approach offers simplicity (i.e. creation of randomized identification can be

done independently and quickly), coordination of names becomes troublesome and the chances of collision increase. Attempts to correct this can be addressed using schemes such as those based on UUIDs. Even though unique, the subsequent identification strings are not easily tied to the reality of a given topology; the lack of human readable names will make this approach unappealing to the user base we are targeting.

.4 Identification Scheme

The network topology identification scheme has been designed to be globally-unique, human-readable and extensible; the construction is in the style of Uniform Resource Names (URNs). The namespace for the URN is "ogf" and the subnamespace is "network". All identifiers must begin with "urn:ogf:network:" in this format. The remainder of the identifier consists of a series of name and value pairs. These fields provide the hierarchy and the flexibility offered by this schema.

There are five major fields defined for network entities:

- domain
- node
- port
- link
- service

The *domain* field describes the administrative domain in which the *network* element is located. If this field is included, it will always be positioned first. The value is the globally unique **DNS** name for the domain. While other options exist for this value (e.g. **AS** number), the simplicity and availability of the former makes this more desirable.

The *node* field may be a host, router, or even a larger abstraction such as a site. The position of this element will always be second with respect to the *domain*. There are no specific rules governing value of this element, allowing maximum flexibility when encoding.

The *port* field describes the interface between a *node* and a *link*, and commonly describes **Ethernet** interfaces, **IP** interfaces, or listening **TCP** sockets. The overall structure of the identifier will be dictated based on whether the interface is physical, logically inside a single device or logically constructed across multiple devices. If the interface is physical or logically inside a single device, the field will appear after the *node*. If the interface is a logical interface for a domain, the field will appear after the *domain*. A prudent choice for the value is the physical interface from a networked device, in the case of an interface in a physical device, or a logical name, for logical interfaces in a domain.

The *link* field can describe logical or physical *link*s, regardless of direction. Bidirectional *link*s (both physical and logical) must appear directly after the *domain* designation since they will logically connect multiple *node*s. Unidirectional *link*s, regardless of direction, will appear after the *port* that is able to transmit using that link. As in previous descriptions, there is no constraint placed on the value, but care should be taken to select a name with meaning to the overall infrastructure.

The *path* field can be used for circuits or other named paths. Named paths can occur globally or within a domain; the field can appear either first or immediately after the *domain* field. The value can be anything, but should be globally unique.

The *service* field is used in identifiers for software services offered by network elements. These can be used to describe high-level services like Web Services, or low-level services like **ICMP** responders or optical converters. This field will appear after the field for the element providing the service. For example, a service offered by a domain would appear immediately after the *domain* field. The value can be anything, but should be a human-readable description of the service provided.

Example Identifiers	
Domain	urn:ogf:network:domain=internet2.edu
Host	urn:ogf:network:domain=internet2.edu:node=packrat
Bidirectional Link	urn:ogf:network:domain=internet2.edu:link=WASH_to_ATLA
Interface	urn:ogf:network:domain=internet2.edu:node=packrat:port=eth0
Logical Interface	urn:ogf:network:domain=internet2.edu:port=InterfaceToGeant
Service	urn:ogf:network:domain=internet2.edu:node=packrat:service=OpticalConverter
Unidirectional Link	urn:ogf:network:domain=internet2.edu:node=packrat:port=eth0:link=WASH_to_ATLA
Circuit	urn:ogf:network:path=IN2P3_Circuit

Fig. 3. Examples of valid identifiers

```
<nmtb:domain id="urn:ogf:network:domain=domain1.edu">
 <nmtb:node id="urn:ogf:network:domain=domain1.edu:node=router1">
  <nmtb:address type="hostname">router.domain1.edu</nmtb:address>
  <nmtb:name type="logical">domain1's router</nmtb:name>

  <nmtl2:port id="urn:ogf:network:domain=domain1.edu:node=router1:port=eth0">
   <nmtb:name type="logical">eth0</nmtb:name>
   <nmtl2:address type="mac">00:16:a4:bb:3a:38</nmtl2:address>
   <nmtl2:capacity>1000M</nmtl2:capacity>
   <nmtl2:mtu>1500</nmtl2:mtu>
   <nmtl2:link id="urn:ogf:network:domain=domain1.edu:node=router1:port=eth0:link=1">
    <nmtl2:remoteLinkId>urn:ogf:network:domain=domain2.edu:node=host1:port=eth0:link=1</nmtl2:remoteLinkId>
   </nmtl2:link>
  </nmtl2:port>

  <nmtl3:port id="urn:ogf:network:domain=domain1.edu:node=router1:port=192.168.1.1">
   <nmtl3:address type="ipv4">192.168.2.1</nmtl3:address>
   <nmtl3:relation type="over">
    <nmtb:portIdRef>urn:ogf:network:domain=domain1.edu:node=router1:port=eth0</nmtb:portIdRef>
   </nmtl3:relation>
  </nmtl3:port>
 </nmtb:node>
</nmtb:domain>
```

Fig. 4. Layer 2 and Layer 3 interface descriptions

.5 Element Relations

Network elements do not exist without context. Layer 3 links pass over Layer 2 infrastructure, Layer 4 ports exist inside of a specific devices. In order to correlate data at different layers, the schema must be able to accurately capture these and other relationships.

The notion of containment is the first relationship explored. Almost all defined entities will contain smaller units and, with the exception of a *domain*, multi-domain *link*s, multi-domain *path*s and multi-domain *network*s, most elements will be contained within something else. A physical interface, for example, can only exist inside a single network device and can be thought of as being "contained" inside that device. This relationship can be captured by placing the definition for the port inside the node definition. While not as common, the schema provides for element's definitions outside of their containing element. This relationship is retained through the use of the hierarchical identifier scheme described in Section .3.

To model new relationships, the schema includes an extensible property, *relation*. These relationship properties are typed to specify what relationship is being modeled, and contains references to the set of elements related in the specified way to the network element. An example of relation types are the *source* and *sink* relations in the context of ports connected by a unidirectional link.

.6 Example

In Figure 4, the aforementioned pieces are combined to describe a network element. This structure creates a *domain* with a single device, which contains two interfaces (Layer 2 and Layer 3). The Layer 2 interface is connected to the host "host" in domain "domain2.edu". The Layer 3 interface is used to describe the **IP** address. Since the Layer 3 interface runs over Layer 2, this linking is described by the relation property in the Layer 3 port structure.