CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2009; **21**:1069–1086
Published online 11 February 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1412

A general encoding framework for representing network measurement and topology data

A. Brown[1,*,†], M. Swany[1] and J. Zurawski[2]

[1] *Department of Computer and Information Sciences, 103 Smith Hall, Newark, DE 19716, U.S.A.*
[2] *Internet2, 1150 18th Street NW, Suite 1020, Washington, DC 20036, U.S.A.*

SUMMARY

Scientific applications are evolving rapidly and rely heavily on the network for data movement, communication, control and result collection. This reliance on the network has increased the breadth of consumers of network status information while the increasing complexity of networks has expanded the diversity of the producers of this information. To enable this wide variety of producers and consumers to effectively interoperate, relevant standards are paramount.

In this paper, we present a general model used to represent both network measurements collected from performance tools as well as describing the physical and logical characteristics of the underlying network. This system is currently being standardized in the Open Grid Forum to enable other uses within the wider grid and distributed computing community (Open Grid Forum, http://www.ogf.org/). Copyright © 2009 John Wiley & Sons, Ltd.

Received 22 April 2008; Revised 6 August 2008; Accepted 1 November 2008

KEY WORDS: networks; monitoring; measurement; topology; XML; schema; web services

1. INTRODUCTION

The advent of high-speed data networks throughout the world such as *ESnet*, *GEANT2* and *Internet2* have created a surge in the deployment of scientific software dependent on this medium to function effectively and efficiently [1–3]. International efforts such as the Large Hadron Collider (LHC) project from the European Organization for Nuclear Research (CERN) rely heavily on the underlying

[*] Correspondence to: A. Brown, Department of Computer and Information Sciences, 103 Smith Hall, Newark, DE 19716, U.S.A.

[†] E-mail: aaron@internet2.edu, brown@cis.udel.edu

network to facilitate research between multiple parties [4,5]. This emphasis on global collaboration makes paramount the proper behavior and health of the network.

Many different tools exist to gather the various characteristics of a network, such as *bandwidth*, *delay* and *loss* [6]. This diversity in tools and characteristics means that the process of collecting network measurements may vary substantially between domains. To facilitate interoperability, the results of these measurements need to be described in a fashion that is independent of the tool that collected them. This uniform representation eases the storage and consumption of the data at the possible cost of losing some of the data measured during the conversion. To minimize the number of conversions, this format should be utilized from the time of collection, and should allow for storage and exchange that limits the amount of conversion required between tools.

Monitoring frameworks, such as *perfSONAR*, the performance-oriented Service-Oriented Network monitoring ARchitecture, expose a wide variety of network data and make it available through a standardized interface [7]. With extensibility (i.e. new measurement types should not cause a re-design of the original format) and flexibility (i.e. all options can be easily enumerated) being the driving factors, the formats presented in this work served as the basis for the construction of this framework.

The consumers of the data will be even more varied than the producers. These could extend from simple visualization tools to complex multi-domain alert and reporting systems to applications that tune their workflows for current network conditions. The needs of these consumers will likely evolve over time as new applications come online or changes in networks enable the consumption of new network characteristics. Standardization of the information representation for these network characteristics enables independent groups to construct new consumer applications that can interoperate with the wide range of producers of network status data.

To effectively use this network status information, significant knowledge of the underlying network topology is required. In the course of diagnosing end-to-end performance issues, knowledge of the communication ‘path’ can be of significant help. The bulk of this diagnostic work is performed by network administrators with intimate knowledge of the network. This approach to debugging becomes problematic in transfers that span multiple network domains. Coordination across these logical boundaries becomes problematic, especially when physical barriers related to geographical position and political affiliation are factored in.

Tools such as **traceroute** can provide to a remote user a *network layer* view of the domains over which their data are flowing. This information is sufficient to provide rudimentary diagnosis of many performance issues. When debugging problems on an end-to-end circuit **traceroute** may not provide the required information. Many circuit provisioning tools tunnel the traffic through a series of Layer 2[‡] connections, creating the illusion of a direct connection between disparate routers [8,9]. This false view will prevent users from effectively ‘drilling down’ to find which of the hidden Layer2 connections might be causing the problem. The topological schema presented in this work provides simple constructs, capable of describing the complex underpinnings of communication networks.

The remainder of this work will proceed as follows. Initially, we will discuss the schema used to describe network measurements as well as some methods used to enhance and extend this basic

[‡] Referring to Layer 2 of the ISO’s Open Systems Interconnection model.

work. Next, we will discuss our proposed network topology representation. Finally, we will touch on how to use these two approaches in concert to obtain relevant performance information.

2. BASE SCHEMA

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. This section will describe some background on the matter before moving into an extensive explanation of the actual data model.

2.1. Motivation

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 [10] has made it clear that defining an interface and storage format is difficult, and there are many different environmental issues to consider. Any solution that 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.

2.2. Schema preliminaries

The requirements take care not to specify a specific technology or methodology with which to address the goals of this data format. The working group came to an early conclusion that the use of XML [11] and the related schema tools and languages provided the best support for a solution. XML provides the capability to produce self-describing documents. The advantages of this format,

DOI: 10.1002/cpe

such as encoding a clear description and purpose into each instance and the aforementioned need for interoperability, far outweigh the disadvantages, most notably efficiency.

2.3. Schema

We will examine each major part of the 'Base Schema' in turn, focusing on the largest scope that would be common to any measurement or topological element. As such, we will present the construction of both *metadata* and *data* elements, followed by the common representation of time used to represent all measured information. Although an integral part of the framework, we save descriptions of network topological elements until Section 5.

2.4. Metadata

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 the *metadata* to be linked or 'chained', further reducing the storage and exchange overhead; this concept will be explored in Section 3.

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 5. 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 very malleable, and does not have a very specific structure leaving the implementations the ability to define it as they wish.

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 s* 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 s, Window Size of 32 kb.

DOI: 10.1002/cpe

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>
```

2.5. Data

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

We omit a proper XML instance at this time until we fully explain the concept of time, as the above elements rely upon it heavily. The obvious question at this stage is ‘why is time treated in a special manner?’, and not included in the proper ‘Base’ documents. The members of NM-WG originally decided that time needs to be expressed in a variety of different ways for various use-cases. For this reason, time is defined separately to make it extensible. This approach may lead to an influx of separately defined time instantiations by various parties; the NM-WG is confident that extraneous and unnecessary instances will be filtered out by the major adopters.

2.6. Time

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 information that is associated with a single time range or timestamp.

The time-related elements reside in a separate namespace from the base and are able to make the definition of time more portable. This separation allows for more natural re-use and extension to current and future uses as well as added flexibility of allowing the time representation to change independently of the base namespace.

DOI: 10.1002/cpe

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>
```

3. PRINCIPLES OF CHAINING

Metadata elements can refer to each other for both expressiveness and efficiency. We refer to this relationship as ‘chaining’. There are two ways *metadata* elements can be chained to fully describe associated *data*. The first form allows for portions of a complete *metadata* to be specified incrementally so that components (e.g. *subject*) can be re-used. We refer to this as ‘Merge’ chaining; components are logically merged to fully describe *data*. The second form is used to fully describe the ‘provenance’ of *data* as it involves operations applied to the underlying *data* set. We refer to this as ‘Operation’ chaining.

3.1. Merge chaining

Merge chaining logically merges partial *metadata* to form a complete structure. Recall from Section 2.4 that *metadata* elements can be linked together via the *metadataIdRef* attribute. This allows for more efficient representations as duplicate information may be passed ‘by reference’ rather than ‘by value’. Conceptually, these information blocks form a chain that forms a complete *metadata* block. While the chain is linked from most-specific to least-specific, reference-wise, we can refer to the chain as having a ‘root’ end and a ‘leaf’ end (making reference to the fact that these reference chains are, in fact, often part of a tree of information). Conceptually, information closer to the leaf end overrides information specified in the root end of the chain, as the leaf is more specific.

Figure 1 shows an example of how merge chaining might be used. In this example the goal is to get utilization, errors and dropped packets for a specific Layer 3 interface. This could be done in one of two ways. The first attempt eliminates chaining and includes the *subject* in each of the *eventType*’s *metadata* elements. A drawback to this is seen when the *subject* contains a large amount of information; the resulting query would consume a significant amount of space with redundant information.

Figure 1 shows the other approach where this redundant information is factored out. In this example, a *metadata* is created that contains only the subject of interest. Three *metadata* instances are also created, absent of a *subject*, but do specify the *data* sets to be retrieved. When these are merged with the subject *metadata*, the resulting element will contain the *subject* along with the

```
<!--specify the port of interest-->
<nmwg:metadata
  <nmwg:subject>
    <nmtl3:port>
      <nmtl3:addresstype="IPv4">140.232.101.101</nmtl3:address>
    </nmtl3:port>
  </nmwg:subject>
</nmwg:metadata>

<!--Use merge chaining to select the measurements of interest for the specific port-->
<nmwg:metadata id="meta2" metadataIdRef="meta1">
  <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:eventType>
</nmwg:metadata>

<nmwg:metadata id="meta3" metadataIdRef="meta1">
  <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/errors/2.0</nmwg:eventType>
</nmwg:metadata>

<nmwg:metadata id="meta 4" metadataIdRef="meta1">
  <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/drops/2.0</nmwg:eventType>
</nmwg:metadata>

<nmwg:data metadataIdRef="meta2"/>
<nmwg:data metadataIdRef="meta3"/>
<nmwg:data metadataIdRef="meta4"/>
```

Figure 1. ‘Merge Chaining’ to select different *data* sets from a single *subject*.

eventType specifying the *data* set. This operation preserves the relationship as if they had been specified as one unit.

3.1.1. Merging metadata

When merging two *metadata* blocks, each of the top-level elements in a *metadata*, *subject*, *eventType* and *parameters*, must be merged with their corresponding elements in the other *metadata*. Owing to the differences in semantics between the top-level elements, each of the elements has slightly different sets of rules.

The first example involves *eventType*s. *EventType*s define the set of data that a specific *metadata* wishes to receive about a given *subject*. If a *metadata* were allowed to contain two *eventType*s which referred to different data sets, the resulting data would be very difficult to differentiate. There can be only one context under which *metadata* with differing event types can be merged: if the two *eventType*s are simply different names for the same type of data.

The next example involving merging is *parameters* merging. There may exist multiple *parameters’* elements, each possessing differing namespaces. If multiple *parameters* blocks are encountered, only those *parameters* with the same namespaces **should** be merged. The resulting *metadata* must contain, at minimum, one *parameters’* block for each namespace encountered.

In the common case, merging two *parameters* blocks can be done in a data agnostic fashion. The *parameter* elements consist of name and value pairs. Most will have a ‘value’ portion consisting of a simple string. The logical operation of merging two *parameter* elements with the same name involves replacing the existing value with the new value.

DOI: 10.1002/cpe

The final two areas of merging may require domain and situational knowledge: the merging of *subject*s and the merging of *parameter* elements containing structured data. In Section 4.5, we will discuss with greater specificity how these types of elements can be merged.

3.2. Operator chaining

Often it is desirable for data to be summarized and analyzed before being presented to an end-user. ‘Operator Chaining’ involves the application of an operator to a set of data described by some *metadata*. The most common case for operator chaining is a selection function, e.g. to define the time range similar to an **RDBMS** query. In this context the first *metadata* is used to select a broad range of data, and subsequent *metadata* that are used to reduce the set to requested range.

Operator chains can be used to modify the data before returning it to the client. An operator *metadata* might be defined to request the jitter for a given set of *data*. Once retrieved from the storage and manipulated by any previous operations, the calculation of the jitter may be performed and returned, eliminating the costly and unnecessary return of an entire *data* set.

Figure 2 shows an example of how operator chaining might be used. The utilization for a given Layer 3 address is obtained. The time range for this set of *data* is then reduced to only

```
<!-- get the utilization information about the specific port-->
<nmwg:metadata id="meta1">
  <nmwg:subject>
    <nmtl3:port>
      <nmtl3:address type="IPv4">140.232.101.101</nmtl3:address>
    </nmtl3:port>
  </nmwg:subject>
  <nmwg:eventType>utilization</nmwg:eventType>
</nmwg:metadata>

<!-- select from the time range of interest data described above -->
<nmwg:metadata id="meta2">
  <select:parameters id="param1c"metadataIdRef="meta1">
    <nmwg:parameter name="startTime">1193876292</nmwg:parameter>
    <nmwg:parameter name="endTime">1194045045</nmwg:parameter>
  </select:parameters>
  <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0/</nmwg:eventType>
</nmwg:metadata>

<!-- get the average of the data in that time range -->
<nmwg:metadata id="meta3">
  <select:parameters id="param1d"metadataIdRef="meta2">
    <nmwg:parameter name="consolidationFunction">AVERAGE</nmwg:parameter>
  </select:parameters>
  <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0/</nmwg:eventType>
</Nmwg:metadata>

<!-- request to only receive the data in the specific time range-->
<nmwg:data metadataIdRef="meta3"/>
```

Figure 2. ‘Operator Chaining’ to select a range of data and perform an operation.

include data in the specified range. An aggregation function is performed on the resulting *data* set. The returned datum will consist only of the average utilization over the time period of interest.

4. EXTENDING THE BASE SCHEMA

With a wide variety of interesting subjects to measure along with evolving computing environments and techniques, we are careful to include extension mechanisms when designing these formats. The use of XML *namespaces* is the standard approach when defining new XML schemas [12]. We reuse this functionality to allow independent extensions of this schema to co-exist without central coordination or 'vetting'. The basic approach replaces the elements defined in the 'Base Schema' with newly defined instances of the same name in **different** namespaces. This construct allows for the addition of new elements or perhaps differing semantics that may be extended to current elements.

4.1. Datum extension

A common need for extension is to provide access to a new type of data. The basic *datum* element does not offer specific attributes or elements currently; complex software that may be used to monitor disparate items such as temperature of a CPU or operational status of a network link would benefit from more specific definitions. Schema extension makes adding these additional requirements possible (either in the form of attributes or elements) and allows this new format to coexist with current offerings free of data collision.

4.2. Subject extension

The concept of a subject 'requires' the notion of extension to be useful. The *subject* element, as presented in the base, lacks many of the features needed to tie it to legitimate measurements. Each measurement type will no doubt require a specific topology element to be used as the subject; eliminating from contention others that are not required. Examples of this include end-to-end measurements between a pair of end hosts or measurements about a network interface collected using **SNMP** [13].

4.3. Eventtype extension

The *eventType*s of a schema are linked to the remaining support elements in a special way: each must contain a specific type of *subject* and *datum* as defined by the schema extension. This ensures that related *eventType*s may easily share a construction. Consider **SNMP** measurements that deliver data about an interface. This specific type of data can also be described as a characteristic, such as *utilization*; it may also be that this characteristic can be measured by other tools. To ensure that the data remains usable, despite *eventType* differences, a common format is required.

EventType descriptions are constructed hierarchically to help ease the burden of schema construction. At the upper levels characteristics such as bandwidth and latency reside. Individual tools can then 'inherit' the properties of these *eventType*s in order to implement extensions. This allows specific instantiations of the utilization characteristic, such as those collected using **TL1** [14] or those collected using **SNMP**, to start with the same underlying definitions.

4.4. Parameter extension

The base namespace imposes very few rules sets on *parameters*; extension is possible through the use of a new namespaces. The overall structure of the *parameter* sets and the individual *parameter*s will look the same. *Parameter* sets will contain a collection of *parameter*s, and each individual *parameter* will contain a name field for differentiation. Unlike the base namespace, the author of the new namespace has wide latitude in deciding which types of *parameter*s are valid in that namespace and what structure the values for each *parameter* can take.

4.5. Merging extended subjects and parameters

The base namespace defines the following algorithm for merging *subject*s and complex *parameter*s.

1. If a given XML element exists only in the parent or child, copy that element to the resulting *metadata*.
2. Otherwise, if the element only has text content, copy the element from the child to the resulting *metadata*.
3. Otherwise, enter into the XML structure, and apply these rules on each child element.

The goal of the algorithm is to 'overlay' information that comes from the children directly into the parent; there may be *subject*s or *parameter*s where this algorithm does not provide the necessary merging semantics. The semantic difference exists when a namespace defines two XML elements as being the same. The base namespace defines two XML elements as the same if both namespaces and tag names match.

The merge semantics become more complicated when two XML elements can have the same tag and namespace, but different meaning. The most obvious example would be a *parameter*. In this schema, *parameter*s all share the same tag name and many share the same namespace. The differentiating factor in these cases is the attribute 'name'. The rule for testing *parameter* equality requires that not just the namespace and tag name for the two XML elements match, but that all their attributes match.

To help clarify the merge process, namespaces must also define the rules for merging. These rules can be as simple as to check if a certain attribute is equal in both XML elements before defining equality, or they may consist of complex dependencies between the XML elements. In most cases, the namespace can simply reuse the default set of rules.

4.6. Namespace versioning

The namespace construct can be used to represent different versions of the same tool or different schema versions. By including a version number when defining the namespace, such as

DOI: 10.1002/cpe

`http://ggf.org/ns/nmwg/event/status/base/2.0/`, this extension is possible. This allows ease of transition between extension namespaces in the face of changing tools and measurements.

5. NM-WG TOPOLOGICAL REPRESENTATION

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.

5.1. Base elements

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.

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.

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

DOI: 10.1002/cpe

of situations. For instance, many networks have been provisioning virtual circuits. 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.

5.2. 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.

The technology-specific namespaces are defined in an object-oriented fashion to extend the base topology, allowing for the definition of elements with technology-specific aspects. Properties added by a specific namespace will be inherited by any extensions. Each namespace defines the properties that are common to anything in the technology. This consistency ensures that the common properties will all be described in the same manner.

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. When a new namespace is allocated, the author takes the URI that the new namespace inherits from and adds on a new name and version number. 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/'. This construction is easy to use for applications and involves simple checks against known namespace constructions.

An example of this kind of namespace extension might be the definition of a namespace for Layer 4 network elements. A Layer 4 namespace might extend the port, link, network and path elements. A port could then be thought of as a listening socket on a given host, and a link might be a reliable or unreliable connection between two sockets. A network could then describe a logical overlay network, and the path would be used for a series of transport level connections between a source and destination.

DOI: 10.1002/cpe

All Layer 4 elements have an associated port. Creating a property ‘port’ or ‘portNum’ would allow this information to be described. Each of these elements has an associated transport protocol; a reasonable addition to the namespace would be an element or attribute that described the relevant transport protocol. Beyond simply adding new properties, the new namespace can modify existing properties, so long as the modifications were not incompatible with the definitions in the parent namespace.

The address for a Layer 4 interface needs more information, such as the protocol and port number. The namespace might define a new type of address ‘ip:port’ or ‘protocol://ip:port’ which encodes the required information for contacting this specific element. Since the addresses are typed, if a client only knew about the base elements, it could simply look at the type of the address in the Layer 4 port and conclude that it did not understand the new address type. The client could then either throw an error, or, simply treat the new address as opaque.

5.3. 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 that 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 of 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 [15]. 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.

DOI: 10.1002/cpe

6. IDENTIFIER 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) [16]. 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 make 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 such as Web Services or low-level services such as **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.

DOI: 10.1002/cpe

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

6.1. Element relations

Network elements do not exist without context. Layer 3 links pass over Layer 2 infrastructure, Layer 4 ports exist inside of 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 5.3.

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. Although many such relationships can be defined, the schema includes two definitions to describe two common relationships between elements.

Oftentimes, network elements exist 'over' lower layer network elements. The common case is when a higher layer element makes use of a lower layer element. For example, a Layer 3 interface uses a Layer 2 interface underneath. This 'over' relationship allows for the separation of the Layer 3 and the Layer 2 port definitions while still maintaining the relationship.

Users and administrators tend to think of networks as consisting of logical devices. These logical devices must be linked in some way to physical devices that comprise them. The most prominent example would be a circuit that provides a single logical Layer 2 link made up of a series of Layer 2 links. To properly capture this type of relationship, the schema defines a 'comprised-of' relation. The contents of the element consist of all the network elements that make up the logical device. In the case of a circuit, this might consist solely of Layer 2 links.

DOI: 10.1002/cpe

```
<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>
```

Figure 3. Layer 2 and Layer 3 interface descriptions.

6.2. Example element

In Figure 3, 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.

7. CORRELATION

These two schemas can be effectively used in concert to aid in the debugging of problems whose cause spans multiple network elements, network layers or even domains. Through the use of monitoring tools such as *perfSONAR*, domains can expose both performance and topological information about their networks. Client applications with knowledge of this framework, and permission to access the data, can take full advantage of this data.

Initially, a client would obtain the topological descriptions from each domain of interest. This information would be combined with information it already has, like a network path or a pair of endpoints, to extract the elements of importance for debugging the issue. These high-level elements may not provide a full view of the state of the network. For example, a circuit will provide a logical Layer 2 link between two topological points. Having the endpoints of this link will provide only minimal information about why a problem may be occurring on the link. If this minimal information does not provide an answer, the client must then try to ascertain the ‘related’ elements that might be causing the problems. These might include the lower-layer network elements that a higher-layer element makes use of. The related elements might also include those elements directly attached to the main elements. Understanding which elements are related to a given network element is a non-trivial task and will likely require domain-specific knowledge. However, all these relationships potentially exist in the topological description allowing clients to search for the relevant related elements.

Once the client has the set of topological elements relevant to the query, it would be able to find network measurements conducted over those elements. This could include active measurements like a bandwidth test between two points or passive measurements like the current utilization for the element. With this set of topological and measurement information, the client could then begin searching for the cause of the problem.

8. CONCLUSION

We have presented an extensible framework for representing and exchanging network performance information. This schema forms the basis for the protocols used by the *perfSONAR* project [17]. The *perfSONAR* project is working to define a set of protocols and develop compatible software implementations that enable interested clients to locate and retrieve network performance information. The *perfSONAR*-compatible software suites have seen deployments across a wide variety of networks including *Internet2*, *ESnet* and *GEANT2*. The *perfSONAR* infrastructure is also going to be used to help monitor the network connecting the LHC sites.

The Network Markup Language Working Group is also being influenced by the ideas in this work. The goal of the Network Markup Language group is to define a schema to standardize the descriptions of arbitrarily complex network topologies. A number of the features and properties of the topology schema described above are being incorporated into the standard produced by the working group.

This work has seen growth far beyond its original intent. While developed to provide network information for Grid computing applications and agents, this work is now being used by research and education network operators all over the world, and is influencing schemas and projects that may prove even more important in the future.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the Open Grid Forum Network Measurement working group, both past and present, for their efforts and input. This work has been shaped by those discussions and by many observations and insights over the years.

REFERENCES

1. The Energy Sciences Network. http://www.es.net/ [17 March 2008].
2. GANT2. http://www.geant2.net/ [17 March 2008].
3. Internet2. http://www.internet2.edu/ [17 March 2008].
4. The Large Hadron Collider. http://public.web.cern.ch/Public/en/LHC/LHC-en.html [17 March 2008].
5. CERN—European Organization for Nuclear Research. http://public.web.cern.ch/Public/Welcome.html [17 March 2008].

DOI: 10.1002/cpe

6. Lowekamp B, Tierney B, Cottrell L, Hughes-Jones R, Kielmann T, Swany M. Enabling network measurement portability through a hierarchy of characteristics. *Proceedings of the Fourth International Workshop on Grid Computing*, 2003.
7. Hanemann A, Boote J, Boyd E, Durand J, Kudarimoti L, Lapacz R, Swany M, Trocha S, Zurawski J. PerfSONAR: A service oriented architecture for multi-domain network monitoring. *Service-Oriented Computing—ICSOC 2005*, 2005; 241–254.
8. Guok C, Robertson D, Thompson M, Lee J, Tierney B, Johnston W. Intra and interdomain circuit provisioning using the OSCARS reservation system.
9. AutoBAHN. http://www.geant2.net/server/show/ConWebDoc.2544 [17 March 2008].
10. Open Grid Forum. http://www.ogf.org/ [17 March 2008].
11. Extensible Markup Language. http://www.w3.org/XML/ [17 March 2008].
12. Namespaces in XML 1.1. http://www.w3.org/TR/xml-names11/ [16 August 2006].
13. RFC 1157—Simple Network Management Protocol (SNMP). http://www.faqs.org/rfcs/rfc1157.html [17 March 2008].
14. Telcordia GR-831-CORE: Operations Application Messages—Language for Operations Application Messages. Telcordia Technologies [November 1996].
15. A Universally Unique IDentifier (UUID) URN Namespace. http://www.faqs.org/rfcs/rfc4122.html [17 March 2008].
16. RFC 2141—URN Syntax. http://www.ietf.org/rfc/rfc2141.txt [24 March 2008].
17. The perfSONAR Consortium. http://www.perfsonar.net/ [17 March 2008].

DOI: 10.1002/cpe