Use Cases

Information Services Use Cases

The following are compiled use cases for Information Services:

Control Plane Use Cases

Cases specific to the world of control plane functionality. Note that some cases here can easily be generalized to other domains (e.g. performance monitoring, distributed computing) but are presented through this community.

ESnet SDN (Science Data Network) - ...

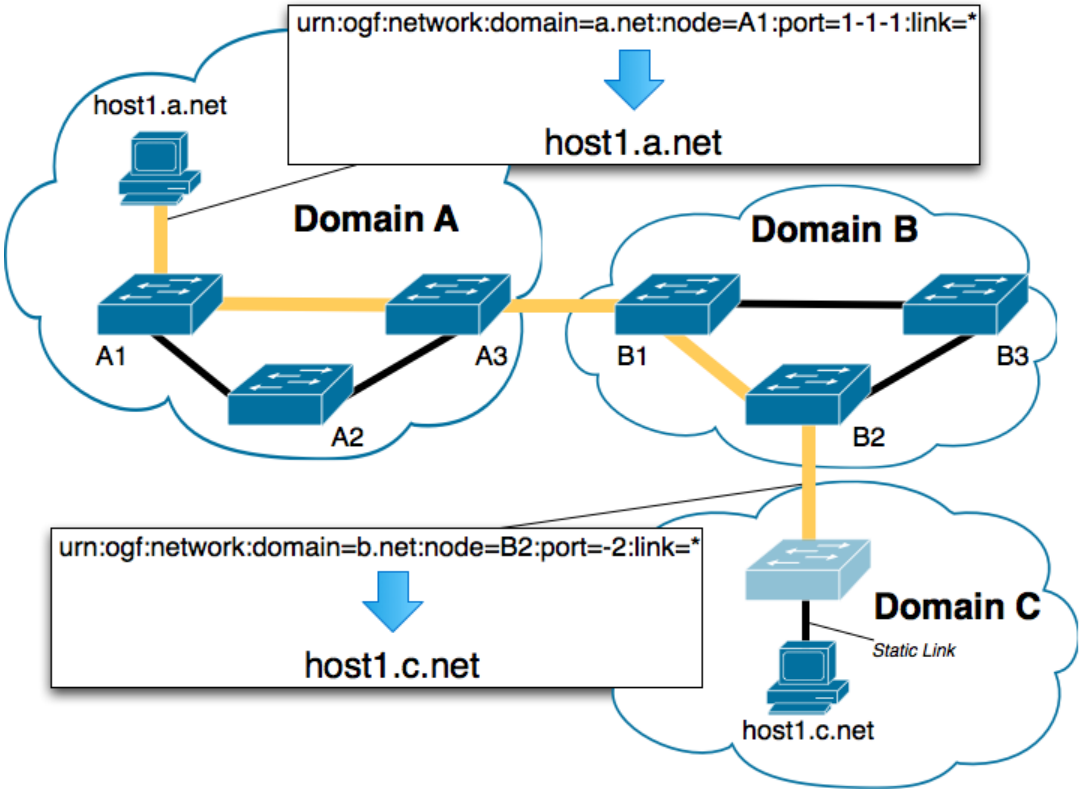
Author: Evangelos Chaniotakis - ESnet

TBD

Internet2 DCN (Dynamic Circuit Network) - Edge mapping

Author: Andy Lake - Internet2

Dynamic circuit networks implementing the Interdomain Controller (IDC) protocol require requesters of circuits to specify the edge links of a dynamic connection. The IDC protocol uses the NMWG topology schema to describe those links as network uniform resource names (NURNS). An NURN describes a hierarchy of domain, node, port, and link that is well suited for machine processing but can result in rather long identifiers. In practice, requesters that need to manually enter endpoints (i.e. via a form on a web page) found these identifiers hard to remember and prone to typos. Similar to how DNS maps human-readable names to IP addresses, the Information Service can alleviate this issue by mapping human-readable names to NURNs. The figure below shows an example of a dynamic connection between two hosts both with human-readable identifiers registered with the Information Service:



The diagram demonstrates a few important points of this use case. The thick orange line represents the dynamic portion of the connection. Some hosts are directly connected to a node that can be dynamically provisioned such as the host in Domain A. In this case we can generate the human-readable name *host1.a.net* that maps to an NURN identifying the link between the host and the dynamically configured node. It should be noted that the name *host1.a.net*looks like a DNS name but it may or may not be registered in DNS. DNS-style names are used because they are common on the web and therefore familiar to users. Also, DNS-style names provide a concise structure that can be summarized for global discovery by a distributed set of Information Services.

The host in Domain C represents a more complicated yet common case in current dynamic circuit networks. Domain B is capable of dynamic configuration but the node in Domain C is not; therefore the host in Domain C is directly connected to node C1 with a static connection. The last dynamic link is that between Domain B and Domain C (as shown by the orange line). When we create a name like host1.c.net we do not want to map it to the static link since the dynamic circuit software cannot provision that link. Instead, we want to map it to the last dynamic connection- the link between domain B and C. A side-effect of this approach is that if domain C ever becomes dynamically configurable the mapping of the name can be changed to the link between the host and node on domain C transparently to the requester. The requester will continue to use *host1.c.net* to provision the circuit but it will automatically map to the new link.

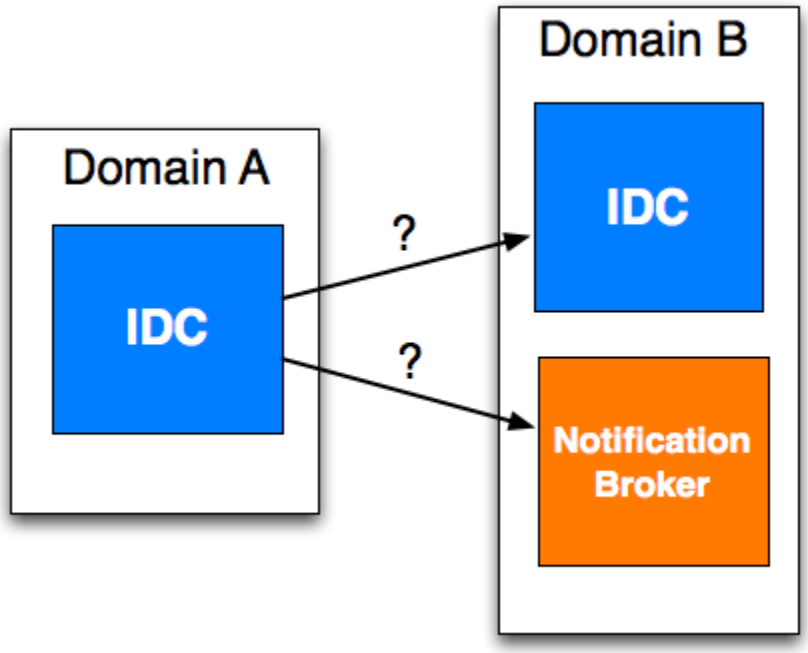
The cases above work well currently but a few questions still remain such as:

- Can user-friendly names be used with non-edge link to aid with processes such as edge discovery?
- Names and their NURN mappings are currently kept in the Lookup Service and information about the link that the NURN identifies is in the Topology Service. Would it be beneficial to have both of these pieces of information in an integrated Information Service?

Internet2 DCN - Service Discovery

Author: Andy Lake - Internet2

The Interdomain Controller (IDC) protocol requires that messages be passed between services in different domains. The OSCARS IDC implementation requires a domain to interact with two services: it must forward messages to another Interdomain Controller and it must subscribe to notifications from a NotificationBroker. The IDC sends these message to a URL and the IDC must be able to discover this URL. The figure below shows an IDC in Domain A that needs to interact with services in Domain B but does not know their locations:



The current discovery options are as follows:

1. Manually enter the URL of each neighboring IDCs IDC and NotificationBroker
2. Each service registers with the Lookup Service and neighbors automatically discover the URL of the registered services

The second case is clearly more desirable as its easier to add new neighbors, discover changes to a URL if a service moves, and avoid human error. In practice it has eliminated some of the more confusing configuration steps required when deploying an IDC.

The lookup service not only allows a domain to ask questions like "Where is my neighbor's IDC service?" but can also provide other useful information about the neighbors service(s). For example, as the IDC protocol continues to change and interact with other projects it becomes increasingly important to determine **what protocol** (or version of the IDC protocol) another domain accepts. Current IDC implementations do not check the protocol but they likely will in the very near future as all the facilities to do so exist in the current lookup service. For example if an IDC discovers another domain speaks an older version of a protocol it may send a different message than it would otherwise. Information like this will continue to be important as the project grows.

Performance Monitoring Use Cases

Cases specific to the world of performance monitoring. Note that some cases here can easily be generalized to other domains (e.g. control plane, distributed computing) but are presented through this community.

perfSONAR - Measurement Archive Queries

Author: Jason Zurawski - Internet2

perfSONAR Measurement Archives (MAs) are services designed to store performance measurement data over time. Typical MA design allows a single type (or several related types) of measurement information to be stored in the same facility, usually dictated by back-end storage. Some examples:

1. RRD MA - Backend storage consists of Round Robin Databases which normally store network data collected from SNMP. Data types include utilization, discards, errors and other related types.
2. SQL MA - Backed storage consists of SQL typed database (e.g. MySQL, PostgreSQL, SQLite) which can store almost any type of measurement data. Structure is limited by table complexity.
3. perfSONAR-BUOY MA - Backend is a MySQL database and measurement data can be OWAMP or BWCTL data.

The query interface of all current perfSONAR services is highly dependent on the XML representation as dictated by the standards discussed in the OGF's NM-WG. For example an RRD MA stores information in different RRD files for each interface of a network device, e.g. for network device **somerouter** on the **Internet2** network there will be many interfaces. Gigabit Ethernet port 3/2 would be stored in file:

```
/data/somerouter.internet2.edu--ge-3_2_0.rrd
```

The MA service needs a way to tie the description of this specific interface and device to this file. The well formed XML would be similar to:

```
<nmwg:metadata xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="m-in-netutil-1">
  <netutil:subject xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/" id="s-in-netutil-1">
    <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
      <nmwgt:ifAddress type="ipv4">10.10.10.1</nmwgt:ifAddress>
      <nmwgt:hostName>somerouter.internet2.edu</nmwgt:hostName>
      <nmwgt:ifName>ge-3/2/0</nmwgt:ifName>
      <nmwgt:ifIndex>2</nmwgt:ifIndex>
      <nmwgt:direction>in</nmwgt:direction>
      <nmwgt:capacity>1000000000</nmwgt:capacity>
    </nmwgt:interface>
  </netutil:subject>
  <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:eventType>
  <nmwg:eventType>http://ggf.org/ns/nmwg/tools/snmp/2.0</nmwg:eventType>
</nmwg:metadata>

<nmwg:data xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="d-in-netutil-1" metadataIdRef="m-in-netutil-1">
  <nmwg:key id="k-in-netutil-1">
    <nmwg:parameters id="pk-in-netutil-1">
      <nmwg:parameter name="eventType">http://ggf.org/ns/nmwg/tools/snmp/2.0</nmwg:parameter>
      <nmwg:parameter name="eventType">http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:parameter>
      <nmwg:parameter name="type">rrd</nmwg:parameter>
      <nmwg:parameter name="file">/data/somerouter.internet2.edu--ge-3_2_0.rrd</nmwg:parameter>
      <nmwg:parameter name="valueUnits">Bps</nmwg:parameter>
      <nmwg:parameter name="dataSource">ifinoctets</nmwg:parameter>
    </nmwg:parameters>
  </nmwg:key>
</nmwg:data>
```

The ***metadata*** portion contains the higher level description, namely how we could identify this specific network functionality from a very high level view. It is a simple interface on some network device and this specific description only cares about interface utilization (e.g. normally collected via SNMP and reading the ifInOctets/ifOutOctets counters). The ***data*** portion is more concerned in describing how we may reach the archived measurement observations. In this cae the RRD file has a name, and we are able to describe some of the internal portions (e.g. the ***datasource*** name, and the units we are measuring).

This format takes advantage of some of the structure of the XML to provide the semantics on how to link the information together: namely through the use of **ids** and **idReferences**. In the example above, the metadata (e.g. ***m-in-netutil-1***) is linked to the data (e.g. ***d-in-netutil-1***) through the use of these references:

```
<nmwg:metadata xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="m-in-netutil-1" />

<nmwg:data xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/" id="d-in-netutil-1" metadataIdRef="m-in-netutil-1" />
```

All perfSONAR MA services feature a simple XML based query system to locate the stored information. Request messages (also constructed in XML) are used to locate information for a given service. To build upon the previous example, a interested client application may want to know about all interface utilization on **somerouter2.internet2.edu** and may think that a certain MA has this information. A request will be sent similar to this:

```
<?xml version="1.0" encoding="UTF-8"?>
<nmwg:message type="SetupDataRequest" id="setupDataRequest1"
              xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
              xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
              xmlns:snmp="http://ggf.org/ns/nmwg/tools/snmp/2.0/">

  <nmwg:metadata id="meta1" xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/">
    <netutil:subject xmlns:netutil="http://ggf.org/ns/nmwg/characteristic/utilization/2.0/" id="s-in-netutil-1">
      <nmwgt:interface xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/">
        <nmwgt:hostName>somerouter2.internet2.edu</nmwgt:hostName>
      </nmwgt:interface>
    </netutil:subject>
    <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:eventType>
  </nmwg:metadata>

  <nmwg:data id="data1" metadataIdRef="meta1" xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"/>

</nmwg:message>
```

The service will perform simple matching against the internal storage to see if there is anything matching this request. Missing fields are assumed to be ***wild cards*** in this system. There are several advantages to this query interface:

1. Easy to construct APIs that are capable of constructing the proper message format
2. Flexible - allows arbitrarily complex queries based on several field names
3. Simple - requests and subsequent responses from the service are easy to interpret for results

Drawbacks also exist, many that limit the usefulness of perfSONAR MA services:

1. Must be aware of internal storage formats - requires knowledge of XML schemata
2. Each MA will use a different schemata, APIs help but this requires knowing many XML dialects
3. Cannot construct queries based on higher level needs, e.g. give me information on domain ***internet2.edu*** or IP range ***192.168.0.0/16***

perfSONAR - Lookup Service Queries

Authors: Jason Zurawski - Internet2

The perfSONAR Lookup Service comes in two flavors:

- **Home Lookup Service**
- **Global Lookup Service**

A brief overview of each follows:

Home Lookup Service

The Home Lookup service (or ***hLS***) is meant to serve as a *lightening rod* to attract complex queries away from individual MA and MP services. For instance if a client is looking for measurement data in the **internet2.edu** domain there may be 4 MAs to choose from. How does this client know which to search? There are two ways to achieve the goal:

1. Search **all** services. This will consume lots of resources
2. Search the **correct** service. This is more efficient, but knowing which service is correct also takes some domain knowledge.

The hLS accepts *registrations* from all services offering perfSONAR data: simply put this is a copy of the metadata storage that each service maintains. This replication of information enables the hLS to know the availability of all data for a specific domain. Each hLS is able to combine the information from these registrations into a single ***summary***. This process tries to extract some important bits of information from each XML registration for a given service:

1. Data Types - These are pulled via the eventType information.
2. Domains - pulled from host specification
3. IP Addresses - pulled from address specification (v4 and v6). We aim to perform CIDR summaries on this when all are collected so we can advertise ip *ranges*. E.g. if we have ***192.168.0.1***, ***192.168.0.2***, and ***192.168.0.3*** we can advertise having information on ***192.168.0.0/30***
4. Keywords - User defined *tags* that describe a dataset. For instance if the data was collected for a project such as ***USATLAS*** or on a network such as ***ESnet***.

The query interface to the hLS is similar to that of the perfSONAR MA and MP interface. At the core it is an XML message but there are two distinct kinds of queries:

1. ***Classic Query***: Relies on knowledge of the internal storage structure of the hLS which is really just an XML database. This query allows client applications to pass raw XPath and XQuery statements
2. ***Discovery Query***: Utilizes the summarization process described above to search for specific elements that may be present in the database.

The original query message has a format similar to this example:

```
<?xml version='1.0' encoding='UTF-8'?>
<nmwg:message type="LSQueryRequest"
              id="msg1"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
              xmlns:xquery="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/xquery/1.0/">

  <nmwg:metadata id="meta1">

    <xquery:subject id="sub1">

  declare namespace nmwg="http://ggf.org/ns/nmwg/base/2.0/";
  /nmwg:store[@type="LSStore"]/nmwg:metadata

    </xquery:subject>
    <nmwg:eventType>http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/xquery/1.0</nmwg:eventType>

    <xquery:parameters id="param.1">
      <nmwg:parameter name="lsOutput">native</nmwg:parameter>
    </xquery:parameters>

  </nmwg:metadata>

  <nmwg:data metadataIdRef="meta1" id="d1"/>

</nmwg:message>
```

The goal of this message is to return all metadata elements in the XML database where the ***nmwg:store*** element has a type equal to ***LSStore***. As noted earlier, this is a property of the actual storage. Note that the contents of the ***subject*** may contain any valid XQuery or XPath statement. There are several advantages to this query interface:

1. Easy to construct APIs that are capable of constructing the proper message format
2. Flexible - allows arbitrarily complex queries based on XQuery and XPath

Drawbacks also exist:

1. Must be aware of internal storage formats - requires knowledge of XML schemata and the organization of the database
2. Due to arbitrary nature, APIs cannot offer specific functions but can simple expose the XPath/XQuery to be sent
3. Cannot construct queries based on higher level needs, e.g. give me information on domain ***internet2.edu*** or IP range ***192.168.0.0/16***

To address some of the drawbacks, the Discovery protocol was designed. The Discovery message has a specific format as seen here:

```
<?xml version='1.0' encoding='UTF-8'?>
<nmwg:message type="LSQueryRequest"
              id="LSDiscoveryRequest"
              xmlns:perfsonar="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
              xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
              xmlns:psservice="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/1.0/"
              xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
              xmlns:summary="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/summarization/2.0/"
              xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/"
              xmlns:nmtl3="http://ogf.org/schema/network/topology/l3/20070828/">

  <nmwg:metadata id="meta1">
      <summary:subject xmlns:summary="http://ggf.org/ns/nmwg/tools/org/perfsonar/service/lookup/summarization/2.
0/" id="subject.1">

<!-- can have multiples of each, note that this creates an 'or' relationship -->
        <nmtb:address xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/" type="ipv4">128.
4.133.167</nmtb:address>
        <nmtb:address xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/"type="ipv4">128.4.100.45
</nmtb:address>

        <nmtb:domain xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/">
          <nmtb:name type="dns">edu</nmtb:name>
        </nmtb:domain>

        <nmtb:domain xmlns:nmtb="http://ogf.org/schema/network/topology/base/20070828/">
          <nmtb:name type="dns">udel.edu</nmtb:name>
        </nmtb:domain>

        <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/utilization/2.0</nmwg:eventType>
        <nmwg:eventType>http://ggf.org/ns/nmwg/characteristic/errors/2.0</nmwg:eventType>

        <summary:parameters>
          <nmwg:parameter name="keyword">project:Geant2</nmwg:parameter>
        </summary:parameters>

<!-- the combination of all things is an 'and' relationship, this entire subject is therefore:

('128.4.133.167' or '128.4.100.45')  and
('udel.edu' or 'edu') and
('http://ggf.org/ns/nmwg/characteristic/utilization/2.0' or 'http://ggf.org/ns/nmwg/characteristic/errors/2.0')
and
('project:Geant2')

-->

      </summary:subject>

    <!-- need this... -->
    <nmwg:eventType>http://ogf.org/ns/nmwg/tools/org/perfsonar/service/lookup/discovery/summary/2.0</nmwg:
eventType>


  </nmwg:metadata>

  <nmwg:data metadataIdRef="meta1" id="d1"/>

</nmwg:message>
```

In general these messages offer a greater expressiveness than is available through XQuery. These 2 query methods serve an important purpose for the LS infrastructure as well as perfSONAR as whole.

Global Lookup Service

The Global Lookup service (or ***gLS***) acts as a globally accesible directory of Information Services. The design and internal workings of this service are exactly the same as the Home Lookup Service (***hLS***) with a single exception: MA and MP services register with an hLS and in turn the hLS will register with the gLS. hLS registration involves take the summarized data (discussed above) and sending this off to one or many gLS instances. gLS instances are then capable of sharing the hLS information amongst themselves to offer complete information coverage.

The two aforementioned query types are available in this service as well, offering a complete solution to information location at the local and global levels.

perfSONAR - Circuit Monitoring Queries

Authors: Aaron Brown - Internet2

TBD

perfSONAR - Finding Closest MP (Measurement Point)

Authors: Marcos Portnoi - UDel, Jason Zurawski - Internet2

In a network measurement infrastructure such as perfSONAR, Measurement Points (MPs) are the devices responsible for running the proper tools and performing the collection of measurement data. When a certain measurement needs to be executed between two end points, one or more MPs (depending on the desired kind of measurement) are activated in order to conduct the mensuration. Depending on the network topology and end points involved, the MPs might (a) lie totally in the path between the end points, or (b) lie outside the path.

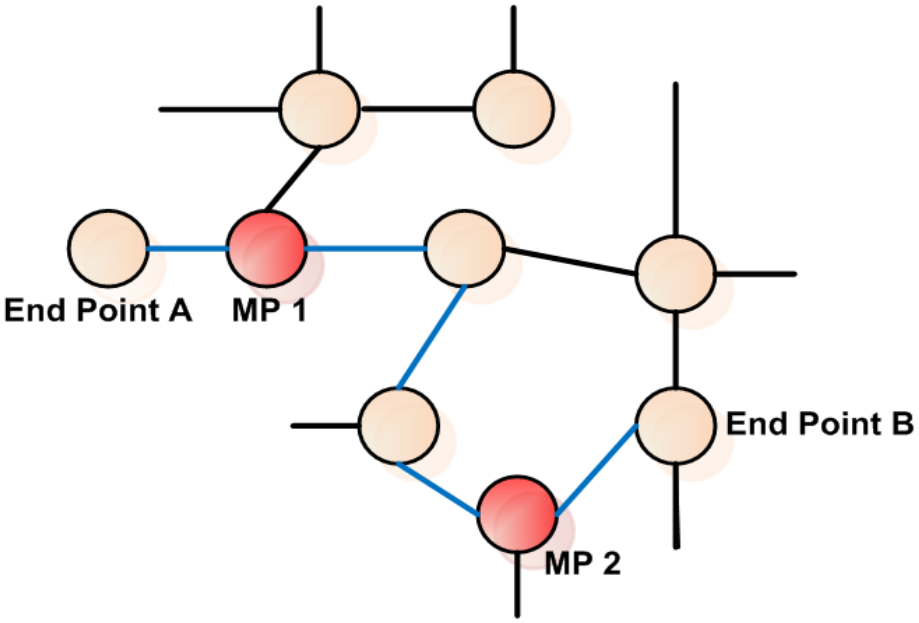


Figure 1: Example topology where MPs lie within a path connecting end points.

Case (a) is depicted in Figure 1. The desired measurement between end points A and B is conducted by MPs 1 and 2, which lie within the same path (shown in blue) that connects the end points. In case (b), illustrated in Figure 2, one of the MPs (MP 1) is located outside the path that connects end points A and B. But there is a path that connects MP 1 to MP 2, and this path includes one or more segments of the path that connects end points A and B. This way, the mensuration can still be carried. (Notice, also, that there is more than one path connecting end points A and B.)

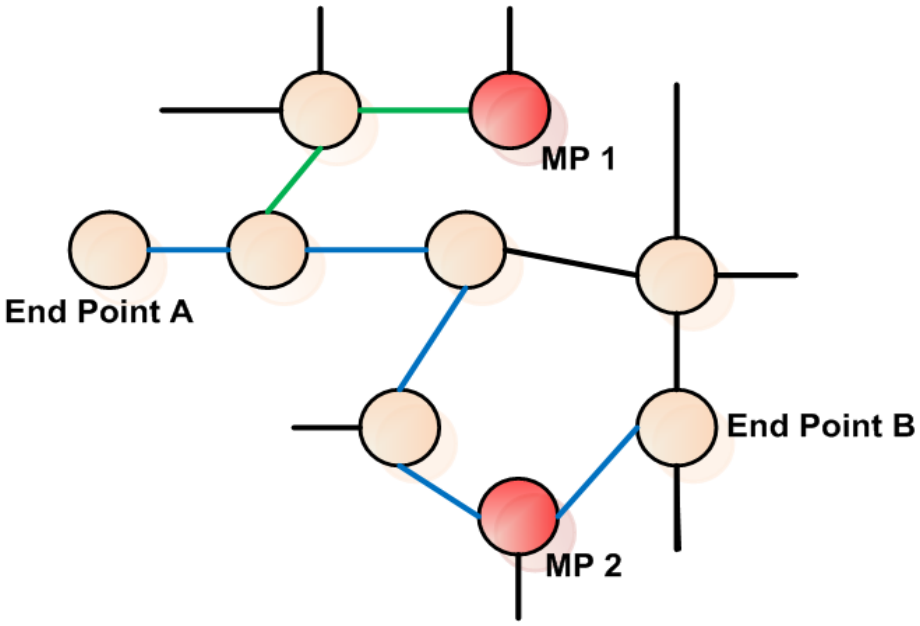


Figure 2: Example topology where one MP lies outside a path connecting end points.

Both cases (a) and (b) can pose their own problems to the measurement being performed. In (a), for example, the path between MPs 1 and 2 does not cover the entire path that connects end points A and B; the segments connecting end point A to MP 1, and end point B to MP 2 are left out of the measurements. Thus, the influence of these segments, such as latency, bandwidth, or loss, will not be computed in the overall results. In case (b), on the other hand, not only those segments won't be included in the measurement, but extra segments (shown in green in Figure 2) that connect MP 1 to the path between end points A and B will now contribute with their own characteristics to the overall results. Since these extra segments do not belong to the original path between end points A and B, the measurements in case (b) might contain noise. Conversely, it can also be said that the measurements in case (a) will not contain all the expected ***noise***.

An additional case is shown in Figure 3. Here, there are three possible MPs that can conduct the measurements; MPs 2 and 3 lie directly within the path connecting end points A and B, whereas MP 1 lies outside the path. So a choice must be made typically between MPs 1 and 3, and MP 2 remains an obvious choice.

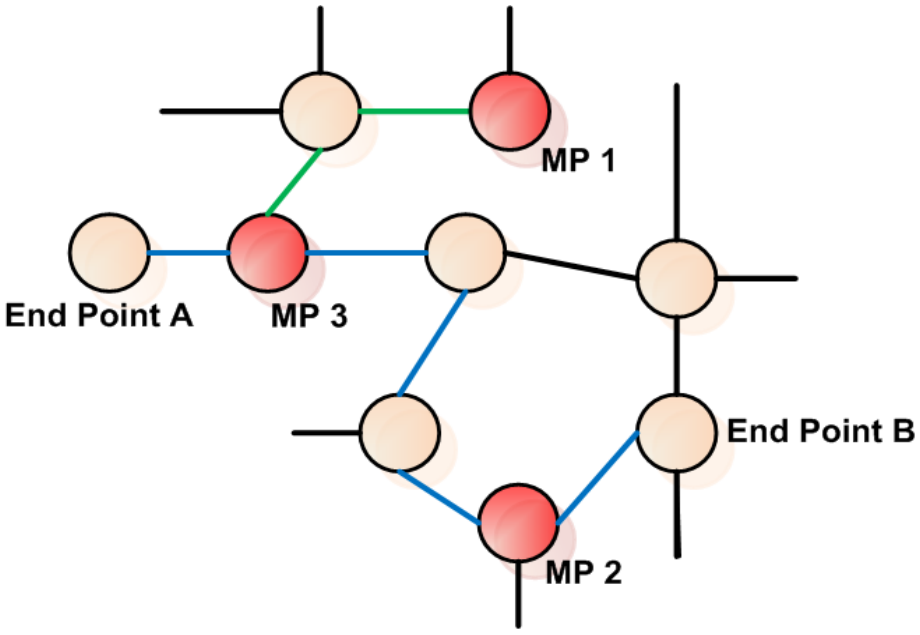


Figure 3: Example topology where several MPs are candidates for conducting a measurement.

In order to manage network measurements in a infrastructure such as perfSONAR, not only MPs are needed, but also their location in respect to the topology must be known, so as to allow choosing the most appropriate MPs in regard to the measurement. In general, the most appropriate MP is the one closest to the end point in the topology. (We can generally assume that the ***closest*** MP is the one with the least number of hops to a certain end point; this metric is flexible, however.)

The proposed service named Closest MP shall fulfill the need to optimize the measurement services performed by MPs, by facilitating the discovery of MPs and detecting their actual location within a topology. Simultaneously, by querying the metrics that can be obtained from each candidate MP (this query is made to the Lookup Service), this may result in a better choice of MPs which will be responsible for collecting the desired measurement.

OSCARS Software Suite - Current OSCARS/perfSONAR Interdomain Pathfinder

Author: Aaron Brown - Internet2

The current OSCARS release has two components that interact with the current perfSONAR Topology Service. The IDCs register their topologies with a Topology Service. The IDCs can then query these Topology Services to download the topologies and build an inter-domain graph. This inter-domain graph can be used to lookup the next domain along the path instead of having static routes configured for each possible destination domain.

The current steps used are sketched out in the following figure.

blocked URL

1. The IDCs register their topologies with their respective Topology Services
2. The Topology Services register which domains they contain information about with the Lookup Service infrastructure
3. When the IDC for domain A gets an incoming request, it begins to do path finding starting with its local topology. When it encounters a link into another domain, the IDC queries the Lookup Service infrastructure to find which Topology Service contains that domain's topology.
4. The IDC contacts that Topology Service and downloads the topology for that domain
5. The IDC continues doing the pathfinding until it finds the appropriate next domain, and connects to that domain.

This approach allows users to deploy their own Topology Service, or to re-use a centralized one. Current deployments are using a centralized Topology Service housed at Internet2, but there's nothing that prevents them from setting up their own.

Topology Abstraction

One feature of the current registration component is that it allows the IDC to register a full topology as well as an abstracted topology that does not include any of the internal links. This allows domains who would rather not divulge the structure of their networks to only register the external interfaces. The pathfinding component, when it sees one of these abstracted domains, assumes that a path can be constructed between any of the external facing interfaces.

Performance Considerations

There are a couple features that improve the pathfinding performance. The IDC will grab a domain's entire topology in one go instead of downloading it piecemeal while the pathfinding is being performed. This speeds up the pathfinding at the expense of increased memory use on the local machine. Were the complexity of the topologies too high or were there too many domains to search, the memory use might become excessive, but for the near term, this optimization works well.

Another way the pathfinding component improves performance is by caching the topologies it downloads as well as the services from which it downloaded the topology. Since the pathfinder will likely often calculate paths through a similar set of domains, the pathfinder can shortcut some of the lookups above. It can reuse the cached topology, bypassing any need to contact other hosts, or reuse the saved Topology Service URL allowing it to bypass the Lookup Service queries.

Requested Features

There have been some features requested of the pathfinder. The most notable is that there's not a way to create an authoritative representation for a domain. As long as there are no bad actors, this shouldn't pose much of a problem. Longer term, it would be useful to give users or IDCs some way to verify the data they receive from the Topology Service. How this would be done is an open question, but possible options would be to have domains sign their topologies, or to let domains define authoritative Topology Services for their topologies.

Another open question is whether there are better forms of topology abstraction that can be done. It might be useful to be able to register different levels of abstraction for different requesting parties. It might also be useful to register more complex abstract topologies, like one that has abstract internal links defining the possible paths that do exist through the network.

Virtual Organizations - Application monitoring

Specific communities use the network to get their job done. This includes scientific groups such as LHC, LIGO, Climate etc... For these groups the network is simply another resource they want to monitor in a similar way that they monitor their compute-clusters and storage-clusters.

Abstract locations for Pop's, Peering/Exchange points, compute clusters, storage clusters...

From an applications perspective, information such as throughput or latency information can largely be abstracted as a a location to location perspective. They want to know how long it takes packets to get from the storage cluster to the compute cluster. They do not care so much about specific hosts in either cluster.

From a network perspective things are similar. In general, Internet2 does not really care which of the hosts within a particular POP was used to perform a throughput test with an end site, or with another POP. The interesting information is the performance from location to location.

Requested Features

The ability to abstract a location without respect to IP/netmask in a only semi-coordinated fashion. A client application should be able to pose abstract queries such as: Latency from Internet2/CHIC to Internet2/LOSA. Note, I'm not saying this has to be supported directly via the MA. This could easily be a two-stage query where all 'hosts' associated with a particular 'location' are first found, and then queries are made to find all data related to a particular 'location'. The issue here is that an 'abstract' location entity seems to be needed at some level.

Authors: Jeff Boote - Internet2

TBD