Grid High Performance Networking Research Group	George Clapp *Telcordia Technologies*
GRID WORKING DRAFT	Tiziana Ferrari *Instituto Nazionale di Fisica Nucleare*
draft-ggf-ghpn-netservices-1	Doan B. Hoang *University of Technology, Sydney*
http://forge.gridforum.org/projects/ghpn-rg/	Mark J. Leese *Daresbury Laboratory*
Comments to ghpn-wg@gridforum.org	Paul Mealor *University College London*
Category: Informational Track	Franco Travostino *Nortel Networks Labs*

Status of this Memo
This memo provides information to the Grid community in the area of high performance networking. It does not define any standards or technical recommendations. Distribution is unlimited.

Comments: Comments should be sent to the GHPN mailing list (ghpn-wg@gridforum.org).

Grid Network Services 1
1 Introduction 1
2 Overview of Grid Network Services 1
3 Grid Network Services 5
3.1 The Store-and-forward Data Mover 5
3.1.1 Purpose, usage, and limitations 5
3.1.2 Service functional specification 5
3.1.3 Service components 5
3.1.4 Other Grid and general network services used 5
3.1.5 Interfaces to the Grid 5
3.1.6 Security 5
3.2 Another Service 5
3.2.1 Purpose, usage, and limitations 5
3.2.2 Service functional specification 5
3.2.3 Service components 5
3.2.4 Other Grid and general network services used 5
3.2.5 Interfaces to the Grid 5
3.2.6 Security 5
4 Grid Network Services or Network Services? 6
4.1.1 Network Information Service (a proxy to NM-WG) 6
4.1.2 Network Monitoring Service (a proxy to NM-WG) 8
AAA control 11
Delegation 11
Discovery 11
Requests for existing results 13
Predictions 14
Client requests for new measurements 14
Querying requests for new measurements 15
Service requests for new measurements 15
4.1.3 Data Transport Service with Network Quality of Service 19
4.1.4 Network Advance Reservation Service and Resource Management Service 20
4.1.5 Connectivity Service 20
4.1.6 Network Cost Estimation Service 20
5 Common Interface Design Principles for Grid Network Services 20
5.1 Interface Design Principles 20
5.1.1 Web service/Grid Service Definition 20
5.1.2 Grid Service Interface definition 22
5.1.3 Design Guidelines 23
6 Security Considerations 24
7 Authors Information 24
8 Intellectual Property Statement 24
9 Full Copyright Notice 25
10 Bibliography 25

Grid Network Services

1 Introduction

[Franco Travostino]

Network services specialize in the handling of network-resident or network-related resources such as QoS classes, policy enforcement points, topology data, network usage metrics, etc. Examples of network services include data transport service, network advance reservation service, network QoS service, network information service, network monitoring service, AAA[1] service, etc..

This informational draft describes how several network services combine and yield a rich mediation function between grid applications and legacy networks. Through these services, the network resource is seen joining CPU and storage as a first-class, grid-managed resource (and handled, as such, by a community scheduler, or other OGSA services).

A network service is further labeled as a **Grid Network Service** whenever the service has roles and/or interfaces that are deemed to be specific to a Grid infrastructure. The dominant foci of this GHPN effort are a) the relationship between Grid Network Services and the known services forming the Grid infrastructure, b) the functional characterization of each Grid Network Service, and c) the interplay among Grid Network Services and d) their interoperability requirements across multi-domain extents. The specification of any particular Grid Network Service (e.g., in terms of its actual portTypes) is out of scope. The breadth exercise captured by this document is meant to spawn depth work around several Grid Network Services, resulting in standard-track documents homed in either existing working groups or new working groups within the GGF.

2 Overview of Grid Network Services

[Franco Travostino]

Network services assist a grid infrastructure in different ways. In the simplest setup, a grid application (or a grid infrastructure on its behalf) consults a Network Service as if it was an omniscient oracle (e.g., a directory service) using a plain question/answer style of interaction. In more complex setups, Network Services interact with one another to realize one or more end-to-end feedback loop cycles (as in: observe + request + provision + act). Application requirements, policy considerations, and broker's directives are continuously injected into these feedback loops via expressive languages and machine interfaces (as opposed to, say, point-and-click sessions driven by operators).

[1] Authentication, Authorization and Accounting

Figure 1 shows an example of notional network services engaged in a fairly complex set of feedback loops[2]. Applications' demands, policy, and network's observed metrics (capacity, latency, jitter, loss, etc.) are continuously mediated, resulting in data marshalling and provisioning actions upon the network and/or the end-systems.

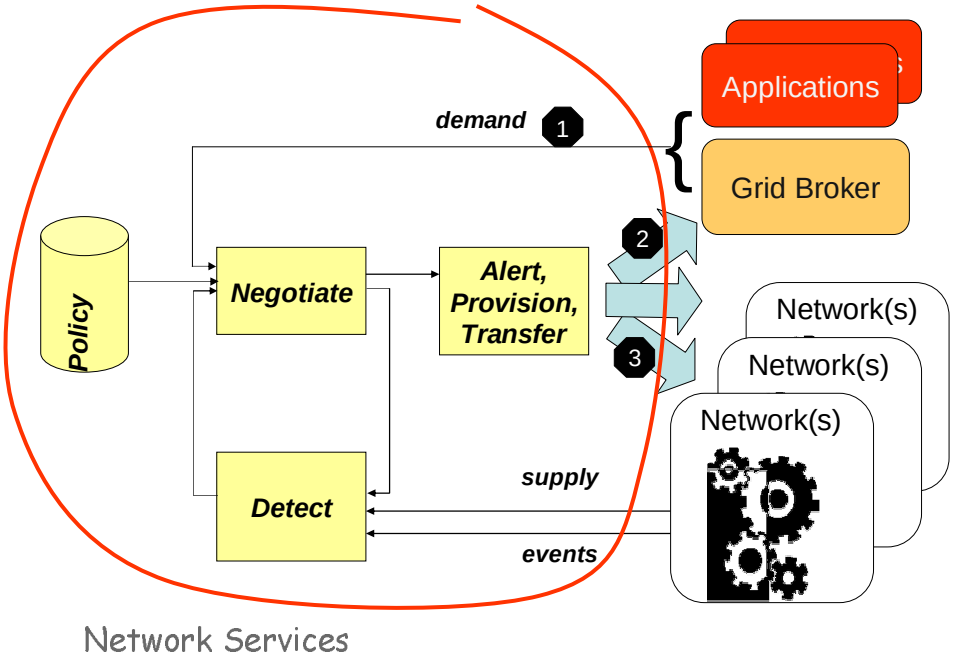


Figure 1. An example of Network Services in action
(e.g., for an hypothetical bulk data mover)

The various flows defined by boxes and edges must operate in a secure fashion across 1…N administrative boundaries. For some of the edges, there may be WS-Agreement Initiators and Providers at the opposite ends of the edge.

In Figure 1, the edge labeled 1 is meant to capture the following concept: there are mechanisms for the application (or the Grid infrastructure in its behalf, e.g. a broker) to invoke services, and pass on to these services parameters like data rate profile (time vs. rate), total data amount remaining (estimation or actual), and other characteristics associated with the data stream and/or the service request. In turn, these parameters aid network services in predicting and optimizing the utilization of network resources, resulting in greater satisfaction and/or cost efficiencies to the end user.

With regard to the edge labeled 2, a designated service must notify an application (or the Grid infrastructure) of those events that the application has negotiated and registered

[2] This picture was inspired by earlier QoS research as part of the DARPA Quorum effort. http://www.dist-systems.bbn.com/projects/QuOIN/FinalReport/QuOINFinalReport.pdf.

for. It must tell an application if it is admission-controlled out (be it a capacity or a policy issue). It must provide timely notifications of SLA violations to an application.

With regard to the edge labeled 3, when appropriate, credited services can dynamically (re)provision network aspects (e.g., to tap on either traffic engineering fixtures or TDM/WDM circuits upon a very large bulk transfers).

This document focuses on the role of boxes such as Policy, Negotiate, Alert, Adapt, and Detect[3], and the directed edges connecting them.

The **Grid Network Services** are those boxes and their derivatives which are directly exposed to elements of the Grid infrastructure (such as a universal Grid broker for all resource types). For these elements to interoperate, a GGF-sanctioned Grid interface is necessary. Examples of Grid network services include: an end-to-end bandwidth reservation service, a store-and-forward data mover service.

On the contrary, the network services are represented by those boxes and their derivatives which are not directly exposed to elements of the Grid infrastructure (e.g., they only interact with other network services, network control planes or management planes). For these, a GGF-sanctioned interface is a sufficient albeit not necessary implementation choice. Examples of network services include: a domain-specific bandwidth broker, a network directory service.

It is appropriate to think of the network services forming a practical underlay to the actual Grid Network Services, as pictured in Figure 2.

[3] As shown throughout this document, the NM-WG already has efforts underway to define several aspects of the “Detect” box.

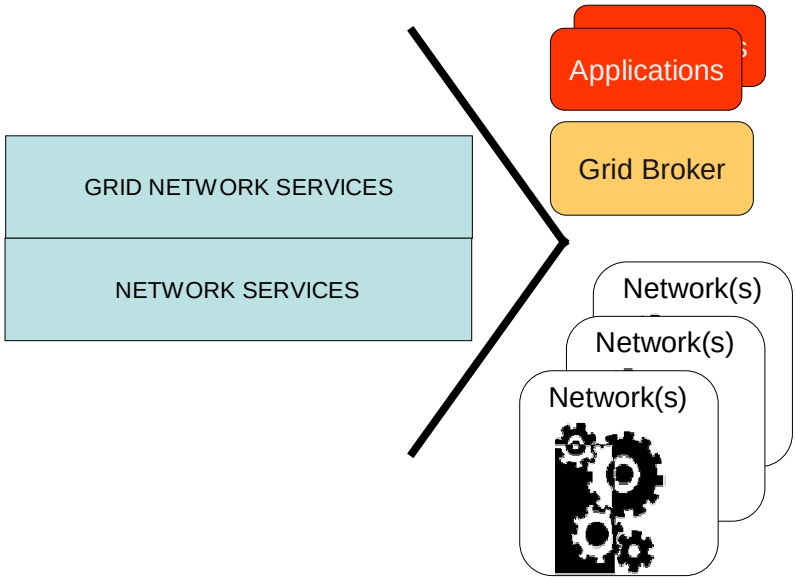


Figure 2. Grid Network Services vs. Network Services

In scoping Grid Network Services, we adopted a rigorous top-down approach. Supported by field experts, we documented Use Cases of the "black-box" type, in which we focus on requirements and the user's experience, while we strenuously avoid anticipating any implementation detail. The Use Cases cover a reasonably broad spectrum. While HEP is a most popular and fertile ground for Grid Use Cases, we have attempted to blend in less mainstream Cases, such as the one of sensors with intermittent connectivity. The outcome of our exercise is captured in the Use Cases draft [ref].

From Use Cases, the next step is to carve out Grid Network Services that have purpose and applicability within some key areas. Throughout this effort, it is crucial to avoid the syndrome of making "least common denominator" choices that can dissatisfy everyone. As well, it is important to avoid exceedingly specialized choices that are self-serving to one Use Case only, or one discipline only.

Once a Grid Network Service is sketched out, several questions need to be answered. What's the interface that the Grid network service exposes to a Broker or another element of Grid infrastructure (i.e., the northbound interface)? What's the schema for the attributes to the service? Which other network services does this service depend upon, are they available today, or are they beyond the state of the art? How is the service broken up in multiple instances across multiple administrative domains, do we need standards for inter-domain service exchanges?

The following sections introduce Grid Network Services, and attempt to answer these questions.

3 Grid Network Services

3.1 The Store-and-forward Data Mover

What it does. Why is this a useful service, and to whom?

3.1.1 Purpose, usage, and limitations

3.1.2 Service functional specification

3.1.3 Service components

3.1.4 Other Grid and general network services used

3.1.5 Interfaces to the Grid

3.1.6 Security

3.2 Another Service

What it does. Why is this a useful service, and to whom?

3.2.1 Purpose, usage, and limitations

3.2.2 Service functional specification

3.2.3 Service components

3.2.4 Other Grid and general network services used

3.2.5 Interfaces to the Grid

3.2.6 Security

4 Grid Network Services or Network Services?

[FT:

This text predates the Use Cases analysis.

Q: Do we believe that these are Grid Network Services rather than Network Services?

If the latter were to be true, is there a reason left why GGF should standardize them (as opposed to IETF, WS, DMTF)

I kept these two sections by Paul and Mark, because they ask a number of good questions, which may help us answering my earlier Q.

/FT]

4.1.1 Network Information Service (a proxy to NM-WG)

[Paul Mealor, Mark Leese]

[Request for feedback (Mark and Paul)
While we are clear on the collective role of the network information and monitoring services, we are much less clear on their division of responsibility. Our section currently has very little (if any) separation between the two.

1. The network information service provides an interface to the monitoring service. This appears to be the position adopted in other sections of the document. However, if the information service is merely some form of "wrapper" to the monitoring service, it begs the question, why not just send requests to the monitoring service. What extra value does the information service provide?

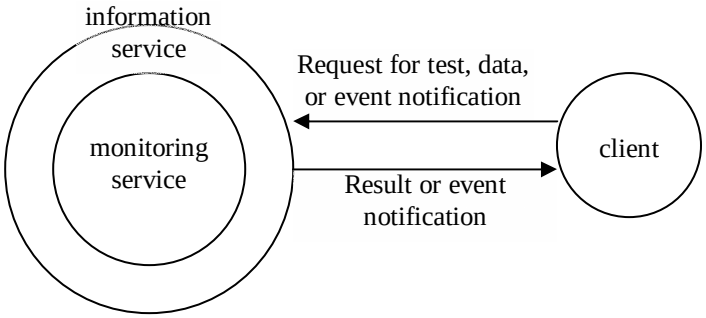


Figure 4. Information service as a "wrapper" to monitoring service

2. In the second option, the client interfaces to both services: the monitoring service to request new measurements (and possibly an event notification), and the information

service to request historic data or the results of a test that has just been run on the client's behalf.

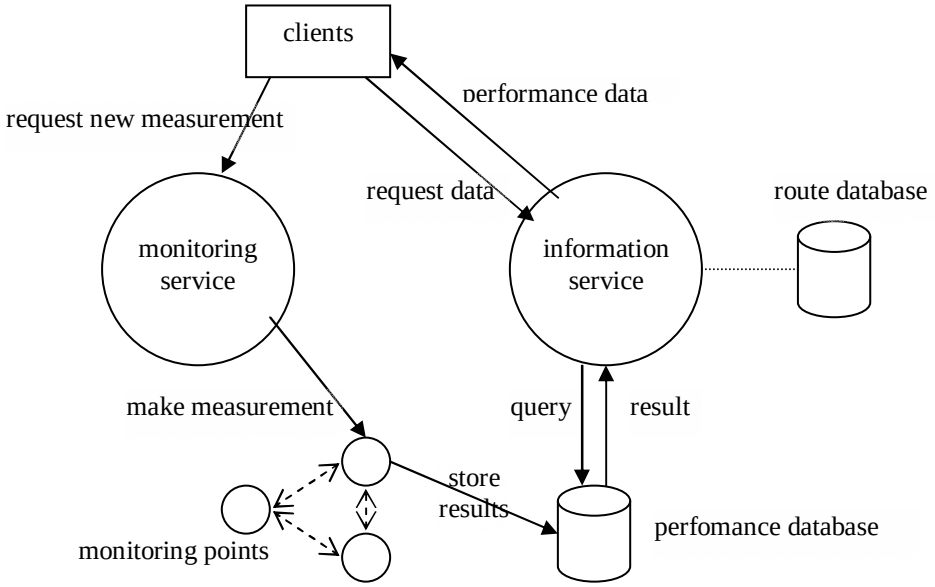


Figure 5. monitoring service for tests, information service for data

Figure 5 attempts to summarise the idea. Requests for new measurements are sent to the monitoring service, which co-ordinates monitoring nodes in performing tests. Once a test is complete, a monitoring node stores the results in a database. The monitoring service may then indicate to the client that the results are ready for collection. The client can then request the data, via the information service, and can if it wishes request non-monitoring information, such as details of routes and topology.

How this is achieved is largely irrelevant at this stage. We are only interested in what each service does.

3. The third option is to divide the services based on the information they provide, for example:

Information Service	**Monitoring Service**
Network topology	Measurement data
Route information	Event notification

Table 1. information and monitoring service versus information types

4. A précis of Franco's proposal is:
   The difference lies in the lifecycle of the data stored. A network information service is a directory service. Data changes very infrequently. The data that is stored is considered as authoritative. For a network monitoring service however, data changes

all the time, and that data can be considered as only a reasonable indication of what's happening. It is (as Franco puts it) “yesterday's news” and should be treated as such.

5. It could even be decided that having two separate services serves no real benefit, and that a single “information and monitoring service” service would suffice.

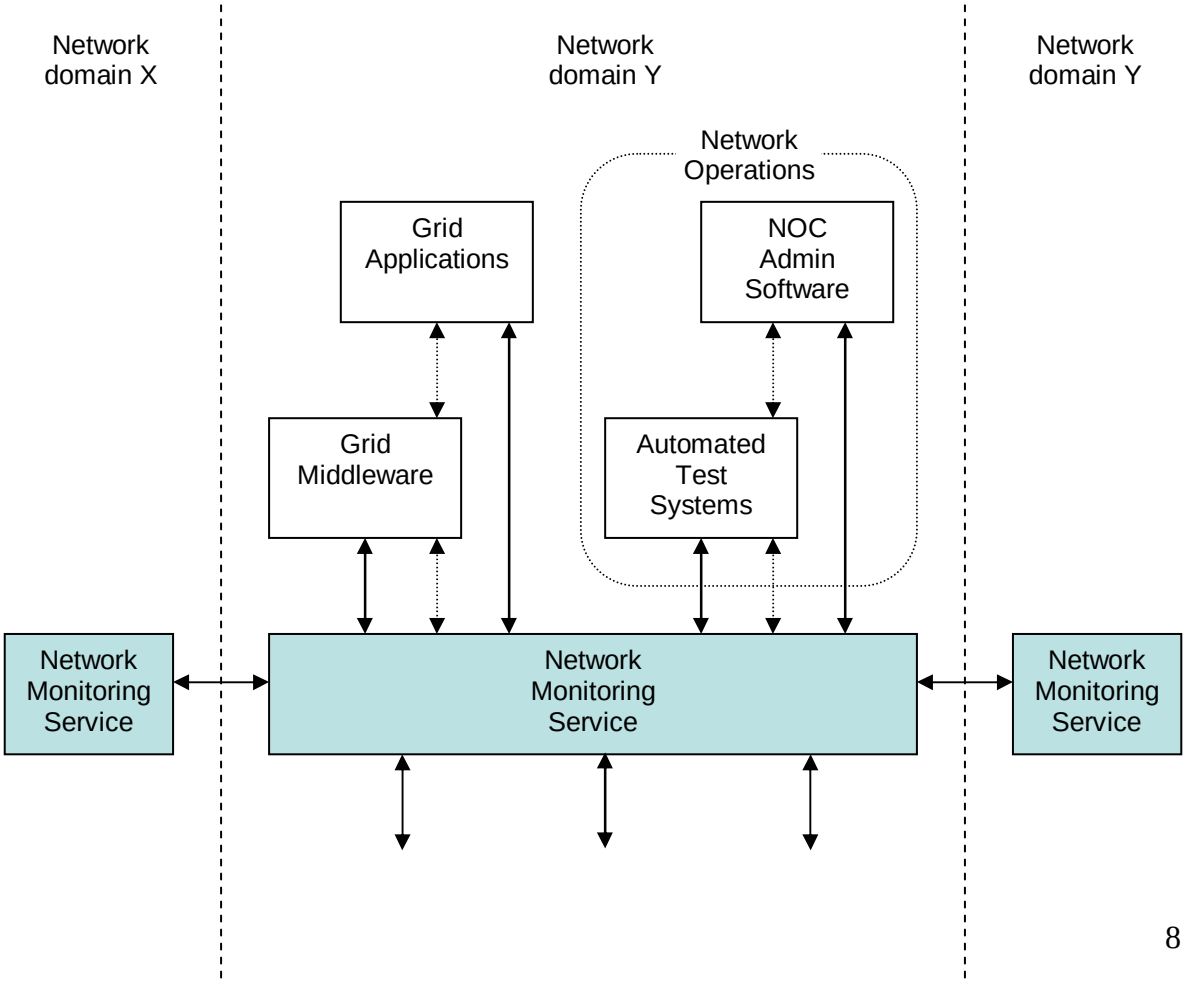
End-of-comment]

4.1.2 Network Monitoring Service (a proxy to NM-WG)

[Paul Mealor, Mark Leese]

Traditionally, network monitoring has been driven by the need for fault detection and performance prediction. While this remains true in Grid environments, a significant new concept is introduced, that of publishing performance data to Grid applications, middleware and the network fabric. This radical change will allow systems to both adapt to changing network conditions, thus optimising performance, and also provide support for the Grid’s much touted self-healing capability.

As figure 6 shows, the service’s potential clients are numerous and varied: Grid middleware and end-user software (Grid applications), other network services (e.g. network cost function), network administration software, such as admin tools used by human administrators in ‘network operation centre’ environments, automated test systems (e.g. [17]), and finally, corresponding monitoring services in other network domains.



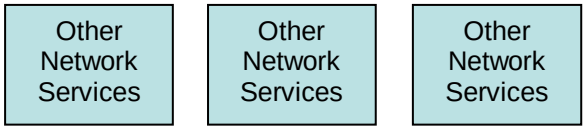


Figure 6. Clients of the network monitoring service

This section introduces the principle requirements for a Grid-enable network monitoring service, first in general overview, including the high-level goals of system, and then in more detail. It will also, where appropriate, make suggestions about how these services can be provided.

4.1.2.1 General Requirements and High-level Aims

In reference to the overall aim of GHPN's network services, the network information and monitoring services are to provide the functionality of the "Detect" box shown in figure 1. As suggested elsewhere in this document, these services will answer questions concerning network status and performance from grid applications and middleware, other grid services, such as a network-cost function, and the network fabric. The information and monitoring services are expected to fulfil the network-related aspects of 'observe' in the observe-request-provision cycle typically associated with Grid resource usage.

Of existing network monitoring efforts, Clarke in the "Grid-Network Interface" section of [18] comments that there are many excellent monitoring initiatives operating throughout the world, but highlights that although many are based on the same core set of monitoring tools, none lend themselves to either:

a. being used collectively to provide information along a complete network path, or
b. being used as low level monitoring services, providing network information to higher-level functions such as resource scheduling, or even SLA monitoring

Agreed interfaces into these architectures are clearly needed to allow access for resource schedulers and the like, and other network monitoring systems, in the latter case to achieve the somewhat utopian state, whereby sets of heterogeneous monitoring infrastructures, in different administrative domains, can interact to provide information for network paths spanning the globe.

We would now appear to reach a cross roads, where we must decide between developing Grid network monitoring services *de nouveau*, or Grid-enabling present architectures. New services could be carefully tailored to the Grid, but at the expense of being unable to leverage current architectures, e.g. for existing monitoring node deployment.

It would seem sensible at this stage to leave our options open, and attempt to consolidate the views of the Grid specific and wider network monitoring communities. To do this we must consider the heterogeneous nature of current monitoring architectures, and as a result, to a large extent ignore how Grid monitoring services could be implemented, focusing instead on defining the functionality and interfaces to access that functionality that monitoring architectures will need to provide to be used in a Grid environment.

It is possible of course to define how Grid network monitoring services could be implemented, as there may be some inherent performance gain in this. However, providing such an implementation should not be enforced. It is sufficient that the required behavior and interfaces are supplied.

So, in general terms, a network monitoring service should allow authenticated and authorised users to request:

1. historic performance data, from the running of previous tests
2. real-time performance data
3. new measurements, which may lead to the running of tests
4. future performance data, based on the assumption that a test is already scheduled
5. future performance data, as a prediction
6. event notifications, as a similar concept to SNMP traps

Even at this high-level view of requirements, there are already several points of note:

- In relation to points 1 and 4 above, regularly scheduled tests will need to be performed to provide users with data or predictions relating periods where they have not requested the running of tests.
- Points 3-5 above imply that it should be possible for a user to select whether a data request will ever result in the running of a test.
- As figure 6 suggests, it is expected that requests will work across multiple administrative domains. In addition to this direct requirement, it is clear monitoring services will also need the ability to discover further monitoring services.
- So far, we have discussed the monitoring service as a single entity. It is entirely possible that it will decompose into several sub-services.
- Further, many of the detailed requirements will make reference to “services” and “monitoring points”. These services could be the network monitoring service as a whole, or one of its possible sub-services. Services control “monitoring points”, the entities which make actual performance measurements. Services and monitoring points have one-to-one or one-to-many relationships.

And in terms of making requests and receiving results:

- Requests for data and tests, and the publication of performance data should in the main make use of the work of the GGF NM-WG group [19], who have defined XML schemas for such tasks.
- Internally, a monitoring service can use any communication method deemed appropriate, but the NM-WG approach should be supported externally. An example of internal communication is that between a monitoring service and its monitoring points (the nodes that actually make measurements).
- Interim communication, that taking place between a request being made and a result being returned, is yet to be addressed.
- A means for requesting event notifications is yet to be defined. In the strictest sense, event notification is a monitoring not measurement task, and may be deemed by NM-WG to be outside their scope.

4.1.2.2 Detailed Requirements

AAA control

1. It must be possible to restrict access to a service, or any part of that service, based on the client's identification. Put another way, it must be possible to control which users have access to a service, and what they are able to do.

   High-level access restrictions could include whether or not users can request inter-domain tests, or the frequency with which test requests can be made. Low-level restrictions could include controls on the duration of iperf [20] tests, or the number of parallel TCP streams that can be used during those specific tool tests.

   The list of possible access restrictions is potentially very large. For flexibility, the granularity with which access is controlled should be at the discretion of those implementing the service.
2. Services must be able to authenticate and authorise between different administrative domains.
3. The service should report if a request is to be refused.
4. Explaining the reasons for refusal, and the detail given in any explanation is at the discretion of those implementing and operating the service. It is expected that some implementations may want to explain to the client the reason for denial, whilst other implementation, perhaps for security reasons, may not.

Delegation

1. It must be possible for taking of measurements, and any prior negotiation, to be delegated to other components within the system.

   An example is given by the existing Internet2 piPEs architecture [17]. The principal building blocks of piPEs system are PMCs (Performance Measurement Controllers) which direct PMPs (Performance Measurement Points), the nodes which make actual performance measurements. A request for a new measurement will be sent to a PMC. The PMC decides which PMPs should be used to make the measurement, before forwarding on the request to one of the selected PMPs. The PMP receiving the request then negotiates with the remaining PMP to schedule the measurement.
2. In general, it must be possible for services to handle requests for measurements that do not directly involve the hosts on which those services run. In more simple terms, the monitoring service and the monitoring points for which it is responsible do not have to be hosted on the same machine.
3. It should be possible for services to *refer* clients to other services. Once a *referral* has been received, the client can contact the further service directly, freeing the initial service for other duties.

Discovery

1. It must be possible to discover the services responsible for a particular host (given the proper authorisation). This is perhaps best illustrated with a real world example, as shown in figure 7.

Let us assume that a large dataset from the planned Large Hydron Collider (LHC) facility at CERN is available for distribution to other institutions, such as Rutherford Appleton Laboratory, where it will be processed. A machine at RAL is acting as an LHC Tier-1 server. Given the amount of data to be transferred, it is quite reasonable for the RAL server to request information about network performance between itself and the LHC data store.

It is infeasible to perform tests directly between these machines, or to be holding past performance data directly associated with them, on which predictions of future network performance could be based. In reality these systems could be any node in the RAL and CERN domains, and so this would require test functionality to be installed on every node in those networks, and regular scheduled tests to be run between all nodes to build up a collection of historic performance data.

Instead, the required performance information is approximated to the RAL-CERN performance data held by the network monitoring services or monitoring points "nearest" to the tier-1 server and data store. Therefore, for a given host, it should be possible to find the network monitoring service with the most appropriate data for that host.

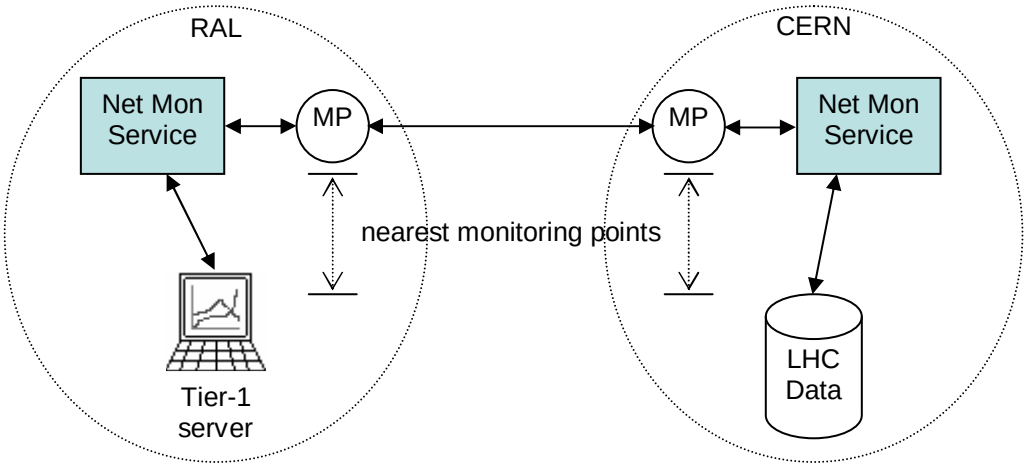


Figure 7. Monitoring point discovery

The discovery mechanism should be flexible, with two options shown in figure 8. The left hand example shows the client as responsible for locating the most appropriate monitoring service, via some form of discovery service. In the right hand example, the location task is part of the monitoring service functionality, meaning the client need only make a simple test request (the location aspect is transparent to them). Service implementers should not be forced into either option.

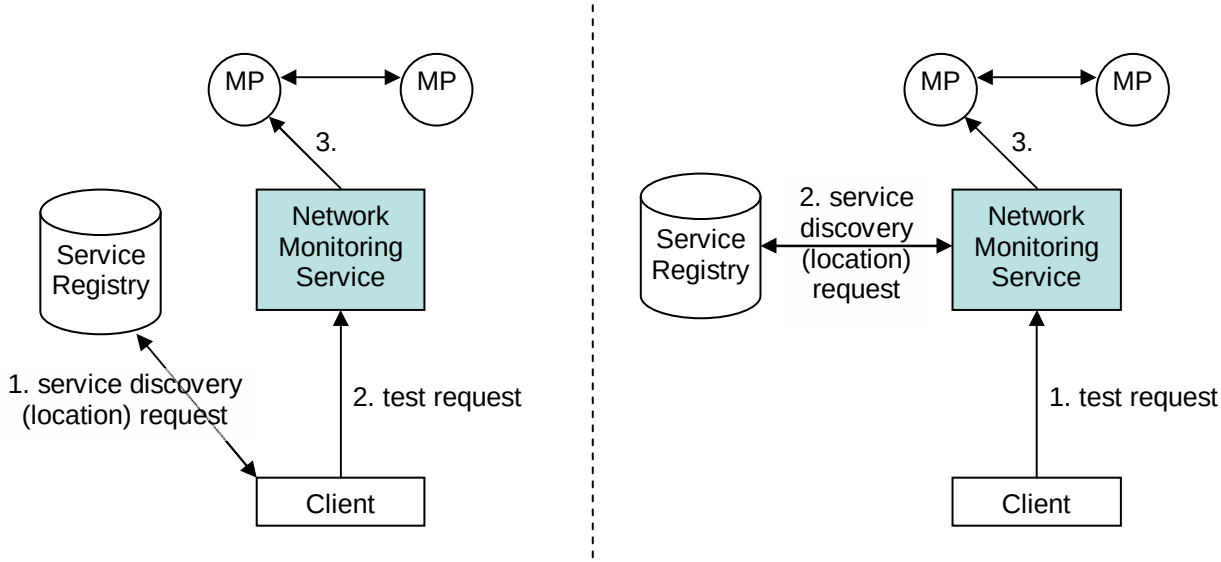


Figure 8. Possible discovery mechanisms

2. It should be possible to discover the types of measurements (characteristics) available from a particular service, and the parameters that can be set for those measurements, and the acceptable values of those. This information is likely to be used in a resource and capability discovery context when searching for services.

Requests for existing results

1. It should be possible to request the value of any measurement as long as it is available. When a measurement is not available, it should be possible to request the running of a new test, a prediction based on existing data (see later "Prediction" section), or no further action.
2. The service should not be constrained to currently defined characteristics, i.e. it should be extensible.
3. Measurement results should be available encoded using the NM-WG NetworkMeasurement schema [19] (or another appropriate form). However, it should be possible to specify that results are transferred in any form supported by both service and client. It is recognised that XML, as used by the exising NM-WG schemas, may not be the perfect medium for transferring extremely large volumes of data. Other, more compact formats should therefore be supported.
4. It should be possible to request measurements which were made with particular parameters, including ranges or choices of parameters.
5. It should be possible to request statistical summaries of data. However, which summaries are supported is matter for service implementers.
6. It should be possible to request any number of measurements within a particular time-range.

7. It should be possible to request and receive notification of a change in the status of a network. Initially, only notifications of new measurements need be supported. However, this functionality can be extended in the future, as appropriate.

Predictions

Monitoring architectures such as the Network Weather Service [22] are able to make predictions on the future state of networks. Where a prediction capability is available, services should be provided that meet the following requirements.

1. It should be possible to request predictions for the value of measurements that have not actually been made.
2. Requests for both future and past predictions are acceptable. In the case of predictions for past measurements, the prediction should be generated by interpolating "nearby" actual measurements.

   Under normal circumstances, a request for past data would be expected to return the chronologically closest measurement. The following example hopefully highlights the validity of the past prediction option: A user involved in performing a file transfer on a Thursday afternoon at 17:00 would likely request a prediction of performance based on measurements made on previous Thursday afternoons at 17:00. If measurement data was only available for 16:00 and 18:00 however, an interpolated prediction may prove more representative than either the 16:00 (within working day) or 18:00 (working day over) data.
3. Requests for predictions must be distinct from requests for new measurements or historic data. This allows users to select whether they receive real data or a prediction. And there are further advantages for future data requests. Firstly, users gain some control over whether a test will ever be performed (it will not if a prediction has been requested). And secondly, requesting a future prediction allows users to receive a value, albeit a prediction, without having to wait for a real measurement to be made.

Client requests for new measurements

1. It should be possible to for users to request new measurements.
2. Further, it should be possible for users to request schedules of one or more measurements, made at different times, with different parameters, between different hosts, or any other setting.
3. Users should understand that measurements may not be made exactly when or how requested. This may be because a component contributing to the measurement is unavailable at the requested time, or because of access restriction issues (as outlined in the "AAA control" section). Rules governing acceptable deviation from a schedule of measurements may be required.
4. It should be possible for users to be able to negotiate the settings of requested measurements, with the service handling their request. Negotiating the exact details of measurements in advance may be impossible, quite simply because some existing systems (which we wish to bring within the scope of this document) cannot guarantee that test requirements are met e.g. they perform "last minute" scheduling and do not know in advance if a requested test can be performed at exactly the desired time.

5. It should be possible to update or remove a particular schedule of measurements, given the proper authorisation.
6. It should be possible to track the progress of measurements in a schedule. That is, it should be clear when a measurement:
   a. has not yet been made;
   b. has been delayed for some (any) reason;
   c. has been cancelled;
   d. has been made; or
   e. is in some other state.

   Clearly, in a schedule, especially a repeating schedule, this reporting could be quite complicated.

Note that users are not expected to interact with schedules in every case. This could take place via the network monitoring service. For example, a user would not request the creation of a schedule of tests. They will simply request a new measurement, and an appropriate schedule may then be created on their behalf.

Querying requests for new measurements

1. Services should be possible to find out if any measurements are to be made at any particular time. This will aid in the scheduling of tests. Whether users have access to this information is at the discretion of the monitoring service's implementers.
2. Users and other services should also be possible to find out if any measurements were made at any particular time. A possible use for this facility is in fault detection. If a user or service detects an anomaly in performance at a particular time, they can verify if an event (such as a bandwidth intensive test) took place at that time, accounting for the anomaly. Of course, this would apply to all nodes along a test path, not just those at the ends.
3. As a collective option, it should be possible to examine an overall (or as close as possible) view of all measurements made or to be made.
4. It should be possible to narrow these queries to particular services, hosts, routes or some other criteria.

Service requests for new measurements

In most instances, making new measurements requires that the nodes at both ends of a test path are actively involved in making the measurement. An iperf [20] test for example involves the local and remote ends executing iperf client and server applications respectively. As a result, local and remote ends will be required to negotiate over the running of tests. In the main, this will be to check that a requested test is permitted, and that the required resources are available.

Even tests where one might not expect both ends to be actively involved may require negotiation. One could expect a ping test to require no negotiation, citing that the node receiving the ICMP echo requests is expected to respond automatically. However, such a test may require temporary holes opening in a site firewall, since many sites block ICMP traffic. Further, it could be desirable to check that other tests are not being performed: a

bandwidth intensive test to the other machine having the potential to skew the path's RTT for example.

As detailed requirements:

1. Services must be able to negotiate the timing of a measurement such that it does not interfere with other measurements. Where agreement cannot be reached, it is acceptable for requests to be refused. There are three possible solutions to this problem, classified by which ends of a test path have the relevant resources reserved: none, one or both.

   No reservation
   Schedule negotiation need only happen on a last-minute basis. That is, the negotiation need only start at the time when the measurement is to be made. In this scenario the monitoring service contacts the local monitoring point to ask if it is available to run the required test. If "yes" the monitoring service or monitoring point contacts their remote peer (service to service, or MP to MP) to ask if the remote end can also run the test. If also "yes" then the test can proceed. If either is "no", the request can either be rejected, or reattempted after some arbitrary delay.

   This method works on the assumption that measurements will be infrequent, of short duration (≤ 30 seconds) so that any delay will be minimal, and that in any event, a delay can be tolerated.

   The clear advantage of this method is that of the three options, it is the simplest to implement. The disadvantage is that measurements can be delayed or in the worst case, cancelled.

   Both ends reserved

   It would also be possible for two services to negotiate a schedule in advance of the actual measurement, if both services supported this. An example is shown in figure 9.

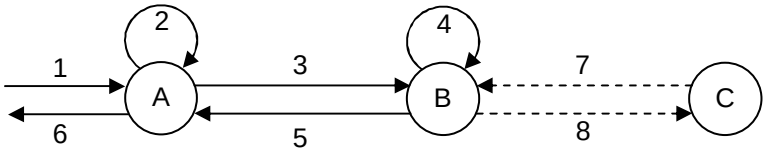


Figure 9. Test path end node reservation

1. Request for test at 12:00 received by monitoring point A
2. 'A' reserves the required resources for 12:00
3. 'A' forwards same test request to 'B'
4. 'B' is also available, so also reserves the required resources
5. 'B' confirms to 'A' that it is reserved for 12:00
6. 'A' confirms to the requesting system that the 12:00 test can proceed
7. 'C' forwards a test request for 12:00 to 'B'

8. ‘B’ responds that it is busy at 12:00, thus ‘C’ cannot disrupt the plans of ‘A’

The advantage in this case is that measurement resources are reserved in advance, allowing a test requestor to be confident that its measurements will be made, and at a time at least approximately equal to that requested. This comes at the expense of providing the implementation to do so, the possible complexity of which should not be underestimated.

One end reserved
The final option is the compromise, whereby just one of a test path’s end nodes, the node associated with test requestor, is reserved. This method provides some guarantee to the requestor, in that the local node will be reserved for their request, but again works on the assumption that tests will be infrequent, so that in all probability, the remote monitoring point will be available at the required time.

In summary, the decision over which method to support will most likely be governed by how important the measurements are deemed to be. For example, if a Grid middleware considers it crucial that it has new performance data available every 10 minutes, the functionality to reserve both ends of the required test paths must be available and exercised, to ensure that the required resources are available at the required 10 minute intervals. If however, the middleware operatives believe that delays in obtaining such performance data can be tolerated, then there is no need to reserve either end of a test path, and no such functionality need be provided.

A complicating factor in this decision is that at the time of writing, there are few Grid applications, middleware or monitoring architectures making requests for network data. As a result, it is difficult to estimate the frequency with which network tests are likely to be requested in Grid environments, and thus what the probability is of a measurement being delayed because of test contention. In any case, the decision is at the discretion of the service’s implementers.

2. Services must be able to discover the capabilities of other services in order to choose the best tools and parameters for measurements.
3. It must also be possible however for a service to negotiate the tools and parameters to be used in making a measurement, such that it does not interfere with other measurements. For example, a system may allow 30 second iperf tests to be run, but not during periods of severe loading, when a shorter test duration must be negotiated, dependant on the level of loading.

It should be noted that bringing existing monitoring architectures under the umbrella of this document may require services to have the ability to indicate that time and test parameters are to be negotiated via an out-of-band (i.e. non-Grid services) mechanism. An example of this would be a proprietary MP to MP protocol, such as that used by the Internet2 piPEs architecture’s [17] OWAMP tool, which has an internal mechanism for “last minute” test scheduling.

4.1.2.3 Use Cases

To be added.

4.1.2.4 Network Monitoring Service Components

In most cases it is undesirable to functionally decompose the tasks of a web or Grid service into several sub-services, the overhead of increased inter-service communication making the collective service highly inefficient. Sub-services are used in this case for flexibility, and to prevent monitoring architectures being forced to provide functionality that is either unnecessary or inappropriate. The UK e-Science network monitoring architecture [22] for example, makes regularly scheduled tests, but has no mechanism for accepting test requests. In this case the architecture would need to implement a 'results service' for providing access to collected measurements, but not a 'scheduling service'.

This approach complements the OGSI service requirement for coherence, in that each portType should accomplish one clear task only (see section 3.1.3).

Figure 8 shows the results of decomposing the monitoring service, where MPs are Monitoring Points (the nodes which make performance measurements).

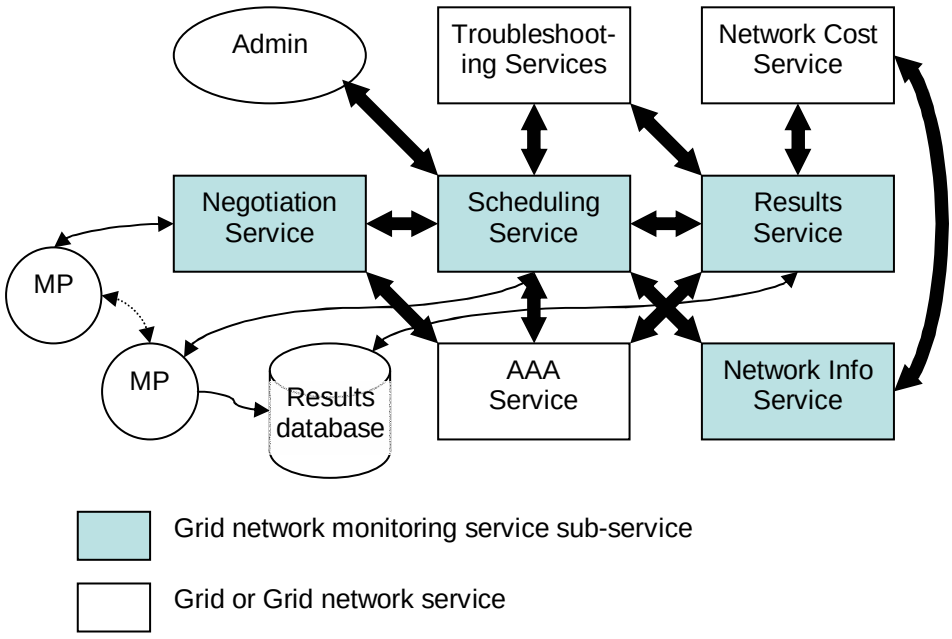


Figure 8. Network monitoring services and their interactions

A *Network Monitoring Schedule Service* provides an interface to control a single schedule of measurements, with the option of retrieving measurement results as they are made, possibly via a notification scheme. In order to make a measurement, the NMSS locates the Network Monitoring Negotiation Service Factory (see below) responsible for the sink of the measurement using a Network Information Service. The NMSS then acts as a client to the Negotiation Service. The NMSS can report the status of the scheduled measurements.

The *Network Monitoring Negotiation Service* allows monitoring services across many administrative domains to negotiate for new measurements to be made, and settings, tools and timing that would be mutually acceptable. A single NMNS corresponds to the interaction between two services making a measurement, as they agree the settings, tools and timing of a measurement. A client wishing to make a measurement locates an NMNS factory using the Grid/Network Information System, and requests an instance of the NMNS ahead of time. The client may negotiate a time slot, the appropriate tools to use and acceptable parameters for the measurement. Some measurement systems do some negotiation out-of-band, so the service must be able to indicate fuzzy guarantees and indeed non-guarantees. *Note:* The Network Monitoring Negotiation Service may be described as a specialisation of the WS-Agreement Services or an Advance Reservation Service. How this fits with the NMWG request schema work needs to be a matter for some discussion.

A *Network Monitoring Results Service* provides an interface to extract measurement results according to some query, and the option of retrieving measurement results as they are made, possibly via a notification scheme. Each NMRS handles a single query expression, but may provide data for some time as new matching data is measured. The soft-state lifetime management of a Grid Service can be used to limit the amount of time an otherwise open-ended query would take. If necessary, the NMRS could invoke the Network Monitoring Schedule Service to make up-to-date measurements if none are available, or could even add or change schedules.

A *Network Monitoring Predictions Service* provides an interface to extract predicted values of measurements if they are available. Interpolated historical data might also be counted as a prediction (i.e. “If a measurement was made at that particular time, what would its value be?”). This service responds to measurement requests in a very similar way to the NMRS, and could provide up-to-date information as new historical data is available, and so both should be derived from a common ancestor.

Clients of the Negotiation and Scheduling Services will be restricted in the level of service they are authorised to access: they might be restricted to certain values of measurement parameters, service instance lifetimes, test frequency or any other restriction. These policy-based restrictions will be handled by a separate AAA Service. Clients of the Results Service may also face policy-based restrictions that should also be handled by a separate AAA service.

4.1.3 Data Transport Service with Network Quality of Service

[FT: the original text can be found in draft-ggf-ghpn-netservices-0. Given the top-down approach that we’ve adopted, it’s premature to talk about this service in the terms that it was written in netservices-0.]

4.1.4 Network Advance Reservation Service and Resource Management Service[4]

[FT: the original text can be found in draft-ggf-ghpn-netservices-0. Given the top-down approach that we've adopted, it's premature to talk about this service in the terms that it was written in netservices-0.]

4.1.4.1 Network resources: the Path example

[FT: the original text can be found in draft-ggf-ghpn-netservices-0. Given the top-down approach that we've adopted, it's premature to talk about this service in the terms that it was written in netservices-0.]

4.1.5 Connectivity Service

[FT: the original text can be found in draft-ggf-ghpn-netservices-0. Given the top-down approach that we've adopted, it's premature to talk about this service in the terms that it was written in netservices-0.]

4.1.6 Network Cost Estimation Service[5]

[FT: the original text can be found in draft-ggf-ghpn-netservices-0. Given the top-down approach that we've adopted, it's premature to talk about this service in the terms that it was written in netservices-0.]

5 Common Interface Design Principles for Grid Network Services

5.1 Interface Design Principles

[Doan Hoang]

The section is organized as follows. First, some useful definitions of Web Services and Grid Services are put forward to set the context for discussing interface design principles (or design guidelines). Second, a Grid Service Interface is defined. Finally, some guidelines are provided.

5.1.1 Web service/Grid Service Definition

Web services are self-contained, self-describing, modular "applications" that can be published, located, and *typically (but not necessarily)* invoked using standard HTTP over port 80. Web services can perform functions which are anything from simple requests to complicated business or scientific procedures.

[4] Some of the ideas described in this section are borrowed from the GARA Advance Reservation toolkit [1] and they reflect the work developed in the framework of the IST project: DataTAG [2].

[5] The service described in this section was defined and implemented in the framework of the IST Project DataGrid [11].

The W3C Web services Architecture working group provides the following definition: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [?]. The main difference between a normal remotely-invoked application and a Web service is that the latter has an XML-based interface description that enables it to be self-describing. Once a Web service component is deployed, other applications can discover and invoke the published service via its interface.

A *Grid service* is a WSDL-defined service that conforms to a set of conventions relating to its interface definitions and behaviors. OGSA specifies three conditions for a Web service to be qualified as a Grid service. First it must be an instance of a service implementation of some service type as described above. Second, it must have a Grid Services Handle (GSH), which is a type of Grid URI[6] for the service instance. The GSH is not a direct link to the service instance, but rather it is bound to a Grid Service Reference (GSR). The idea is that the handle provides a constant way to locate the current GSR for the service instance, because the GSR may change if the service instance changes or is upgraded. Third, each Grid Service instance must implement a port called "GridService portType." This portType is analogous to the base Object class within object-oriented programming languages such as Smalltalk or Java in that it encapsulates the root behavior of the component model. The behavior encapsulated by the GridService portType is that of querying and updating against the serviceData set of the Grid service instance, and managing the termination of the instance. The portType has 5 operations:

1. *GridService::findServiceData*. This operation allows a client to discover more information about the service's state, execution environment, and additional semantic details that are not available in the GSR. In general, this type of reflection is an important property for services. It can be used by the client as a standard way to learn more about the service.
2. *GridService::setServiceData*. This operation allows for the modification of a service data element's values.
3. *GridService::requestTerminationAfter*. The request specifies the earliest desired termination time.
4. *GridService::requestTerminationBefore.* The request specifies the latest desired termination time.
5. *GridService::Destroy*. This operation explicitly requests destruction of this service.

OGSA framework demands that a service be represented as a self contained, modular entity that can be discovered, registered, monitored, instantiated, created, and destroyed with some form of life cycle management. To assist the messaging, discovery, instance creation and lifetime management functions required by a Grid service, the OGSA defines a number of standard Grid Service ports: *NotificationSource,*

[6] Universal Resource Identifier

NotificationSubscription, NotificationSink, HandleResolver, Factory, and ServiceGroup. A Grid service hence always requires a hosting environment to provide supplementary functions including Global Information Services and Grid Security Infrastructure and to ensure that the services it supports adhere to defined Grid service semantics.

It is clear from these definitions that Web services emphasize on stateless interactions and Grid services concentrate on stateful resources that must be shared and managed. Stateless interaction enhances reliability and scalability: a stateless Web service can be restarted following failure without concern for its history of prior interactions, and new copies of a stateless Web services can be created (and subsequently destroyed) in response to changing load [2]. However, to deal with shared resources within a dynamic environment, it is desirable to model resources as a stateful entity that can be discovered, shared, and managed. OGSI chooses to adopt this model. OGSI models a Grid service as a stateful entity that can be pointed to, operate upon, and managed in a manner similar to an object. The Grid service specification, however, does not require, nor does it prevent, implementations based upon object technologies.

5.1.2 Grid Service Interface definition

A Grid service's interface is defined by its service description; comprising its portTypes, operations, serviceData declarations, bindings, messages, and types definitions. A Grid service description describes how a client interacts with service instances. The description is independent of any particular *Grid service instance*. The service description is meant to capture both interface syntax as well as semantics. Interface syntax is described by WSDL portTypes. Semantically, the interface is defined in some specification documents or through some formal descriptions.

- *portType*: defines a group of input, output, and fault messages that a service is prepared to accept or produce and the message exchange patterns (operations) in which it is prepared to participate.
- *operation*: a named end point that consumes a message as input and optionally returns a message as output.
- *message*: may be composed of many parts, where each part can be of a different type. The message parts can be thought of as input and output parameters.
- *types*: defines the collection of all the data types used in the Web service as referenced by various message part elements.
- *binding*: describes the concrete implementation of message: that is a data encoding, messaging protocol, and underlying communication protocol.
- *serviceData declarations*: serviceData element definitions are referred to as serviceData declarations. serviceData elements are named and typed XML elements encapsulated in a standard container format. Service data elements provide a standard representation for information about service instances. The service data declaration is the mechanism used to express the elements of publicly available state exposed by the service as part of its service interface. ServiceData elements are accessible through operations of the service interfaces such as those defined in this specification. Private internal state of the service is not part of the service interface and is therefore not represented through a service data declaration. Since WSDL

defines operations and messages for portTypes, the declared state of a service MUST only be externally accessed through service operations defined as part of the service interface. To avoid the need to define serviceData specific operations for each serviceData element, the Grid service portType provides base operations for manipulating serviceData elements by name.

A given Grid service implementation is an addressable and potentially stateful instance that implements one or more interfaces described by WSDL portTypes. Each instance can be characterized as state coupled with behavior published through type-specific operations. Each service instance is made accessible to client applications through a global name, a Grid Service Handle, which resolves into a pointer to a specific Grid instance hosted in execution environment.

It is clear that Grid Service model share the same fundamental characteristics of a traditional distributed object model, even though a number of object-related issues are not addressed within OGSI: implementation inheritance, service mobility, development approach, and hosting technology.

5.1.3 Design Guidelines

As mentioned earlier, OGSI shares many fundamental characteristics of a distributed object system; hence it is no surprise that Object-Oriented Design Methodologies will be helpful in the design of a Grid service interface. However, we have to be mindful of the fact that object-oriented infrastructures for building distributed applications are more suitable for closed systems since they encourage tight integration of distributed components. This is one of the main reasons why many distributed object applications have failed in the past when they have had to operate across enterprises, platforms, and languages. To achieve its goal for distributed system integration, designers of Grid services should eliminate aspects that are detrimental to interoperability. Furthermore, good Grid interface design principles can be extracted from design principles of systems engineering and software engineering. Some of the general guidelines are discussed below.

Abstraction: One of the most important tasks in designing an interface is to find the right abstraction for the task at hand. Abstraction means that we can forget nasty details of some parts of the system while we concentrate on other parts of the system which do not require understanding part-details. This job is best done over a period of time and in discussion with other people. Abstraction enables us to build more complex systems.

Simplicity: Always strive for simplicity. If one can think of a simpler and clearer way to do a task, one improves the chances that all components will understand the task and how it fits into the whole system more reliably.

Loose coupling: Statelessness tends to enhance reliability and scalability. If statelessness is unavoidable, the next best property of Grid services is loose coupling. Strong dependency between a user and Grid services or between Grid services make it difficult to build open distributed systems. Loose coupling facilitates construction of complex

services. Loose coupling also implies that if an application requires access to a stateful resource, it should only deal with a Grid service that manages the resources and not directly invoke the resource. As a result, the service requestor and the Grid service manager can interact in a stateless or loose coupling manner.

Coherence: Each portType should accomplish one clear task only. If several operation are necessary, they should be closely related and access the same set of resources. For example, portTypes within a serviceGroup should be closely correlated and share the same set of resources. This also helps serviceGroup portType operations to access service data elements of the Grid service easily and consistently.

Naming: Naming of the interface, portType, service group, operation, service data element should be clear, consistent, unambiguous, and reflect the functionality and characteristics of the service. An indication that one has defined a set of cohesive portType, operations, etc., is that one can think of good names for each portType or operation because it does one task.

Form: Implementation of a service is often determined by its structure. Structure of the implementation is reflected in its form. That is to say, the form of the interface (i.e., the structure of the service XML document) plays an essential part in defining a contract between service requestors and a service. For example, service data elements (SDEs) within the service XML document allows state information to be accessed.

Information hiding:

Resuse:

Others:

6 Security Considerations

TBD

7 Authors Information

TBD

8 Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or

users of this specification can be obtained from the GGF Secretariat. The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contacts information at GGF website).

9 Full Copyright Notice

10 Bibliography

1. Foster, I.; Roy, A.; Sander, V.; A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation; Proc. 8th International Workshop on Quality of Service 2000.
2. *Demonstration of Advance Reservation and Services*, DataTAG Deliverable 2.5, Dic 2003 (https://edms.cern.ch/file/431913/1/D2.5-1.3.pdf).
3. Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W.; *An Architecture for Differentiated Service;* RFC 2475, Dec 1998.
4. K. Nichols, V. Jacobson, and L. Zhang, *A Two-bit Differentiated Services Architecture for the Internet,* Work in Progress, (ftp://ftp.ee.lbl.gov/papers/dsarch.pdf)
5. Campanella, M.; Ferrari, T.; Leinen, S.; Sabatino, R.; Reijs, V.; *Specification and Implementation plan for a Premium IP service*, GÉANT Deliverable 9.1, March 2001 (http://archive.dante.net/tf-ngn/GEA-01-032.pdf).
6. *The QBone Premium Service*, http://qbone.internet2.edu/premium/.
7. Davie, B.; Charny, A.; Bennet, J., C., R.; Benson, K.; Le Boudec, J., Y.; Courtney, W.; Davari, S.; Firoiu, V.; Stiliadis, D.; *An Expedited Forwarding PHB (Per-Hop-Behavior)*, RFC 3246, Mar 2002.
8. Heinanen, J.; Baker, F.; Weiss, W.; Wroclawski, J.; *Assured Forwarding PHB Group*, RFC 2597, Jun 1999.
9. *The Qbone Scavenger Service (QBSS)*, http://qbone.internet2.edu/qbss/.
10. Chown, T.; Ferrari, T.; Simar, N.; Sabatino, R.; Venaas, S.; Leinen, S.; *Experiments with LBE Class of Service*, GÉANT Deliverable 9.9, Aug 2002 (http://archive.dante.net/tf-ngn/D9.9-lbe.pdf).

11. *Final Report on Network Infrastructure and Services,* DataGrid project IST-2000-25182, Deliverable DataGrid-07-D7-4.
12. Ferrari, T.; Giacomini, F.; *Network Monitoring for GRID Performance Optimization*, INFN CNAF Tech. Report n. X, (http://www.cnaf.infn.it/~ferrari/papers/myarticles/comp-comm2002.ps).
13. The Network Cost Estimation Service, http://ccwp7.in2p3.fr/nces/
14. Stockinger, H.; Stockinger, K.; Harakaly, R.; Vicat-Blanc Primet, P.; Bonnassieux, F.; *Replica Access Optimisation in a Data Grid Using Network Cost Estimation Service*, submitted for publication to the Journal of Grid Computing
15. Kunszt, P.; Laure, E.; Stockinger, H.; Stockinger, K.; *Advanced Replica Management with Reptor*, in Proceedings of 5th International Conference on Parallel Processing and Applied Mathematics, 2003, Czestochowa, Poland, September 7-10
16. The OGSA Replication Services Working Group, Global Grid Forum, (https://forge.gridforum.org/projects/orep-wg/)
17. Internet2, End-to-End Performance Initiative Performance Environment System, (http://e2epi.internet2.edu/E2EpiPEs/)
18. Fox, G. ; Walker, D.; *e-Science Gap Analysis, Appendix "UK Grid Services and Activities"*, (http://www.nesc.ac.uk/technical_papers/UKeS-2003-01/Appendix30June03.pdf)
19. The Network Measurements Working Group, Global Grid Forum, (http://forge.gridforum.org/projects/nm-wg/)
20. Iperf, TCP & UDP bandwidth performance tool, (http://dast.nlanr.net/Projects/Iperf/)
21. Network Weather Service, (http://nws.cs.ucsb.edu/)
22. UK e-Science network performance measurement architecture, GridMon, (http://www.gridmon.dl.ac.uk)
23. Y. Gu, R. Grossman, "SABUL(Simple Available Bandwidth Utilization Library)/UDT (UDP-based Data Transfer Protocol)", (http://sourceforge.net/projects/dataspace)