Grid High Performance Networking Research Group	George Clapp *Telecordia Technologies*
GRID WORKING DRAFT	Tiziana Ferrari *INFN*
draft-ggf-ghpn-netservices-1.0	Doan B. Hoang *University of Technology, Sydney*
Category: Informational Track	Gigi Karmous-Edwards *MCNC Institute*
http://forge.gridforum.org/projects/ghpn-wg/	Tal Lavian *Nortel Networks Labs*
	Mark J. Leese *Daresbury Laboratory*
	Paul Mealor *University College London*
	Inder Monga Nortel Networks Labs
	Volker Sander *Forschungszentrum Jülich*
	Franco Travostino *Nortel Networks Labs*

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

Grid Network Services

1 Introduction

[Franco Travostino]

Network services are services that specialize in the handling of network-related or network-resident resources. Examples of network services are data transport service, network advance reservation service, network Quality of Service (QoS) service, network information service, network monitoring service, and AAA[1] service.

This informational draft describes how several network services combine and yield a rich mediation function—a resource manager—between grid applications and legacy networks. Complements of 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 three dominant foci of this GHPN effort are a) the relationship between network services and the known elements of grid infrastructure, b) the functional characterization of each grid network service, and c) the interplay among grid network services. The definition of any particular grid network service (e.g., in terms of 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

2.1 Conceptual Description

[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 were 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). 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).

For example, the TCP protocol already defines automated, end-to-end feedback loops at the Layer 4 of the OSI stack. These loops are necessary and sufficient to protect the network from congestion. They are not sufficient, however, to factor in application

[1] Authentication, Authorization and Accounting

requirements, policy considerations, and broker's directives properly. For these, there is a need to operate outside of any individual Layer 4 session and reason in terms of a macro-picture with multiple transport connections over multiple routes and possibly over multiple providers with different Service Level Agreements (SLAs). Hence the need for grid network services above Layer 4 and for their automated orchestration of network resources end to end.

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 capacity are continuously mediated, resulting in provisioning actions upon the network as well as the system and middleware layers at the end-systems.

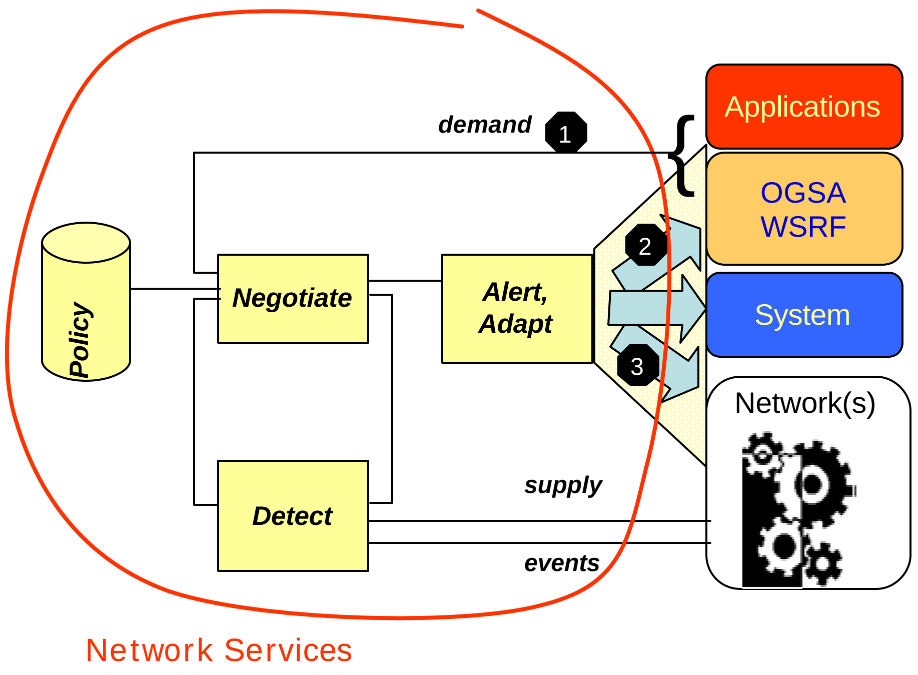

Figure 1. Abstraction of Network Services

This document focuses on the role of boxes such as Policy, Negotiate, Alert, Adapt, and Detect[3] as well as on the directed edges connecting them. For some of these boxes—the actual grid network services—a grid-friendly interface (be it WS or WSRF or JSDL) is necessary and sufficient. For some others—the more general network services—a grid-friendly interface is a sufficient albeit not necessary implementation choice (e.g., the service only interacts with other network services, and standard legacy protocols suffice).

[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.

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

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.

As a general rule, the boxes must conform to the end-to-end design principle. That is, a network service does not know whether an application is working properly or has failed. It does not know whether an application is legitimate or is a worm exploiting a security breach. It does know, however, whether an application is inside or outside of its agreed-upon SLA envelope.

The stateful boxes must also conform to the so-called fate-sharing principle. This principle argues that it is acceptable for a network service to lose its state information associate with an application if at the same time the linkage to the application is lost.

Still 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 communicate with services factors like data rate profile (time vs. rate), total data amount remaining (estimation or actual), and other characteristics of the data stream to help the network fabric to optimize and predict load, which in turn may result in greater satisfaction 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 for which it has registered. 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).

The sections that follow give specific meanings to the boxes and edges represented in Figure 1. They do so through use cases which show a distinguishing and realistic use of the network as an actively managed grid resource.

2.2 Relationship with Standards

2.3 Case Studies

[Franco Travostino]

2.3.1 High Throughput File Transport

2.3.2 Baseline High Throughput File Transport

2.3.3 High Throughput File Transport with a Scheduled Connectivity Service

[Tal Lavian]

Network elements such as routers/switches, end devices, and physical links are essential for creating connections between end users. A Grid network infrastructure is essentially an overlay network over physical networks for connecting end systems belonging to a Virtual Organization. Connectivity between end systems is thus an essential resource that glues the infrastructure together.

With packet switched networks, connectivity is assumed always available under best-effort service and statistical multiplexing. A connection is never really denied but quality of the connection degrades progressively depending on the traffic conditions at the time of the connection. With most telecommunications networks (circuit-switched), a connection can be denied or blocked if its QoS requirements are not met or network resources are not available.

To provide a degree of stability and predictability for Grid applications connectivity should be treated as a scheduled service, a Grid Scheduled Connectivity Service. With Grid Scheduled Connectivity Service, Grid applications can utilize a WS-Agreement service to negotiate connectivity resources that satisfy their QoS requirements before their actual deployment. Furthermore, Grid Scheduled Connectivity Service allows network resources to be utilized flexibly and efficiently.

Connectivity services can be classified into several types depending on the specific type of service provisioning. DS-MPLS, pure DiffServ, and Lightpath provisioning, to name a few. In a dynamic optical network environment, connectivity between end systems can often be established by concatenating of lightpath segments between two end points. In a DiffServ network, connectivity can be established by concatenating logical hops of the same class or DS codepoint.

A Grid Scheduled Connectivity Service is just like any other Grid service; it has to expose its service through a Grid interface. For example, an interface for requesting lightpath connectivity should be rich enough to allow applications to express their own flexibility via under-constrained (or loose-constrained) requests. This allows for optimal scheduling, and for automatic rescheduling if necessary. The scheduling connectivity service considers the flexibility in the requests, the flexibility inherent in any conflicting current reservations, and other factors such as job priorities or predictive load balancing. It provides guarantees or advance reservations for channel availability between specific endpoints, with certain constraints. The reservations may be periodic, and may be many days in advance. They may be policed by various resource reclamation policies, such as periodic application resiliency requirements. When rescheduled within the bounds of the under-constrained request to meet new requests, these changes are reported via a middleware query or notification mechanism.

Once WS-Agreement service is created for the connectivity resource, higher level scheduled services such as data transfer, storage, computation, instrumentation and visualization can be activated to support an application.

A File Transport service can be scheduled between end systems by deploying a scheduled storage service and a Grid connectivity service. Often data-intensive applications require transfer of a massive amount of data sustained over a considerable period. For this reason, Grid Scheduled Connectivity service is essential, and some form of high throughput transport protocol us required. GridFTP is often used to provide high throughput file transport. Alternatively, an optimized transport protocol like SABUL/UDT [23] can also be used.

2.3.4 High Throughput File Transport with a Deadline

[Volker Sander]

A particular challenge that arises in Grid infrastructures is the coordinated use of multiple resources. Here, workflows with potentially complex interdependencies have to be mapped to a distributed environment. A grid network service that assures the local existence of remote data could be used in workflow management frameworks to synchronize the coordinated use of resources and thus to avoid unnecessary blocking times due to missing staging data.

GridFTP was proposed to achieve an extremely high throughput. It is based on parallel streams and implemented by striped TCP sockets, as also applied by many Web browsers. Using several TCP streams in parallel circumvents TCP congestion control to some extent. Each of the individual TCP connections performs congestion control by adapting its data rate. Thus each TCP connection on its own can be considered to be TCP-friendly, whereas the sum of the rates achieved with parallel TCP connections is higher than the rate of a single connection. In order to increase fairness towards applications that use only a single TCP stream, GridFTP should ideally apply a Scavenger Service. On the other hand, as stated above, Grid computing often requires that data is delivered fulfilling certain deadlines. Thus a reservation of network capacities is required, which can be addressed with a Guaranteed Rate Service that assures a certain bandwidth. As a consequence, deadline file transports will probably rely on both: a Scavenger Service for using unused bandwidth and a Guaranteed Rate Service that assures a negotiated level of service. Of course, the challenge of effectively using the guaranteed rate remains. Pacing the low-level traffic by using traffic shaping mechanisms has been approved as an appropriate solution to this.

While the user – either an end-user or a high-level service such as a super-scheduler – could calculate the required bandwidth to meet the deadline, the question arises whether a Grid Network Service should provide more advanced solutions, particularly in the context of aggregated bandwidth reservations that were partly unused. Unused aggregated reservations are a potential waste of resources, particularly in a Layer 1 optical network. However, also a shared Layer 2 / Layer 3 infrastructure could still benefit from a service concept that automatically adapts the assigned guaranteed rate to the available rate (that is typically limited by an SLA term). In using unused resources in multiple service classes while only reserving the bandwidth required to meet the deadline, the load injected to value-added services is reduced and admission control procedures that decide about the access to advanced services receive more flexibility for their decisions.

2.3.5 High Throughput File Transport with Optical Bypass

[Tal Lavian]

Optical Bypass for IP traffic is an alternative mechanism to off load capacity from the L3 routers, in case the required capacity exceeds the available router capacity. It does this in the edge device, without requiring application-level changes to end-systems. Once router traffic grows beyond a certain point, IP becomes inefficient and optical bypass is a preferred method for high throughput file transport. While IP services are best suited for the many-to-many paradigm, optical bypass service is ideal in the few-to-few occurrences of large data transfers for data-intensive applications.

The development of optical transport has brought a huge supply of network bandwidth, while the cost per bit is about one order of magnitude more expensive for IP traffic than for the optical transport. Optical Bypass is a network service that sets up an L1 optical shortcut directly between designated end-points and directs data traffic over the shortcut to bypass the IP cloud. An optical shortcut, for example, a point-to-point wavelength, provides a big bandwidth pipe for high throughput file transport in Grids. Grid applications can utilize WS-Agreement to negotiate with Optical Bypass service to satisfy the requirements and network characteristics of high throughput file transfer. It is assumed that the bypass negotiation will be based on some level of policy and AAA.

An optical shortcut is neither a fixed optical network nor a leased optical link. It is dynamically created to satisfy high throughput data transfer. It is torn down when the transfer is ended. A Grid application first requests a large data transfer, a middleware service then sets up the optical bypass, and then traffic flows via the optical network instead of the IP cloud. The data transfer can be transparent with no changes in the applications, end-systems, or the IP network. The new bypass capabilities are in the edge devices. The Optical Bypass service mechanisms and the edge device must conform to the end-to-end design principle and the fade-sharing design principle, and must not add any requirement on the application side. Control plane, optical network intelligence, and interaction with the Grid middleware is required to determine the right path out of the edge device, namely the IP cloud or the optical bypass. The optical control plane needs to discover, identify and set the lightpath.

In some cases, the lightpath is provided by a different entity, administrative domain, or via the fibers owned by the user; hence, the data transfer bypasses the public network. With the help of intelligent network services in both Grids and optical networks, an optical bypass will be set up with sufficient bandwidth between the source and destination to support the requested transfer. Moreover, the data traffic of the file transport is routed via the edge device to the optical bypass. Once the file transport completes, the optical bypass is released or redirected for other Grid uses.

2.3.6 Visualization Session

[Inder Monga]

Visualization is one of the key methods used to represent data (raw or processed) and is used extensively by almost all fields of specialization for instance e-sciences, medicine, engineering and digital art. A visualization session can be trivial utilizing data-sets available locally or require significant amount of grid resources (including from the network) for applications like collaborative virtual-reality, distributed CAD, tele-immersion, distributed simulation analysis and haptic collaborations. This use-case refers to the network services required for the latter set of compute and data-intensive sessions. For visualization and collaboration to be achieved with good performance, low latency and jitter are the two most important parameters needed from the network service to ensure the different media streams are delivered within the specified delay constraints.

An sample workflow of a non-real time visualization session includes receiving raw data from one or many sources, processing it over a grid infrastructure and then streaming it to one or more remote locations for visualization. Depending on the quantity of raw data to be transferred, a high-throughput data transport service with or without guarantees will be required by the application. In order to view the final data as visualization in real-time, a guaranteed bandwidth service with a low-latency QoS from the network is needed.

In a collaborative visualization scenario, real-time data is distributed between a subset of multi-point to multi-point clients. The scenario typically consists of multiple data, audio, video streams with possibly strict delay constraints between the streams and possibly varying network QoS requirements per stream. In order to provide a flexible collaborative session, the control of this session needs to be passed between the sites while ensuring the response of the visualization session at each site feel instantaneous. This particular scenario requires the ability to request and get a low-latency network service between the participating locations. The network requirements will probably change from the initial rendering to subsequent changes to the visualization session due to collaboration, and thus the network must support the requesting application varying the QoS requirements from the network through the life of the session.

On a broader level, collaborative and visualization grid sessions might be pre-scheduled or held spontaneously. The network services should provide advanced reservation service as well as support on-demand requests. These sessions may also be configured to be one to many (broadcast) or many-to-many (collaborative). In the former case, the bandwidth requirements may not be symmetrical, thus requiring the network to provide a service that allows differentiation of QoS requirements including bandwidth allocated uni-directionally.

2.3.7 Point-to-Multipoint Session

2.3.8 Transparent Optical Channel

[Gigi Karmous-Edwards]

Having an all-photonic network connection provides the following advantages: i) a unique capability where only the two end-point transceivers need to understand the format, protocol, data rate, etc. of the data transmitted, ii) low latency across the network (assuming application level latency and jitter requirements are handled at the edges) due to no OEO and no buffering, iii) no OEO resulting in reduced network CAPEX and OPEX. Although there are many benefits to not having OEO in an end-to-end connection, one area that will require more attention than connections with OEO is the degradation of signals due to physical layer impairments. As signals travel longer distances without OEO regeneration, the accumulated effects on BER will increase. Therefore, physical layer optical monitoring becomes more critical in an all-photonic network for connection SLA assurance and fault detection. It can be concluded that a Grid network service providing an all-photonic connection should interact closely with a grid service which provides optical physical layer monitoring information on a per channel basis.

2.3.8.1 Grid Service Scenarios for All-Photonic End-to-End Connections

Having a grid service which can provide an all-photonic end-to-end connection may provide capabilities that are of interest to the Grid community. Today, the majority of data transfers within the Grid community involve large file transfer between sites using IP applications like GridFTP. However, other more latency sensitive applications are starting to appear more in the Grid community, i.e., remote visualization steering, real-time multicasting, real-time data analysis and simulation steering. Collaboratories analyzing a data set from remote instrumentation may be inclined to send raw digital data across the network via an all-photonic connection to remote locations where processing of data can be done. This will require compatible transceivers at the end points, and the network will be completely unaware of the contents of the transmitted payload. Other usage examples include:

- Raw data sent from instrumentation to remote processing systems
- Non-IP applications (e.g. HDTV) as well as IP applications
- Analog data
- Medical imaging data

2.3.8.2 All-Photonic Grid Network Service Concepts

It is assumed that a WS-Agreement with an end user has occurred during the first phase of establishing a Grid network service and that policy matters such as AAA and pricing for the different QoS levels have been negotiated.

The grid service shall provide the following operations for Grid applications:

- Is the destination address is reachable via all-photonic connection?
- Can the all-photonic connection to the destination meet the minimum requested BER?
- Is an end-to-end connection to the destination is available
- Provide a sign-up for a push notification service from Grid network monitoring services, e.g., violations of SLAs.

Potential input parameters of interest for such a service may include:

- DestinationAdress
- wavelength
- QoSData
  - Minimum BER
  - Restoration times
  - Priority and pre-emption
- Bandwidth

- Duration
- DataSize
- Protocols

2.3.8.3 Issues and Questions[4]

The establishment of a Virtual Organization over the Grid (VO/Grid) may involve the following: multiple network service providers, multiple network technologies (e.g., SONET and all-photonic networks), and multiple signaling protocols (e.g., GMPLS, UCLP, and JIT [5]), and multiple control planes for one end-to-end connection within a Grid. Each provider will have an agreement with their individual Grid members (GUNI-agreements), and these providers must also have agreements with each other (G-NNI-agreements). Some Grid providers may not even be aware of Grid members; for example, a transit domain may interact only with other service providers. This leads to the following questions:

- Will only the access (GUNI) provider to an individual Grid member be involved in that user's GUNI agreement?
- Will unsolicited GUNI notifications only reach a Grid member from their perspective access (GUNI) provider?
- Will a Grid network service have an instantiation for each client or for each Grid/VO?
- Will there be a common policy repository that includes the individual and common "rules" per VO/Grid?
- If a Grid has a network quality monitoring service running, will it be responsible for the entire Grid or will there be an instance per client connection or service/GUNI agreement?
- Will the Grid monitoring service get feeds (quality monitoring information) from each of the domains as necessary?

The following is a suggested starting point:

- We introduce an End-to-End Agreement Provider that is distinct from a Network Service Provider, and we may have non-GUNI related events that trigger the End-to-End Agreement Provider and as a result, the user might be informed.
- There might be a common policy repository for the VO; this is first per domain (a domain might serve multiple VOs), and there could be an end-to-end agreement provider that composes.
- The End-to-End Agreement Provider serves a particular VO; policies that are associated with this could either be tunneled (i.e., transported by GUNI) or shared via some VO specific policy repository.

[4] This text is based on an email conversation that Volker initiated.
[5] GMPLS: Generalized Multiprotocol Label Switching; UCLP: User Controlled Lightpath Provisioning; JIT: Just In Time.

3 Requirements and Definitions of Grid Network Services

3.1 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.

3.1.1 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.

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:

[6] Universal Resource Identifier

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, 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.

3.1.2 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.

3.1.3 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:

Please comment whether we are on the right track. Please make further contributions. Doan

3.2 Related Standards

3.3 Service Definitions and Requirements

[Tiziana Ferrari]

The following diagram tries to represent the interaction among the Grid network services currently described in this draft and with other general-purpose Grid services.
[Note: the diagram is incomplete and needs to be updated/modified as soon as the currently missing services are specified]

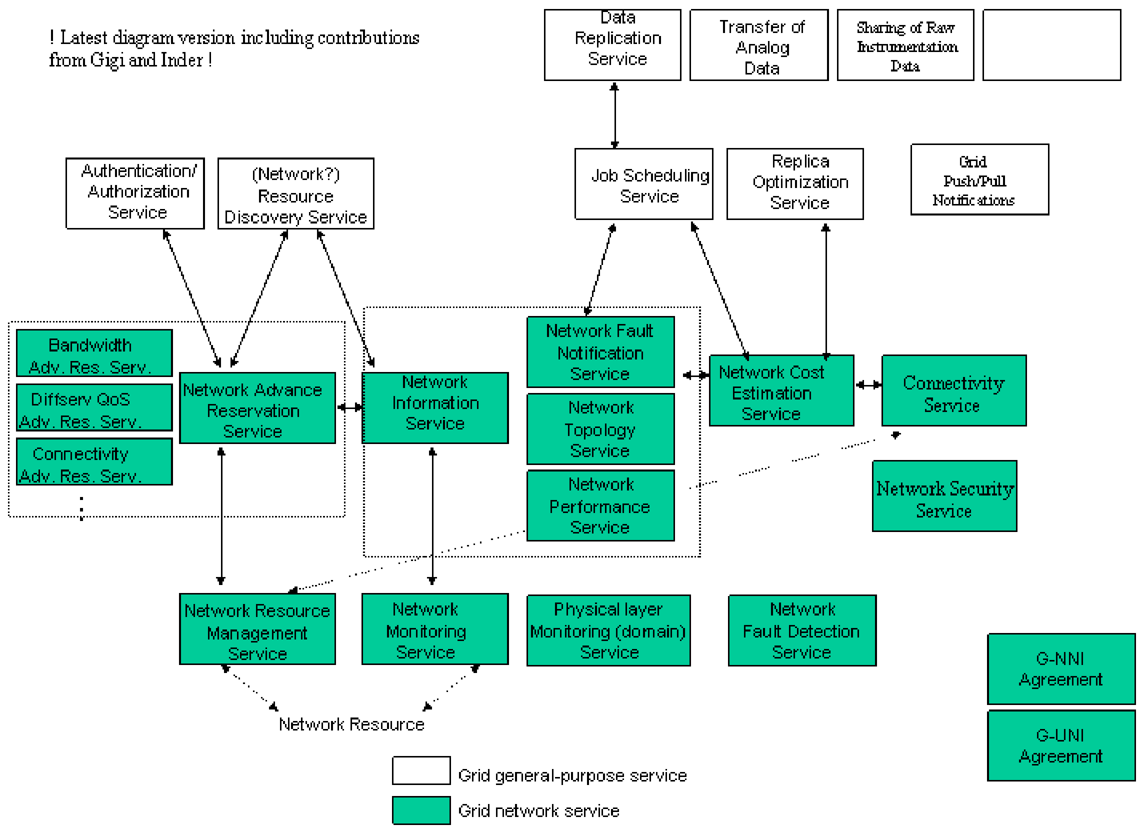
! Latest diagram version including contributions from Gigi and Inder !
Data Replication Service
Transfer of Analog Data
Sharing of Raw Instrumentation Data
Authentication/ Authorization Service
(Network?) Resource Discovery Service
Job Scheduling Service
Replica Optimization Service
Grid Push/Pull Notifications
Bandwidth Adv. Res. Serv.
Diffserv QoS Adv. Res. Serv.
Connectivity Adv. Res. Serv.
Network Advance Reservation Service
Network Information Service
Network Fault Notification Service
Network Topology Service
Network Performance Service
Network Cost Estimation Service
Connectivity Service
Network Security Service
Network Resource Management Service
Network Monitoring Service
Physical layer Monitoring (domain) Service
Network Fault Detection Service
G-NNI Agreement
G-UNI Agreement
Network Resource
Grid general-purpose service
Grid network service

Figure 2. Relationship between Grid Network Service and general-purpose Grid services

3.3.1 Data Transport Service with Network Quality of Service

[Tiziana Ferrari, George Clapp, Doan Hoang, Volker Sander]

[Comment (Tiziana): I have moved the content of that I originally intended to put here into section 3.3.2.2 under the paragraph "DiffServ QoS Advance Reservation Service." This reflects my understanding of the "Data Transport Service with QoS." But of course we can restructure section 3.3.2.2 in order to have a section for each adv res type described there. In any case, the other people who expressed interest in contributing here may have different views. So please comment!]

3.3.2 Network Advance Reservation Service and Resource Management Service[7]

The *Network Advance Reservation Service* allows the user to negotiate and claim specific network capabilities for a specified time interval. An advance reservation request contains:

- The full characterization of the resources needed through a set of resource-specific attributes (*Resource Type, Domain-specific Attributes, etc)*;
- the time information needed to identify the time span of the resource usage (*Start Time, Duration, End Time, etc*);
- Additional run-time information (provided at a later stage).

An instance of the **Network Advance Reservation Service** (NARS, in what follows) can be modeled as the instance of a more general Advance Reservation Service, where its domain-specific agreement terms provides a high-level specification of the type of resource that the reservation service instance supervises. In our case the domain-specific agreement terms where related to network services.

A resource acquisition process immediately follows the user-application request. The first phase of the process can be performed by relying on the Network Resource Discovery Service (a Resource Discovery Service instance of type network), which discovers the (network) resources that match the requirements and preferences specified in the request, and returns them in a list to the NARS instance. Note that this step is likely to be community specific, i.e. the Network Resource Discovery Service uses its knowledge about the composition of the Virtual Organization and thus serves as a community broker. At this point, the second phase starts and the NARS performs the allocation on the best-matched resource.

The Network Resource Discovery Service queries, either directly or indirectly, the resources to know their current status and availability by accessing the *Grid Information Service*. As the two-step allocation procedure is expected to frequently fail (for example when a proper Network Resource Discovery Service instance of Grid Information Service instance is not available), the NARS should be designed and implemented to be

[7] 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].

resilient to such failures. The Grid Information System offers vital information in a number of reservation phases, in fact it provides not only the list of resource instances that satisfy the user's requirements, but also for each instance, the information about its properties and the handle of the corresponding authentication/authorization service instance.

The user application who acts as agreement initiator, i.e. who issues an advance reservation request, has to be authenticated and authorized on the basis of a set of policy rules. Authentication and authorization can be performed by relying on a separate *Authentication and Authorization Service***,** which can be implemented for example according to a generic AAA Architecture. An authorized request can be satisfied only if a resource that satisfies the user's requirements is available in the specified time frame. The related agreement terms are then called to be "observed".

The actual resource allocation can be performed by a direct service invocation model where the end-system is part of the control plane signaling. However, this service invocation model requires appropriate authorization mechanisms to be included in the control plane signaling. The alternative approach that nicely fits to the Grid environment is an indirect service invocation model. Here, claiming an agreed service is performed by invoking the **Network Resource Management Service** instance coupled with a network resource element, which hides the complexity of the resource-specific allocation tasks. Different service instances are available depending on the specific type of of service provisioning. DS-MPLS, pure DiffServ, and Lightpath provisioning are some examples of Resource Management Service types.

Each resource element is handled by a single Resource Management Service instance, and a service instance can control one or more resource elements. Multiple Resource Management Service instances can run in a given domain, e.g. services such as a guaranteed bandwidth can be build by multiple technologies. . In the multiple administrative domain scenarios, more resource managers are needed to deal with the reservation of an end-to-end path. Again, two different models exist to implement end-to-end reservations. The control plane signaling approach relies on a consistent end-to-end signaling in the control plane. Here, signaling messages have to be intercepted for authorization and further policy decisions. The other approach is following the concept of composed agreements. Here, a particular end-to-end service of a specific community would perform the negotiation with the individual NARS, and, following the indirect service invocation model, it would also contact the individual resource management service instances for service allocation. The Resource Management Service instance keeps track of the use and the reservation of a given resource over time.

The NRES is a client server of the Network Information Service, as information about the actual performance experienced by the application which performed the advance reservation request, should be continuously notified back to it.

The interaction between the NRES and the related services is illustrated in Figure 3.

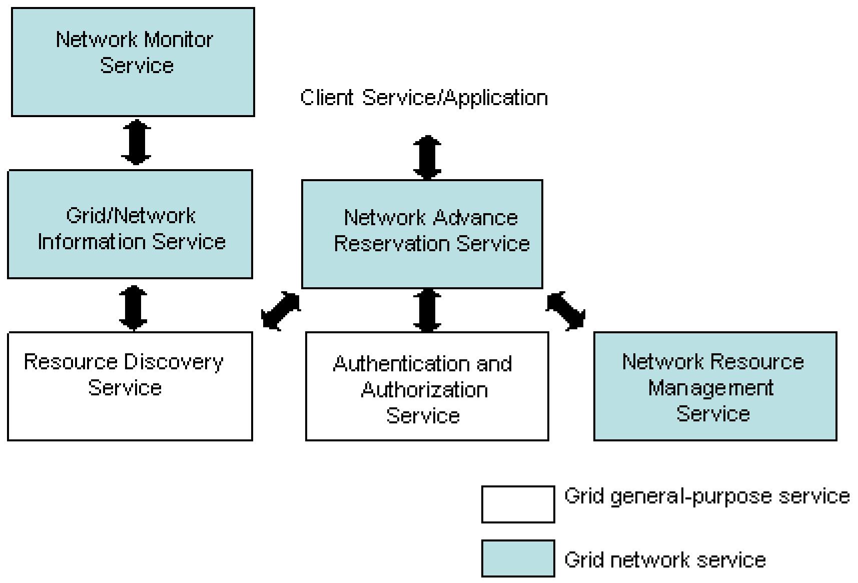
Network Monitor Service
Client Service/Application
Grid/Network Information Service
Network Advance Reservation Service
Resource Discovery Service
Authentication and Authorization Service
Network Resource Management Service
Grid general-purpose service
Grid network service

Figure 3. Network Advance Reservation Service: relationship with Grid network services and other Grid general-purpose services

At the start of the reservation time, the client can use the reserved resource to perform the task if the reservation has been granted; for example, a network reservation can be used to initiate a data transfer. Of course, agreement terms can be renegotiated with the Advance Reservation Service, eve non previously granted reservations.

3.3.2.1 Network resources: the Path example

The definition of Network Advance Reservation Service relies on the abstraction of the *Network Resource*, which is shared or exclusively allocated to a service client and the element the Network Resource Manager acts upon. In a general scenario, several types of network resource can be handled and new types can be defined at any time according to the evolving characteristics of the network.

Network resources can be dynamically discovered by the Grid Resource Discovery Service. Information about the network resource elements is provided by the Grid Information System. In the following we provide an example of network resource abstraction that we call *Path*.

Path is the name that defines the general network entity providing unidirectional connectivity between a source and destination network node. The network node can represent a single device (e.g., an end-system, router, or switch) or a network domain (e.g., an Autonomous System, IP network, or LAN). A path can cross one or more administrative boundaries, and it offers a specific packet treatment service that is described by a set of parameters providing information about its characteristics (e.g., bandwidth, packet loss, and one-way delay statistics). The monitoring of the performance experienced by traffic crossing a given path is implemented by the Network

Monitoring Service (Section 3.3.4) and the corresponding information is available from the Network Information Service (Section 3.3.3).

Paths connecting domains can be *collective*. In this case, a set of multiple users can share it where traffic sources and destination belong to the source domain and destination domain respectively, which characterize the collective path (the path connects sets of hosts rather than a single source-destination couple). On the other hand, *on-demand single-user* paths are only visible to the requesting application. A given couple of network domains can be connected by zero, one, or more collective paths.

Each Path is managed by a specific *Network Resource Manager*, which depends on the path type, and is controlled by a specific Authentication and Authorization Service instance. The path is described by attributes such as: the Path ID, the type, the handle of the corresponding Authentication and Authorization Service instance, the source/destination, and the set of path-specific parameters describing its properties, such as the maximum bandwidth. The source/destination can be a node or a domain.

The Path can be end-to-end or per-domain. End-to-end connectivity can be provided by an end-to-end path or by a chain of per-domain paths[8] (each holding its own ID). DiffServ, MPLS, and Lightpath are possible path types. Typically, the DiffServ Path and an MPLS Label Switch Path are end-to-end (in fact, they require dynamic configuration only at the path head-end node/domain), while a Lightpath constitutes of an end-to-end chain of per-domain paths. A per-domain path may consist of one or more Path Elements identified by their resource manager. In general, a path has to be viewed as a chain when the configuration and/or the bandwidth reservation on the path requires interventions in each transit domain.

Paths can be allocated statically or dynamically. A static path is manually configured by a network administrator, and users can reserve a portion of its residual bandwidth. In the dynamic path case, on the other hand, it is the user who requests the configuration according to the specified requirements. Static paths are typically needed to provide QoS support to a community of users as they contribute to minimize the path set-up overhead.

3.3.2.2 Network Advance Reservation Service types

The type of path advance reservation request that a server client can express depends on what a specific path resource offers, for example *bandwidth*, a given *differentiated services packet forwarding behavior* [3], *physical connectivity* between two nodes, or other.

In the first case, bandwidth is one of the main path parameters, and it defines the *maximum bit transmission rate* allowed, and users can request the exclusive use of a fraction of this rate during a given interval. The path can be characterized by other parameters such as the maximum traffic burst length allowed. At any point in time, the path *residual bandwidth*, i.e., the remaining available bandwidth that can be allotted to

[8] An end-to-end path crossing a single domain is equivalent to a per-domain path.

users at a given point in time, can be computed by the resource management service. The NARS offers a **Bandwidth Advance Reservation Service**.

If a particular Layer 3 IP packet forwarding behavior is of interest, then the client application may specify the packet treatment required in terms of, for example, minimum, average, or maximum one-way delay allowed, minimum, average, or maximum packet loss probability allowed, or through any other combination of packet forwarding performance metrics. The resource management service associated with the NARS discussed in this case can be based on the *Differentiated Services* architecture [3] and we call the resulting service the **DiffServ QoS Advance Reservation Service.**

For example, a client application may request a given amount of traffic to be treated so that the average packet loss, one-way delay, and packet delay variation are minimized (or guaranteed to vary in a specified range). In this case, a suitable path offering for example the Virtual Leased Line (VLL) service (also known as Premium service [4,5,6]) – based on the Expedited Forwarding Per Hop Behavior [7] – could be used to satisfy the request. Similarly, guaranteed rate-based services can be supported through the Assured Forwarding DSCP [8] and the configuration of rate-based markers. *Less Than Best Effort* (also known as *Scavenger*) is another known DiffServ service used in some production networks [9,10] that could be requested by client Grid applications.

Another type of resource that can be requested is the *connectivity* between some network nodes and the corresponding NARS is the **Connectivity Advance Reservation Service**. Connectivity is a general resource type that can be provided through various technologies at different OSI protocol stack layers. For example, Layer 1 connectivity can be provided through the on-demand configuration of a lightpath that is cross-connected in an optical network. Similarly, an extended Ethernet VLAN can be configured to provide Layer 2 connectivity through the configuration of MPLS LSPs that perform the encapsulation of IEEE 802.1Q frames into MPLS packets. MPLS is one of the protocols that can be used for the dynamic configuration of Layer 3 VPNs, which are another important example of connectivity resource that could be requested by a client application.

[Comment (Tiziana): the paragraphs above about advance reservation service types could be developed further if there is consensus on this approach. In any case, the relationship with the "Connectivity Service" that George should describe in this draft, has to be understood.]

3.3.3 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?

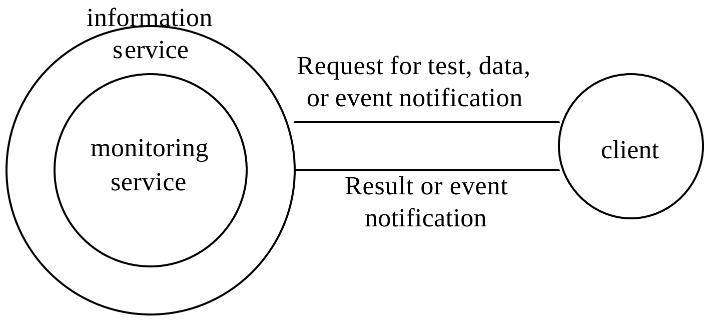
information service
monitoring service
Request for test, data, or event notification
Result or event notification
client

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.

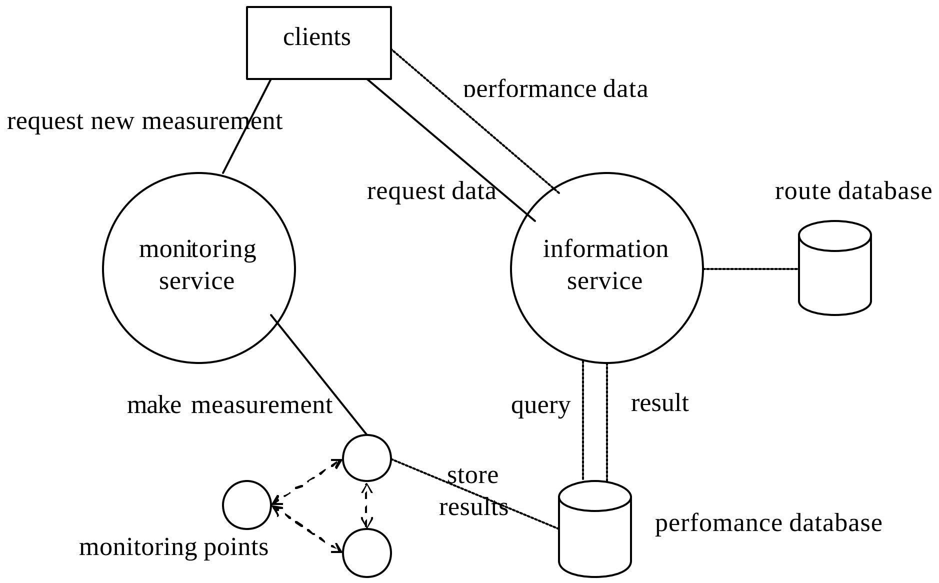

Figure 5. monitoring service for tests, information service for data

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]

3.3.4 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.

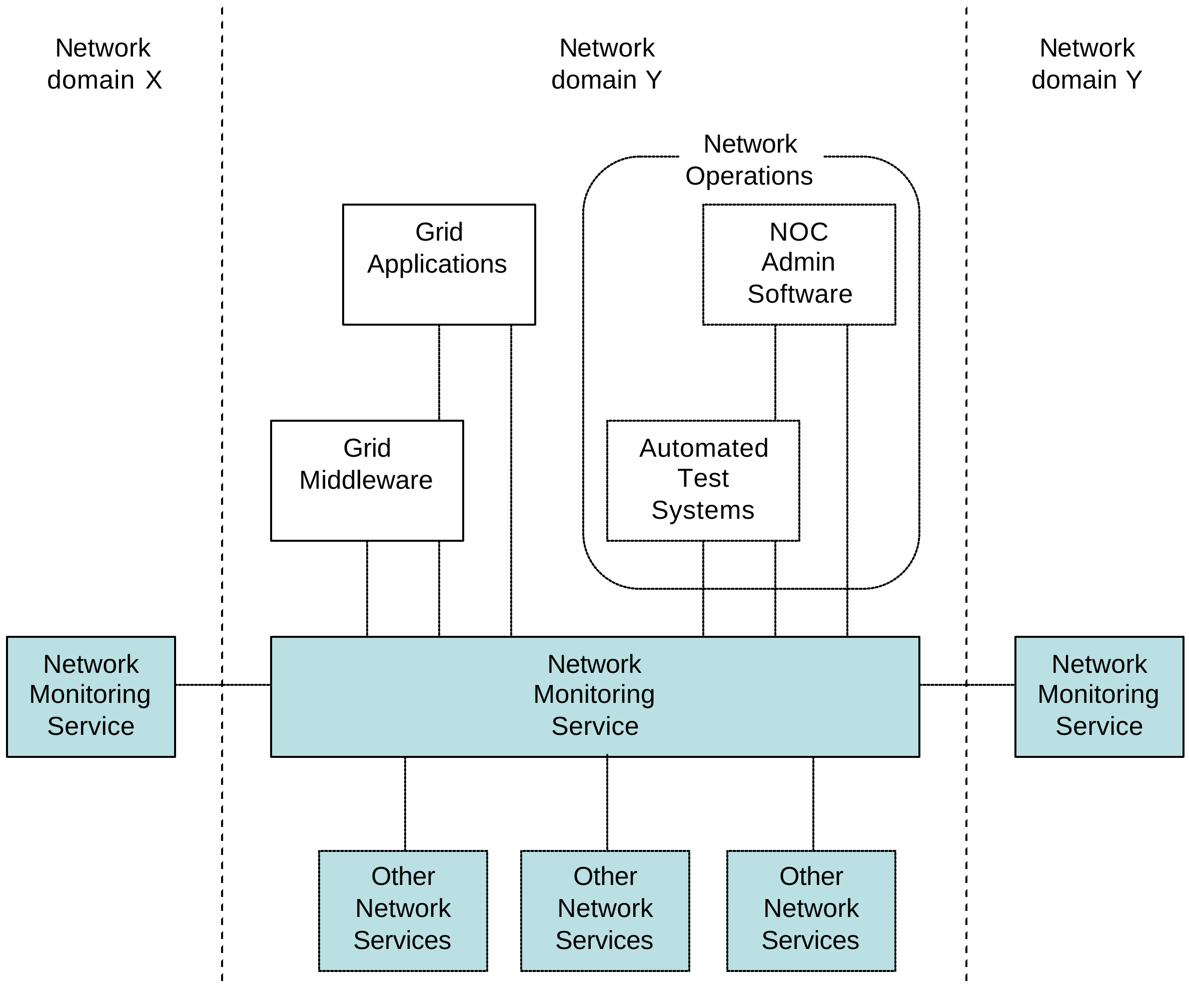

Figure 6. Clients of the network monitoring service

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.

3.3.4.1 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 <u>may</u> 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.

- o 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.
- o So far, we have discussed the monitoring service as a single entity. It is entirely possible that it will decompose into several sub-services.
- o 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:

- o 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.
- o 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).
- o Interim communication, that taking place between a request being made and a result being returned, is yet to be addressed.
- o 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.

3.3.4.2 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.

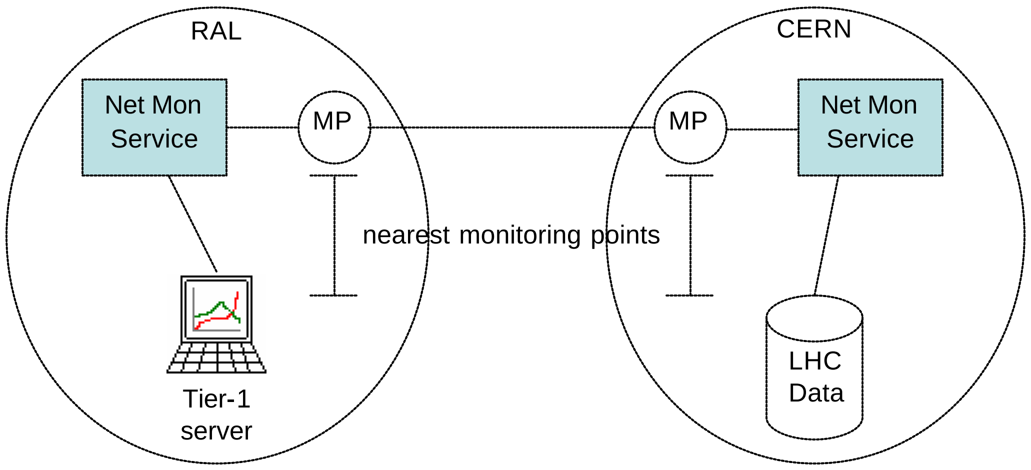

Figure 7. Monitoring point discovery

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.

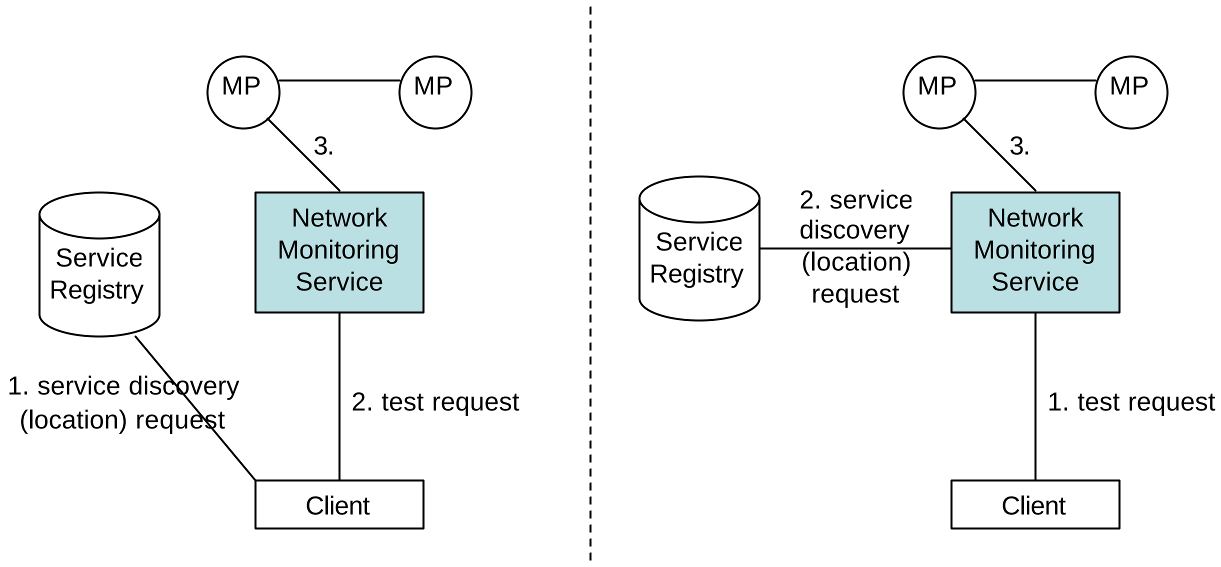
MP
MP
3.
Service Registry
Network Monitoring Service
1. service discovery (location) request
2. test request
Client
MP
MP
3.
Service Registry
2. service discovery (location) request
Network Monitoring Service
1. test request
Client

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 <u>exactly</u> 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.

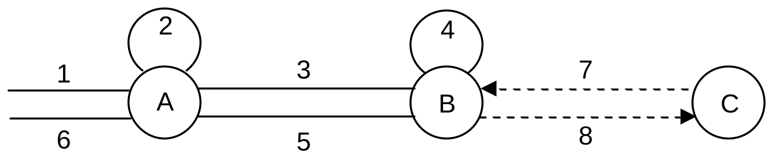

Figure 9. Test path end node reservation

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.

3.3.4.3 Use Cases

To be added.

3.3.4.4 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).

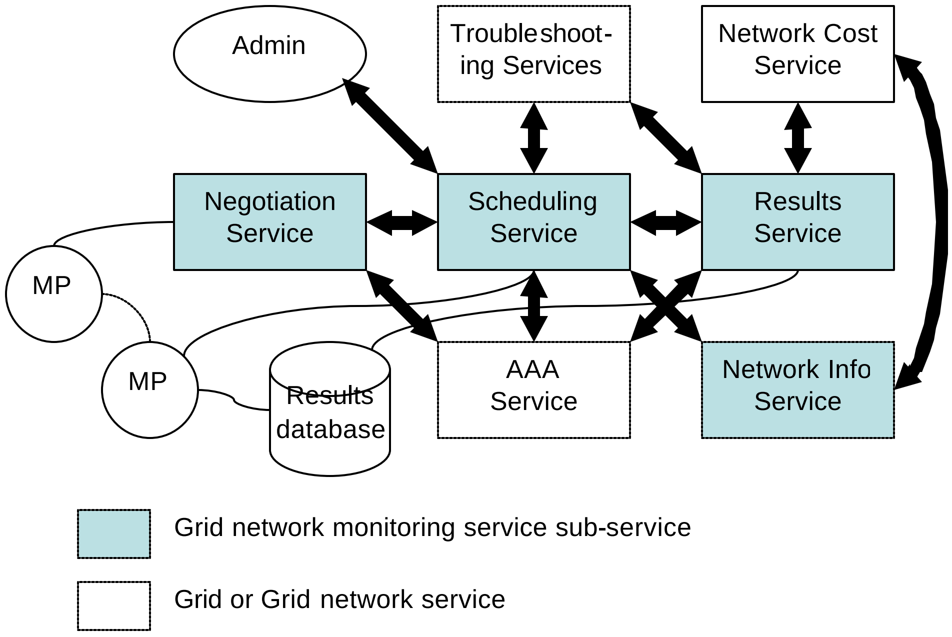

Figure 8. Network monitoring services and their interactions

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.

3.3.5 Connectivity Service

[George Clapp, Inder Monga]
Connectivity stands for the notion of one network element having the ability to exchange information with a number of other network element that it can reach. Connectivity thus includes the concepts of reachability along with methods associated with establishing connections to establish that reachability. For example, a personal computer with a telephone modem and a phone number to an Internet Service Provider cannot reach any other node on the internet unless it establishes the connectivity by dialing and authenticating with the service provider. On the other hand, a personal computer with connectivity into the internet may not necessarily be able to reach and communicate with another personal computer also connected to the internet, because of a network fault partitioning the network or due to security settings causing the other computer to be on separate virtual network.

This service is accessed either directly through the Grid Services or accessed by the other advanced network services like Network Advanced Reservation Service (NARS).
The two sections below go over further detail on these two services:

{Note: Testing for rechability may become the part of Network Information Services?}

3.3.5.1 Reachability Service

Reachability Service is a service instance queried by the clients to determine reachability to remote grid resources being selected. For example, before the client kicks of a GridFTP session it can use the Reachability Service to test for connectivity between the end points. If the results of the reachability request is not positive, then the client can look into dynamically establishing connectivity between those two endpoints through mechanisms existing in the network at a different layer.

3.3.5.2 Connectivity Establishment Service

Connectivity Establishment Service uses mechanisms existing in the network to provide reachability between two network elements. Options include

- establishing Layer 1 connectivity using GUNI requests and GNNI protocols, and/or
- establishing Layer 2 connections using GMPLS control plane protocols and/or
- configuring both elements to belong to the same virtual network by configuring them to be a part of the same VPN (Layer 1, 2..n)

3.3.6 Network Cost Estimation Service[9]

The Network Cost Estimation Service [11,12,13] allows the Grid to enhance its performance through the use of information on the status and transmission behavior of its network links. It provides an estimate of the transmission quality between two or more Grid nodes as specified in the query, and it can be accessed by any existing Grid service or application. Through the Network Cost Estimation Service, Grid services have the possibility to use monitoring information for dynamic adaptation to the Grid status at any given time.

The Network Cost Estimation Service instance is created on request of the client application, and it can be destroyed after that the requested network cost is computed (through either an explicit destroy operation or a soft-state approach). The invoking service client specifies the internal *cost model,* which defines the *cost function* the service will use for the computation of the cost. The cost model is specific to the service client and it expresses its cost vision [14]. The Network Cost Estimation Service relies on the availability of raw information on network performance provided by the Network Information Service and collected by the Network Monitoring Service, and it produces a high-level view of network performance through an internal *cost functions*. The cost function is defined according to the cost model of the application. Different service instances can be created. The cost function uses as input a set of basic network metrics and produces a compound high-level metric.

For example, a *Replica Optimisation Service* (ROS) [15] can be defined to support the Data Replication service [16] for which the cost model assumes the cost to be expressed as the time needed to transfer a file of known size between two given end-points. The estimate of this cost may be used to identify the data storage node that offers the minimum retrieval cost of a requested file copy or the optimal file transfer cost if new data items have to be created. In this example the network cost function used by the Network Cost Estimation Service instance is based on the available TCP/UDP bandwidth, the packet loss, and the Round Trip Time metrics. These metric values are supplied to the service instance at creation time as input parameters. The cost model of

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

interest is specified in the form of a service creation parameter when the Network Cost Estimation Service instance is created by the invoking client.

Other application scenarios of this service are Resource brokerage, Data replica management, and Remote file access.

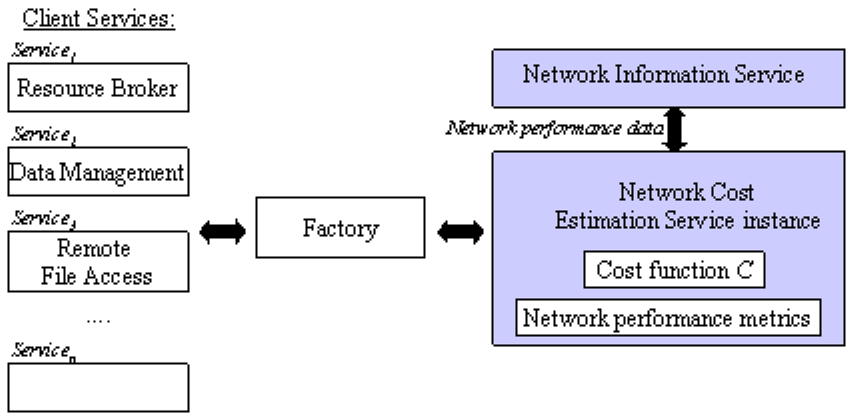

Figure 4. The Network Cost Estimation Service and its application

Service application scenarios

Several different application scenarios of the Network Cost Estimation Service can be envisaged according to the number and type of Grid services available in a given distributed computing platform.

The integration of network information can be extremely useful for the optimization of the decision taking process of a Job Scheduling Service, which has to select from a list of candidate computing nodes – often geographically distributed – the best computing element for the execution of a given job. The decision can be taken according to primary and secondary selection rules. A primary selection rule is defined according to the requirements of the job (such as the software environment available on a given Computing Element, the amount of free disk space, available CPU, etc), while the secondary rules are adopted to make a choice out of a list of equivalently good execution platforms. The execution of a job may require one or more input files and produces output data; thus, given the distributed nature of the databases, the input/output process can produce considerable data traffic across the Grid. Computing and Storage Elements can be selected so that the amount of traffic to be exchanged is minimized and/or the nodes with a suitable network connectivity performance are given higher priority.

Similarly, the Network Cost Estimation Service can be used to improve data management among different Storage Elements (SEs), e.g., the selection of the best replica of a given file (if there are copies in different SEs), the identification of the most appropriate SEs when a given amount of data has to be replicated, and the management of input/output data fragments in a single SE. In the last example, it may happen that input/output data of a given job is fragmented and distributed among a number of SE s. If the fragments need to be gathered into a single SE, then the most appropriate SE has to be identified. In this case the cost model can be based on principles such as the minimization of the amount of data exchanged between SEs, the identification of the SE with the lowest packet loss probability or with the maximum available bandwidth.

Finally, in case of adaptive remote file access there are situations where an application decides what file/files it needs to access only at run time, implying that in those cases that information cannot be used by a Resource Broker to statically allocate suitable Computing and Storage Elements to the application. In this case it becomes important to provide the application itself with a method that allows it to optimize the access to remote files at run time. The optimization is based on the dynamic adjustment of the Storage Element set that the application is using as the file access pattern changes.

4 Security Considerations

TBD

5 Authors Informatio

TBD

6 Intellectual Property Statement

7 Full Copyright Notice

8 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)