DataGrid

DATAGRID NETWORK MONITORING SCHEME PROPOSAL

WP07: Network Services

Document identifier:	**DataGrid-07-TED-nnnn-0_0**
Date:	**13/08/2001**
Work package:	**WP07: Network Services**
Partner(s):	**YTL, PDM**
Lead Partner:	**PELC**
Document status	**DRAFT**
Deliverable identifier:	**DataGrid-Dxx.y** (if deliverable)

Abstract: The application of computer processing and information retrieval has lead scientists to an increasing need to transfer petabytes of data across the internet. With the proposal of the GRID, the storage and retrieval of data across the GRID is increasingly integral to its design, so much so that it has become necessary to store performance data to identify the data transfer qualities of the GRID from one host to another.

In this paper, we will describe the basic Globus Data Grid architecture, give an account of some basic networking monitoring utilities and their application to GRID environments and then propose a method of storing such information for retrieval on LDAP.

Delivery Slip

	Name	Partner	Date	Signature
From	Y. Li	P. D. Mealor		
Verified by				
Approved by				

Document Log

Issue	Date	Comment	Author
0_0		First draft	Y. Li (HEP UCL), P. D. Mealor (HEP UCL)

Document Change Record

Issue	Item	Reason for Change

Files

Software Products	User files
Word	DataGrid-07-TED-xxxx-0_0-DataGridNetworkMonitoringSchemeProposal

CONTENT

1. INTRODUCTION

1.1. OBJECTIVES OF THIS DOCUMENT

The objectives of this document are to propose a scheme for publishing network-monitoring and performance data about the GRID on a Lightweight Directory Access Protocol server. Later drafts should also include its application to real GRID environments through the use of local GRIS and remote GIIS servers.

1.2. APPLICATION AREA

The area(s) of interest for this document are,

WP07 - Network Monitoring.

WP03 – Grid Monitoring Services.

WP05 – Mass Storage Management.

1.3. APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS

Applicable documents

[NETWORKMONITORING]	http://icfamon.dl.ac.uk/papers/WP7/netmon-requirements.htm
[RepSel]	Replica Selection in the Globus Data Grid. Sudharshan Vazkudai, Steven Tueke and Ian Foster. NB. Has since been updated.

Reference documents

[GLOPERF]	A Network Performance Tool for Grid Environments, Craid A. Lee, JamesStepanek, Rich Wolski, Carl Kesselman and Ian Foster.
[GRIDFTP]	The Globus Project White Paper: GridFTP Universal Data Transfer for the Grid. September 5, 2000.
[GridMonArch]	A Grid Monitoring Architecture, Document GWD-Perf-16-1, Ruth Aydt, Dan Gunter, Warren Smith, Martin Swanym Valerie Taylor, Brian Tierney, Rich Wolski, July 2001.
[HighPerfDistComp]	A Directory Service for Configuring High-Performance Distributed Computations, Steven Fitzgerald, Ian Foster, Carl Kesselman, Gregor von Laszewski, Warren Smith and Steve Tueke.
[IPERF]	http://dast.nlanr.net/Projects/Iperf/
[MRTG]	http://ee-staff.ethz.ch/~oetiker/webtools/mrtg
[NETPERF]	http://netperf.org/
[PingER]	http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html
[RFC1242]	Benchmarking Terminology for Network Interconnection Devices by S. Bradner, Harvard University, July 1991.
[RFC2330]	Framework for IP Performance Metrics, V. Paxson, G. Almes, J. Mahdavi, M. Mathis, May 1998
[RIPE_NCC]	http://www.ripe.net/ripencc/mem-services/ttm/index.html

[NIMI] http://www.ncne.nlanr.net/nimi/
[SURVEYOR] http://www.advanced.org/csg-ippm/

1.4. DOCUMENT AMENDMENT PROCEDURE

1.5. TERMINOLOGY

Definitions

DN	Distinguished Name
DIT	Directory Information Tree
GIIS	Grid Index Information Service
GRIS	Grid Resource Information Server
ICMP	Internet Control Message Protocol
IPDV	Instantaneous Packet Delay Variation
LDAP	Lightweight Directory Access Protocol
MDS	Metacomputing Data Service
RTT	Round Trip Time

Glossary

2. EXECUTIVE SUMMARY

{NONE}

3. GRID ARCHITECTURE

3.1. LDAP

LDAP is a form of 'database' that utilises a directory structure and allows the storage of data in attribute fields. These attribute fields are defined in object classes from which a hierarchy and properties of the attributes are defined. A schema is used to represent this information such that the data stored within it are useful and define the following:

· An unique naming schema for object classes

· Which attributes the object class must and may contain.

· Hierarchical relationships between object classes.

· The data type of attributes.

· Matching rules for attributes.

The schema is then what makes data valuable to applications.

Globus are implementing its own schema for describing computing resources such as CPU load. However, work to describe the volume of data and network-monitoring details have been minor thus far.

Typically, as with LDAP servers, the Globus schema can be easily modified. However, in order to govern consistency amongst the various GIIS's infrastructures, the structure is lead by an authority.

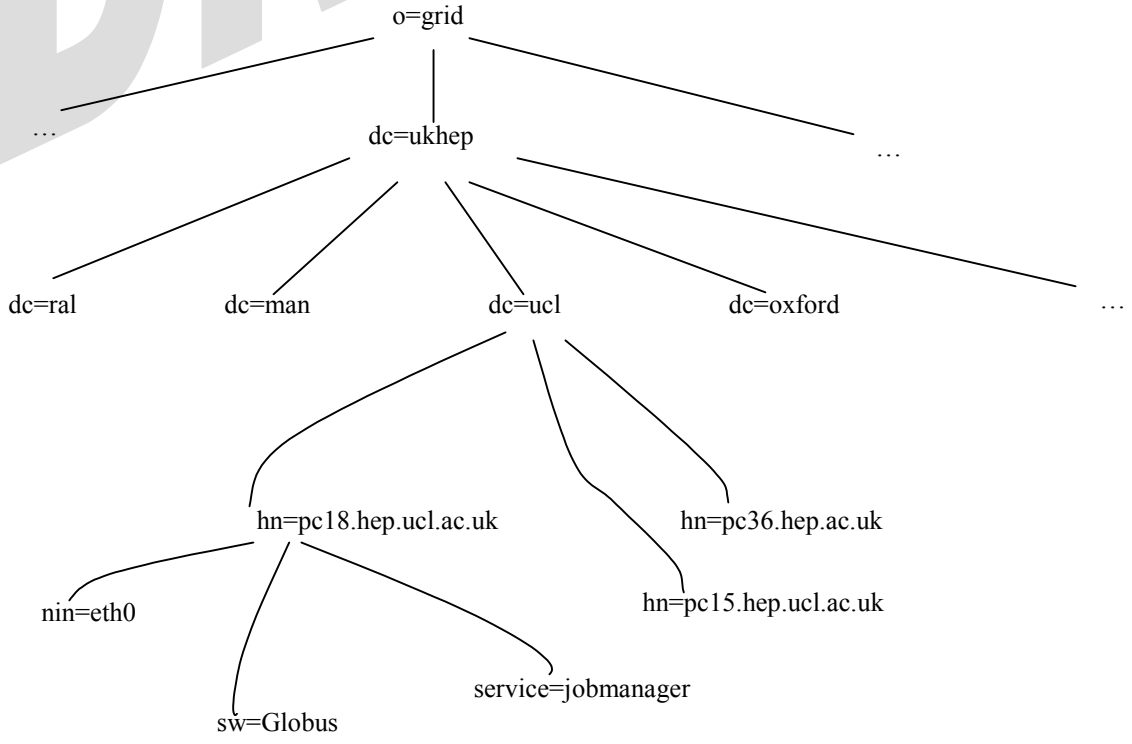

Figure 1: An example of a DIT representing some UK HEP hosts

The significance of using LDAP is that it is extensible and flexible [HighPerfDistComp]. An example of a Directory Information Tree (DIT) is shown in Figure 1.

3.2. GRIS AND GIIS

The Globus grid architecture proposes two layers from which higher-level services can be built on top of core services. This should enable the re-use of services across applications and tools such that Storage Request Brokers (SRB) can access data which can be used on an application wide level.

The research and design of the Globus project has lead to the development of a GIIS, formally the Metacomputing Directory Service (MDS) that consists of a LDAP representation in which directory structures, data representations and APIs are defined.

The discussion of a directory service will be reference in this paper in LDAP format and terminology, however, should be just as applicable in GRIS and GIIS format.

In order for a 'consumer' to find out about the network condition to a 'producer' [GridMonArch], it is proposed that a central directory service be implemented. In our model, this is presented as the LDAP engine from which a producer (i.e. the local machine containing a GRIS) will store and publish 'event' information that a consumer will receive and use. The information from these local LDAP/GRIS servers would then be updated on a local GIIS server that would communicate with other GIIS servers in order to supply the metric.

Currently, under the DATAGRID networking context, it is proposed that information should be stored on each machine on a GRIS. Within each clique a GIIS server is implemented which polls information from the GRIS servers assigned to its clique. This GIIS server would then communicate with other GIIS servers. The hierarchy of the GIIS structure is proposed as follows,

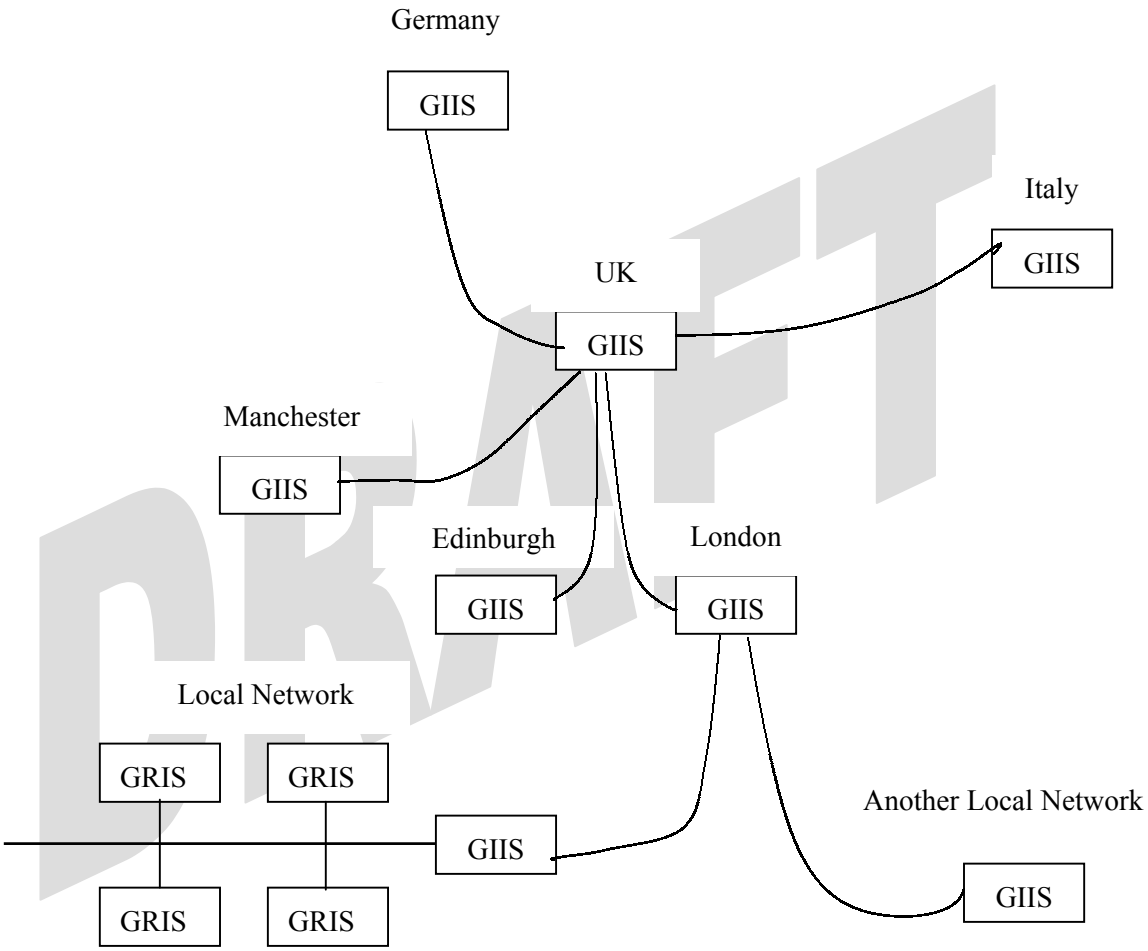

Figure 2: Proposed structure of GRIS/GIIS relationship.

As such, by only having GIIS servers at each clique communicating with other GIIS servers, we alleviate the $O(n^2)$ problem when we monitor performance from every endnode to endnode. By using geographical assignment of GIIS servers to towns/cities and countries, we only perform (possibly resource intensive) tests between the GIIS servers and hence lower the impact of network monitoring. As such, 'town GIIS' will have to communicate with a 'country GIIS' that is nominated. This 'country GIIS' will then communicate with other countries on the GRID network, also via a nominated 'country GIIS'. As such network-monitoring measurements will only be conducted between elected GIIS servers and not all individual nodes that are connected to that server.[1]

Implementations currently use GRIS servers that also act as GIIS servers. However, this may pose problems as the machine may become heavily overload resulting in poor performance. This is especially true if the server has to gather information from many servers at once. It may be better to have dedicated GIIS servers at each location.

[1] This only applies to active networking.

4. NETWORK MONITORING

4.1. REQUIREMENTS

In order to obtain information about the network, certain tests may need to be performed. These tests can be broadly classified into two categories:

- Active: Where test data is sent through the network in order to discover the properties of the end-to-end connection.
- Passive: Where useful data, such as a required file, is transferred across the network and the resultant transfer properties used as a test benchmark.

This current proposal only implements active forms of monitoring, with the resultant test results stored on the 'sending' machine. The specifics of the storage requirements are discussed later in this paper.

To update the information from network monitoring on the LDAP server, two forms of networking monitoring initialisation are proposed:

- Streamed: Periodic updates of each network property is updated,
- Query/Response: Where a 'user' can ask for specific network properties and be supplied an immediate response (or as long as it takes to conduct that measurement).

Our initial implementation shall use a query/response scheme to query network-monitoring information collated from logs and store them in an LDAP server. Future incarnations shall also implement automatic streamed updates either through,

- Push: Where a LDAP 'backend' will automatically interface with the applications and update accordingly – this may also be considered as 'piping'.
- Pull: Where information will be stored locally (or remotely) and then periodically polled by the GRIS to update the tree information.

There is current concern over the feasibility of using a 'push' technique in real life GRID environments, as servers may be flooded with so much information they have little resources to do anything else.

The process of data gathering in our proposal is similar to a method adopted by Alex Martin with his ftree program: it is suggested that the programs that gather or produce the data shall be separate from the GRIS and interfaced via a LDAP backend.

For early implementations, the network monitoring application(s) would produce a log file from which the network metric values are extracted and converted to a format (such as LDIF) that can then be published on the LDAP server of that machine. This is shown in Figure 3.

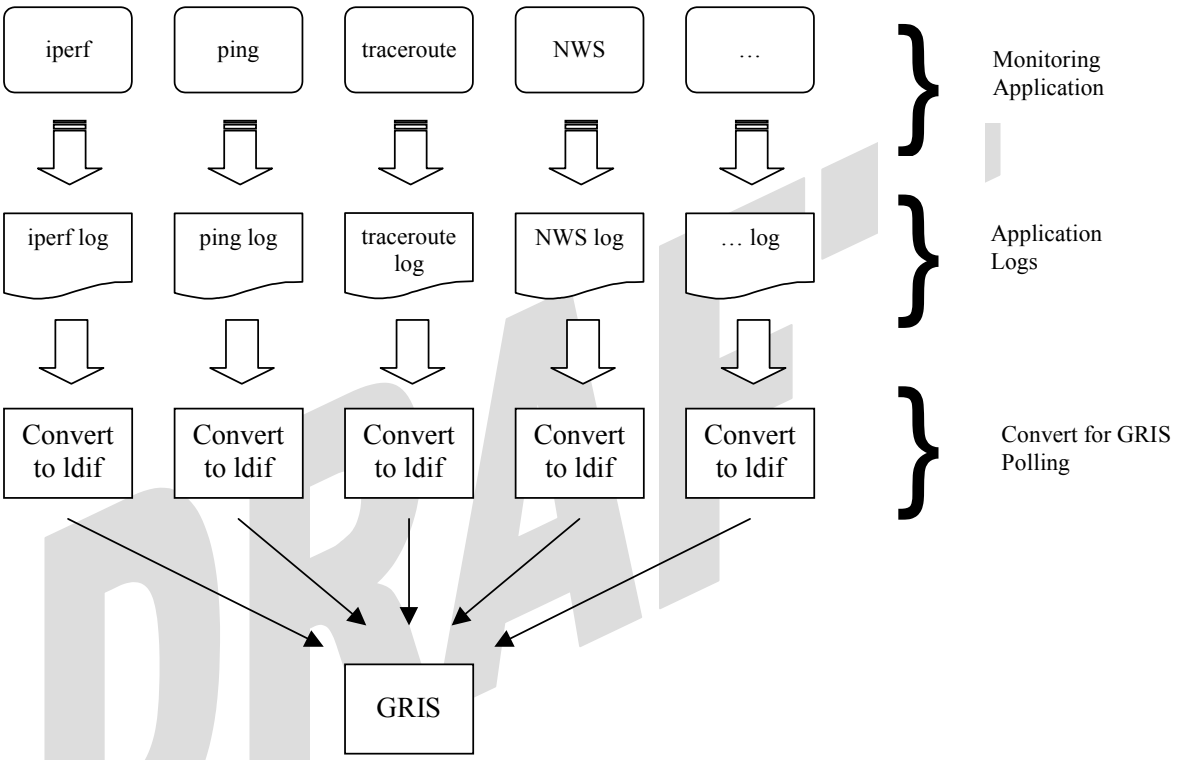

Figure 3: The relation between data produced by network monitoring programs and the LDAP representation of that data.

4.2. LDAP APPLICATION

By adopting the scheme shown in Figure 3, the use of any user specified network-monitoring package could be used and the information stored on the LDAP directory – as long as an appropriate backend is implemented on the LDAP. This has the benefit of flexibility as administrators can choose to implement preferred network monitoring tools, however, for the initial testbeds the type and number of applications should be limited to certain widely available tools such as those described later in this document.

Different performance measuring tools produce different results for the "same" test, due to the difference in the exact method and internal parameters they use. For this reason, it may be beneficial to classify performance measurements within the DIT according to the tool used. It is expected that this would produce a deeper tree, but it may be useful to consumers to know how a piece of information was measured so that they can act accordingly.

By allowing the user to specify the network monitoring applications used, it is necessary to identify the type of tools that could be used. This may, for example, pertain to bandwidth measurements, one-way delay etc. In LDAP, we can classify the type of network monitoring data using object classes. By specifying the basic form of data that is required, such as the RTT value of a measurement, we can inherit these attributes and possibly include further information measured by the program such as the deviation of the measured sample. This therefore means that in order to implement the use of a network-monitoring program, one must inherit the metric type (e.g. RTT) object class - adding it as a subclass. It is therefore important to store information such as the network monitoring application used and also the version number of that program. This also has the benefit of storing performance metrics

of different tools separately, without conflict. This feature may also become valuable in comparing the quality of different tools under the same conditions.

4.3. BACKENDS

The production of log files will no doubt differ in terms of content. It is the purpose of a LDAP backend to convert the log information containing networking metrics into something that can be published on the LDAP. This, in principle implies that every network-monitoring program will require its own LDAP backend to convert the data for publication on to the GRIS.

Is it proposed by groups in the US to adopt a standard NetLogger log file format to store the networking monitoring information. This, if implemented, should allow a single backend to enter network-monitoring information into the LDAP.

Other groups also propose to use 'application wrappers' that can be used to encapsulate each network monitoring application. This has the benefit of a centralised interface to setup update frequency and maintenance. Applications such as NIMI support this, whilst NWS required specifically written modules to gather such information.

4.4. INFORMATION STORAGE

Our current proposal is to only store the most recent data for each metric. This may also contain basic statistical information such as min, max and averages. It is recognised that historical information is important for network monitoring and forecasting should also be implemented on a later release.

In terms of historical information storage, the current idea is to include an attribute that contains the number of records (or individual measurements) to keep for that 'branch' and to simply overwrite the earliest entry should the number of records become larger than this value. While this is in principle possible, it may require hacking of the LDAP Protocol. This type of queuing can be envisaged to one that is similar to one used by NWS and would require each entry to be identified by date as part of its dn. The format of this historical recording should also be compatiable with the archiving service that is being developed to store network monitoring readings.

For forecasting, it is proposed that perhaps the information can be set up on the GIIS machine and not locally on the GRIS. This is similar to the methods used for centrally gathering information on NWS networks. Currently, Rich Wolski and Martin Swany in the US are implementing a method of publishing such information on LDAP format, and it is hoped that this can be incorporated into our scheme.

5. NETWORK PERFORMANCE METRICS

As discussed above, network performance metrics can be classified into two areas, passive and active measurements. Please note that for immediate implementations, only active metrics are considered.

5.1. ACTIVE

5.1.1. RTT & jitter

The RTT is the time taken to traverse a path from one host to another and back again to the original host. Tools that measure RTT usually send ICMP Echo messages, that allows a packet of a user defined length to a remote node and have it echoed back. The time the echo request takes to come back is regarded as the RTT. Technically, RTT is regarded as the sum of the propagation times at each host plus the time occurred from delays from the hops associated with a particular path.

A measure of the variation of the RTT should be also be considered (and consequently the variation in the arrival times of successive samples of RTT, i.e. jitter or IPDV). This can be calculated with a sample of RTTs from a given pair of endpoints to give a better indication of the status of the internet.

As the state of the internet can not be known at any instance, the RTT can be influenced by router queuing, unreliable remote hosts or packet damage. Hence, frequent measurements (e.g. once every 15 minutes or less) would yield a good indication of the network performance quality.

5.1.2. Packet loss

Packet loss provides a good measure of the quality of the route between end points and is especially important for connection-orientated protocols such as TCP/IP. As such, knowing how 'lossy' a particular path is would be valuable information in selecting a particular transfer path.

As packet loss can be attributed to many different factors (such as router queuing from network traffic or the lack of receiver-side acknowledgement) the amount of loss on a path can change quickly. Hence it is proposed that measurements be taken frequently (as frequently as RTT) without being too intrusive and that statistical measurements be made to record the minimum, maximum and average values for various sized packets. Information such as the deviation in loss percentages would also be valuable.

5.1.3. Aggregate network throughput (bandwidth)

The bandwidth is the total maximum throughput of data from one endpoint to another. This may measure the current utilization as a rate (volume/time) or as a proportion of the total bandwidth capability within the path across the network.

In order to supply these 'active' measurements of the throughput, we need to measure network capability with tools that throttle the network with as much data as possible and measuring the resulting bandwidth of data successfully transferred. As such, active measurements of bandwidth are resource intensive, and hence highly intrusive. It is therefore suggested that the frequency of this measurement should be kept low, perhaps performed only once or twice every day during off-peak times.

5.1.4. Number of hops

This metric is a measure of the number of hops required for a local host to reach a remote host. The information provided within this metric will highlight potential problems with, say, an increase in RTT, which could be due to differing routing tables implemented by routers along the path.

As it is often rare that routing tables change, the measurement of the number of hops should be infrequent – perhaps once a day. It is worth noting that programs such as traceroute, which supplies information about the total number of hops can also supply information about the RTT.

This metric would become important with the implementation of Quality of Service and DiffServ/MPLS routing.

5.2. PASSIVE

5.2.1. Per flow application throughput with GridFTP

GridFTP [GridFTP] is an extension of the standard FTP mechanism for use in a Grid environment. It is envisaged that it will be used as a standard protocol for data transfer. Per flow application throughput defines the throughput measured for a specific GridFTP transfer between specified end-points. It will be based on a 'passive' measurement of the data transferred - i.e. only the information transferred - not additional (test) data.

The passive measurement of GridFTP transfers means that information collected from a GridFTP transfer should be stored locally on the sending machine and then be polled by the GRIS server for storage into the LDAP DIT.

The data that shall be stored regarding a GridFTP transfer is as follows,

- Source
- Destination
- Total Bytes / Filesize
- Number of Streams Used
- TCP buffer size
- Aggregate Bandwidth
- Transfer Rate (Mbytes/sec)
- Time Transfer Started
- Time Transfer Ended
- Total Time

A schema proposed by Sudharshan Vazhkudai [RepSel] utilizes a patched version of GridFTP in which transfers are recorded and summary data stored; this is not incorporated into our scheme as he proposes information storage in a different format to ours. However, it is hoped that once a standard form of network monitoring schema is formed, both schemes will be unified.

Future incarnations of GridFTP propose to incorporate features such as automatic negotiation of TCP buffer/window sizes and parallel data transfer, and reliable data transfer. These should make some of the object class and associated attributes subject to modification.

5.2.2. Grid volume

To supply a measurement of the volume of data transferred on the grid as a function of time (daily/weekly/monthly/yearly), we propose that information from data transfer across the grid be aggregated as transfers proceed. Should GridFTP be used for data transfer, then the various variables shall be obtained from GridFTP measurements and simply added. Otherwise the software used for transfers should account for this. This may or may not include data transferred from active (test) data.

The results from such measurements shall be stored on the sending site as the receiver is assured by acknowledgements and shall be updated as frequently as GridFTP transfers (and or active measurements) are opened.

6. NETWORK MONITORING TOOLS

This section highlights some common tools that can be used to monitor network conditions. This is by no means an exhaustive list and should be regarded as generic applications that could interface with the LDAP server through a backend to collate data.

6.1. PINGER

PingER can be used to measure the RTT, packet loss percentages and the variability in response time between a pair of end-nodes. Being well established in the HEP community, it currently has hundreds of sites in many countries. As such, it is favoured above other tools such as Surveyor [SURVEYOR], RIPE NCC [RIPE_NCC] and MRTG [MRTG].

6.2. NWS

Network Weather Service (NWS) is a distributed system that periodically monitors and dynamically forecasts the performance of various network and computational resources over given time interval. It can also measure metrics such as TCP bandwidth etc. that can be provided with modules. It utilises a central reserve from which data is collated and used for network forecasting.

6.3. PIPECHAR

Pipechar is a program that probes the network to find out a bottleneck link across a network. It can also supply information regarding RTT and jitter times. It does, however, require superuser privileges in order to run. As the tool throttles the network to each hop in the connection, it can be a very intrusive application to use on a network.

6.4. IPERF

Iperf [IPERF] is a tool for measuring maximum TCP and or UDP bandwidth associated with a link, reminiscent of ttcp and nettest. It has been written to overcome the shortcomings of those ageing tools. It attempts to throttle a network with TCP or UDP traffic - discovering the maximum transfer throughput (bandwidth) between two nodes in a network, without monitoring in-between nodes/routers. It can also utilise parallel-streamed transfers if the appropriate libraries are installed and have the benefit of using user-specified window size for network transfers.

6.5. NETPERF

Netperf provides a general measure of performance of a network. It can give a measure of latency between request and response of generic transactions across a TCP/IP network by conducting unidirectional throughput.

6.6. TRACEROUTE

Traceroute is a program that analyses the path of an end-to-end connection. It can supply information regarding the ping times to the intermediate routers and the names (IP/DNS) of the routers. As such it can give information about the route that an end-to-end connection establishes.

6.7. NIMI

NIMI is an application wrapper that uses existing network monitoring applications to unify these applications to a central interface. This can then be used to set scheduling methods and information gathering. It can also unify publication of such information.

7. EU 'ACTIVE' NETWORK MONITORING

Whilst our current proposal stores all the information locally to describe network metrics to any other host, it is important that this information should be available for universal GIIS publication. As active measurements can be quite resource intensive and hence intrusive, it does not make sense to conduct node-to-node measurement of all nodes on the grid. As such, a sole GIIS should be elected at each clique. Hence, only one machine in each country should conduct tests to one other machine in each country.

This would not give exact metrics for end-to-end connections, but should however allow enough information to give indication on network performance from country to country or from town to town.

Performance on passive measurements, i.e. that performed by applications such as GridFTP, are currently under development by Sudharshan Vazhkudai. His work is currently not implemented in this proposal. It is hoped, however, that future implementations of our scheme will incorporate his work.

7.1. VIRTUAL ORGANISATION

It was recommended by Alex Martin, that instead of building another DIT for EU active network monitoring, we could in principle create the structure requested by WP07 for active measurements by implementing a 'virtual organisation' that contains referrals to the appropriate GRIS/GIIS machines on the GRID network.

As such, performance metrics which have already been conducted are simply referenced on this virtual organisation to the appropriate node of the DIT of the GIIS/GRIS servers.

The benefit of this method is that the locally stored information will remain as it is, with only the structure of this virtual organisation defining the active tests which are needed to be performed. Should the appropiate test parameters not exist on the local GRIS/GIIS, then the virtual organisation LDAP could interface with the LDAP backend to perform the tests to acquire the performance metrics sought.

The structure of this 'virtual organisation' is hoped to be in the form of a list of countries under a root dn. Each branch will then contain another list of countries which would represent network monitoring tests from the branch above to this branch. Underneath this level would then contain the results of the network performance metrics – possibly similar to that presented for locally stored attributes.

7.2. IMPLEMENTATION

This is currently work in progress.

8. FUNCTIONALITY OF STORAGE SYSTEM

Each measure of network performance is affected by a great number of parameters, some of which are under the control of the tools used. The information each tool produces can be divided according to these parameters - which gives more detailed information to a consumer about how to optimise a data transfer. It can also be aggregated across all the different values of the parameters – which gives an overview of the performance which can be expected for a data transfer. These parameters variously include packet size, buffer sizes and the number of streams used, and not all measures of network performance are affected by all of them.

A performance-measuring tool might make many measurements of the same variable, and this all this information can be collated into something more manageable. How the information is collated depends on its type, and is explained in more detail below.

8.1. DIT AND INHERITANCE

The meaning of many object classes is, in part, defined by their position in the directory information tree (DIT). The proposed DIT is shown in Figure 4.

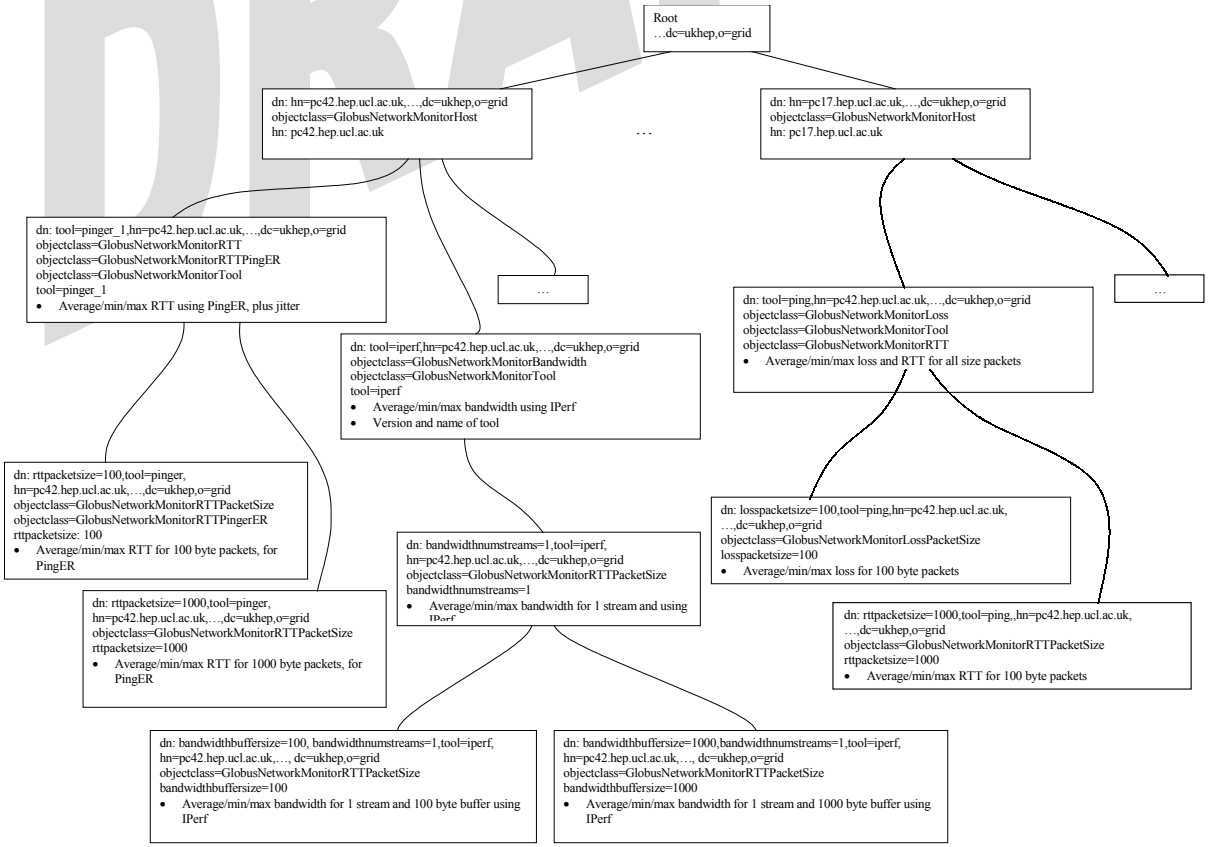

Figure 4: Proposed DIT for organising network monitoring measurements

It is assumed that either the physical location of the LDAP server or the position of the data within another DIT will indicate the host at which the measurements were made. The `GlobusNetworkMonitorHost` object (annex 10.1.5) object is used to store the hosts to which the measurements were made.

Aggregated information for each tool is stored in the tree at the level below the remote hostname to which that information pertains. For example, the average RTT measured with ping (for any packet

size) to a host at 192.168.0.12 might be found in the attribute named `rttavg` under the DN "tool=ping, hn=192.168.0.12,dc=ours, dc=ukhep, o=grid", where the name of the tool is given by the `tool` field of `GlobusNetworkMonitoringTool`. Below that level, information is divided according to the parameters used to define it. For example, to find the average RTT for a packet size of 100 bytes, the DN would be "rttpacketsize=100, tool=ping, hn=192.168.0.12, dc=ours, dc=ukhep, o=grid". To search for the ping measured by any tool, a search starting at the base DN of "hn=192.168.0.12, dc=ours, dc=ukhep, o=grid" for any entries with an object class given by "objectclass=GlobusNetworkMonitorRTT" might be made.

Entities in the DIT can be of more than one object class - an entity that has two object classes will have all the attributes from both classes (although some or all of them may be optional attributes). Some tools may produce more than one type of data as classified by the object classes given in section 10. In this case, the entries for that tool may have many object classes, corresponding to the different types of data that it produces.

Object classes which inherit attributes from other object classes do not necessarily attach exactly the same meaning to those attributes. In the following section, most of the inheritance is from an object class representing an overview of performance data to an object class representing the same performance data for a particular value of a parameter.

8.2. SCHEMA

Below are described the attributes of each object class associated with each type of network monitoring metric.

8.2.1. RTT

Round trip time is affected mainly by packet size, so as well as the mean, maximum, minimum etc RTT, entities are stored below that to give per-packet size information. The objects used to publish RTT measurements are shown in annex 10.1.

The `GlobusNetworkMonitorRTT` object is used to store information about the RTT for all packet sizes. The minimum and maximum RTTs encountered are stored in `rttmin` and `rttmax`, and the average of all measurements made of RTT is stored in `rttavg`. Some versions of 'ping' produce a value known as `mdev` (mean deviation), and the mean `mdev` for all packet sizes is also stored. The last time information was entered is also stored in `rttlastmodified`; this simply shows how up to date the information is.

`GlobusNetworkMonitorRTTPacketSize` is used to store information about the RTT for a specific packet size. It inherits all of the fields from `GlobusNetworkMonitorRTT` and adds a new field: `rttpacketsize`, which is used to record the packet size.

8.2.2. Loss

The number of packets lost en-route to a host is also affected by packet size. Thus there are two object classes similar to those for RTT, one of which contains aggregate packet loss information, and the other which contains packet loss information for a particular packet size.

The objects used to publish loss measurements are shown in annex 10.1.3.

The `GlobusNetworkMonitorLoss` object is used to store information about the RTT for all packet sizes. The minimum, maximum and average losses are stored in `lossmin`, `lossmax` and `lossavg`, respectively.

`GlobusNetworkMonitorLossPacketSize` inherits attributes from `GlobusNetworkMonitor-Loss` and adds `losspacketsize`. It is used to store the packet loss for a particular packet size.

8.2.3. Hop count

The number of hops between two hosts is unlikely to change often, and is not affected by anything other than the topology of the network connecting them. For this latter reason, only one object class exists for publishing hop count data. The object class records the minimum and maximum number of hops counted in `hopsmin` and `hopsmax`, plus the current value in `hopslast` (annex 10.1.4). The attribute `hopslastmodified` indicates the time at which the number of hops was last counted.

8.2.4. Bandwidth

Bandwidth can be measured using a variety of different tools, such as Iperf [IPERF] or Netperf [NETPERF]. Although these tools are different, they produce similar classes of data and are affected by the same protocol parameter changes. As such, the bandwidth data is classified by the number of streams used in the transfer, and the buffer size of the local machine. (annex 10.1.2).

`GlobusNetworkMonitorBandwidth` is used to store information from all numbers of streams and buffer sizes. It stores the maximum and minimum bandwidths achieved with any number of streams and any buffer size in `bandwidthtransmax` and `bandwidthtransmin`, and the average bandwidth in `bandwidthtransavg`. `bandwidthlastmodified` contains the last time this record was modified, in other words the last time a bandwidth measurement was made on this host.

`GlobusNetworkMonitorBandwidthStreams` is used to store the average, maximum and minimum bandwidths for a particular number of streams. It inherits attributes from `GlobusNetworkMonitorBandwidth` and adds the attribute `bandwidthnumstreams` which is used to record the number of streams.

`GlobusNetworkMonitorBandwidthBufferSize` is used to store the average, maximum and minimum bandwidths for a particular number of streams and a particular buffer size. It inherits attributes from `GlobusNetworkMonitorBandwidth` and adds an attribute to store the buffer size, `bandwidthbuffersize`. The number of streams is indicated by the object's position in the directory tree, below a `GlobusNetworkMonitorStreams` object.

8.2.5. GridFTP

This section of the schema (annex 10.1.5) is the same as that currently in use by Sudharshan Vazhkudai for publishing logging data from GridFTP [RepSel]. The patched version of GridFTP produces logs on a per-logical-volume basis, rather than a per-host basis. For this reason, the GridFTP object classes are in a separate branch of the tree to the rest of the network performance objects.

9. FUTURE WORK

A number of different areas were not considered for this report, or were touched upon only lightly.

The main area that requires some work is integrating the proposed DIT into the standard GRIS tree structure. It is conjectured that the network monitoring data could be placed below either the description of a particular host, or below the description of a particular network interface. Alternatively, the monitoring data could be stored elsewhere and referenced as a service available on a host. It should also be noted that dynamic data could be easily added to a GRIS after modifying a simple configuration file.

The proposed DIT is not entirely complete. For example, the mean bandwidth to a host for a particular buffer size but any number of streams is not published. This work is simply a refinement of the DIT. Also, object identifiers (OIDs) need to be assigned.

The statistical calculations carried out on many of the data types could be reviewed: The statistics that might be required by a consumer have not been fully explored, but any modifications to the proposed scheme are expected to be fairly trivial. For example, the `mdev` value calculated for RTT may not be as suitable as jitter as a measure of the stability of the RTT.

The software package, NIMI, is a tool designed to coordinate an arbitrary collection of network monitoring tools [NIMI]. It can be used to periodically run these tools and collate their output. It may be possible to use this program to generate LDIF format files, which can be easily published through an LDAP server.

10. ANNEXES

10.1. ACTIVE MONITORING SCHEMA

10.1.1. RTT object classes

This schema is used to publish RTT data.

```
OBJECTCLASS 'GlobusNetworkMonitorRTT' {
    INHERITS FROM { GlobusNetworkMonitorHost }
    OID { }
    DESCRIPTION {
        'Contains attributes defining RTT. These are values against a source-ip (if no
transfers are found per hn)'
    }
    MUST CONTAIN {
    }
    MAY CONTAIN {
        rttmax:: single-valued, cisfloat, { Maximum rtt   }
        rttavg :: single-valued, cisfloat, { Average rtt  }
        rttmin :: single-valued, cisfloat, { Minimum rtt  }
        rttmdev :: single-valued, cisfloat, { Mdev specified by some versions of ping }

        rttlastmodified       :: single-valued, cisfloat, { Time rtt was last measured }
}

OBJECTCLASS 'GlobusNetworkMonitorRTTPacketSize' {
    INHERITS FROM { GlobusNetworkMonitorRTT }
    OID {}
    DESCRIPTION { 'Stores min/max/avg etc. RTT for each packet size' }
    MUST CONTAIN {
    }
    MAY CONTAIN {
        rttpacketsize::single-valued, integer { Packet size in bytes?bits? }
    }
}
```

10.1.2. Bandwidth object classes

```
OBJECTCLASS 'GlobusNetworkMonitorBandwidth' {
    INHERITS FROM { GlobusNetworkMonitorHost }
    OID { }
    DESCRIPTION {
        'Contains attributes defining the Bandwidth (eg from iperf or netperf). These are
values against a source-ip (if no transfers are found per hn)'
    }
    MUST CONTAIN {
    }
    MAY CONTAIN {
        bandwidthtransmax:: single-valued, cisfloat, { Maximum transfer bandwidth  }
        bandwidthtransavg:: single-valued, cisfloat, { Average transfer bandwidth  }
        bandwidthtransmin:: single-valued, cisfloat, { Minimum transfer bandwidth  }
        bandwidthlastmodified::single-valued, cisfloat, { Time last bandwidth measurement was
made }
    }
}

OBJECTCLASS 'GlobusNetworkMonitorBandwidthStreams' {
    OID {}
    DESCRIPTION { 'Contains bandwidth information per number of streams' }
    INHERITS FROM { GlobusNetworkMonitorBandwidth }
    MUST CONTAIN {
        bandwidthnumstreams::single-valued, integer { Number of streams used in transfer }
    }
    MAY CONTAIN { }
}
```

```
OBJECTCLASS 'GlobusNetworkMonitorBandwidthBufferSize' {
    OID {}
    DESCRIPTION { 'Contains bandwidth information per buffer size, classified by number of
streams by directory structure' }
    INHERITS FROM { GlobusNetworkMonitorBandwidth }
    MUST CONTAIN {
        bandwidthbuffersize::single-valued, integer { Buffer size in bytes used in transfer }
    MAY CONTAIN { }
}
```

10.1.3. Loss object classes

```
OBJECTCLASS 'GlobusNetworkMonitorLoss' {
    INHERITS FROM { GlobusNetworkMonitorHost }
    OID { }
    DESCRIPTION {
        'Contains attributes defining the Loss. These are values against a source-ip'
    }
    MUST CONTAIN {
    }
    MAY CONTAIN {
        lossmin :: single-valued, cisfloat, { Minimum loss  }
        lossmax :: single-valued, cisfloat, { Maximum loss  }
        lossavg :: single-valued, cisfloat, { Average loss }
    }
}

OBJECTCLASS 'GlobusNetworkMonitorLossPacketSize' {
    INHERITS FROM { GlobusNetworkMonitorLoss }
    OID {}
    DESCRIPTION {
        'Contains attributes defining the loss for a given packet size. These are values
against a source ip' }
    MUST CONTAIN {
        losspacketsize::single-valued, integer { Packet size in bytes?bits? }
    }
    MAY CONTAIN { }
}
```

10.1.4. Hop count object class

```
OBJECTCLASS 'GlobusNetworkMonitorHops' {
    INHERITS FROM { GlobusNetworkMonitorHost }
    OID { }
    DESCRIPTION {
        'Contains attributes defining the number of hops to a source ip'
    }
    MUST CONTAIN {
    }
    MAY CONTAIN {
        hopslastmodified :: single-valued, cisfloat {Time that last traceroute was performed }
        hopslast:: single-valued, integer {Last measured number of hops}
        hopsmin:: single-valued, integer {Maximum number of hops recorded}
        hopsmax:: single-valued, integer {Minimum number of hops recorded}
    }
}
```

10.1.5. Information object classes

The GlobusNetworkMonitorHost class is used to store the name of the remote host to which measurements are made. It is expected that any entities of this object class would not be of any other.

```
OBJECTCLASS 'GlobusNetworkMonitorHost' {
    INHERITS FROM { GlobusTop }
    OID {}
    DESCRIPTION {
        'Standard attributes that will be required by all the transfer information objects'
    }
    MUST CONTAIN {
        hn::single-valued,cis { Hostname of monitored ip/dns }
    }
```

```
    MAY CONTAIN { }
}
```

The GlobusNetworkMonitorTool would be used to store the name and version of a tool used to make measurements. It should be used in conjunction with other classes which are used to actually store the measurements made by that tool.

```
OBJECTCLASS 'GlobusNetworkMonitorTool' {
    INHERITS FROM { GlobusTop }
    OID {}
    DESCRIPTION {
        'Standard attributes used to indicate the type of tool used'
    }
    MUST CONTAIN {
        tool::single-valued,cis { Tool used to make measurements, syntax {toolname}_{version}
}
    }
    MAY CONTAIN {
        toolname::single-valued,cis { The name of the tool }
    toolversion::single-valued,cis { The version of the tool }
    }
}
```

10.1.6. Example tool-specific object classes

Certain tools may produce more data than the generic object classes can handle. In this case, new object classes can be inserted into the DIT to publish those data.

```
OBJECTCLASS 'GlobusNetworkMonitorRTTPingER' {
    INHERITS FROM { GlobusTop }
    OID {}
    DESCRIPTION {
        'PingER also measures jitter'
    }
    MUST CONTAIN { }
    MAY CONTAIN {
        rttjittermin::single-valued,cis { The minimum jitter }
        rttjitteravg::single-valued,cis { The average jitter }
        rttjittermax::single-valued,cis { The maximum jitter }
    }
}
```

10.2. PASSIVE NETWORKING SCHEMA

10.2.1. GridFTP logging schema

These object classes are used by Sudharshan Vazhkudai to publish logging data from a patched version of GridFTP

```
OBJECTCLASS 'GlobusStorageServerVolume' {
        INHERITS FROM { GlobusTop }
        OID { 1.3.6.1.4.1.3536.2.4.3.10 }
    DESCRIPTION {
        'Contains a bunch of static (disk seek times, administrator policies) and dynamic
attributes'
    }
    MUST CONTAIN {
        volume          :: single-valued, cis,     { Volume name }
        totalspace      :: single-valued, INTEGER, { Disk volume }
        availablespace  :: single-valued, INTEGER, { Available space }
        percentused     :: single-valued, INTEGER, { Percent used }
        gridftpurl      :: single-valued, cis, { GridFTP URL including port # }
    }
    MAY CONTAIN {
        disktransferrate :: single-valued, cisfloat, { Disk transfer time }
        drdtime          :: single-valued, cisfloat, { Disk read time in milli secs  }
        dwrtime          :: single-valued, cisfloat, { Disk write time in milli secs }
        requirements     :: single-valued, cis,     { Administrator policies }
    }
}
```

```
OBJECTCLASS 'GlobusStorageTransferBandwidth' {
        INHERITS FROM { GlobusStorageServerVolume }
        OID { 1.3.6.1.4.1.3536.2.4.3.11 }
    DESCRIPTION {
        'Contains attributes defining performance. These are performance values for the entire
site against a source-ip (if no transfers are found per volume)'
    }
    MUST CONTAIN {
    }
    MAY CONTAIN {
        maxrdbandwidth :: single-valued, cisfloat, { Maximum read bandwidth   }
               avgrdbandwidth :: single-valued, cisfloat, { Average read bandwidt     }
        minrdbandwidth :: single-valued, cisfloat, { Minimum read bandwidth    }
        last_rd_bandwidth :: single-valued, cisfloat, { Minimum read bandwidth    }
        numrd          :: single-valued, INTEGER,  { Number of read transfers  }

        maxwrbandwidth :: single-valued, cisfloat, { Maximum write bandwidth   }
        avgwrbandwidth :: single-valued, cisfloat, { Average write bandwidth   }
        minwrbandwidth :: single-valued, cisfloat, { Minimum write bandwidth   }
        last_wr_bandwidth :: single-valued, cisfloat, { Minimum write bandwidth    }
        numwr          :: single-valued, INTEGER,  { Number of write transfers }
    }
}

OBJECTCLASS 'GlobusSourceTransferBandwidth' {
        INHERITS FROM { GlobusStorageTransferBandwidth }
        OID { 1.3.6.1.4.1.3536.2.4.3.12 }
    DESCRIPTION {
        'These are performance values per volume agaist a source-ip (if no transfers are found
per volume, then entire site averages are returned)'
    }
    MUST CONTAIN {
    }
    MAY CONTAIN {
    }
}
```