Publication of network monitoring data in the European DataGrid

Paul Mealor (pdm@hep.ucl.ac.uk)
University College London, Gower Street, London WC1E 6BT, United Kingdom

Robert Harakaly (robert.harakaly@ens-lyon.fr)
ENS Lyon, 46, Alle d'Italie, 69364 LYON Cedex 07, France

Franck Bonnassieux
Formerly of: ENS Lyon, 46, Alle d'Italie, 69364 LYON Cedex 07, France

Peter Clarke (clarke@hep.ucl.ac.uk)
University College London, Gower Street, London WC1E 6BT, United Kingdom

Abstract. Efficient allocation of computing, storage and network resources in a Grid environment where large volumes of data need to be moved about requires good knowledge of the state of the networks involved. Comprehensive and accessible information about the current and historical state of networks allows easier diagnosis of network problems. The European DataGrid Network Monitoring Architecture includes monitoring points running new and existing network monitoring software modified to publish monitored information using the Relational Grid Monitoring Architecture (R-GMA); the Probes Coordination Protocol that provides distributed scheduling of network measurements; software to publish the logged information from file transfer tools such as GridFTP into R-GMA; and tools to archive the published information for later use.

Keywords: European DataGrid, network monitoring, monitoring information publication

Abbreviations: EDG – European DataGrid; R-GMA – Relational Grid Monitoring Architecture; API – Application Programming Interface; PCP – Probes Coordination Protocol; NCES – Network Cost Estimation Service

1. Introduction

Information about the state of networks has many uses in a Grid, and all the more so in a Data Grid where large volumes of data are moved around and processed. In such an environment, there are two main areas where there state of the networks is particularly useful: for resource selection and for trouble-shooting.

Information on the state of the networks must therefore be available to Grid middleware, users, applications and adminstrators.

1.1. Resource selection in the Grid

In the European DataGrid environment, the major resources available to Grid jobs are storage and computation. Experimental data and data generated in simulation jobs can be stored at the storage resources. These files may then be replicated across the Grid to be available for further analysis by other jobs and users. Thus, a single logical file may exist as several replicas on the Grid.

1.1.1. *Replica selection*

The selection of a particular replica of a logical file for use by a computational job and for further replica generation is very important. In EDG, otherwise identical replicas are selected by the "cost" of retrieving them. The cost could be based on many factors, but in EDG, it is based primarily on the time until the file can be used; this cost may be dominated by the time taken to transfer the file over the Internet.

1.1.2. *Job distribution*

Orthogonally, the selection of computational resources for a grid job may be based on the availability of data replicas.

Grid Jobs may consume or generate very large files. Users may specify the storage system at which to store large output files, and large input files must be retrieved from other storage systems. In this case, the resource brokering system can use the cost associated with storing or retrieving these files to decide on the optimum computational resources on which the job should be run.

1.2. Network trouble-shooting

Network trouble-shooting requires long-term monitoring of the state of networks. Comprehensive information allows identification of long- and short-term trends to flag problems before they occur. Sudden changes can be associated with configuration changes, even after the fact with historical data.

2. EDG network monitoring architecture

The network monitoring publication scheme consists of extensions of existing software and new software distribute measured network information via Grid information services. The main tools making measurements of network performance are three similar active monitoring tools: PingER[11], IperfER and UDPMon, described in more detail in

§2.1; the GridFTP[1] system used to transfer data in a secure and Grid-aware fashion is also used as a source of passively measured performance information, this is described in §2.6.

2.1. Monitoring tools

The monitoring tools used within the European-DataGrid were mostly adapted from other work. A brief overview of the tools is given here:

PingER, IperfER and UDPMon are monitoring tools used in EDG, and are based upon PingER[11], originally developed by the Internet End-to-End Performance Monitoring project (IEPM) at SLAC.

2.2. PingER

PingER is based on a number of scripts run at set intervals by *cron*, and a number of CGI scripts which provide a web interface.

The *cron* scripts are run at regular intervals, performing a number of multi-packet *ping*s of all machines listed in a configuration file. Each group of *ping*s can be performed with different packet sizes. Ping measures round-trip latency and packet loss with ICMP ECHO packets. The measurements are stored in text files on the measuring machine.

Data measured by PingER is presented to users via a web interface. From this interface, the most recent measurements can be displayed, as well as graphs of historical information. Raw data may also be extracted in a more machine-readable format for use by automated tools.

Because the IEPM PingER software requires manual configuration, measurements to new sites take time and manpower. Conflicting tests (which are especially pertinent to Iperf and UDPMon test) can only be avoided with careful planning. Also, there is no standard method of discovering pre-existing measurements.

A number of changes were made to the basic PingER system for its use in EDG: Initially, the lists of hosts to measure was downloaded from a central repository to try to avoid simultaneous measurements (this was more useful for IperfER and UDPMon) and to allow new measurement targets to be added to the entire system easily. This system was eventually substituted for the Probes Coordination Protocol, which provides better guarantees that simultaneous measurements will not occur.

PingER was also adapted to allow it to publish measurements into R-GMA via *edg-netmon2rgma*; this is explained more fully in §3.

2.3. IperfER

IperfER is a conversion of the PingER tool that makes use of Iperf[7] to make regular TCP throughput tests. As they are based upon the same codebase, both IperfER and PingER have similar functionality: graphical and data display, PCP integration and the ability to push measurements into R-GMA.

2.4. UDPMon

UDPMon refers to two pieces of software: a custom built measurement tool and another monitoring infrastructure based on PingER[10].

The measurement tool sends UDP packets to a target server at regular intervals. The server can calculate achievable UDP throughput, packet loss, and inter-packet delay variation (jitter), and the results are returned to the client.

The UDPMon monitoring infrastructure used in the European DataGrid has the same functionality as PingER: graphical and data display, PCP integration and the ability to push measurements into R-GMA.

2.5. Probes Coordination Protocol

The Probes Coordination Protocol (PCP) is a flexible group communication protocol specially designed for Grid network monitoring services. This protocol was developed within Work Package 7 of the European DataGrid project. The PCP control topology is based on a named ring that is traversed by a token. PCP integrates different concepts that have been proposed in a heterogeneous context involving networking, distributed algorithms and Grid security in a simple and open tool.

PCP enables the definition of a set of logical sensor groups (cliques) and also the interaction between the cliques. Scalability issues can be easily solved by creating a hierarchical structure of measurements; the basic requirement of mutual exclusivity between measurements is always satisfied as measurements are only allowed when the token is present. PCP also implements security mechanisms as well as many other features. A detailed description of the Probes Coordination Protocol can be found in [9]. The generalised version of PCP called Grid Coordination Protocol can also be used for configuration coordination and other distributed actions, and is detailed in [8].

The main principles of the protocol are as follows: clique members have a position on the ring, and each member can receive the token or pass it on. For simplicity, we define the ring as an ordered list of site agents like

$$Ring = [S_0, S_1, S_2 ... S_{n-1}, S_0]$$

The token definition contains the specification of the ring and the action (f) that will be actived on receipt of the token by each site agent. A list of arguments can be added to the action

$$Argl = [Arg_1, Arg_2, ..., Arg_n]$$

The token is defined with required and option temporal parameters such as a periodicity, and timeout and the authorised delay deviation.

When the token has been generated and initialised with its arguments, it is sent to the first member agent. When the token is received by a member, it is registered locally, the action is performed and the current time is saved. Then the token is transmitted to the next member node.

PCP is robust: if a token is lost, the next agent in the ring will time out and a new token is generated; duplicated tokens are discarded.

PCP is used for the scheduling of all network monitoring actions in the European DataGrid application testbed.

2.6. GridFTP

GridFTP[1] is an extension of FTP to allow various Grid-specific functions, such as Grid security, plus multi-stream and striped transfers. The implementation of the GridFTP daemon used in EDG was adapted from NCFTP. The adapted GridFTP daemon logs in two formats: a general text format and a NetLogger[12] format; the latter consists of key/value pairs separated by an equals sign (=), and provides more information than the former. A new log entry is added once every transfer to or from the daemon is complete.

3. Publication and archiving

Publication of network monitoring information in EDG is accomplished using the R-GMA system[2]. The R-GMA architecture is shown in Figure 1, along with network monitoring-specific components, which are explained below. R-GMA components are shaded. Clients publish information via an the Producer API. The Producer API and associated Producer servlets deal with advertising the producer through the distributed Registry, and the servlets deal with connections from Consumers.

To query R-GMA, a client uses the Consumer API. The API hides the complexity of fulfilling the query from the client, by presenting relational database-like tuples to the client. The Consumer servlet actually queries the Mediator to evaluate which Producers are generating data needed to fulfill the client's query. The Consumer servlet can then set up connections to each of the Producers to retrieve the data from them. Data can either be streamed from the Producers as it is updated (a push model), or requested with once-off queries (a pull model), depending on the type of Producers used. Streaming Producers store a short history of tuples in a buffer, to try to ensure that no information is lost when lots of new tuples are added in a short time.

The Mediator makes use of predicate information generated by the Producers. These predicates indicate limits on what data the Producer can and will produce. They are expressed as SQL WHERE clauses.

3.1. Publication from active tools

Producers need to be fairly persistent, so that data can be retrieved at any time by Consumers; however, the three active measurement tools are based on short scripts run at regular intervals (§2.1), and exit after all measurements in a batch have been made. There is no way to extend the lifetime of the Producers beyond the lifetime of the containing scripts. To circumvent this problem, a daemon program was created that would handle all of the producers required for a particular host. The daemon, *edg-netmon2rgma*, maintains connections to the servlets while the scripts are not running, ensuring that the connections do not time out by setting their expiry times beyond the time at which the next measurement is expected to be made. The layout is shown in Figure 1. Solid lines indicate the path of measurement data; dashed lines indicate the path of meta-data and queries. The components in the top left part of the diagram run on the network monitoring machine while the associated servlets (centre) may run on on any nearby machine. The bottom left box contains the archiving system. Registry and Mediator components hidden from the user are shown on the bottom right. The top right represents more Producers and associated servlets.

edg-netmon2rgma creates a Unix named pipe (or FIFO, for first-in first-out) that is used by the measurements scripts to contact the daemon. The scripts can send commands to the daemon; each command consists of a command name, plus any number of parameters separated by spaces.

Each measurement script, for PingER, IperfER and UDPMon, was modified to contact *edg-netmon2rgma* with each batch of measurements, to declare the information it will produce, and then again after each measurement at the same time as data is stored in the tool's internal database.

In the declaration phase, the tools send an "addmetric" command to *edg-netmon2rgma*. This command has a number of parameters: the

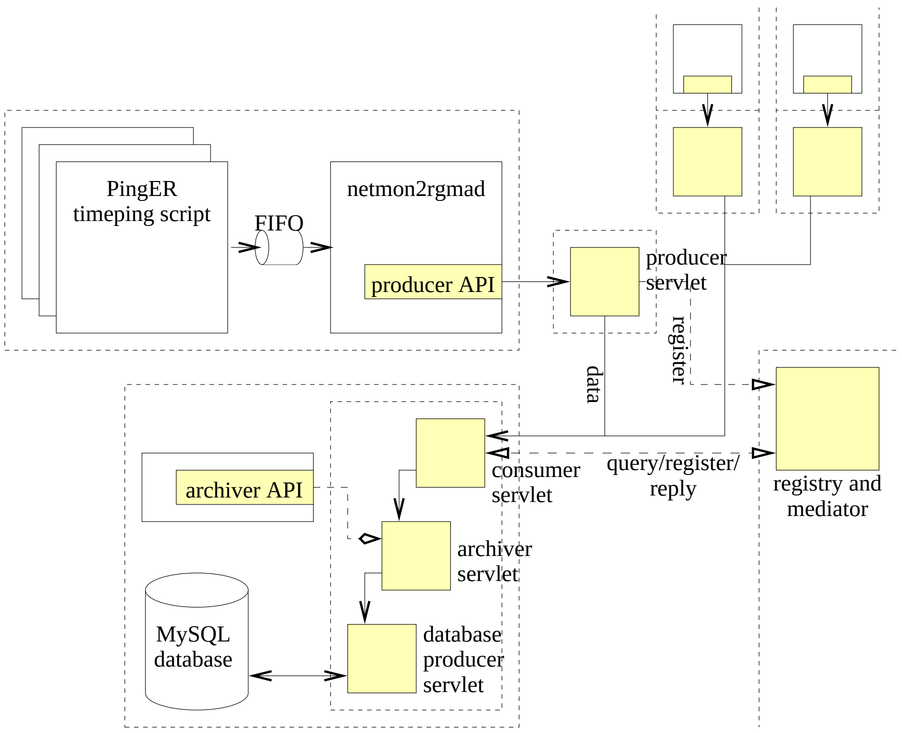

Figure 1. *edg-netmon2rgma* and associated components in the EDG monitoring architecture. The client in this case is an archiver. Solid lines indicate the path of measurement data; dashed lines indicate the path of meta-data and queries; shaded boxes are R-GMA components.

name of the tool, the metric being declared, an estimate of the number of measurements to be made at each go, and a list of fixed values that this tool will always produce. The combination of tool and metric is used to identify the tool in later communications. The list of fixed values is validated in a similar way to new results, see below. The fixed rules are used to generate the predicate which will be advertised by a Producer. Each *edg-netmon2rgma* daemon can support any number of Producers simultaneously, and so any number of measurement tools can rely on on daemon. A new Producer is created whenever a) the daemon has no Producer that produces the required table (or tables), b) the daemon has no Producer that matches the predicate for the data from the tool. Even if no new Producer need be made, the daemon ensures that the Producer buffer size is large enough to hold every measurement expected each time that tool's scripts are run.

As each script runs, it sends the results of each measurement to *edg-netmon2rgma*. Each measurement is sent in a "measurement" command. With each measurement result, the identifying tool and metric are transferred, as well as name/value pairs indicating the source and destination of the measurement, the parameters used in the measurement, and the result. The tool and metric are used to pick a Producer and are used to validate the name/value pairs. The name/value pairs are validated against general rules of the validity of these declarations, and rules specific to the metric being declared, in particular the valid names available, and the values they are allowed. IP address values are also converted to DNS hostnames.

Each script may send multiple measurements to *edg-netmon2rgma* each time it makes a measurement if more than one metric is measured. For example, a simple ping by PingER measures both round-trip time and packet loss.

3.2. Archiving

The Producers used by *edg-netmon2rgma* store a very short history. This history is maintained so that Consumers may always retrieve all information from the Producer, even if the Producer is updated more quickly than the Consumer can retrieve each update. However, historical data is useful for various purposes: network trouble-shooting (see §1.2) is a good example. The calculation of uncertainty on measurements and the calculation of predictions might also be useful uses of historical data, although these were not implemented in EDG. Historical data is stored in an archiving system. R-GMA provides an archiving system, where a streaming Consumer constantly evaluates a query. New tuples created by any Producer that match the Consumer's query are stored in a database for later retrieval. The tuples are then republished by a (non-streaming) Producer.

The network monitoring archiving system is designed to be distributable: the queries on each Archiver may be set so that only a subset of all possible network information is stored there. The R-GMA Producers themselves advertise a predicate indicating the content of the data they produce, so that the Mediator can calculate which Producers are required to fulfill a particular query. In this fashion, multiple Archivers may be configured to store different information for load-balancing purposes.

3.3. Publication from GridFTP

GridFTP statistics publication is handled differently from publication for active tools. The GridFTP server application generates a log file in NetLogger format containing information about each file transfer made to or from the server. The log file contains: the two hosts involved in the transfer, the direction of the transfer, the TCP buffer size, the

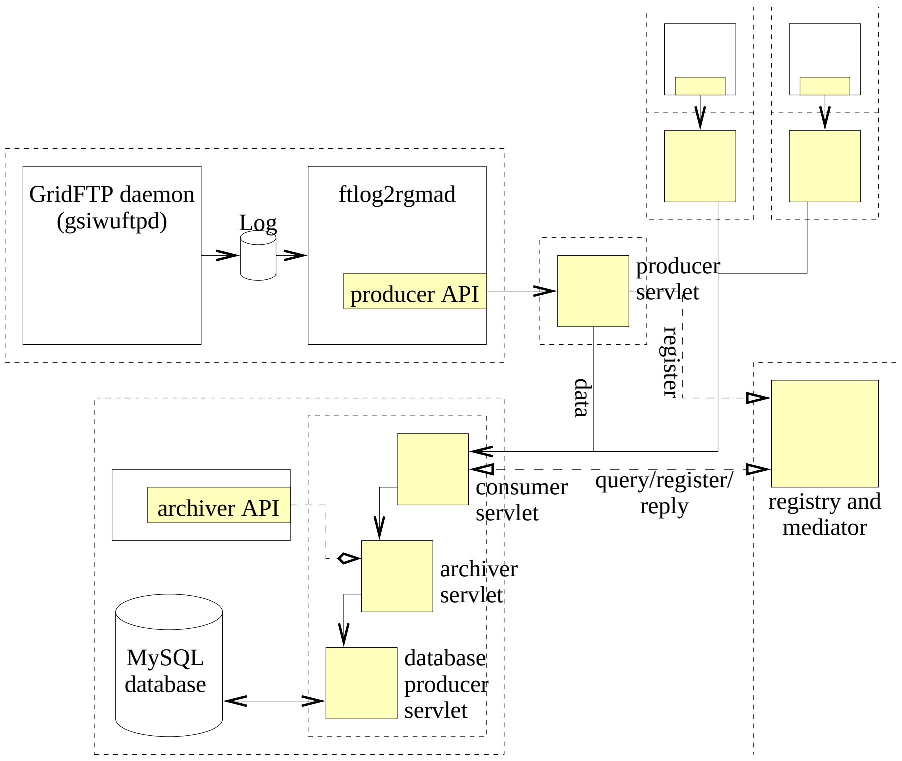

Figure 2. ftlog2rgma and associated components in the EDG Network Monitoring Architecture.

number of bytes transferred, the number of parallel TCP streams used to make the transfer and the number of stripes used. Each log entry also contains a start and end timestamp for the transfer, plus the name of the file transferred (which is ignored).

The GridFTP publication architecture is shown in Figure 2. A separate daemon, *ftlog2rgma*, periodically checks the log file for changes: if the logfile is extended (that is, the file size increased since the last check), *ftlog2rgma* assumes that new entries have been appended, and attempts to publish those entries into R-GMA; if the logfile is reduced in length, *ftlog2rgma* assumes that the log has been emptied by, perhaps by a utility such as *logrotate*, and so any entries in it are counted as new and therefore are published into R-GMA. Because it is a persistent daemon, *ftlog2rgma* can maintain the Producers it uses.

3.3.0.1. *Linking information* In order to be useful to Grid middleware, network performance information must be associated with Grid components other than the network monitoring machines. Two further tables of data are published by *netmon-info* daemons.

4. Publication schema

The publication schema is split into two types: measurement information and linking information.

Each measurement information table contains a value of a single metric, plus information about the measurement such as the source and destination hosts and the tool and protocol parameters used when making the test. For round trip time (as measured by *ping*), the maximum, minimum and mean of a small number of measurements is published rather than a single value.

Protocol parameters depend upon the metric and protocol:

TCP throughput The TCP buffer size used, the number of parallel streams, the duration of the transfer. Iperf measures TCP throughput.

ICMP packet loss The size of packets sent including the ICMP and IP headers. This is measured by Ping.

ICMP round trip time The size of packets sent. This includes the ICMP and IP headers. This is measured by Ping.

UDP throughput The packet size, the number of packets and the gap between each packet sent. This is measured by UDPMon.

UDP loss The packet size, the number of packets and the gap between each packet sent. This is measured by UDPMon.

UDP inter-packet delay variation The packet size, the number of packets and the gap between each packet sent. This is measured by UDPMon.

File transfer achieved throughput The filesize, the TCP buffer size and the number of parallel streams. GridFTP reports this.

Every tables also includes the time at which the measurement was finished, and the tool used to make the measurement.

File transfers can be made to or from the transfer protocol server, so neither the source or destination on a single Producerare fixed: this makes it impossible to create a predicate to aid the Mediator, create a selective archiver or for trouble-shooting information for an administrator. Therefore file transfer table also includes the hostname which hosts the GridFTP daemon. This field is always the same for a given server, so a more precise predicate can be generated.

There are two linking tables: one that links Storage Element identifiers to measurement machines, and one that links Computing Element

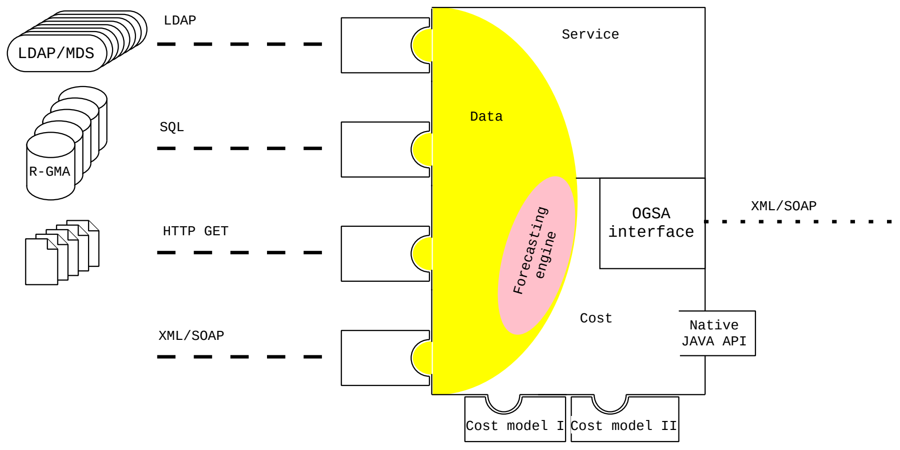

Figure 3. Modular architecture of network cost suite. Data storage plugins (left side) for communication with different data storages.

identifiers to measurement machines. These tables simply consist of pairs of Computing or Storage Element identifier and the name of a measurement machine.

5. Uses of network information: Network cost estimation service

The Network Cost Estimation Service [6] (NCES) was developed and deployed within the European DataGrid project to enable efficient estimation of network quality for network-based optimisation. NCES provides an estimation of the network quality for a given network communication. Quality is expressed as a cost, which might be any abstract measure of network quality. Due to the wide variety of different network communication types, different measures of quality must be used.

NCES is also designed to solve interoperability problems. A set of input service plugins enables access to any source of monitoring data, such as LDAP, MySQL or R-GMA. The service also publishes through different interfaces, such as via a Web Service or an OGSA compatible interface. Finally, a Java interface library is available that hides network communication from the application. Thanks to this design feature, the NCES can interoperate with storage and services based upon different standards. These features are illustrated in Figure 3.

5.0.1. *Use of the Network Cost Estimation Service*

Both resource brokering and replica selection in the EDG using NCES may be shown by a simple example.

A user wishes to submit a job j that requires a logical file F, which exists as replicas f_1, f_2, f_3 at storage elements S_1, S_2, S_3, respectively. The job may be run at one of two computing elements C_1 or C_2. The user submits the job to a resource broker. The resource broker discovers the locations of the replicas, and calculates the total cost for each combination of computing and storage element as $c_{11}, c_{12} \ldots c_{23}$ as the sum of the network cost for the transfer of a file from the storage element to the computing element and any other costs associated with the transfer of the file and the cost of doing the computation at that computing element.

The network cost for each pair could be estimated from file transfer information if it is available. With other instrumented applications or middleware, costs for different cost models could also be calculated. If no passively collected information is available, the network cost for each pair is calculated by first querying the linking tables to find the network monitoring nodes associated with the computing and storage elements. An estimate of the network cost between the storage element and the computing element can then be calculated from actively measured data between the two network monitoring nodes.

The resource broker chooses the computing element with the lowest total cost, and queues the job there[3].

When the job actually starts, it searches for the nearest replica in order to start using it using the *getBestReplica* function[4]. It compiles a list of costs for each storage element at which the replica resides c_1, c_2, c_3. Conceivably this could be a different set of storage elements if conditions have changed since the job was queued. The job then retrieves the best replica.

Acknowledgements

None of this work would have been possible without R-GMA, which was written by the members of Work Package 3 of the European-DataGrid. Their guidance and support has been invaluable.

References

1. Allcock, W., J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, 'GridFTP: Protocol Extensions to FTP for the Grid'. http://www.globus.org/datagrid/gridftp.html. Internal draft.
2. Cooke, A., W. Nutt, J. Magowan, P. Taylor, R. Byrom, L. Field, S. Hicks, et al.: 2003, 'R-GMA: A Relational Grid Information and Monitoring System'. In: *Proceedings of the 2nd Cracow Grid Workshop.*

3. EU-DataGrid Work Package 1: 2004, ‘Work Package 1 Final Evaluation Report’. CERN EDMS Id: 414780.
4. EU-DataGrid Work Package 2: 2004, ‘Work Package 2 Final Evaluation Report’. CERN EDMS Id: 406379.
5. EU DataGrid Work Package 7, ‘WP7 Network Metrics Archive Browser’. http://ccwp7.in2p3.fr/wp7archive/.
6. EU-DataGrid Work Package 7: 2004, ‘Final report on network and infrastructure and services’. CERN EDMS Id: 414132.
7. Gates, M., A. Tirumala, J. Ferguson, J. Dugan, F. Qin, and K. Gibbs, ‘Iperf’. http://dast.nlanr.net/Projects/Iperf/.
8. Harakaly, R., P. Primet, and F. Bonnassieux: 2004, ‘Grid Coordination by Using the Grid Coordination Protocol’. To be published in: *proceedings of the 4th IEEE/ACM International Symposium on Cluster Computing and the Grid*.
9. Harakaly, R., P. Primet, F. Bonnassieux, and B. Gaidioz: 2002, ‘Probes Coordination Protocol for Network Performance Measurement in GRID Computing Environment’. In: *proceedings of ISPDC, Iasi, Romania*. pp. 278–286.
10. Hughes-Jones, R., ‘UDPMon’. http://www.hep.man.ac.uk/u/rich/net/.
11. Matthews, W. and L. Cottrel, ‘PingER’. http://www-iepm.slac.stanford.edu/pinger/.
12. Tierney, B. and D. Gunter, ‘NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging’. LBNL Tech Report LBNL-51276.