DataGRID

NETWORK SERVICES:

REQUIREMENTS, DEPLOYMENT AND USE IN TESTBEDS

Document identifier:	**DataGRID-07-D7-3-0113-1-5**
EDMS id:	**(if diferent than nnnn)**
Date: (use "update field" Word function, right mouse button)	**06/06/2002**
Work package:	**WP7: Network**
Partner(s):	**CERN, CNRS, INRIA-RESO, INFN, PPARC, SARA, University of Amsterdam**
Lead Partner:	**INFN**
Document status:	**DRAFT**
Deliverable identifier:	**D7.3**

Abstract: *This document describes enhanced network services that address Grid middleware and Grid applications requirements for better end-to-end performance and optimised network transactions. Three service types are proposed and illustrated in detail: Differentiated packet forwarding, Transport protocol monitoring and configuration, Network-based Grid Optimisation.*

Delivery Slip

	Name	Partner	Date	Signature
From				
Reviewed by	Moderator and reviewers			
Approved by	PTB			

Document Log

Issue	Date	Comment	Author
1-0	May 27 2002	First draft	Editor: T.Ferrari Authors: S.Alessandrini, H. Blom, F. Bonnassieux, T. Ferrari, R. Hugues-Jones, M. Goutelle, R. Harakaly, Y.T. Li, M. Maimour, C.P. Pham, P. Primet, S. Ravot.
1.3	May 30 2002	Third draft	Editor: T.Ferrari Authors: S.Alessandrini, H. Blom, F. Bonnassieux, T. Ferrari, R. Hugues-Jones, M. Goutelle, R. Harakaly, Y.T. Li, M. Maimour, C.P. Pham, P. Primet, S. Ravot.
1.4	May 31 2002	Fourth draft	Editor: T.Ferrari Authors: S.Alessandrini, H. Blom, F. Bonnassieux, T. Ferrari, R. Hugues-Jones, M. Goutelle, R. Harakaly, Y.T. Li, M. Maimour, C.P. Pham, P. Primet, S. Ravot.
1.5	June 03 2002	Fith draft	Editor: T.Ferrari Authors: S.Alessandrini, H. Blom, F. Bonnassieux, T. Ferrari, R. Hugues-Jones, M. Goutelle, R. Harakaly, Y.T. Li, M. Maimour, C.P. Pham, P. Primet, S. Ravot.

Document Change Record

Issue	Item	Reason for Change
1.3	Structure of document	Experimental sections moved to Annex, changes to Deliverable according to P.Primet's comments.
1.4	Packet forwarding services	Earth observation use-case added, several typos and language problems fixed
1.5	Active reliable multicast and experimental results	Section about active reliable multicast removed; addition of section about Experimental Activities, Reference list completed, introduction about transport services added.

Files

Software Products	User files / URL

Word	DataGRID-07-D7-3-0113-1-5.doc (use "update field" Word function)

CONTENT

1 INTRODUCTION

1.1 OBJECTIVES OF THIS DOCUMENT

This document addresses three distinct but related goals. Firstly, it defines application and middleware requirements in three main areas: packet forwarding services, high-performance data transport and network-based optimisation of Grid middleware and applications.

For each category a set of services is defined and their applicability to the Grid is described. While most of the proposed services are ready for adoption in the DataGRID testbed, for a few of them more research is needed for a complete service definition and implementation plan. In this case the document illustrates the future work programme in that area.

Lastly, the document reviews the main achievements of the experimental and implementation work carried out for each service category in order to evaluate the effectiveness and the viability of what is proposed.

1.2 APPLICATION AREA

Services illustrated in this document are mainly applicable to Workload Management and Data Management, for enhanced and optimised middleware performance, and to user applications from High Energy Physics, Biology and Earth Observation.

1.3 APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS

Applicable documents

EDG-D7.1 *Network Infrastructure for Testbed 1;* DataGRID DataGrid-07-D7.1-0109-1-1, Work Package 7; http://edmsoraweb.cern.ch:8001/cedar/navigation.tree?cookie=1066787&p_top_id=1665390983&p_top_type=P&p_open_id=1411186599&p_open_type=P

EDG-D7.2 *Grid network monitoring. Demonstration of enhanced monitoring tools;* DataGRID DataGrid-07-D7.2-0110-8-1, Work Package 7. See previous reference for URL.

EDGrb *Definition of architecture, technical plan and evaluation criteria for scheduling resource management, security and job description;* DataGRID datagrid-01-d1.2-0112-0-3, Work Package 1; http://edmsoraweb.cern.ch:8001/cedar/navigation.tree?cookie=1066787&p_top_id=1665390983&p_top_type=P&p_open_id=1403460582&p_open_type=P

EDGreplica Stockinger, H.; Samar, A.; Allcock, B.; Foster, I.; Holtman, K.; Tierney, B.; *File and Object Replication in Data Grids*, Proceedings of 10th IEEE Symposium on High Performance and Distributed Computing (HPDC-10), San Francisco, California, Aug 2001.

EDGopt Bell, W.,H.; Cameron, F.,G.; Capozza, L.; Millar, P.; Stockinger, K.; Zini, F.; *Design of a Query Optimisation Service* – WP2 Data Management, WP2 internal document, work in progress.

GCOL Ferrari, T.; Hughes-Jones, R.; Primet, P.; *Collaborative investigations between DataGRID WP7 and DANTE;* May 2002.

Reference documents

DSarch Blake, S. et alt.; *An Architecture for Differentiated Service,* RFC 2475

DS Nichols, K. et alt.; *Definition of the Differentiated Services Field (DS Field) in the IPv4 and*

IPv6 Headers; RFC 2474

OWD — G. Almes et alt.; *A One-way Delay Metric for IPPM,* RFC 2679

IPDV — C. Demichelis et alt.; *IP Packet Delay Variation Metric for IPPM,* IPPM Working Group, work in progress

EF — Jacobson, V. et alt; *An Expedited Forwarding PHB*, RFC 2598

STO01 — Stockinger, H.; Samar, A.; Allcock, B.; Foster, I.; Holtman, K.; Tierney, B.; *File and Object Replication in Data Grids;* Proceeding of the 10th Symposium on High Performance and Distributed Computing (HPDC-10), San Francisco, California, August 7-9 2001

PAR93 — Parekh, A.H; Gallager, R.,G.; *Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case*; IEEE Transactions on Networking, Vol 1, N.3, 1993

DRR — Shreedhar, M.; Varghese, G.; *Efficient Fair Queueing using Deficit Round Robin.*

EDGwrr — Ferrari, T.; Mangiarotti, A.; *QoS testing on the Catalyst 6500*;
http://server11.infn.it/netgrid/index-dg.html

EDGdrr — Andreozzi, S.; Ferrari, T.; *Testing of MDRR scheduling on the CISCO GSR router*,
http://server11.infn.it/netgrid/index-dg.html

QBSS — *The QBONE Scavenger service*, http://qbone.internet2.edu/qbss

TEQUILA — Goderis, D. et alt; *Service Level Specification Semantics and Parameters;* draft-tequila-sls-02.txt, Aug 2002, work in progress.

IPPdef — Campanella, M.; Ferrari, T.; Leinen, S.; Sabatino, R.; Rejis, V.; *Specification and implementation plan for a Premium IP service;* Deliverable 9.1 of the GÉANT project (IST-2000-26417)

IPPimp — Campanella, M; *Implementation architecture specification for the Premium IP service;* Deliverable 9.1 – Addendum 1 – of the GÉANT project (IST-2000-26417)

FECH — Ferrari, T.; Chimento, P, F.; *A Measurement-Based Analysis of Expedited Forwarding PHB Mechanisms*; Proceedings of IWQoS 2000, Pittsburgh June 2000, IEEE 00EXL00, pp. 127 - 137

AF — Heinanen, J. et alt.; *Assured Forwarding PHB Group*, RFC 2597.

ARATE — Seddigh, N. et alt.; *An Assured Rate Per-Domain Behaviour for Differentiated Services;* draft-ietf-diffserv-pdb-ar-01.txt, work in progress

GOME — *GOME: the Global Ozone Monitoring Experiment* home page; http://auc.dfd.dlr.de/GOME/

NPERF — *The Netperf home page*; http://www.netperf.org/netperf/NetperfPage.html

EDGecn — Alessandrini, S.; Ferrari, T.; *Experimental evaluation of TCP performance with ECN optimization;* DataGRID test report, http://www.cnaf.infn.it/~ferrari/tfngn/tcp/ecn/

Iperf — *The Iperf traffic generator*, http://dast.nlanr.net/Projects/Iperf/

Mgen — *The Multi-Generator toolset,* http://manimac.itd.nrl.navy.mil/MGEN/

Tcptrace — *Tcptrace*, http://www.tcptrace.org/

igmp — W. Fenner;. *Internet Group Management Protocol, version 2.* IETF Request for Comment, RFC 2236

XTP — XTP Forum. *Xpress Transport Protocol Specification* , March 1995

DEE90 — S. E. Deering and D. R. Cheriton. *Multicast Routing in Datagram Internetworks and Extended LANs.* In ACM SIGCOMM'88 and ACM Transactions on Computer Systems, Vol. 8, No. 2, May 1990.

FJL95 S. Floyd, V. Jacobson, and Liu C. G. *A reliable multicast framework for light weight session and application level framing*. In ACM SIGCOMM'95, pp342--356.

PS97 S. Paul and K.K. Sabnani. *Reliable multicast transport protocol (rmtp)*. IEEE JSAC, Special Issue on Network Support for Multipoint Comm., 15(3):407--421, 1997

HSC95 W. Holbrook, S. K. Singhal, D. R. Cheriton. *Log-Based receiver-Reliable Multicast for Distributed Interactive Simulation*. In Proceeding of ACM SigComm'95

YGS95 R. Yavatkar, J. Griffioen, and M. Sudan. *A reliable dissemination protocol for interactive collaborative applications*. In ACM Multimedia'95 .

CHIU98 Dah Ming Chiu, Stephen Hurst, Miriam Kadansky and Joseph Wesley. *TRAM: A Tree-based Reliable Multicast Protocol*. Sun Microsystems Inc Tech. Report TR-98-66

PPV96 Christos Papadopoulos, Guru M. Parulkar, and George Varghese. *An error control scheme for large-scale multicast applications*. In IEEE INFOCOM'98 , pp1188--1996.

RC *Regional Centers for LHC computing;* the MONARC Architecture Group, http://monarc.web.cern.ch/MONARC/docs/monarc_docs/1999-03.html

GIS K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman; *Grid Information Services for Distributed Resource Sharing;* Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

PLM Koodli, R.; Ravikanth, R.; *One-way Loss Pattern Sample Metrics;* draft-ietf-ippm-loss-pattern-07.txt, IPPM Working Group, work in progress.

EIC93 T. von Eicken. *Active Messages: an Efficient Communication Architecture for Multiprocessors*. PhD thesis, University of California at Berkeley, November 1993.

PKC97 Pakin, S., Karamcheti, V., Chien, A. Fast messages (FM): Efficient, portable communication for workstation clusters and massively-parallel processors. IEEE Concurrency , 1997.

GPPTW01 P. Geoffray, C. Pham, L. Prylli, B. Tourancheau, and R. Westrelin. *Protocols and software for exploiting myrinet clusters*. In Proceedings of the International Conference on Computational Science (ICCS 2001) , number 2073&2074 in LNCS

PT98 L. Prylli and B. Tourancheau. *BIP : a new protocol designed for high performance networking on myrinet*. In Workshop PC-NOW, IPPS/SPDP98 .

VJ88 V.Jacobson, M.J.Karels; *Congestion Avoidance and Control*; ACM SIGCOMM-87, Aug 1988

UDPmon *The UDPmon toolkit*, http://www.hep.man.ac.uk/~rich/net

GEProbe Kolya, S.; *The Gigabit Ethernet Probe*;
http://www.hep.man.ac.uk/~scott/projects/atlas/probe/probehome.html

MotherNIC Hughes-Jones, R.; *Initial Performance Measurements on Gigabit Ethernet NICs with 64 bit PCI Motherboards*, Presentation at CeBIT 2002, Hanover.
http://www.hep.man.ac.uk/~rich/net/nic/Rich_GigEth_tests_Boston.ppt

IPPMf Paxson, V. et alt; *Framework for IP Performance Metrics;* RFC 2330

IPPMb Mathis, M; *A Framework for Defining Empirical Bulk Transfer Capacity Metrics;* RFC 3148

COT Cottrell, L.; *Bulk throughput measurement;*
http://www-iepm.slac.stanford.edu/monitoring/bulk/index.html#duration

VLBI *The VLBI Homepage at Jodrell Bank Observatory,* Dep. Of Physics and Astronomy, University of Manchester

ECN Ramakrishnan, K. et alt.; *The Addition of Explicit Congestion Notification (ECN) to IP*; RFC 3168.

RED Floyd S.; Jacobson, V.; *Random Early Detection Gateways for Congestion Avoidance*; IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.

WEB100 *The Web100 Project home page*; http://www.web100.org/

TTRACE *Tcptrace home page,* S. Ostermann, *http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html*

OPT02 Ferrari, T., Giacomini, F.; *Network Monitoring for GRID Performance Optimization,* Submitted for publication on Computer Communications, March 2002.

NLOG *The NetLogger Toolkit: End-to-End Monitoring and Analysis of Distributed Systems*, Data Intensive Distributed Computing Research Group; http://www-didc.lbl.gov/NetLogger

1.4 DOCUMENT AMENDMENT PROCEDURE

1.5 TERMINOLOGY

Glossary

AF	Assured Forwarding
API	Application Programming Interface
BE	Best-effort
CBR	Constant Bit Rate
CIR	Committed Information Rate
DIFFSERV	Differentiated Services
DRR	Deficit Round Robin
ECN	Early Congestion Notification
EF	Expedited Forwarding
GIIS	Grid Information Index Service
GRIS	Grid Resource Information Service
IPDV	Instantaneous Packet Delay Variation
LBE	Less-than-Best-Effort
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base
NIC	Network Interface Card
NRN	National Research Network
OWD	One-Way Delay
PIR	Peak Information Rate

QoS	Quality of Service
QBSS	QBONE Scavenger Service
RED	Random Early Discard
RTO	Retransmit TimeOut
RTT	Round Trip Time
SACK	Selective ACKnowledgement
SLA	Service Level Agreement
TCP	Transport Control Protocol
TRAM	Tree-based Reliable Multicast Protocol
UDP	User Datagram Protocol
WRR	Weighted Round Robin

2 EXECUTIVE SUMMARY

WP7 proposes the introduction of advanced network services for the DataGRID testbed to address middleware and application requirements in three different areas. The services we propose are: *Differentiated packet forwarding*, *Transport protocol monitoring and configuration* and *Network-based grid optimisation*. The above-mentioned services have been defined according to the requirements specified in collaboration with WP1, WP2, WP3, WP8, WP9 and WP10 of the DataGRID project, and on the basis of the application needs specified in [EDG-D7.1] and of the testbed performance results collected so far according to the monitoring infrastructure specified in [EDG-D7.2]. In this deliverable we refine the application requirements and show how the metrics gathered by our monitoring architecture can be used for enhanced Grid functionality.

For each of the three areas identified above this document presents the corresponding application and middleware requirements and proposes services that are described in terms of features, benefits and applicability. The viability of the proposed service is supported by a set of test activities that are documented in Appendix. The experimental work summarized in this document is a fundamental part of WP7 activities. The test programme is still ongoing and part of it will be carried out in collaboration with the European project GÉANT. Novel results will be documented in future deliverables.

The services we propose are the following.

1. **Differentiated packet forwarding**. Being the Internet mainly based on the connectionless communication mode of the IP protocol, the network protocol has no inherent mechanisms to delivery guarantees according to traffic contracts. Consequently, flows tend to experience varying network conditions that affect the original traffic profile. In order to guaranteed the predictability of Grid network transactions, techniques for Quality of Service support, like classification, metering, policing, marking, scheduling and shaping need to be introduced. Different services can be provided depending on the DataGRID application and middleware requirements, which can be grouped in four categories:
   - *Applications handling audio/video/image content* requiring low packet loss, one-way delay and IPDV minimization;
   - *Short-lived, reliable data transactions* requiring data transfer reliability and maximization of the number of completed transactions over time. Packet loss and delay minimization are important to reduce the number of retransmitted data units and for timely recovery in case of congestion;
   - *Bulk data transfer* requiring maximization of throughput. Packet loss rate needs to be minimized especially at high speed, given the penalty introduced by rate reduction every time a data unit is lost.
   - *Test applications and middleware* requiring total separation from production traffic in order to avoid resource contention.

   Experimental analysis of policing, marking, classification and scheduling (Weighted Round Robin and Deficit Round Robin) on high-end switch and router platforms indicate that Quality of Service techniques show satisfactory performance and provide effective traffic isolation. For this reason we propose the configuration and use of three services: *IP Premium*, *Less-Than-Best-Effort* and *Assure Rate*.
2. **Transport protocol monitoring and configuration**: most of the Grid applications and middleware components analysed by WP7 are based on *reliable* data transfer. TCP is the transport protocol most commonly used. However, it has been proven that the current congestion avoidance and congestion control algorithms supported by the protocol limit the efficiency in network resource utilization especially on high-speed data paths given the conservative approach adopted in case of congestion.

DataGRID applications are particularly sensitive to packet loss, which according to our experimental results is rather frequent independently of the connection distance (the loss distance is normally less than a few seconds) and characterized by loss bursts of up to 10 or more packets. This loss pattern applied to high-speed long-distance can severely limit the overall performance. For this reason, experimentation of mechanisms for proactive congestion avoidance like ECN has been carried out. Results prove the effectiveness of the mechanism in presence of multiple congestion points and in particular with interactive short-term TCP sessions. The interim results are very promising.

TCP-based performance is also critically dependent on windowing mechanisms whose tuning is fundamental whenever sessions are characterized by long round trip times. The effectiveness of signalling from destination to source, which is the basis of reliability and congestion control/avoidance, is penalized in case of long round-trip latency. For this reason we propose for any host critically dependent on TCP performance, the installation of a script for tuning of the default TCP kernel configuration and we propose the installation of tools for TCP monitoring and tuning at the application level like Web100.

In some application scenarios reliability is also needed when packets are multicast to a number of receivers. The *Tree-based Reliable Multicast Protocol* (TRAM) has been tested to verify its adequacy to the Grid. Interim test results show good functionality and scalability both in local and wide area Grids.

3. **Network-based optimisation of the Grid.** Grid optimisation, the third service we propose, addresses the problem of how the Grid can enhance its performance through the use of information on the status and transmission behaviour of its links. We propose the introduction of a novel Grid server, the *Network-based Optimiser,* which is a Grid functional block that can be queried by any existing Grid service or application and that provides an estimate of the transmission quality between two or more Grid nodes. The Network-based Optimiser relies on the availability of raw information on network performance provided by the Grid monitoring system and produces a high-level view of network performance through a set of internal *cost function*.

   Network-based optimisation can be applied to a large number of use cases, in the deliverable we focus on the following scenarios:

   - *Integration of network information into Resource Brokerage*;
   - *Gathering of distributed data in a single Storage Element*;
   - *Adaptive remote file access*.

   A prototype of network-based optimiser has been implemented to support two different cost functions for application of optimisation to Resource Brokerage and Data management.

We plan to evaluate the actual benefits of the proposed Differentiated forwarding services in collaboration with GÉANT by testing them on the network production infrastructure in combination with traffic produced by DataGRID applications. Experiments carried out in the transport protocol area will assist in the definition and implementation of the Assured rate services. In addition, TCP performance on very high-speed connections will be tested on the GÉANT infrastructure according to what defined in the GÉANT collaboration proposal [GCOL] and in coordination with the European project DataTAG. Research will be carried out to evaluate the effectiveness of protocol extensions and more advanced services for automatic TCP tuning will be analysed. Finally, the Network-based Grid optimisation prototype will be used to evaluate the advantages of grid optimisation, the appropriateness and accuracy of the cost functions implemented and the performance of the Grid Information Service. New achievements will be documented in future WP7 deliverables.

3 PACKET-FORWARDING SERVICES FOR NETWORK QUALITY OF SERVICE

Grids rely on the Internet for wide area communication. Being the Internet mainly based on the connectionless communication mode of the IP protocol, the network protocol has no inherent mechanisms to delivery guarantees according to traffic contracts. IP routers on a given data path from source to destination may suffer from congestion when the aggregated ingress rate directed to an output interface exceeds the output capacity. Consequently, flows tend to experience varying network conditions that affect the original traffic profile.

With term *Packet forwarding service* we indicate the capability of a packet network infrastructure, or of a set of independent networks, to transmit data packets in a differentiated fashion according to a number of distinguishing properties like priority, type of application that generated the packet, source, destination etc. Packet-forwarding services give the possibility to characterize the type of treatment packets are subject to, by defining the transmission profile experienced by a given *flow* or *traffic class* and by enforcing differentiation through a set of Quality of Service (QoS) function blocks:

1. *Metering*: the action of estimating the amount of traffic (instantaneous/average rate, burstiness etc) offered at a given metering point by one or more traffic classes, where the traffic class can correspond to a single *micro-flow* – identified for example by source/destination IP addresses, by the application port numbers etc – or to a set of individual micro-flows.
2. *Policing*: the action of limiting traffic through packet drop or remarking in order to make it comply to a given profile agreed by the customer and by the service provider.
3. *Classification and marking*: the action performed for each data unit that consists in the identification of the packet through a combination of IP header fields and in the binding of the packet to a given traffic class. Given the binding between a packet and a class, each packet can be tagged with a code-point that univocally identifies the class and simplifies the classification task in the following network nodes. According to the Differentiated Services architecture [DSarch], which will be described in the following paragraphs, the packet tag is called the *Differentiated Services Code Point* (DSCP) and is placed in the DiffServ field of the IP header [DS].
4. *Queuing:* the action of binding a packet to a given output queue in a network element egress point and of dropping packets according to a given specific algorithm (tail drop, Random Early Discard etc).
5. *Scheduling:* the policy adopted to select a packet for transmission out of a number of queues at a given time.
6. *Shaping:* the action of buffering data units in a packet memory area and of transmitting them according to a new traffic profile, so that compliancy of the original stream to a given traffic contract is enforced.

Unfortunately, the greatest majority of current network providers only support one basic packet treatment, the so-called *best-effort* service, according to which packets are forwarded identically assuming that all of them have the same priority. The best-effort service refrains from providing any type of transmission guarantee: packets simply share the same queuing structures at forwarding points and are scheduled for transmission according to the First-In-First-Out (FIFO) policy.

Despite of the fact that many network infrastructures are still best-effort, several research networks both in the US and in Europe (for example Abilene and GÉANT) are already providing advanced services and many European National Networks are planning to introduce packet differentiation. We think that the wide range of Grid-enabled applications and Grid middleware could greatly improve their reliability and performance if supported by specific forwarding services.

In this section we review the specific QoS requirements of DataGRID applications and middleware and we describe the QoS services and mechanisms needed to address such needs.

3.1 APPLICATION AND MIDDLEWARE REQUIREMENTS

Given the extremely large range of Grid-enabled applications and Grid middleware components, packet-forwarding requirements can be grouped in categories. We have identified four preliminary groups according the type of requirements specified by the Application Work Packages of this project (WP8, WP9 and WP10):

1. *Applications handling audio/video/image content*. Applications like videoconferencing, remote visualization real-time remote analysis of images, tele-immersion are examples of applications belonging to this group. In general this category is identified by applications performing remote processing of large databases of images and requiring remote visualization of the processed result. In addition, traditional videoconferencing applications producing audio and video traffic are used for computer-supported cooperative work. Medical applications (DataGRID WP10) belong to this category.
   In this case three are the critical parameters that can effect performance: packet loss frequency – for good video and audio quality and image resolution –, one-way delay [OWD] – for timely delivery of images and voice – and Instantaneous Packet Delay Variation (IPDV) [IPDV] for good audio quality.
   Expedited Forwarding [EF] in Differentiated Services is the standardized Per-Hop-Behavior (PHB) that exactly addresses these requirements.

2. *Short-lived, reliable data transactions*: Data-oriented applications accessing large amounts of small data portions like in remote file analysis – mainly for HEP applications (WP8) and Earth Observation (WP9) – and client/server transactions in GRID middleware fall in this category. Applications in this group require data transfer reliability and are particularly affected by packet loss, which reduces the data rate if congestion control and avoidance algorithms are used at the transport level. In addition, packet loss reduces the number of completed transactions over time, a parameter that is more critical than throughput itself, given the relatively small amount of data exchanged. One-way delay minimization is also important for timely communication between servers and clients.
   Active queuing management techniques like ECN and RED seem very promising to reduce packet loss recovery and can detect congestion proactively. In addition, the availability of guaranteed bandwidth could help applications in this category to avoid contention in bandwidth usage. Experimental results are described in ANNEX A.

3. *Bulk data transfer*. Data management operations causing replication of large database portions [STO01] and jobs accessing very large data collections are examples of middleware software and applications moving a very large amount of packets across the network. The difference between this group and the previous is related to the amount of data exchanged and the frequency of network transactions – bulk transfers are likely to be rather infrequent but with well-defined scheduling. In this case throughput achieved for each data transfer is the critical parameter since it determines the data exchange completion time and the efficiency in network resource utilization. Throughput with reliable transfer protocols like TCP is critically influenced by the packet loss rate and loss pattern experienced during transmission. In fact, for every lost data unit the output rate at the source is dynamically reduced in order to cope with congestion, for this reason packet loss is highly undesirable, especially when running on high-speed infrastructures. The rate penalty introduced by congestion control algorithm is proportional to the output rate reached at congestion time. Then, the packet loss pattern – isolated packet loss vs. packet loss bursts – determines the efficiency in resource utilization at packet retransmission time.

For this group the guarantee of a minimum bandwidth is important to estimate the transaction finish time and to avoid network bottlenecks produced by multiple concurrent transactions for a given data source. Also, parallel data transfer and optimizations of the TCP stack could greatly help to improve performance.

QoS needs are expressed in terms of *guaranteed delivery* of a complete data file or in throughput predictability, while the priority of individual packets is transparent to the application. The type of required services differs from classical QoS offerings in that the application or middleware may want to specify the ultimate delivery time or a target throughput or a stability-level for data-delivery.

The ultimate goal is to allow job schedulers and brokers to co-schedule data transfer/processing and to determine the data sources from which information can be delivered to users and jobs more efficiently.

Assured Forwarding [AF] in Differentiated Services is the Per-Hop-Behavior (PHB) that addresses the above-specified requirements and can be used to differentiate in-profile packets, from out-of-profile ones according to three different code-points. ECN and active queue management can be used to avoid congestion proactively in order to reduce the amount of lost packets.

4. *Test applications.* Any application and middleware component running for test purposes and producing contention with production traffic falls into this category. It is desirable that during test sessions, applications producing large amounts of test traffic and using a considerable portion of network resources at network bottlenecks are isolated, so that production traffic is always given higher priority in case of resource shortage. Nevertheless, test traffic still needs a minimum amount of guaranteed resources in order to ensure correct functionality of test applications and middleware.
   Non-elevated services like LBE address this specific class of requirements. For this type of service any experimental (i.e. non-standardized) code-point can be used.

Annex A illustrates the experimental activities that were carried out in order to investigate functionality and performance of the techniques requested to address the packet forwarding requirements herein specified. Experimental results prove the viability of techniques for packet differentiation, even when applied to high-speed network infrastructures.

The following paragraphs illustrate two DataGRID Quality of Service use-cases. They provide an example for the Middleware and Applications community of how application requirements can be combined with Differentiated packet forwarding techniques for enhanced performance.

The earth observation use-case

The *Global Ozone Monitoring Experiment* (GOME) [GOME] algorithms developed by ESA and KNMI are representative of a typical Earth Observation Application Model. Five years of GOME data have been archived at ESA (Italy) and KNMI (The Netherlands). A typical data processing run analyses one-year data to compute a Global Ozone Profile Map. The amount of data involved in the processing phase consists of 3650 data files of 15 Mby each, i.e. a total amount of information equal to 55 Gby.

In order to run data processing in a distributed environment, files need to be replicated to DataGRID Storage Elements and processed using DataGRID Computing Elements. Initial data sources are in Frascati (Italy), KNMI (The Netherlands) and IPSL (Paris), while DataGRID sites supporting Grid facilities include CERN, Lyon, NUKHEF and INFN-CNAF.

A second use case of particular interest is about processing of data acquired by satellite instruments, where the amount of information acquired daily is approximately 300 Gby. In this case data transfer rate is of fundamental importance, since a minimum guaranteed rate of 8 Mbps is needed for timely delivery, while at peak times (for example at night), sustainable rate should be up to 34 Mbps. In this case data is acquired at Kiruna (Sweden), Frascati (Italy) and Matera (Italy) and needs to be available for processing at KNMI (The Netherlands) and DLR (Germany).

Earth observation applications critically depend on throughput and latency. Throughput needs to be maximized especially when very large files have to be exchanged (300 Mby) daily, while low round-trip latency is particularly important for the interactive-life data transactions that involved short files of approximately 1 Mby.

The biomedical use-case

In this paragraph we provide an example of how Grid applications from WP10, namely *Content-based query of medical image databases* and *Remote interactive applications*, can benefit from QoS services in IP networks. The former is based on the distribution of large data sets over the network while the latter requires a constant data flow between a remote site and the user interface.

Medical imaging is data intensive. For instance, for 3D images made of 50 to 100 slices, size can vary from 25 to 50 Mby. Similarly, the typical 3D Magnetic Resonance image size is 16 Mby. Normallly, only no-loss compression techniques can be applied to medical images, with resulting poor compression ratios. The biomedical use-case consists of two distinct applications.

1. *Application 1: Content-based query of medical image databases.* Digital medical images are stored in databases. The data structures to be accessed are simple flat files: one file for each (2D or 3D) image. Meta-data about image files (e.g. patient name, acquisition type etc.) are usually stored in relational tables and used for querying the available data.

   Beyond simple queries on meta-data, physicians may want to search for images close to a sample they are interested in, so that they can study cases similar to the image they are looking at. This involves an analysis of the image contents to determine images from a database that are similar to a sample image. The comparison of a sample image against a complete database may become intractable due to the size of medical databases. However, the computation can be easily distributed on a computation grid since the comparisons of the sample image with each image of the database are independent from each other.

   Computations should be fast enough for a reasonable use in clinical practice (of the order of a few minutes). Given the rather low similarity computation time (from few seconds to 15 minutes on a 1GHz Pentium processor), the transmission of images to computing sites may become a bottleneck. For this reason, guaranteed sustainable throughput is the main requirement, where the actual throughput value depends on image processing time and database size.

2. *Application 2: Remote interactive segmentation.* For some medical image analysis applications require a human supervision both for technical and legal reasons. These applications usually run locally on the user computer so that a visual feedback can be returned and an interactive session is possible. However, some computation-intensive applications run locally and require the grid computing power to execute at an interactive rate. These applications send their visual feedback through the network to the user interface.

   A continuous data exchange between the two machines is needed, where the actual rate varies from 27 Kbps for a refresh every three seconds, to 164 kbps with two refreshes per second, this in case of 2D images. In case of 3D images bandwidth varies from 5.3 Mbps to 32 Mbps.

3.2 PROPOSED SERVICES, APPLICABILITY AND BENEFITS

The delivery of network services for the support of packet forwarding differentiation is of particular complexity since a seamless end-to-end support of guarantees is needed even when a large number of service providers are involved. This means that given the large span of Grid infrastructures, implementation of services has to be adopted in several domains. This implies the introduction of a set of enhanced features in each network node providing packet differentiation, and the establishment of a set of Service Level Agreements between neighbouring network domains.

Despite of the inherent complexity of the problem, we think that the use of advanced packet forwarding services could greatly improve the reliability and the performance of some classes of Grid traffic, as indicated by the good experimentations results collected so far. In addition, given the distributed nature of the Grid, we think that collaboration between network providers and the Grid community could be greatly beneficial to both for a quicker and smooth transition from the plain best-effort service to QoS-capable networks.

In the following sections we present the type of QoS architecture that is requested for the implementation and use of QoS in Grid testbeds and we propose the adoption of three services that have been selected in order to match application and middleware requirements with services currently under analysis/implementation is some major research networks.

3.2.1 Network architecture for QoS support

We propose the adoption of the Differentiated Services (diffserv) architecture. The main advantage stems from the fact that no protocol needs to be supported in *each* network element traversed by priority Grid traffic. Diffserv packet forwarding guarantees are offered to *traffic aggregates* and not to individual *micro-flows*. This approach has the advantage that aggregate packet treatment requires fewer network re-configurations thanks to the statistical multiplexing of multiple flows within aggregates. However, traffic aggregates imply a coarser QoS granularity as no protection between individual flows within the same aggregate is supported.

In Diffserv it is assumed that over-provisioned network nodes, especially at the core of a given domain, do not need any special QoS support. The complexity of QoS support is at the edge of the network domain: at egress points classification, (re)marking and policing is need, while at egress points traffic shaping may be requested to ensure that the profile of aggregates at domain boundaries complies to the service specifications agreed.

The QoS implementation adopted in a given domain is completely independent and transparent to the rest of the network. This means that on a given QoS-capable data path the existence of best-effort domains is possible, provided that the corresponding links are correctly over-provisioned. In general, packet scheduling is only needed at congested interfaces. This simplifies the gradual and incremental support of QoS in the Internet.

At the very edge of a given data path, i.e. in Grid testbed sites, the following actions are requested.

- Traffic using a specific service needs to be classified and marked with the code-point (DSCP) agreed with the neighbouring domain, since the DSCP identifies the class of service to be applied to a given packet. Classification and marking can be either supported in LAN

[1] Examples of policing configuration for the Catalyst IOS 12.1(8b)EX2 are in Appendix.

[2] DRR configuration syntax and examples are provided in Appendix.

[3] Results and conclusions will be part of deliverable D7.4.

switches or at the egress of the LAN, i.e. in gateway routers. The first approach is recommended since it scales better.

- Policing has to be enabled to make sure that only traffic sourced by entitled users is marked with a given DSCP and to enforce compliance of the traffic aggregate to the profile agreed with the neighbouring domain. The latter action is needed since several users can normally inject traffic for a given aggregate. Policing can be enabled at collective points of LAN switches (for example at the output interface connecting the testbed LAN switch to the rest of the network) or at the input interface of the general LAN egress point, i.e. at the gateway router.
- A Service Level Specification (SLS) [TEQUILA] has to be established with the neighbouring domain providing connectivity to the destination host/site priority traffic is destined to. The SLS contains a set of parameters and their corresponding value, where parameters vary depending on the service under consideration (destination host/network, delay, packet loss, sustainable rate, peak rate, etc). In general, the establishment of an end-to-end service requires the definition of a SLS chain between neighbouring domains on the data path from source to destination. Metrics for service level description include:
  - *Bit Rate (r):* the rate at which an application's traffic must be carried by the network;
  - *One-way delay (d):* the delay that an application can tolerate in delivering a packet of data;
  - *Instantaneous packet delay variation (j):* the variation in one-way delay for consecutive packets;
  - *Loss (l)* : the percentage of lost packets.

  Monitoring of such metrics, performed according to the approach proposed in [EDG-D7.2], is important for end-to-end service level auditing.

3.2.2 IP Premium

IP Premium is the service based on the Expedited Forwarding Diffserv PHB [EF] defined and supported by the GÈANT European research network. IP Premium addresses the requirements specified in Application category 1, i.e. it minimizes one-way delay, instantaneous packet delay variation (IPDV) and packet-loss. As such, it's particularly suitable to applications with real-time requirements and requesting very low packet-loss, like critical middleware transactions between client and servers. Packet-loss, delay and IPDV can be controlled through the use of strict-priority schedulers: given a set of independent queues, whenever a new candidate packet has to be selected for transmission the strict-priority queue is checked first and if one or more packets are present, all the packets of that queue are serviced first.

We recommend the use of IP Premium for critical middleware transactions and for applications used in WP9 and WP10. For good end-to-end service the use of strict priority queuing or equivalent scheduling algorithms is recommended when either short or long-term congestion occur at the output interface connecting a Grid testbed site to the respective National Research Network. If possible, also the NRNs involved in the IP Premium end-to-end data path should support the service.

For more information about the IP Premium definition and implementation refer to [IPPdef, IPPim, FECH].

3.2.3 Less than best-effort

Given the good test results collected in experimentations carried out so far, we recommend the use of Less-Than-Best-Effort services[4] like QBSS for isolation of test traffic from production traffic (Application category 4). In particular, we encourage the test of the service as soon as possible, especially in Grid sites suffering from long or short-term congestion on the link connecting the LAN to the NRN, in order to assist middleware and Grid applications during test phases.

In addition, we recommend the use of this service for testing of applications based on parallel ftp or on modified TCP stacks when carried out on production networks, in order to prevent bandwidth contention and fairness problems between modified and traditional TCP applications.

Since LBE services do not provide end-to-end guarantees, LBE can be easily adopted incrementally in congestion points where traffic isolation is needed without requiring the establishment of a chain of SLSs between independent domains.

3.2.4 Assured rate services

We think that services for the provisioning of guaranteed minimum bandwidth is particularly important in order to address the needs of both data and transaction-intensive TCP-based applications. The *Assured rate* service aims at guaranteeing an end-to-end minimum sustainable rate between two defined points and "*ensures that traffic conforming to a committed information rate will incur low drop probability*" [AF, ARATE]. The interest in this service is motivated by the need to improve the reliability of TCP sessions through the harmonization of the access of multiple TCP clients to the same data source.

Unfortunately both the definition and the engineering of TCP-oriented services have received little attention from the Internet Engineering Task-Force and from network providers because of the inherent complexity of the task. TCP applications produce bursty traffic, where burstiness is a function of a large number of application and source-dependent parameters, among which the type of TCP stack in use, Round Trip Time and the type of interface connecting the source host to the network (see section *High-performance and novel transport protocol services*). For this reason, the characterization of a TCP aggregate profile is reflected in the difficult engineering and tuning of a seamless end-to-end service.

WP7 will focus on the research areas specified in the following in order to develop a service definition and a set of implementation guidelines to be proposed to the European research network operators.

The aspects subject to further investigation are the following:

- effect of *aggregation* of hundreds of TCP streams in a single traffic class (different depending on the aggregation degree) on the aggregate traffic profile;
- effect of *systematic errors* like router and operating system bugs on packet loss and achieved throughput;
- frequency of both short and long-term congestion;
- tolerance of policing to *aggregate burstiness* in presence of sources connected with high-speed interfaces for token bucket depth configuration, scheduling queue length and RED threshold tuning;
- evaluation of shaping in terms of benefits – for example as a mean to control burstiness – and penalties – additional queuing delay and buffer size tuning issues.

[4] The definition and experimentation of the LBE service is presented in Annex A.

4 HIGH-PERFORMANCE AND NOVEL TRANSPORT PROTOCOL SERVICES

In this section we present the requirements of applications and Grid middleware in terms of transport services. In particular, we focus on two aspects: the need to efficiently and reliably distribute the same data set to a number of receivers and the need to enhance the existing reliable transport protocols when adopted on high-speed long-distance networks.

Packet duplication can be avoided by a specific packet-forwarding mode called *multicast*, which consists in the simultaneous transmission of data units from one source to multiple destinations. Multicast avoids communication bottlenecks at sources by making sure that only once can each network link be traversed by an instance of a given packet. Reliable multicast is an extension of the basic multicast communication mode that additionally guarantees data integrity through the retransmission of lost or corrupted packets.

Transport protocol performance can be enhanced through a set of different technical approaches. Tuning of transport protocol parameters in the kernel and tuning/monitoring of transport protocol variables *at the application level* are the basic recommendations for protocol performance enhancement. An additional number of technical approaches ranging from packet loss minimisation to proactive congestion detection are analysed and the interim experimental results are presented in Appendix B.

The following paragraphs illustrates the Grid middleware and application requirements collected so far and describe the service proposed to address such needs.

4.1 APPLICATION AND MIDDLEWARE REQUIREMENTS

4.1.1 Reliable multicast

Multicast transmission mode handles one-to-many communications in a wide-area network with the lowest network and end-system overheads, mainly by avoiding packet duplication on data paths. Unlike best-effort multicast, which typically tolerates data loss (traffic produced by real-time audio or video applications is an example), reliable multicast requires that all packets transmitted by the source are safely delivered to all the destination hosts. Desirable features of reliable multicast include, in addition to reliability, low end-to-end delays, high throughput and scalability.

There can be a large variety of Grid infrastructures but in most cases they use a similar network architecture: local computing resources are connected with a variety of LAN technologies – Fast and GigaEthernet, Fiber Channel, SCI, Myrinet etc – and gain access to the wide area network through a sequence of one or more routers.

WAN infrastructures often connect the distributed local Grid sites with high-speed core backbones, such that the Grid site local loop often represents a network bottleneck.

Multicast, both in its reliable and unreliable mode, is a packet communication mode that naturally suites the Grid. In fact, in order to improve robustness and fault tolerance, servers and databases are often replicated in the Grid and in particular multiple instances of peer servers are intertwined by a mesh of communication sessions. In this scenario multicast can improve the efficiency of one-to-many network transactions.

Data management. Data management is one of the central functionalities of most computational grids. Included in this data management general term would be data replication, code and data transfer (job submissions), data communication for distributed applications, databases operations, directory related messages etc. These applications have different communication requirements, but for all of them reliability is the unifying common factor.

Database replication can take great benefit from the use of multicast when copying very large portions of data from a given source site (e.g. from a Tier 1 site) to multiple remote destination sites (e.g. Tier 2 and Tier 3 national sites) [RC]. In fact, data replication sessions need to be sufficiently frequent to grant worker nodes an efficient local access to large portions of data in order to minimize data access latencies and to avoid communication bottlenecks at given Grid sites. However, multicast communication is applicable only in case of *push* transfer mode, according to which it's the data source that coordinates and synchronizes the transmission of a data fraction to multiple remote sites. The push approach requires write access to remote sites. On the other hand, in the *pull* mode it's the remote site that requests a given file to the upper-level file server; this "on-demand" approach does not imply synchronization between remote sites.

While in the push approach reliable multicast can be effectively used thanks to the existence of multiple receivers at the same time, in the pull approach it can be used only if synchronization between independent remote sites exists, so that if multiple requests of the same data set are received in the same timeframe, the central data source activates the data transfer at the same time. Only in this way the interested receivers can be part of the same multicast group. This session initialization procedure is needed to avoid the need of caching for the support of late subscriptions.

In both scenarios multicast communication should be completely transparent to the data server.

For other network requirements such as bandwidth and latency, a periodic data transfer could benefit from high bandwidth, while low latency is not really necessary. Such requirements characterize most of the data-oriented transfer applications. For this reason, packet loss minimization or active queue management techniques for proactive congestion control are of great interest. On the other hand, data transfers for interactive job submissions usually need both high bandwidth and low latency to effectively support interactivity. Distributed applications such as distributed simulations fall into this category.

The efficiency of the packet loss recovery mechanism depends on the multicast scenario adopted, for example on the locality of receivers belonging to the same group. For instance, a distributed application running on the Grid may imply the use of several computing resources per site – for example in presence of job decomposition – and each computing resource could subscribe to the multicast group in order to receive data individually. On the other hand, a periodic data replication service for providing access to scientific data, such as data from the CERN Large Hadron Collider (LHC) physical experiments would generally involve a single source and very few storage servers per recipient site. While the effectiveness of local packet loss recovery is rather limited in the second case – due to the presence of a single receiver per site – recovery latency could be highly reduced in the former one. Given the very large amount of data that needs to be replicated in the latter case, stripping, parallel transfer and check-pointing methods should be supported in addition to reliable multicast.

GRIS-GIIS and inter-GIIS communication. A second interesting application scenario that could benefit from reliable (but also unreliable) multicast is communication between GRIS (Grid Resource Information Service) and GIIS (Grid Index Information Service) instances [GIS]. The GIIS acts as data cache for the information provided by the GRIS and it also organizes a group of GRISes in a coherent way. Every time a registration occurs, the GIIS deletes expired registrations. Expiration is determined by comparing the previous registration TTL with the time of the file containing the registration. Two types of registrations are possible: *direct* and *cachedump*. For direct registrations information the corresponding GRIS has to be queried directly, while in the latter case valid data is extracted from the GIIS cache data cache and only expired data is re-fetched from the GRIS.

In case of multiple *virtual organizations* (VOs) a given local GRIS may belong two or more different VOs. In this case registration to multiple VO GIIS is required. With the support of multicast registration packet duplication and network bottlenecks can be avoided. Similarly, inter-GIIS communication could take benefit from multicast, as a given GIIS can register to multiple GIIS instances. Since the registration process is periodic, unreliable multicast could be sufficient: in case of packet loss transactions could be simply aborted. However, reliable multicast may prove to be more effective in order to reduce registration latency and for a more timely update of data in caches.

4.1.2 TCP and UDP performance

The Applications that will be used in the DataGRID project and the Grid middleware will generate and use large amounts of traffic with different high performance requirements for data exchanged between geographically distributed nodes.

According to our estimation, the initial deployment of grids and user applications, for particle physics alone, will produce very large volumes of data, with at least 200 Mbps of continuous traffic from many sites. In addition, similar amounts of intermittent traffic are expected.

Similarly, many of the middleware components of distributed systems, like authentication, database replication and the exchange of jobs and input/output data, require reliable high-speed communication among grid nodes like computing elements, storage elements, resource brokers and information servers.

The most widely deployed transmission protocol is TCP. However, it has been proven that the current congestion avoidance and congestion control algorithms supported by the protocol limit the efficiency in network resource utilization especially on high-speed data paths given the conservative approach adopted in case of congestion.

One of the purposes of the DataGRID project is to implement a working distributed Grid computing system spanning Europe and serving a large pool of applications in a range of different scientific areas, like high energy physics, earth observation and biology. The DataGRID testbed will be also interconnected with remote grid platforms in the US, so the performance of applications and hence high performance networking over long-distance links is of primary importance.

For these reasons, the optimisation of high performance, high throughput network transport protocols, and especially TCP/IP, both in terms of configuration and stack implementation is particularly important. There is a clear need to maximize the network transport performance and efficiency in order to maximise the compute and network resource utilization. These considerations apply to both the throughput and latency of network connections.

4.2 PROPOSED SERVICES, APPLICABILITY AND BENEFITS

4.2.1 Reliable multicast

In what follows we provide an overview of solutions for support of reliable multicast services covering a range of approaches.

1. *IP multicast protocol suite*: IP multicast and its associated extensions for hosts and routers (RFC 1122 and [DEE90]) enable multi-point communication on the Internet and LANs. Multicast addresses from 224.0.0.0 to 239.255.255.255 (class D addresses) are used to identify logical groups that can contain thousands of participants geographically distributed over the Internet.

Group management is handled by the IGMP protocol (Internet Group Management Protocol, currently in version 2 [igmp]) and subscriptions to a logical group are at the receiver's initiative with the help of the local site router that periodically sends QUERY messages at specific reserved multicast addresses (for instance all multicast-enabled hosts are required to subscribe to the 224.0.0.1 group address). Hosts that wish to join a given multicast group inform the local site router through REPORT messages. The main advantage of IP multicast over the IP unicast protocol is the router-embedded packet duplication functionality at the cost of per-group state information maintenance.

2. *Adding reliability*: IP multicast has no mechanisms for the support of reliability, which consequently has to be added at a higher layer. A number of approaches to reliable multicast relying on complex exchanges of feedback messages – ack and nack messages – have been proposed in the past. XTP, SRM, RMTP and TMTP [XTP, FJL95, PS97, YGS95] are a few examples. Most of them fall into one of the following classes: *sender-initiated* or *receiver-initiated*.

   In the former approach the sender is responsible for both loss detection and the recovery (XTP). This approach suffers from poor scalability due the ACK implosion problem at the source in case of many receivers. On the other hand, in the latter approach loss detection is performed by receivers – NACKs are used instead of ACKs – however, scalability problems may still arise when a large number of receivers subscribe to the multicast session.

   Local recovery is an alternative approach that solves the implosion problem at the source by: (i) using some receivers or dedicated servers as repliers (in some cases with a router-assisted solution), or (ii) using network elements such as routers for caching. In the first category, the replier could be any receiver in the neighbourhood (SRM), a designated receiver (RMTP), TMTP, TRAM [CHIU98], LMS [PPV96], PGM [PGM]) or a logging server (LBRM [HSC95]) in a hierarchical structure. Most of these approaches have no exact solution to scalability problem because of their lack of topology knowledge.

3. *Application-level and end-system multicast:* All the approaches described so far rely on the existence of a dedicated multicast support at the network layer, as proposed by the IP multicast approach (RFC 1112). On the other hand, application-level multicast handles all the aspects necessary for multi-point communications within the application. The mechanisms supported include: group management, tree construction and packet replication (with several unicast connections).

   It is also possible to use the so-called *end-system multicast*, according to which IP multicast support and reliability are added on top of the loss-prone best-effort infrastructure and handled directly by end-systems (hosts or edge routers) with the construction of an overlay network where unicast communications are used. In this way new infrastructures are used (IP multicast for instance) and most mechanisms for error and congestion control are used. Both application-level and end system multicast are suitable to small size or sparse groups and may show poor scalability in large grid infrastructures, but may still prove to be adequate for multicast support to the Grid Information Service scenario.

With regards to packet loss recovery, we recommend recovery to be performed within the same sub-net especially when computing resources are connected at Gbit/s rates to the rest of the Grid infrastructure. Packet loss normally occurs in routers due to limited memory buffer space and/or to output link congestion – packet drop can occur either before the routing decision has been taken or

afterwards. Recent studies show that nowadays packet loss probability is quite small in backbone networks, and losses mainly occur in edge/access.

Now, if a packet is successfully transmitted on a medium-speed LAN (10 to 100Mbit/s), then it is most likely that all receivers in that given sub-net will be able to get it. However, this is not true anymore for high-speed LANs operating at Gbit/s rates where packet loss frequently occurs in the last layers of the protocol stack if standard OS and IP stacks are used [EIC93, KC97, PT98, GPPTW01].

4.2.2 Script-based socket buffer tuning

We propose to include an init script in the EDG software release for automatic tuning of the standard TCP kernel configuration in any EDG machine.

The init script file `socket_conf` , available in the Appendix of this document, can be used to set the socket configuration at initial startup for Linux version 2.2 and 2.4[5]. The script sets a set of appropriate parameters in the `/proc` pseudo file system. For high-throughput TCP connections it is required that the TCP window size – the internal TCP parameter used by the admission control and congestion avoidance algorithms [VJ88] – is appropriately set according to the following equation:

$$TCP\_window = RTT * Bandwidth$$

The motivation behind the formula is that both in sending and receiving end-system a sufficient amount of kernel memory should be reserved to sockets in order to buffer unacknowledged data. Send and receive socket buffer size is also used for the setting of the maximum TCP window allowed. For this reason, the larger the RTT, the larger is the amount of data that needs to be transmitted without interruption to keep the sending process active. Unfortunately, in most of the standard operating system the minimum, average and maximum socket buffers allowed by the kernel is set to a default value which is often not sufficient for long-distance TCP connections (for example 64 Kby). This is why a script run by default at boot time setting the default minimum, average and maximum socket buffer sizes to a much larger value can help to improve the overall performance of long-distance TCP sessions.

4.2.3 TCP monitoring and tuning

One of the issues that TCP-based applications see is the lack of feedback information on the end-to-end performance experienced over time at the TCP layer (like the number of TCP segments retransmitted, the average and instantaneous size of the TCP window etc). For this reason, the availability of tools that provide both means to monitor and tune TCP parameters would be greatly beneficial. Prototypes of such tunnels have been developed recently in the framework of a number of projects. Web100 is one example.

Web100 is both the name of a NSF-funded project[6] and of the tool developed in that framework [WEB100]. The project aims to enable automatic and transparent high throughput transport of TCP/IP.

---

[5] The initial release of the script is for Linux kernel given the fact that Linux is the main platform for EDG software development and testing in the DataGRID testbed. Versions for future Linux kernel releases or for other kernel platforms will be released when/if required.

[6] The Web100 team are currently working with LBNL on a project called Net100 that aims to make Web100 more transparent for automatic performance tuning via a 'network enabled OS'. It is based on the NetLogger [NLOG] interface. We plan to test future tool releases to evaluate the adequacy to the Grid.

The current implementation (version 1.2 alpha) involves a kernel patch (currently only for version 2.4.16) to allow the *live* monitoring of TCP variables at kernel level.

Important TCP variables such as the number and type of packets sent and retransmitted, ECN (RFC 3168) and SACK (RFC 2061) availability, timeouts and the retransmit timeouts (RTO), the slow-start threshold (`ssthresh`) and the congestion window size (`cwnd`) are all provided on a time-basis, unlike tools like tcpdump and tcptrace [TTRACE] that can supply information on a packet-basis. Web100 can also provide an indication of the TCP sliding window size like tcpdump/tcptrace, but does not require root access to the system.

We propose the use of the Web100 tool on any EDG node whose service functionality is heavily dependent on TCP performance, for example the Storage Elements. At a high level, Web100 can be used to simply monitor the instantaneous throughput of a TCP stream and debug simple TCP stack metrics such as sender and receiver window sizes. These values can 'cap' transfer rates due to bottlenecks at the sending and receiving hosts rather than any network related bottleneck.

Whilst much of the variable list available through Web100 (known as the TCP Management Information Base – MIB) is read-only, a few variables allow write access to allow manipulation of the TCP stack in situ of a TCP connection. These variables currently include adjustments of the sender buffer sizes such that the TCP connection can tune itself given the appropriate parameters.

A set of API's has been developed along with the Web100 project that allows manipulation of Web100 variables. These variables, located in `/proc/web100`, can be easily viewed and manipulated. Once installed, all TCP connections are 'logged' in this pseudo directory. Performance degradation of TCP and general system performance as a result of installing Web100 has received no negative comments, however, a research is ongoing.

Benefits to the DataGRID project would include an instantaneous logging facility to any TCP stream, providing a way of characterising and diagnosing all TCP traffic with more than just a throughput/goodput value. Advantages include providing a visual representation of, for example, the effect of different QoS parameters at the host level on a TCP stream.

5 THE NETWORK-BASED GRID OPTIMIZATION SERVICE

This Grid abstraction highlights the fundamental role of the networking infrastructure and in particular of its stability, reliability and quality of service, on which the overall system is founded.

Grid optimisation addresses the problem of how the Grid can enhance its performance through the use of information on the status and transmission behaviour of its network links. The *Network-based Optimiser* is a Grid building block that can be queried by any existing Grid service or application and that provides an estimate of the transmission quality between two or more Grid nodes, as specified by the querier.

The Network-based Optimiser relies on the availability of raw information on network performance provided by the Grid monitoring system and produces a high-level view of network performance through a set of internal *cost functions*. The cost function gets in input a set of basic network metrics and produces a compound high-level metric.

The Optimiser relies on the Grid monitoring system, which provides a global view about the status of a range of Grid resources. This information is extremely useful for a number of purposes, for example, for a periodic check by users and Grid site managers of the system status, for correct use and regular maintenance of the Grid itself. In addition to this, also Grid services have the possibility to use monitoring information for dynamic adaptation to the Grid status at any given time.

A large number of metrics can be regularly tracked by the monitoring system so that a comprehensive and up-to-date picture of the Grid status is always available. Network metrics are an example of parameters that can be used for optimisation of a number of Grid operations in many different *application scenarios*.

5.1 MIDDLEWARE REQUIREMENTS

5.1.1 Network optimisation applied to Resource Brokerage

The integration of network information can be extremely useful for the optimisation of the decision taking process of a *Grid Resource Broker* [EDGrb]. One of the tasks of the Resource Broker is to select from a list of candidate worker nodes – often geographically distributed – the *best* computing element for a given job execution. The decision can be taken according to an internal list of what we call *primary selection rules*. A primary selection rule is bound to the stringent requirements of the job, like the software environment available on a given worker node, the amount of free disk space, available CPU etc. Primary selection rules are defined by the job for correct execution and can be used by the scheduler to identify a list of equivalently good execution platforms available in the Grid. Job execution 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. Given the networked nature of the Grid, the list of candidate worker nodes and databases can be selected further on, so that the amount of traffic to be exchanged is minimized and/or the nodes with better network connectivity are given higher priority. This type of information could be provided by the Optimisation Service, which relies on what we call *secondary selection rules*.

5.1.2 Gathering of distributed data in a single Storage Element

Similarly, the Optimisation Service can be used for the improvement of data management among different Storage Elements in a large number of cases: for the selection of the best replica of a given file if multiple copies of it are present in different storage elements, for the identification of the most appropriate Storage Elements when a given amount of data has to be replicated [EDGreplica] in the Grid, for the management of input/output data fragments in a single Storage Element etc.

For example, in the latter case, it may happen that input/output data of a given job is fragmented and distributed among a number of Storage Elements. If the set of fragments needs to be gathered in a single SE, then the most appropriate storage element has to be identified. The selection criteria can be based on a number of principles like the minimization of the amount of data exchanged between SEs, the identification of the SE with better connectivity to the SE group etc.

5.1.3 Adaptive remote file access

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 that application. In this case it becomes important to provide the application itself with a method that allows it to optimise the access to remote files at run time. The optimisation is based on the dynamic adjusting of the Storage Element set that the application is using as the file access pattern changes.

5.2 PROPOSED SERVICES, APPLICABLITY AND BENEFITS

5.2.1 Cost functions

Different cost functions are possible and could coexist, for example in a scenario that allows dynamic selection of the most appropriate cost function. In what follows we just restricted the service model to the code of a single cost function definition, which we name *Closeness* C_{ij}. Closeness of two nodes N_i and N_j provides a transmission quality estimate for the data path between the two nodes.

According to our definition, $C_{i,j}$ varies in the range [0,1], where 0 indicates that a given node is not reachable, while 1 represents the maximum degree in transmission quality, for example when the two nodes are located in the same LAN – if we assume that the LAN is congestion-free and offers high-speed connectivity to its nodes – or when the two nodes are the same. For a detailed definition of Closeness refer to [OPT02].

5.2.2 Service definition

We define *Network-based Grid Optimizer* (NGO) a novel Grid building block in charge of estimating the quality of network connectivity between any set of Grid nodes. The Optimiser offer a service to any other Grid middleware service or application that is interested in introducing network performance as a parameter in its decision taking process.

As shown in Figure 1, the Network-Based Optimiser relies on the availability of the monitoring information that is regularly collected by the Grid Monitoring Service through the *Grid Information Service*, in other words, the Optimiser can be seen as a primary consumer of Grid monitoring data. The network monitoring infrastructure defined in [EDG-D7.2] will be used. For this reason, given the strong dependency on the monitoring and information service, issues like the frequency of monitoring information updates, the granularity of the measurements and the performance of the query process to the Grid Information Service are fundamental to achieve an effective and reliable Grid optimisation service.

The *NGO* relies on the definition of one or more cost functions that numerically quantifies the quality of network connectivity.

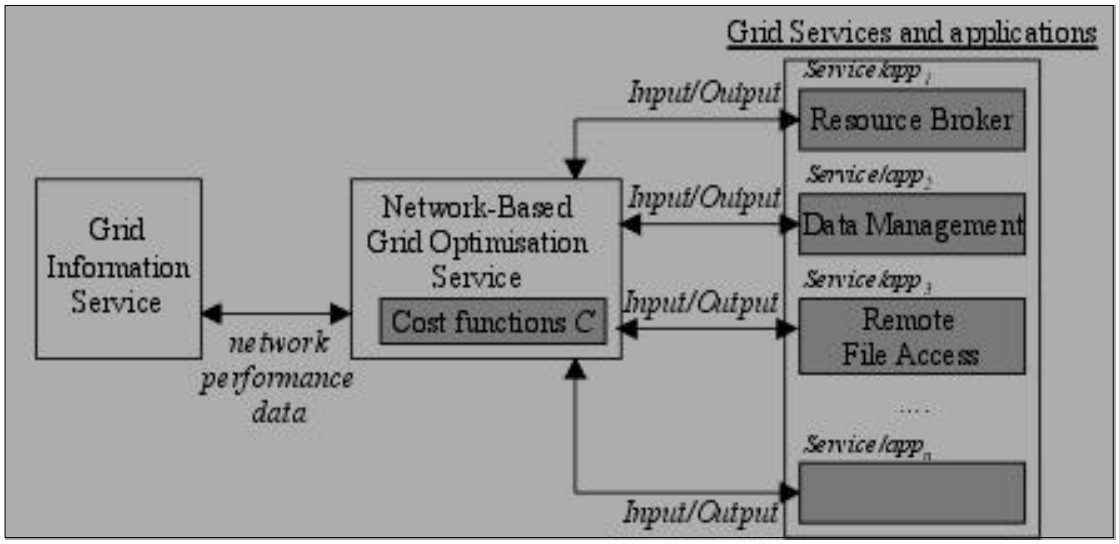


Figure 1: A logical view of the Network-based Grid Optimiser

5.2.3 Application scenarios

Network optimisation applied to resource brokerage

One of the services provided by Grid Resource Brokers is the identification of the most suitable worker node/s that can be selected for job execution and of the corresponding Storage Elements providing the input data requested by the job itself.

In order to complete this task, a number of selection criteria can be adopted. Primary selection rules are needed to identify the worker nodes offering a suitable computing environment, i.e. to produce a list of candidate computing elements like the following:

$$CE_{list} = (CE_1, CE_2, \ldots, CE_m).$$

At this point, the list of input files requested by the job has to be worked out: $f_1, f_2, \ldots, f_n$, so that for each file a list of Storage Elements keeping a copy of the file can be identified thanks to the support of the data management service. In this way, a comprehensive list of SEs can be defined:

$$SE_{list} = (SE_1, SE_2, \ldots, SE_N)$$

where each SEi keeps one copy of at least one input file f_j.

The goal of network optimisation applied to resource brokerage is the definition of secondary selection criteria for the identification of the most appropriate CE and, for each file f_j, and of the corresponding most appropriate SE from which the file can be retrieved. In what follows we focus on selection criteria based on network performance metrics with the assumption that compound rules can be defined if needed, based on a larger variety of resource metrics.

CE_{list} and SE_{list} are the input information provided by the Resource Broker to the Optimiser.

For each input file fj *NGO* needs to identify the most suitable couple (CE, SE). The Closeness function $C_{i,j}$ can be used for this purpose so that node N_i belongs to the CE_{list} set, while node N_j is in SE_{list}. In

particular, given file f_j only the relevant SEs, i.e. only the ones keeping one replica of it, are considered so that only a subset of SE_{list} needs to be analysed.

A given list subset identifies what we call an *optimisation domain*. Each optimisation domain

$$Opt_i(f_j, CE_{list}, SE_{list})$$

is defined by the identity of a given input file and by all the SEs keeping a copy of that given file. It restricts the optimisation problem by considering all the candidate CEs in CE_{list} and only the SEs in SE_{list} relevant to the file under consideration. For each CE_i the ideal SE can be identified. For a detailed description on the solution of the overall optimisation problem refer to [OPT02].

Network optimisation for distributed data management

One of the Resource Broker's tasks is to identify the most suitable computing and Storage Element for job execution. However, input/output data may be partitioned in individual sets among a number of different SEs. If needed, the job may require that data are gathered in a single SE prior to execution. In this case the best SE has to be selected to collect data and one or more data transfer sessions to the selected SE have to be triggered by the Resource Broker.

The candidate SE may be selected according to a number of criteria. Firstly, the selection of the SE keeping the largest information set gives the possibility to minimize the overall amount of data exchanged over the network. At the same time, the candidate SE should be located close to one or more CEs. The SE to be selected should be the one that maximizes the overall *Proximity P* between SEs in order to increase the efficiency of transmission protocols.

While P can be computed in according to a number of different cost functions [OPT02], according to the requirements expressed by the DataGRID WP2, Proximity is determined by estimating the time needed to transfer a file of known size between two different Storage Elements SEid1 and SEid2. Computation of the transfer time is based on the estimated throughput between the two end-points – if a point-to-point measurement is available – or between the monitoring domains the two nodes belong to. If a population of measurement samples is available, then estimation is based on the median of the throughput samples, while the estimation error is expressed as the 68% confidence interval, while if only the most recent sample can be provided by the information system, the error is based on the estimated accuracy of the measurement tool.

The Application Programming Interfaces used for provisioning of the optimisation service to Resource Brokerage and Data management is described in Appendix.

6 EXPERIMENTAL ACTIVITIES

Testing has been carried out both in laboratories and real-life networks. Experiments are needed in order to verify the viability and effectiveness of the services we propose and to investigate an additional number of fields that we think are important for the understanding of the current network problems experienced by Grid middleware and applications and for the analysis of more sophisticated technical approaches that will be the basis of future services.

Experiments are an integral part of the work carried out by WP7 and are important to substantiate the technical solutions proposed in this Deliverable. Results are summarized in Appendix A, B and C for the sake of brevity of the main part of this document.

Tests were carried out in different areas for the evaluation of different approaches to packet forwarding service provisioning, for the analysis of the transport protocol performance experienced on current network infrastructures, and for the evaluation of advanced packet transmission techniques.

The following list provides an overview of the activities documented in detail in Appendix:

- Performance analysis of scheduling algorithms applied to high-speed traffic (Weighted Round Robin and Deficit Round Robin);
- IP Premium applied to medical applications;
- QBSS performance analysis;
- Network interface cards, motherboards and end-to-end performance;
- UDP and TCP Performance over the European WAN;
- Characterization of TCP packet-loss across over medium and long-distance connections;
- Proactive TCP congestion avoidance: ECN;
- Reliable multicast.

We refer the interested reader to the Appendix for a more complete view of the technical addressed in this Deliverable.

7 CONCLUSIONS AND FUTURE WORK

We propose the introduction of advanced network services for the DataGRID testbed to address middleware and application requirements in three different areas: *Differentiated packet forwarding*, *Transport protocol monitoring and configuration* and *Network-based grid optimisation*. The above-mentioned services have been defined according to the requirements specified in collaboration with WP1, WP2, WP3, WP8, WP9 and WP10 of the DataGRID project, and on the basis of the application needs specified in [EDG-D7.1].

We think that the proposed services are a concrete step forward towards the integration of networking into distributed computed platforms, which will improve performance of the DataGRID testbed as well as provide useful feedback to research network providers about new requirements, applicability and performance of the experimental services currently supported. The synergy between the two worlds will be of mutual benefit.

The viability of the proposed service is supported by the preliminary encouraging experimental results documented in Appendix. However, the validity of the services will be addressed and proven by our future test plan, which aims at integrating the proposed services into the DataGRID middleware and application framework.

We plan to carry out an extensive test programme for each proposed service. Differentiated packet forwarding tests will rely on the network production infrastructures (local DataGRID testbed site networks, national research networks and GÉANT) and on the services supported by the research network providers. DataGRID application traffic will be used. In particular we will start with biology applications and in case of good results will extend the test programme to WP8 and WP9 applications. The Assured service will be evaluated in detail, both in terms of definition and implementation, with the assistance of TCP test results carried out on the GÉANT backbone.

The definition of an API for service requirement specification – used for network advance reservation, admission control etc. – will be addressed.

In the high-performance transport area, TCP performance on very high-speed connections will be tested on the GÉANT infrastructure according to what defined in the GÉANT collaboration proposal [GCOL] and in coordination with the DataTAG project. Research will be carried out to evaluate the effectiveness of protocol extensions and more advanced services for automatic TCP tuning will be analysed.

Finally, we will evaluate the performance of cost functions in terms of computation and query overhead. Testing of cost functions will assist the Monitoring Services Work Package (WP3) in the evaluation of the Information Service performance.

The effectiveness of the current cost functions will be evaluated, especially in terms of accuracy, and new ones will be possibly defined to address new specific middleware and application requirements. Mechanisms for dynamic configuration of network cost functions will be considered, while analysis of accuracy will be based on results of the measurement forecasting activity of WP7 [EDG-D7.2].

New APIs will be defined if required and the applicability to new scenarios will be evaluated.

8 ANNEX A: PACKET-FORWARDING SERVICES FOR NETWORK QUALITY OF SERVICE

8.1 EXPERIMENTAL ACTIVITIES

The experimental activities herein documented focus on the following QoS functional blocks: metering, policing, classification and scheduling, when supported on production platforms for high-speed networking. The emphasis on high-speed is motivated by the need to gain knowledge about the feasibility of QoS on the typical network platforms that will be used in future Grid testbeds both in the local and in the wide area.

Our interest is focussed on two services that are particularly interesting for the Grid community: IP Premium (GÉANT backbone and few European National Research Networks) and QBSS (Abilene network), which address requirements of applications in Category 1 and 4 respectively.

8.1.1 Performance analysis of scheduling algorithms applied to high-speed traffic

The following two sections illustrate the performance and the issues of two popular scheduling algorithms, namely Weighted Round Robin (WRR) [PAR93] and Deficit Round Robin (DRR) [DRR] when tested in a LAN and WAN testbed respectively. WRR testing in the LAN was performed on the Catalyst 6500, a common last-generation GigaEthernet switch platform by Cisco, while the Gigabit Switch Router (GSR) by CISCO was chosen for DRR testing. The GSR is a high-end router for high-speed connectivity in provider backbones.

A detailed description of testbed configurations and test results including metering, classification, marking and policing is available in EDG reports on the Web [EDGwrr, EDGdrr]. The following paragraphs only provide a summary of the main achievements.

Quality of service in the LAN: Weighted Round Robin

Policing

Policing on the Catalyst 6500 is based on a dual token bucket and its profile is defined by a set of key parameters:

- *Committed* (CIR) and *Peak information rate* (PIR): they respectively represent the minimum sustainable rate and the maximum rate at burst time accepted in input;
- *Normal burst size*: size of the token bucket serviced at CIR rate;
- *Maximum burst size*: size of the token bucket serviced at PIR rate;
- *Conform, exceed and violate action*: the policing action to be applied when traffic rate is less than the CIR, between CIR and PIR, or larger than PIR respectively. Examples of actions are: transmit, set-dscp-transmit (to mark traffic with a given DSCP code), drop etc.

Policing results related to traffic sourced by GigaEthernet interfaces proved to be substantially different from what seen at lower speed. To start with, the transport protocol in use (e.g. TCP or UDP) is fundamental for proper tuning of the policing configuration. In particular, in case of UDP constant bit rate (CBR) traffic, token bucket size has a negligible impact on performance. In fact, in this case inter-packet gaps are approximately equal and policers run out of tokens only if CIR is underestimated, so that no packet loss is visible when the input traffic rate complies with the policer.

Results are the opposite with TCP traffic: in this case both the normal and maximum burst size have to be sufficiently large in order to absorb the very high burstiness produced by the TCP source when connected at gigabit speed. Unexpectedly, instantaneous TCP burstiness can be extremely high even when the average TCP throughput is of the order of tens of megabits per second. This is due to the very small inter-packet gap that characterizes data units belonging to the same TCP window. In fact,

within a given window, packets can be transmitted at wire rate and in local area networks. If RTT is low, a sequence of end-to-end packet windows can be transmitted without interruption thanks to the timely reception of TCP acknowledgements. This implies that even in presence of small average congestion windows, token buckets have to be sufficiently large to accommodate input bursts. On the other hand, if burst tolerance is too limited, the resulting TCP throughput can be very small or totally negligible, with the result that applications cannot make use of the target minimum rate guaranteed.

This is shown in Figure 2, which shows the dependency of the TCP achieved rate on the normal burst size configured. For a constant CIR rate of 50 Mbps the minimum burst size that has to be configured in order to allow a single TCP flow to achieve the specified target rate is of 2.50 Mby. The optimum burst size to be configured on the switch depends on both the number of TCP flows in the policed class and on the CIR and PIR parameter.

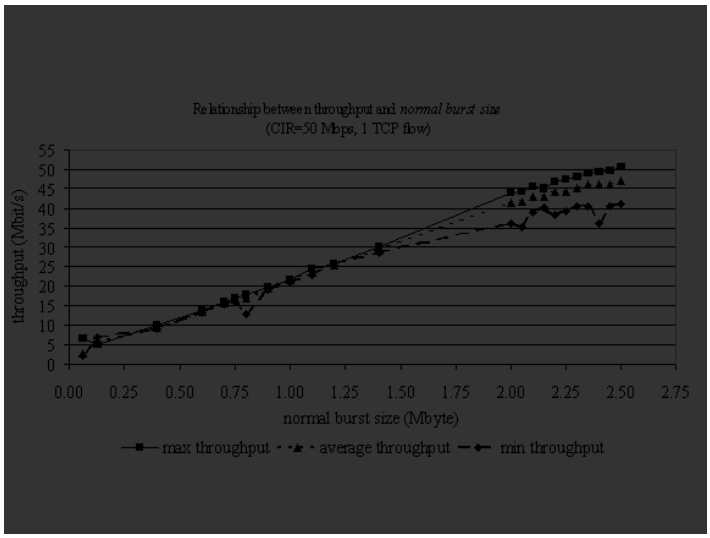


Figure 2: TCP throughput as a function of the normal burst size parameter

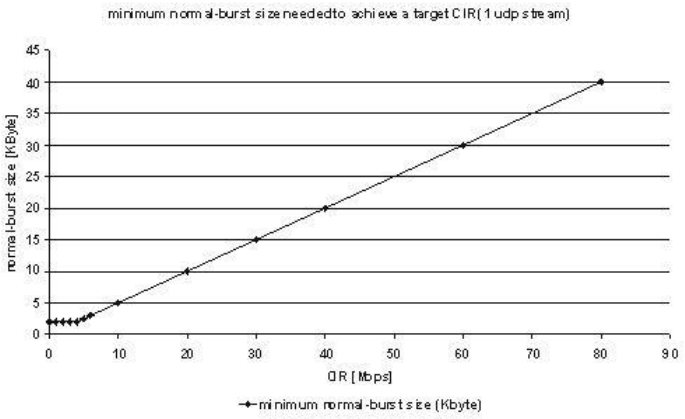


Figure 3: normal burst size needed to achieve a target rate equal to CIR in case of UDP traffic

On the contrary, with UDP traffic, given the constant bit rate nature of traffic used in our tests, the normal burst size impact on goodput is almost negligible. For example, Figure 3 shows that in order to get a goodput at least equal to the configured CIR of 80 Mbps the normal burst size only needs to be equal to 40 Kby.

Queuing

Two types of queuing are supported in the switch under test: *input* and *output*. They are applied to traffic in different stages: the former at ingress in the switch, the latter after crossing of the internal switch fabric. Both input and output queuing on the Catalyst are implemented differently depending on the type of interface in use.

Normally for a given hardware queue several *sub-queues* can be configured so that multiple traffic classes can be associated to the same *main* queue. In particular, sub-queues are defined by means of *thresholds*. Two different types of threshold are supported by GigaEthernet interfaces on the Catalyst 6500: *tail-drop* and *random-detect*. The tail-drop threshold for a given sub-queue defines the percentage of the main queue length at which tail dropping is applied to traffic belonging to that sub-queue, while random-detect thresholds are of two types: *random-detect min threshold* and *random-detect max threshold*. Min threshold identifies the queue size percentage at which the WRED dropping starts to be applied to a sub-queue with a given (non-configurable) probability, while max threshold identifies the queue size level at which tail drops occurs.

According to our tests, while the dependency on min random-drop thresholds is negligible, in case of tail-drop thresholds, queue parameters have to be carefully configured: too small values should be avoided to make the scheduling algorithm more tolerant to TCP burstiness, since larger values give the possibility to make a more efficient use of spare capacity if some classes are not using the amount of bandwidth allocated to them. A general rule for queue size tuning is not available, since queue sizes should depend on the TCP aggregate profile is injected. The TCP and UDP throughput dependency on tail-drop thresholds is illustrated in Figure 4.

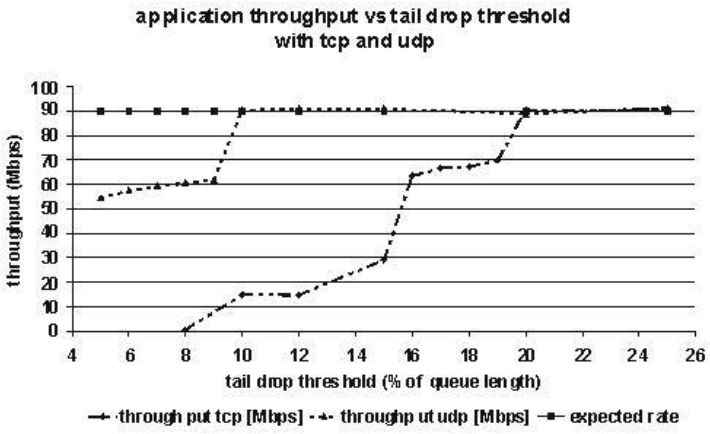


Figure 4: application throughput of a UDP and TCP stream as a function of the tail-drop threshold

WRR queue *weight* tuning was also performed in order to determine the most appropriate weight as a function of the *target rate*, i.e. the amount of (UDP) traffic injected. When configuring a weight, we considered that overhead traffic produced by IP and UDP overhead has to be taken into account – the weight has to be larger than the amount of application data produced by the UDP source.

In WRR weights are assigned to define relative bandwidths of the 2 main queues according to the following formula:

$$bw\_share(queue\_i) = W(queue\_i) / (W(queue\_1)+...+W(queue\_n))$$

We define *theoretical weight* the amount of bandwidth to be assigned to a given queue so that the queue service bit rate equals the amount of offered load for that given main queue.

Test results indicate that the theoretical weight allows the destination to get an amount of traffic rate almost equal to what configured at the source – just a few packets are dropped by the scheduler as indicated by the UDP receiver. In order to avoid packet loss totally, i.e. in order to achieve total isolation between WRR main queues, the weight has to be over-estimated by a factor that is inversely proportional to the weight (for example it is 8.5 for a weight corresponding to 20 Mbps, while it is 3 for 80 Mbps).

Perfect traffic isolation between sub-queues is difficult as bandwidth guarantees are only configured per main queue. This implies that traffic isolation in sub-queues only relies on the threshold parameters adopted. In addition, since packets share the same queuing facility, we suspect that cross-interference of delay and jitter profiles occurs due to the mixing of packets in the same queue. This is subject of future research.

Quality of service in the WAN: Deficit Round Robin

The following aspects of DRR scheduling on the Cisco GSR have been analysed[7]:

- delay and IPDV performance of DRR with variable queue bandwidth over-estimation factors;
- throughput and delay/jitter comparison between DRR and strict priority queuing;
- effect of buffer allocation in buffer memory pools on bandwidth guarantees and traffic class isolation.

Test results show that DRR is an effective mechanism for the provisioning of bandwidth guarantees to queues even in presence of high traffic loads (the maximum traffic load tested was 2.5 Gbps). However, traffic protection can be compromised by the presence of memory buffer pools both at the ingress and at the egress of the router crossbar that are indistinctly shared by different traffic classes. The assignment of a packet to a pool simply depends on the packet size. This implies that in case of congestion at an egress port, packets can be dropped due to the lack of free memory both at input and output shared buffers. Consequently, buffer pools represent contention points for packets of different classes but with size in the same range.

In order to avoid such contention points, active queue management has to be configured on each DRR queue in order to selectively drop packets from the shared buffers in case of incipient congestion. Different drop profiles can be adopted for each class. However, the drop profile (defined by the minimum/maximum threshold and by the packet drop probability) has to be accurately tuned according to the instantaneous and average size of that given queue. A too aggressive packet loss probability or too low thresholds can introduce excessive packet loss that prevents the traffic class from successfully using the total amount of bandwidth guaranteed to it.

8.1.2 IP Premium applied to medical applications

Different test scenarios have been defined for *Content-based query of medical image databases* testing with support of Quality of Service. Each scenario is characterized by a different database size:

- small database made of 124 2D images, 140 kb each;
- medium database made of 238 3D images, from 7 Mb to 15 Mb each;
- large database made of 456 3D images, about 14 Mb each.

[7] DRR configuration syntax and examples are provided in Appendix.

The network will be stressed with transport of data from a few Mby (comparison of few 2D images) to 6.3 Gby (complete largest database). The global computing time achieved on N Grid nodes when using different QoS services will be compared to the equivalent performance seen when the plain best-effort service is in use. Testing will initially rely on the IP Premium service, which will be experimented in collaboration with GÉANT.

In the second application scenario (*Remote interactive segmentation*) qualitative testing will be performed. In particular, the perceived quality when using different QoS services and a range of different bandwidth values will be compared to what experienced in case of best-effort service.

The testing of the Differentiated Services in the context of Grid computing is currently underway. We believe that results will help the network community to better understand the requirements of distributed computing in terms of QoS [8].

8.1.3 QBSS performance analysis

The QBone Scavenger Service (QBSS) is an example of *less-than-best-effort* (LBE) service adopted by the Internet2 [QBSS]. The purpose of the service is rather simple: to make the fraction of link capacity not used by the standard best-effort traffic available to the *less-than-best-effort traffic* class and to protect best-effort traffic from the potential congestion produced by large bulks of *less-than-best-effort* traffic. The QBSS service is well described by the list of requirements that a router supporting the LBE service:

1. packet loss probability is higher for QBSS packets than best-effort packets;
2. a minimum amount of bandwidth for QBSS traffic has to be reserved to QBSS traffic to avoid starvation;
3. unused bandwidth should be available to QBSS traffic.

The QBSS service is suitable in a number of application cases, for example for protection of best-effort traffic from applications exchanging large bulks of traffic. The purpose is to let any unused resource to be allocated to QBSS packets, without preventing best-effort traffic from using any amount of resource that can be offered by the network infrastructure.

Packets are marked with a special DSCP value (001000). Some networks may ignore the packet code-point and treat this traffic just like the default best-effort class.

QBSS testing in local area networks

Service implementation and experimental set-up for QBSS testing in the LAN

While the QBSS service can be supported by production routers adopting suitable scheduling algorithms, the following experimentation is based on Linux platforms. The *iproute2* package, supported in Linux 2.2.x kernel versions and later ones. Iproute2 is an interface for configuration of QoS support in Linux based on the *tc* command, giving the possibility to enable a variety of scheduling algorithms, queuing disciplines and traffic filters.

As indicated in Figure 5, two service classes were enabled: best-effort and QBSS, the former having the possibility to use a minimum of 99% of the link capacity available at a given output interface – no additional resources can be borrowed from other classes in case of congestion – and the latter using a minimum bandwidth guarantee 1% with the possibility to use more in case of resource availability.

[8] Results and conclusions will be part of deliverable D7.4.

Weighted Round Robin and Class Based Queuing are the scheduling and queuing disciplines adopted.

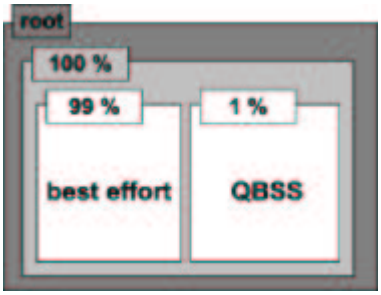


Figure 5: Definition of the two traffic classes: best-effort and QBONE

Experimental results

A cluster of four Athlon machines (850~MHz, 128~Mo RAM) running Debian GNU/Linux 3.0 – 2.4.18 kernel – with 100baseTX Full Duplex connections was used. Two machines run as traffic sources, one host as destination, while the fourth was configured as router. The iperf [iperf]and mgen [mgen] traffic generators together with the tcptrace [tcptrace] tool were used.

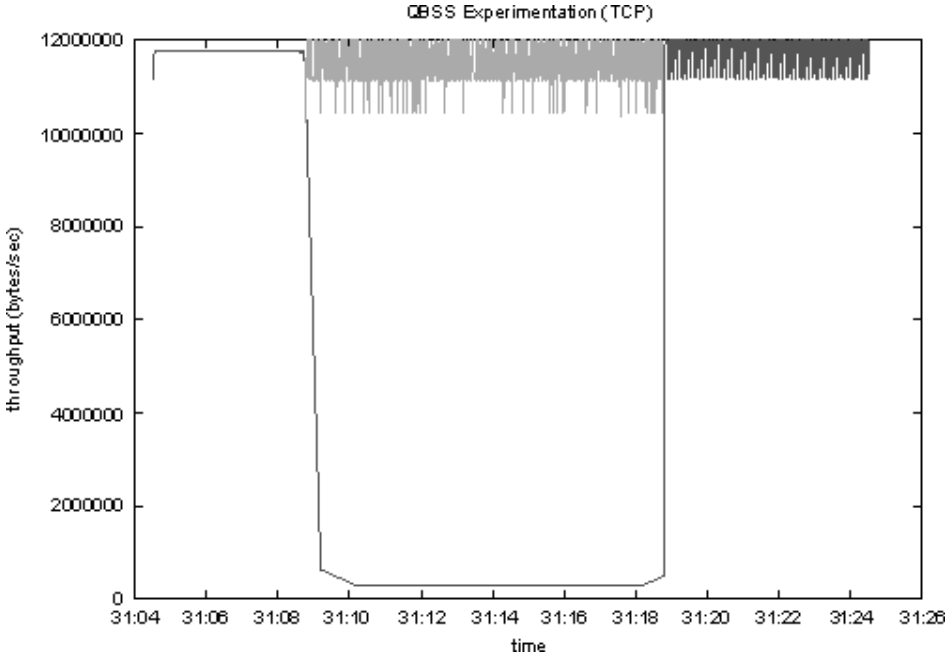


Figure 6: TCP Experimentation with a single best-effort stream

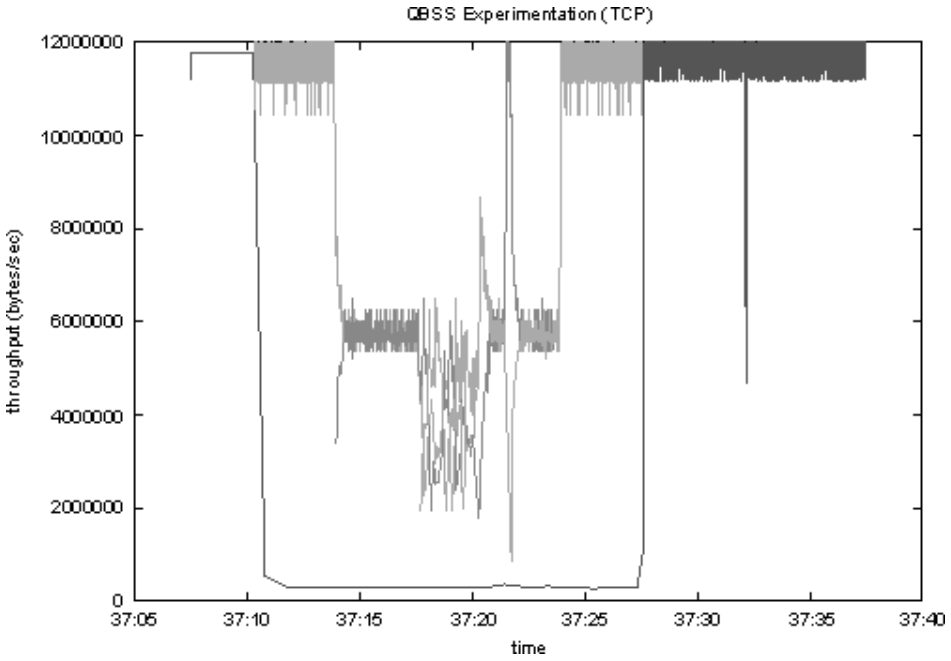


Figure 7: QBSS Experimentation with three best-effort streams

The following test methodology was followed for experiments based on the TCP protocol. A single unicast QBSS stream is run in background for about 30 sec. Then, an additional set of best-effort streams is started from a new source to the same destination used for QBSS traffic. The data rate was monitored during the whole test; the resulting throughput for both classes in shown in Figure 6.

The graphs show that the QBSS stream successfully gets the whole available bandwidth in case of no traffic contention, however, as soon as best-effort packets appear, the overall QBSS rate dynamically shrinks to 1% of the link capacity (the remaining 93 Mbps are allocated to best-effort traffic as specified by the QBSS definition).

The same behaviour is observed in case of more best-effort streams not synchronized in time, with the only difference that in this case an individual best-effort stream has to share the overall best-effort capacity with the other streams of the same class. This is illustrated in Figure 7.

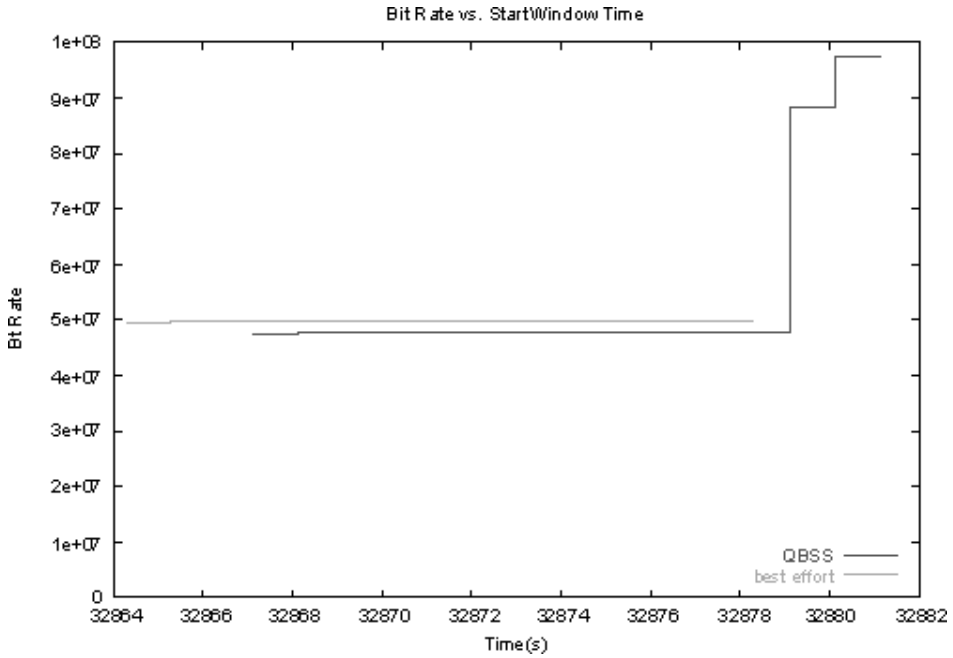


Figure 8: UDP Experimentation with one best effort stream at 50 Mbps

For QBSS testing based on UDP a constant bit rate best-effort stream of 50 Mbps is started in background, while later on a new TCP QBSS source is started. Figure 8 shows that thanks to the TCP congestion avoidance algorithm, the QBSS service class is able to extend its transmission rate to 100% of the link rate as soon as best-effort traffic disappears. While QBSS traffic is highly affected by the presence of best-effort traffic especially in terms of packet loss, best-effort traffic is not influenced by concurrent QBSS traffic.

QBSS testing in wide area networks

Service implementation and experimental set-up

A different test layout has been devised in order to test the same service in a production infrastructure characterized by long propagation delays, unlike the previous scenario. The production CERN/DoE transatlantic link was used for this purpose; or the first time the service has been tested across the Atlantic Ocean. QBSS was analysed in terms of behaviour and impact on other traffic classes.

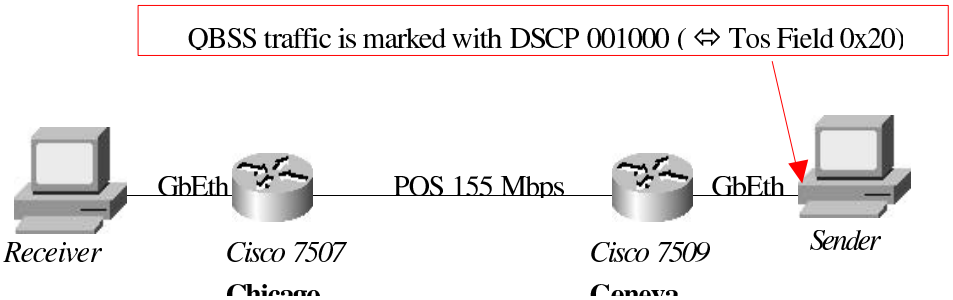


Figure 9: testbed layout of QBSS testing in the wide area

The iperf tool was used to mark packets at the source with TOS code-point 0x20.

Two concurrent streams were produced so that packets belonging to connection 1 were not marked (best-effort traffic) while packets of connection 2 were marked with QBSS code-point (the same test suite was repeated six times to analyse bandwidth sharing over time.

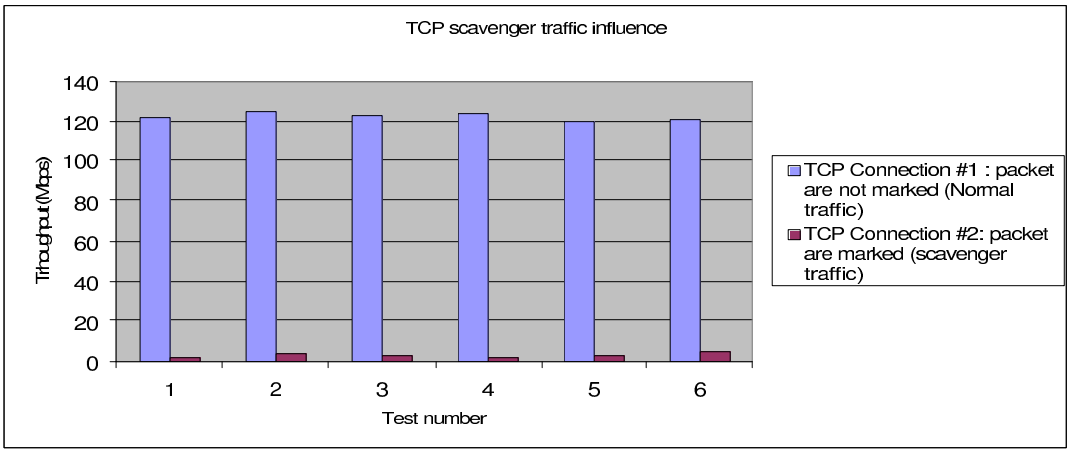


Figure 10: influence of TCP QBSS traffic on the production best-effort class

As indicated in Figure 10, WAN tests confirm the results collected in the previous scenario and prove the capability of dropping QBSS packets before affecting best-effort traffic (Connection 1).

However, for high bandwidth/delay network connections, we noticed that the QBSS traffic was not able to use the whole available bandwidth even in case of no congestion on the link. This may be due to the QBSS scheduling default configuration, which assigns only a small amount of buffer space that may not suffice, depending on the burstiness of TCP sources. Proper buffer tuning is a subject of future research in the framework of the DataTAG project.

Concluding remarks

QBSS proved its feasibility and its suitability to TCP-based non real-time applications injecting large data bulks, which can be effectively differentiated from best-effort traffic. QBSS can be used by UDP-based applications as well, however, adaptive algorithms are recommended in this case in order to tune the output rate and the information-coding scheme. Adaptation can be important to face with both short and long-term congestion: in absence of rate adaptation a considerable amount of resource can be wasted in the data path upstream a given congestion point.

8.2 POLICING AND WRR: CONFIGURATION SYNTAX AND EXAMPLES (CATALYST 6500)

The following configuration samples refer to IOS version 12.1(8b)EX2.

Policing syntax:

```
police <CIR> <normal-burst-size> <PIR> <maximum-burst-size>
       <conform-action> <exceed-action> <violate-action>
```

Example of policing configuration:

```
class-map match-all B1high
       match access-group 151
class-map match-all B2high
       match access-group 150
policy-map Bbandwidth
   class B2high
       police 300000000 2500000 2500000 conform-action set-prec-transmit  2
              exceed-action drop
       class B1high
          police 50000000 2500000 2500000 conform-action set-prec-transmit 1
              exceed-action drop

access-list 150 permit tcp host 131.154.100.3 host 131.154.100.2 eq 50150
access-list 151 permit tcp host 131.154.100.3 host 131.154.100.2 eq 50151
```

WRR configuration with tail-drop thresholds

```
interface GigabitEthernet3/16
 description to cisco 12000 test wrr
 no ip address
 load-interval 180
 speed nonegotiate
 wrr-queue bandwidth 91 9              // prec 1 bandwidth: 90 Mbps
 wrr-queue queue-limit 80 10
 wrr-queue random-detect min-threshold 2 min_th 100     // min_th: of queue 2,1
 wrr-queue random-detect max-threshold 2 100 100
 wrr-queue cos-map 1 1 0 4 5 6 7       // prec 0 in queue 1,1
 wrr-queue cos-map 1 2 3
 wrr-queue cos-map 2 1 1               // prec 1 in queue 2,1
 wrr-queue cos-map 2 2 2
 switchport
 switchport access vlan 3
 switchport mode access
```

WRR configuration with random-drop thresholds

```
interface GigabitEthernet4/6
```

```
description to cisco 12000 test wrr
no ip address
load-interval 180
speed nonegotiate
wrr-queue bandwidth 91 9              // prec 1 bandwidth: 90 Mbps
wrr-queue queue-limit 90 10
wrr-queue threshold 1 100 100         // thresholds of queue 1,1 and 1,2
wrr-queue threshold 2 threshold 100   // threshold: of queue 2,1
wrr-queue cos-map 1 1 0 2 3 4 5 6 7   // prec 0 in queue 1,1
wrr-queue cos-map 2 1 1               // prec 1 in queue 2,1
switchport
switchport access vlan 3
switchport mode access
```

8.3 DRR: CONFIGURATION SYNTAX AND EXAMPLES (CISCO GSR)

The following configuration examples apply to Cisco IOS 12.0(20011210:200219) (QoS modular Command Line Interface is available):

```
class-map match-all prec4
  match ip precedence 4
class-map match-all prec5
  match ip precedence 5
class-map match-all prec6
  match ip precedence 6
class-map match-all prec7
  match ip precedence 7
class-map match-all dscp7
  match ip dscp 7
class-map match-all prec0
  match ip precedence 0
class-map match-all prec1
  match ip precedence 1
class-map match-all prec2
  match ip precedence 2
class-map match-all prec3
  match ip precedence 3

policy-map 5classes
  class prec1
       bandwidth percent 4
       random-detect                                  // WRR is enabled
       random-detect precedence 1 2000 3000 1         // WRR configuration
  class prec2
       bandwidth percent 4
       random-detect
       random-detect precedence 2 2000 3000 1
  class prec3
```

```
     bandwidth percent 4
     random-detect
     random-detect precedence 3 2000 3000 1
class prec4
     bandwidth percent 16
     random-detect
     random-detect precedence 4 2000 3000 1
class class-default
     bandwidth percent 75
     random-detect
     random-detect precedence 0 500 1000 1
```

9 ANNEX B: HIGH-PERFORMANCE AND NOVEL TRANSPORT PROTOCOL SERVICES

9.1 EXPERIMENTAL ACTIVITIES

9.1.1 Network interface cards, motherboards and end-to-end performance

In order to be able to reliably perform sustained data transfers over the network at Gigabit speeds, it is essential to study the behaviour and performance of the end-system compute platform and the network interface cards (NIC). In general, data must be transferred between *system memory* and the *interface* and then placed on the network. For this reason, the operation and interactions of the *memory*, *CPU*, *memory-bus*, the *bridge to the input-output bus* (often referred to as the "chipset"), the *network interface card* and the *input-output bus* itself (in this case PCI) are of great importance. The design and implementation of the software drivers, protocol stack, and operating system are also vital to good performance.

As most of the systems used in DataGRID are currently based on Linux PCs, this platform was selected for making the evaluations. RedHat Linux v 7.2 was used with the 2.4.14, 2.4.16, and 2.4.18 kernels. However, the results are also applicable to other platforms. Two PCs were used with the NICs directly connected together with suitable crossover cables. A variety of high performance or server type motherboards with the PCI or PCI-X I/O bus were used and NICs from different manufacturers were tested in these motherboards. Care was taken to obtain the latest drivers for the Linux kernel in use. UDP/IP frames were chosen for the tests as they are processed in a similar manner to TCP/IP frames, but are not subject to the flow control and congestion avoidance algorithms typical of TCP.

The following measurements were made:

- *Round trip latency as a function of Ethernet frame size using request-response messages*. This provides information about the sum of the data transfer rates for each part of the host-to-host path. It also indicates interrupt coalescence in the NICs and provides information on protocol stack performance and buffer management.
- *UDP throughput between the hosts as a function of the frame size and the transmit spacing between the frames*. The UDPmon program [UDPmon] was used to transmit a stream of suitably spaced UDP packets and the amount of data received and the time taken to receive that data was measured at the destination. The number and distribution of lost frames was measured as well as the throughput.

Activity on the PCI bus of sender/receiver nodes and passage of the Gigabit Ethernet frames on fiber. These measurements were achieved using a Logic analyzer and a specialized Gigabit Ethernet Probe [GEProbe] as shown in Figure 11.

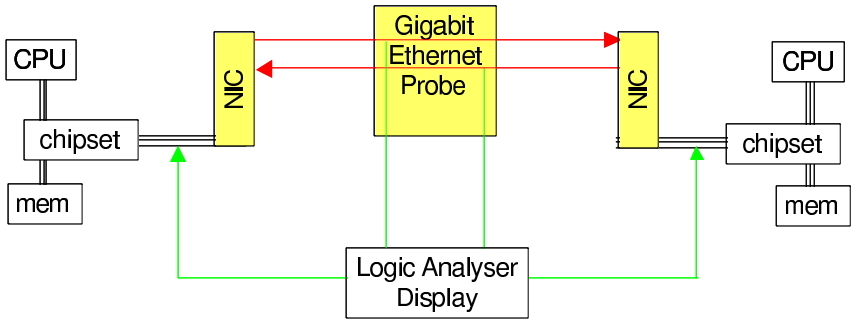


Figure 11: Test arrangement for examining the PCI buses and Gigabit Ethernet link.

Table 1 shows the tests made on the various motherboards, PCI bus configurations and NICs to date[9].

NIC / **Motherboard**	Alteon AceNIC	SysKonnect SK-9843	Broadcom	IntelPro1000
SuperMicro 370DLE; Chipset: ServerWorks III LE PCI 64bit 32 MHz		670 Mbit/s 17 µs		
SuperMicro 370DLE Chipset: ServerWorks III LE PCI 64bit 64 MHz	925 Mbit/s 12 µs	730 Mbit/s 16 µs	In progress	920 Mbit/s 12 µs
SuperMicro P4CD6+ Chipset: Intel i860; PCI 64bit 64 MHz				950 Mbit/s 12 µs
IBM das Chipset: ServerWorks CNB20LE; PCI 64bit 32 MHz		790 Mbit/s 15 µs		930 Mbit/s 11 µs
SuperMicro P4DP6 Chipset: Intel E7500; PCI 64bit 64 MHz		In progress	In progress	950 Mbit/s 12 µ
SuperMicro P4DP6; Chipset: Intel E7500; PCI 64bit 64 MHz				In progress

Table 1: Motherboards and NICs examined. The numbers shown for each condition are the maximum received wire throughput in Mbit/s and the corresponding transmit frame spacing.

The upper graph in Figure 12 shows the UDP throughput as a function of the spacing between the transmitted frames for the Intel Pro 1000 NIC and the Supermicro P4DP6 motherboard with dual Xeon Prestonia 2.2 GHz CPUs. Various frame sizes are given with the length of the user data ranging from 50 bytes to 1472 bytes, the maximum for one frame. As expected, larger frame sizes are more efficient, with a throughput of 950 Mbit/s for user data of 1472 bytes. The lower plot shows the percentage frame loss that occurred when making the throughput measurement. The exact reason why there is frame loss is unclear at the moment, but it has been determined that all the frames generated by the sending software are transmitted and received by the low level drivers. In fact it is the IP protocol layer that claims to discard the frames. Further work is in progress to understand this.

[9] This work is still ongoing, and a detailed report is in preparation. For more information see also [MotherNIC].

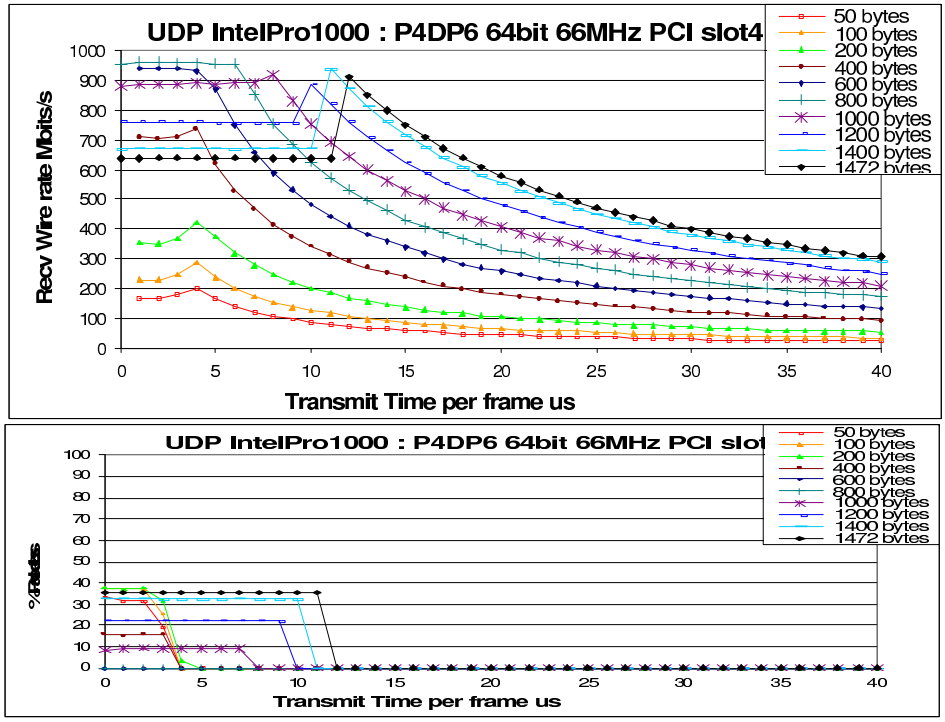


Figure 12: *upper graph*: UDP throughput as a function of the transmit frame spacing for the Intel Pro 1000 NIC and the Supermicro P4DP6 motherboard. *Lower plot*: corresponding frame loss.

Figure 13 shows the signals on the sending and receiving PCI buses when frames with1400 bytes of user data were sent with a spacing of 16 µs. At time *t* the dark portions on the send PCI shows the setup of the NIC card, which then moves the data over the PCI bus. The Ethernet frame is transferred over the receive PCI bus at time *O*, some 17 µs later. The sending bus is in use for ~68% of the time.

Observation of the signals on the Gigabit Ethernet have shown that the PCs can generate back-to-back Ethernet frames when the software spaces the generation of the frames at the time required to transmit them on the Ethernet i.e. for 1472 bytes of user data this separation time, i.e. the time needed to transfer the user data part and the overhead bytes,[10] would be approximately 12.26 µs.

[10] The overall number of overhead bytes is 60: 8 bytes for inter-packet gap, 6 bytes for the preamble, 18 bytes for Ethernet frame overhead and 28 bytes of IP and UDP overhead.

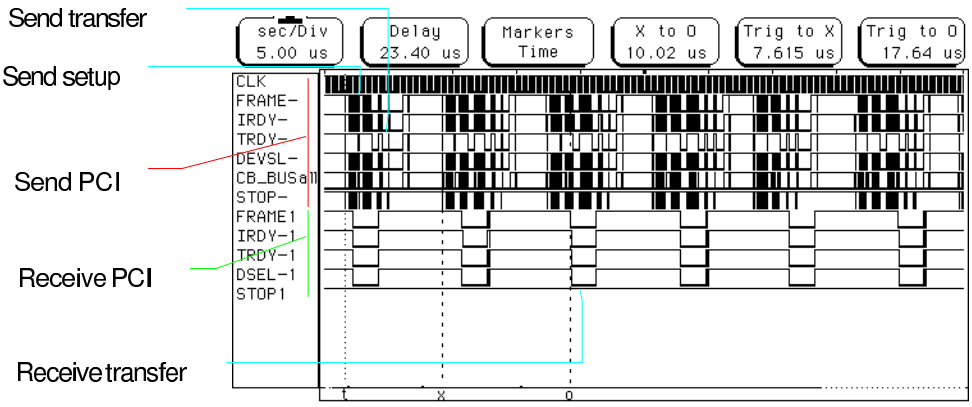


Figure 13: The traces of the signals on the send and receive PCI buses for the Intel Pro 1000 NIC and the Supermicro P4DP6 motherboard.

Concluding remarks

Experiments show that with a 64 bit 66 MHz PCI bus and a modern NIC, these PC platforms are capable of generating large frames (1000 to 1472 bytes of user data) at Gigabit Ethernet line speed and achieving UDP/IP transfer rates of 950Mbit/s. No difference in behaviour or performance has been observed when using Copper or Fibre Gigabit Ethernet links.

To enable Gigabit transfers to high performance disks or RAID sub-systems, the design of the motherboard should allow storage and network devices to be on separate PCI buses. For example, the SuperMicro P4DP6 Motherboard has 4 independent 64bit 66 MHz PCI-x buses, allowing suitable separation of the bus traffic.

The operating system should be configured with sufficient buffer space to allow a continuous flow of data at Gigabit rates.

User applications and Grid Middleware should be designed to minimise data copy and movement to or from its final place of processing; suitable buffer space and time conscious algorithms should be provided.

9.1.2 UDP and TCP Performance over the European WAN

Some initial tests have been made to measure the performance of connections over the current production network. Such measurements are very important to estimate the baseline performance of the infrastructure used by the DataGRID testbed. Tests have been made both within a single National Research Network (NRN) and between NRNs using the GÉANT backbone. The tests required lightly loaded high performance end systems with Gigabit Ethernet NICs as shown by experiments on NIC and motherboards. These nodes have Gigabit connectivity via the local Campus LAN to the MAN and hence to the NRN. Suitable systems were made available as follows:

Location	Access Link Speed	Interface Card
The University of Amsterdam	1Gbit	Alteon / Netgear 620
CERN	1Gbit	
The University of Manchester	1Gbit	Alteon / Netgear 620
Rutherford Laboratory	622 Mbit	

UDP Performance on the WAN

UDPmon [UDPmon] was used to send controlled streams of UDP packets between Manchester and RAL, Manchester and Amsterdam and Manchester and CERN; the number of packets received and the time taken to receive the frames was measured at the receiver. On an unloaded network this methodology would measure the Capacity Bandwidth (BW) [IPPMf, IPPMb] of the link with the minimum throughput, however, on a loaded network the test UDP frames will queue along with other traffic, thus extending the time to receive the data. This gives an estimate of the Available Bandwidth (BW).

Figure 14 shows the measured throughput at the receiver as a function of the time between sending the packets for a series of packet sizes. The maximum rate between Manchester and RAL was 570 Mbps, which was 91% of the 622 Mbps access link between SuperJANET4 and RAL. An Ethernet frame with 1472 bytes of user data would take approximately 21µs at this transfer rate, and for packet transmission spacing of this and greater, little packet loss was observed.

Very good performance was also achieved by connections across Europe. 750 Mbps was measured to Amsterdam using the GÉANT backbone to connect SuperJANET and SURFnet. A smaller rate of 460 Mbps was achieved on the Manchester to CERN tests, but this was due to the fact that the PC at CERN only had a 32 bit PCI bus. A fall in throughput for small times between transmitting the packets was observed: this phenomenon is interface dependent and is not a WAN effect, in fact it can be reproduced in back-to-back laboratory tests [MotherNIC]. This problem is being investigated further.

TCP Performance on the WAN

High throughput measurements were also made using TCP/IP between Manchester and Amsterdam, and Manchester and CERN.

The throughput of the user data, sometimes called *goodput*, was measured at the receiving end of the link using socket programs developed in the framework of the DataGrid Project [UDPmon][11]. Figure 15 shows the throughput obtained as the send and receive socket buffer sizes were increased. To make the tests, the TCP kernel parameters were set to allow use of large socket buffer sixes. As expected and confirmed by experimental results collected in many different test scenarios, throughput increases with the buffer size. Optimum throughput on a TCP connection would be achieved when the TCP window is large enough that data may be continuously sent without having to wait for the acknowledgements. The table below shows the TCP window sizes in Mbytes calculated from the RTT * bandwidth product.

Link	RTT (msec)	Bandwidth measured by UDP (Mbps)	Expected TCP window size considering 1 Gbps of local connectivity of the host (Mby)	TCP window size (Mby)
Man – Ams	14.5	750	1.81	1.36
Man – CERN	21.4	460	1.23	1.23

Table 2. TCP window sizes in Mbytes calculated from the RTT*bandwidth product.

[11] It is planned to extend these programs to use the TCP information provided by the Web100 kernel extensions.

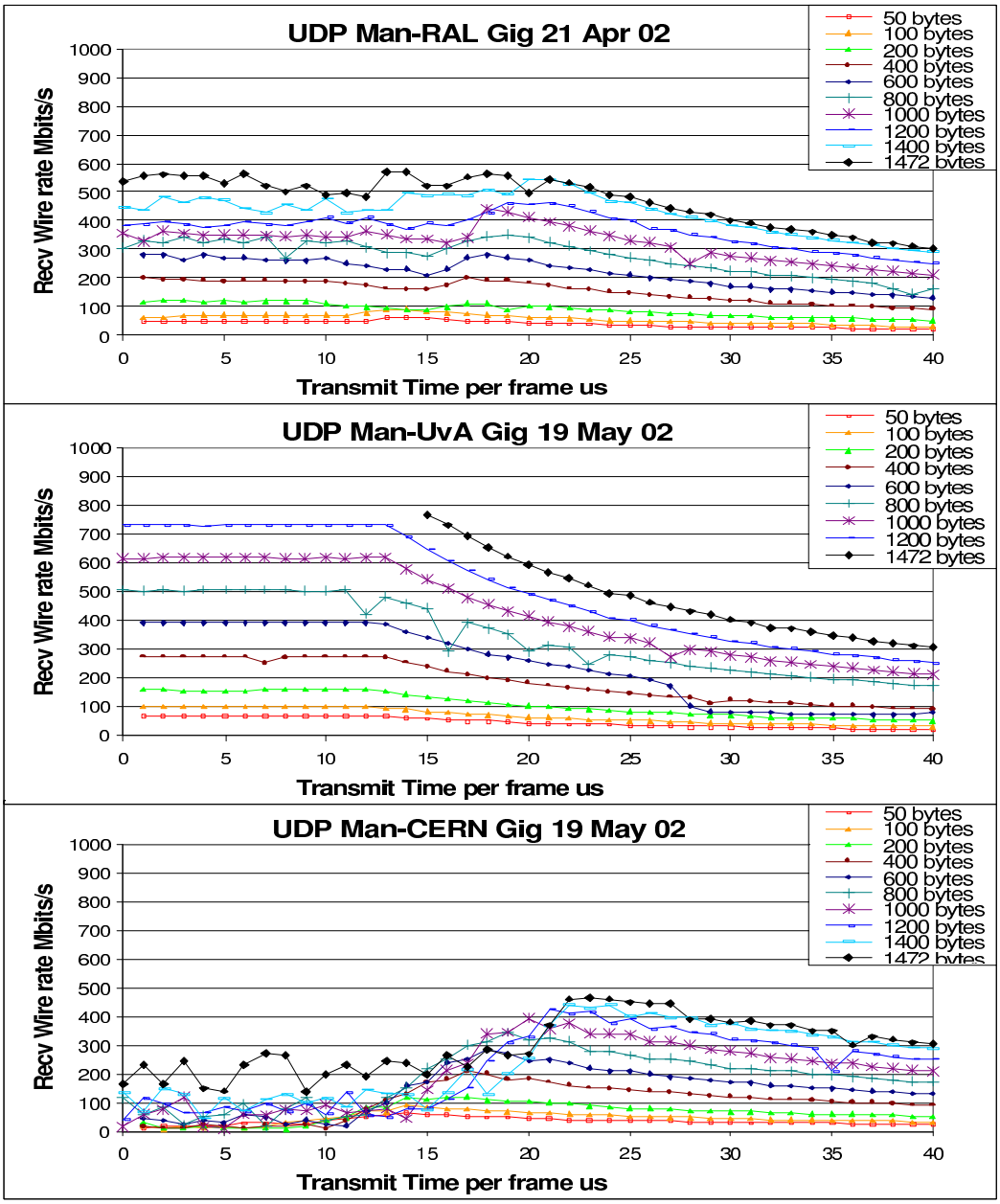


Figure 14: UDP throughput as a function of the time between transmitting frames for various data sizes. Data was sent between Manchester, Amsterdam, CERN and RAL.

The throughputs for both the connections shown in Figure 15 confirm the estimation given in Table 2, as they plateau when the TCP buffer was set to about 1Mbyte.

The 510 Mbit/s measured to Amsterdam agrees favourably with the 750 Mbits/s achieved with a small stream of UDP packet and the 380 Mbit/s to CERN is 82% of the UDP rate.

Time for the TCP transfers was typically 4-20 sec, which should allow the slow start phase of the TCP transfer to be over [COT]. Experimentation of longer tests to check the packet dynamics of the TCP streams if part of future research.

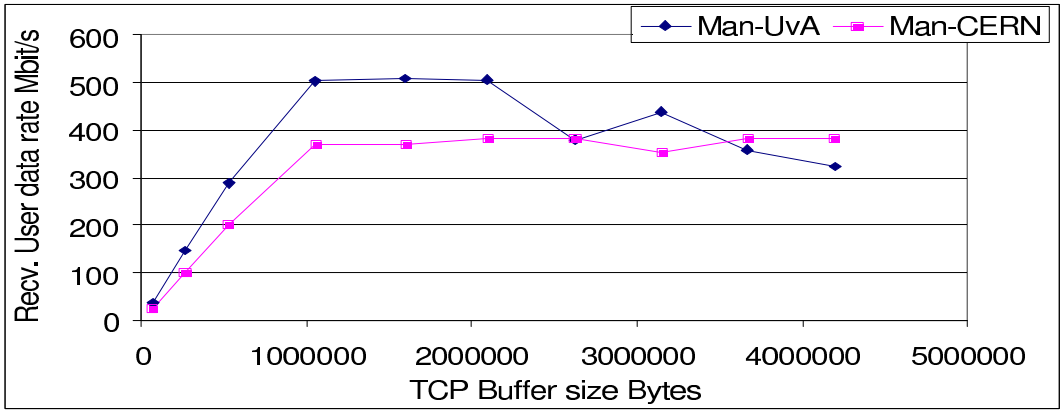


Figure 15: The throughput obtained at the receiver as the send and receive TCP socket buffer sizes were increased. One TCP stream was used.

Additional tests were carried out by varying the number of test streams and results are reported in Figure 16 for different socket size values of TCP connections between Manchester and CERN.

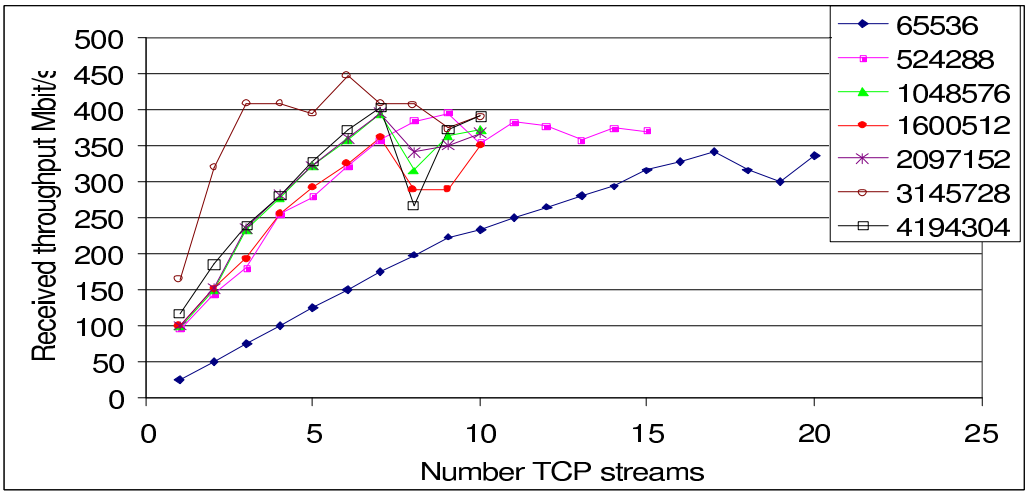


Figure 16: user data throughput as a function of the number of TCP streams used in the transfer

For the default buffer setting (65536, 64KByte) the throughput increases linearly at ~25Mbit/s/stream up to 9 streams when it drops to only approximately 15Mbit/s/stream. The initial rate is 25Mbit/s and the linear increase in throughput up to 9 streams is reasonable as there is ample bandwidth and computing power available. With more than 9 streams in use, it is possible that the streams are competing with themselves for resources.

When the TCP buffer sizes are increased, the rate of increase per stream is larger, and typically plateaus at about 7 streams giving a total throughput of approximately 400 Mbit/s. On notes that although the constrains of the end systems have not changed, the throughputs shown in Figure 16 are somewhat less that that expected from the data shown in Figure 15. The measurements were made at

different times, and it is possible that the average load on the production network was quite different, which may explain the inconsistency in results. This area is under investigation and will be reported in future deliverables.

Discussion

Given the intrinsic intrusiveness on the production network of our tests, in order to minimise possible effects, the tests were run for very brief periods outside the normal working day. It is clear that somewhat longer tests are required to explore the *slow-start* and steady-state phases of the TCP protocol. Suitable discussions are in progress with the NRNs involved.

Results show that to ensure high throughput it is essential to set the TCP kernel parameters to allow large socket buffer sizes to be used. However there are indications in this and other work, that excessive buffer sizes may lead to data being transmitted at such a rate that packets are dropped, TCP then responds by dramatically reducing the congestion window (cwnd) with the overall effect of causing poor throughput. Some form of dynamic tuning of the buffer sizes has been suggested and will be investigated.

Currently the packet loss (and congestion) on the links tested is low, but the loss rate and loss distribution are most important to the dynamic behaviour of the TCP protocol. This would not be true if several high bandwidth flows were to be operated on the same link. A TCP algorithm that was still fair, but more responsive to occasional packet loss and congestion signals, would help in this case. Possible algorithms are being discussed at IETF by the protocol developers and we plan to test some of these ideas when they are available.

9.1.3 Characterization of TCP packet-loss across over medium and long-distance connections

The analysis of TCP performance in different network scenarios (local, metropolitan and wide area networks) is a fundamental step to characterize TCP performance in current best-effort production networks and to evaluate the need of TCP stack enhancements and/or new TCP-oriented packet forwarding services.

In particular, the packet-loss profile if of great importance for packet voice and video applications and non real-time applications using adaptive protocols a la TCP. In fact, different loss distributions can potentially produce widely different perceptions of performance. For example, in voice and video individual missing data units can be recovered without any need of retransmission thanks to redundant information coding. Similarly, isolated TCP segment loss can signalled and recovered in a timely fashion thanks to replicated acknowledgements triggered by the subsequent packets that are correctly received.

Loss is said to be *bursty* when several consecutive packets are lost within a single stream. [PLM] defines a set of packet-loss based metrics that help with the characterization of the packet loss profile:

- *Loss Distance*: the interval between two consecutive lost packets; it can be expressed in time or in sequence numbers
- *Loss Period*: it represents a sequence of consecutive lost packets;
- *Loss Length*: the number of packets in a Loss Period.

Experimentations aimed at characterizing packet loss in two different network scenarios: between INFN (Bologna) and NIKHEF (Amsterdam) and between INFN (Bologna) and the University of Michigan. The Iperf traffic generator was used to source TCP traffic, in addition the Linux 2.4.9 kernel was adopted on end-systems and TCP Selective Acknowledgement was enabled. In both test scenarios TCP socket buffers sizes were set to 4 Mby. Given the low packet loss frequency in both

cases, the packet loss profile was computed by aggregating loss results for several concatenated TCP sessions.

Test results indicate that in both scenarios the average packet loss rate was rather small and of the same order of magnitude: approximately 40 packets every 10000 packets transmitted within Europe, and 10 over 10000 for traffic from the US. However, results show that this small packet loss frequency can have a considerable impact on TCP streams when run on high-speed infrastructures with high packet transmission rates. In fact, in our case test results show that *loss distance* is normally less than 10 sec both within Europe and with the US, which implies very frequent transmission rate adjustments. Figure 17 shows the loss distance within Europe in the interval [2, 14] sec is normally distributed.

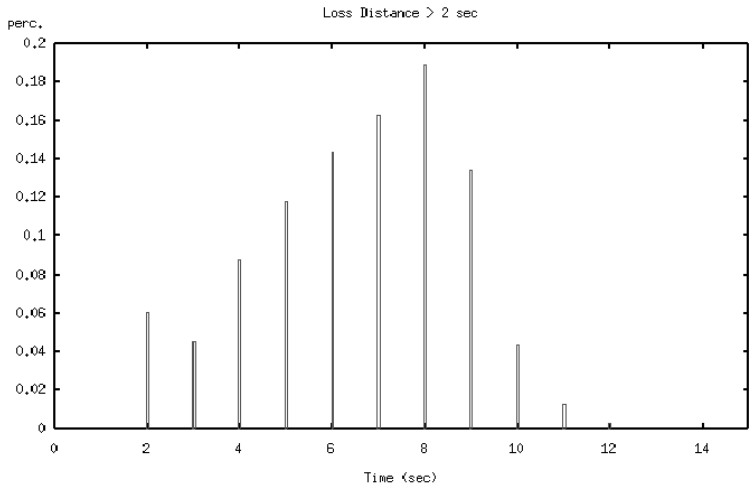


Figure 17: loss distance frequency distribution within Europe

While the loss distance frequency distribution is comparable in the two scenarios, test results show that *packet loss length* can vary significantly, as indicated in Figure 18 and 19, which compare packet loss length for medium and long-distance TCP connections. The graphs show that within Europe individual packet loss is much more frequent than for connections with the US, but in both cases loss length can be up to 10 packets or more. The existence of bursty loss is a clear indication of network congestion, either produced by the congestion avoidance algorithm when the maximum transmission rate is reached by the source, or by external network congestion sources which produce instantaneous traffic peaks.

This bursty packet loss profile clearly indicates the importance of mechanisms in the network for proactive congestion avoidance like ECN and of the availability of services for TCP traffic protection for enhanced TCP performance like the Assured Service.

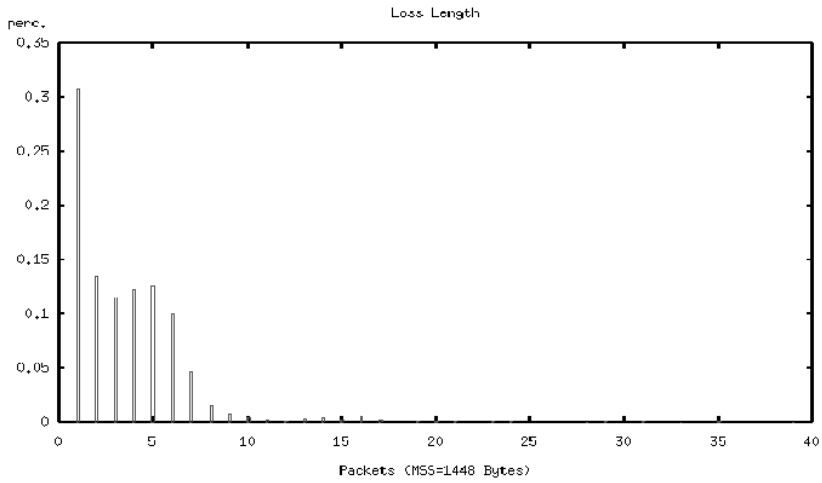


Figure 18: packet loss length for medium-distance TCP connections (EU)

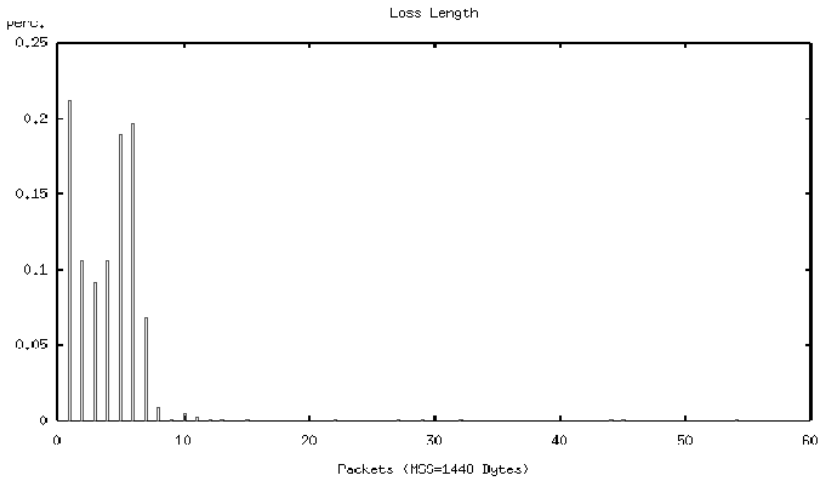


Figure 19: packet loss length frequency distribution for long-distance TCP connections (US)

9.1.4 Proactive TCP congestion avoidance: ECN

Early Congestion Notification (ECN) [ECN] is a type of active queue management technique for the proactive detection of congestion in a given queuing stage of a network element and for its signalling back to the TCP source. Some of the main goals of ECN are the enhancement of TCP performance and a more efficient network resource utilization under light congestion (thanks to the minimization of the number of packets that normally need to be recovered by retransmission in case of congestion).

ECN relies on active queuing mechanism, such as RED or WRED, to detect incipient congestion. A simple extension of RED/WRED is needed so that packets are simply marked instead of being dropped, for example when the average queue size oscillates in the range determined by parameters *min_threshold* and *max_threshold*. If a ECN-marked packet is detected by the destination, the presence of incipient congestion is signalled back to the TCP source through a ECN-specific TCP option in the TCP header of the acknowledgement. Packets are marked with increasing probability when the average queue size is close to the maximum threshold (the mark probability denominator indicates the fraction of packets to be marked when the average is equal to the maximum threshold).

Despite of the use of modified versions of RED/WRED, packet loss can still occur when the average queue size computed by RED exceeds the maximum threshold (RED drop) or when the queue is full (tail drop). The event of a completely full queue is possible, especially when the instantaneous queue size reaches the tail drop threshold before that an ECN cycle is terminated (the destination signals the incipient congestion back to the source and the TCP source consequently reduces its output rate).

Round Trip Time is a critical parameter to test the ECN effectiveness. For this reason tests have been carried out both in an experimental local area network and on a medium-distance production data path between Italy (Bologna) and The Netherlands (Amsterdam). The goal of the tests was to quantify the gain introduced by ECN in terms of throughput and/or number of TCP sessions completed over time.

Laboratory test sessions were carried out to define the RED [RED] region. The ideal RED configuration, expressed in terms of minimum threshold, maximum threshold and drop probability denominator, needs to satisfy the following requirements: a sufficiently large number of packets experiencing congestion is needed in order to quantify the ECN benefit, but the number cannot be too large, since by definition ECN is effective only in presence of light congestion.

The importance of the RED region adopted in test sessions is shown in Figure 20, which plots the gain that can be observed for request/response traffic in presence of ECN for different packet mark denominators.

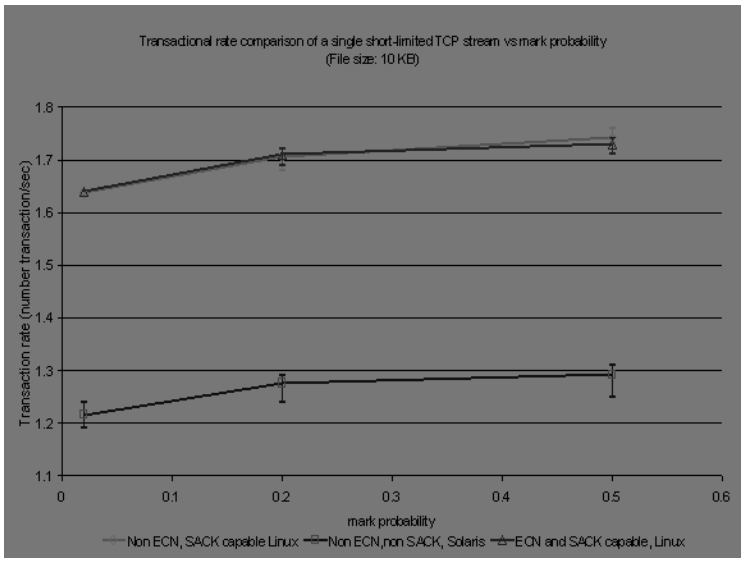


Figure 20: performance gain in terms of number of completed TCP transactions for different ECN packet mark denominators

Different TCP stack implementations – both ECN and non-ECN capable – were compared: TCP New Reno with support of selective acknowledgement (SACK) and with/without ECN support (Linux kernel 2.4.9) and a TCP stack neither ECN nor SACK-capable (Solaris 2.7).

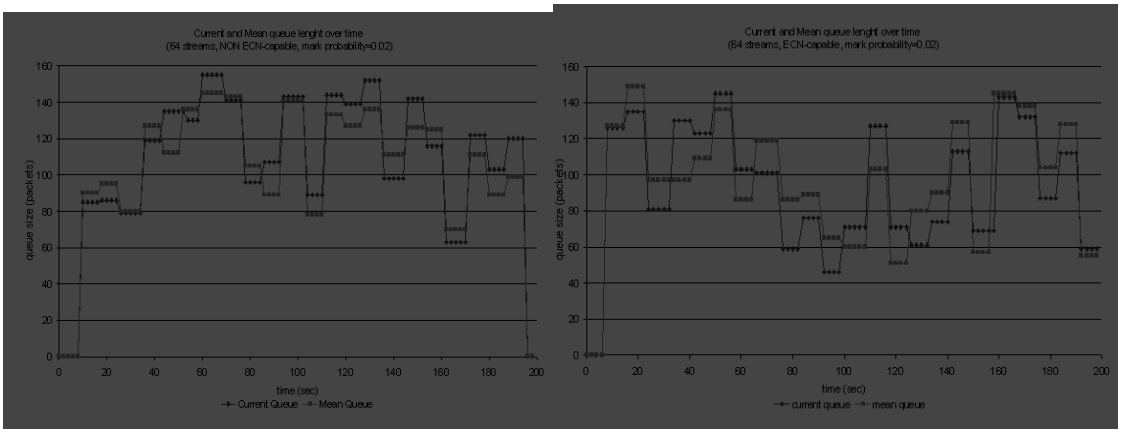
Two different types of TCP sources were used: long-term TCP sessions – to emulate bulk data transfers – and transactional TCP sessions – to emulate TCP traffic produced by interactive TCP-based applications like the Web. For the latter type of traffic the packet generator *Netperf* [NPERF] was used in request/response mode.

Tests in the local area show that the performance gain in presence of TCP is particularly noticeable in presence of transactional TCP sessions, independently of the amount of data transmitted – where performance is expressed by the number of completed TCP sessions over time. In this test scenario the

comparison of two TCP stacks with and without SACK support (both non-ECN capable) shows that great performance gains are also introduced by SACK, which allows the source to identify blocks of consecutively lost packets and consequently to retransmit only the missing data units.

Test results also show that the number of congestion points on the data path and the level of congestion can be fundamental parameter. In our local test scenario we verified that the gain is more relevant in case of multiple congestion points for both transactional and long-term TCP sessions. Running a variable number of concurrent TCP sessions simulates different levels of congestions, as indicated by the SNMP-based tools that we developed to monitor instantaneous and average queue sizes in the router.

Queue length monitoring shows that in presence of ECN the overall burstiness of the TCP aggregate is reduced. Figure 21 (a) and (b) compare the TCP aggregate burstiness monitored with and without ECN for two distinct test session of identical length. The reduced overall burstiness is visible from the number of time intervals for which the instantaneous queue size is the maximum range [140,160]: while with ECN the number of intervals is only approximately 3, without ECN the number of occurrences doubles.



(a) (b)

Figure 21: instantaneous and average queue length for (a) 64 non-ECN capable streams and (b) 64 ECN-capable streams. Drop probability is equal to 1/50 in both cases.

Test sessions on long-distance TCP sessions, which are fundamental for the understanding of the dependency of ECN benefits for different Round Trip Times, are still under way. For a detailed and complete description of the test set-up and results please refer to [EDGecn].

9.1.5 Reliable multicast

TRAM

For preliminary DataGRID experiments on multicast an existing implementation has been selected for an immediate evaluation of reliable multicast benefits: TRAM. We think that receiver-initiated protocols with local recovery are the most promising ones since it's the receiver that can retransmit a repair packet to another local receiver instead of the source. Thanks to this the end-to-end recovery latency can be reduced together with the bandwidth consumption that would be introduced by source retransmissions especially in a large-scale infrastructure.

The TRAM [CHIU98] protocol is a tree-based protocol with the capability of building a dynamic recovery tree, which makes it suitable to the dynamic nature of a multicast group. It supports different types of recovery tree optimised for different application categories and network scenarios without adding complexity to the network infrastructure itself. In addition, TRAM has been designed to scale

to large receiver groups either sparsely or densely distributed and it suites applications producing large data bulks.

TRAM also provides rate-based flow control and congestion avoidance. The data rate is dynamically adjusted according to the feedback information about congestion originally set by the receivers and later on aggregated by *repair heads* through the recovery tree. The rate is lower and upper bounded according to a minimum and maximum threshold configured at the source.

The maximum threshold is used to limit the output burstiness, while the minimum one is used to insure some level of data throughput. Receivers that are too slow (i.e. those that can not support the minimum rate) are pruned off.

TRAM is based on the JRMS package by SUN, which supports a set of libraries and services for building multicast-aware applications. It provides a set of general new services that enable a third party to develop its own reliable multicast protocol. New JRMS services include multicast address allocation and channel management. In addition, a stress test environment is also available for testing of JRMS reliable multicast protocol performance, for example in terms of scalability, transfer rate and throughput, through textual and graphical description of every aspect of the test.

Experimental scenarios and test methodology

Tests have been carried out in different network scenarios with increasing degree of complexity and scale. An initial dedicated experimental testbed has been set-up and has been extended later on to include National Research Networks (NRNs) and the European backbone GÉANT.

LAN set-up: it consists of six sender/receiver nodes and three routing elements that in some cases also acted as receivers. Production platforms (CISCO routers) and *mrouted*-based end-nodes were used for routing. This set-up was used to verify the basic functionality of the package and in particular its capability of supporting reliable transfer. Binary files were exchanged and remotely executed to prove their integrity. The NISTnet tool, a WAN simulator, was also used to introduce variable packet loss and/or Round Trip Time (RTT). Tunnelling of multicast traffic for transparent forwarding over a non-multicast capable network was also used.

WAN set-up: it is based on the European GÉANT core and it spans three countries: Czech Republic (three nodes at CESNET and Masaryk University, Brno) Slovakia (6 nodes at Safarik University and Technical University, Kosice) and England (1 node at RAL, London). Results gathered in this set-up were strongly affected by the performance and configuration of multicast in each of the three NRNs. The non-availability of a homogeneous widespread multicast platform across Europe was a problem for the successful completion of the tests.

Results

Several problems had to be addressed during the test phase. Firstly, some versions of SUN JAVA SDK showed wrong functionality by means of error messages and the JRMS package suffered from hardware dependencies and this affected the application performance at a higher level. In addition, the non-availability of a homogeneous widespread multicast platform across Europe was a problem for the set-up of the test platform in the WAN.

From the hardware and software point of view, the initial tests carried out so far indicate that most of the end-system network interface cards can correctly support IP multicast and SUN SDK 1.4.0 is recommended for good compatibility with the JRMS package.

LAN tests based on NISTnet show that the JRMS package and TRAM correctly provide reliability in multicast file transfer in presence of packet loss. In WAN tests TRAM showed correct functionality and the repair tree type was the same as in the LAN environment thanks to the negligible packet loss observed in the NRN and GÉANT networks, in which bandwidth is largely over-provisioned. We can

conclude that our interim results show a good match between application requirements and the reliability and functionality offered by the architecture under test.

Our future work will cover address in details aspects like scalability, repair tree construction, and the tuning of optimisation of parameters like congestion and ACK windows.

9.2 TCP-TUNING SCRIPT: INSTALLATION AND GUIDELINES

socket_conf

```
#! /bin/sh
#
# socket_conf:  Set the socket parameters
# chkconfig:    2345 96 96
# description:  Set the socket configuration by defining the appropriate \
#               parameters in /proc
#
# Init file to set the socket configuration by defining the appropriate
# parameters in the "/proc" pseudo file system.
#
# This job can probably also be done with a kernel reconfiguration.
#
#!
##
# Define the minimum, default and maximum socket receive & write socket sizes to
# set. In kernel versions other than V. 2.4 only the maximum sizes will be set.
##

RMEM_MIN=4096
WMEM_MIN=4096
RMEM_DFT=87380
WMEM_DFT=65536
RMEM_MAX=4194304
WMEM_MAX=4194304

##
# Source the function library when available (a.o. RedHat). If not alternatives
# for the functions used are defined.
##
if [ -f /etc/rc.d/init.d/functions ]    # Source function library.
then
    . /etc/rc.d/init.d/functions
else                                    # Define alternative functions.
    success() {
        echo -n "."
        return 0}

    failure() {
        return 1}
fi
```

```
##
# Define a function to set the socket sizes in "/proc". Set only the maximum
# values for Linux kernels with versions other than V. 2.4.
##
start()
{
    LINUX_VERSION="`uname -r | cut -d . -f 1-2`"

    echo -n "Configuring socket parameters: "
    echo $RMEM_MAX > /proc/sys/net/core/rmem_max  && \
    echo $WMEM_MAX > /proc/sys/net/core/wmem_max  && \
    ( \
        [ $LINUX_VERSION != "2.4" ] \
        || \
        echo "$RMEM_MIN $RMEM_DFT $RMEM_MAX" > /proc/sys/net/ipv4/tcp_rmem  && \
        echo "$WMEM_MIN $WMEM_DFT $WMEM_MAX" > /proc/sys/net/ipv4/tcp_wmem \
    )  && \
    success  ||  failure
    RESULT=$?
    echo
    return $RESULT
}

##
# Handle finally the specified init action.
##
case "$1" in
    start | restart )    # Set the socket parameters in "/proc".
        start
        exit $?
        ;;
    stop | reload | condrestart | status )    # Ignore.
        exit 0
        ;;
    * )    # Unknown action: give a usage message.
        echo "*** Usage: $0" \
             "{start|stop|status|restart|reload|condrestart}"
        exit 1
esac
```

Script installation and usage guidelines

Script has to be copied to directory `/etc/rc.d/init.d` or `/etc/init.d`, wherever the `init` directory reside in the system. The minimum, default and maximum socket parameters `RMEM_MIN`, ..., `WMEM_MAX` can be set to the desired values.

If the system supports `chkconfig`, one of the two following options has to be used:

```
man chkconfig
info chkconfig
```

It is presumably sufficient to add the following commands:

```
chkconfig --add socket_conf
chkconfig socket_conf on
```

If desired `chkconfig` and `description` fields used by the `chkconfig` program can be edited.

If `chkconfig` has not been used or you do not want to use it, the script can be started from the appropriate run level `<run_level>`, by setting the following symbolic links:

```
cd     /etc/rd.d/rc<run_level>.d
ln     -s    ../init.d/socket_conf S96socket_conf
```

Value 96 may be adjusted according to the order to be followed in setting the socket parameters. The run level where these commands should be set are system-dependant, but in general should include the run levels 3 and 5. See the `init` man and `/etc/inittab` for more information.

10 ANNEX C: THE NETWORK-BASED GRID OPTIMIZATION SERVICE

10.1 API AND PROPOSED EXTENSIONS OF INFORMATION SCHEMA

Two different Application Programming Interfaces (APIs) have been defined for the provisioning of the optimisation service to resource brokerage and data management, each based on a different cost function. The former API is based on the *Closeness* function defined in this document and in [OPT02], while the latter on estimated file transfer time.

Resource Brokerage API

It deploys function `getBestCeSes()` that requires the following input and output parameters:

1. Input parameters:
   - `CElist`: the list of candidate CEid's;
   - `SEmx`: matrix of SEid's, where `SEmx[1,i]` is the list of SEid's of SEs with a replica of file f_i
   - `alpha`: tuning parameter used by function Closeness and specified by the Borker;
   - `timeTolerance`: maximum age of a measurement sample requested by the Broker to consider a given sample provided by the information service valid;
   - `toolName`: list of names of the tools requested to measure RTT, throughput and packet loss.
2. Output parameters:
   - `BestCE`: identifier of best CE;
   - `BestSE`: list of identifiers of best SEs (SE_i corresponds to file f_i);
   - `err`: estimation error;
   - `S`: overall score of bestCE

Data management API

The data management API is based on function `getNetworkCosts()` implementing the computation of the Proximity function. This function provides an estimate of the current network costs between two Storage Elements.

The list of input and output parameters follows.

1. Input parameters:
   - `SEid1`: identifier of the Storage Element from which a given file has to be transferred;
   - `SEid2`: identifier of the Storage Element the file is destined to;
   - `filesize`: size of file in MBy to be transferred.
2. Output parameters:
   - `networkCosts[]`: array of `getNetworkCosts` type, where this a data structure containing information about estimated transfer time and the corresponding estimation error:

```
public class networkCosts  {
   private float cost;
   private float error;
```

```
        private float getCost() { return cost; }
        private float getError() { return error; }
    }
```

In the current release the return value `networkCosts[]` is just one element, and the cost returned is the time to transfer the file in seconds.

To compute the function the history of file transfers logged by the GridFTP application is used. The instrumented version of the tool supporting NetLogger [NLOG] logging is requested. If no measurement history is available, which is the case now, then time is simply computed according to the following formula:

*filesize (in bytes) * 8 / throughput*

where *throughput* is computed by the `edg-iperfer` tool. This tool (based on `iperf` measurements) periodically measures the throughput available between different testbed sites and publishes the results into the information system.

For two Storage Element identifiers, the function extracts the corresponding transfers logs and bandwidth information through LDAP (RFC 2251) searches through the full information service hierarchy. Currently, a full search from the top DataGRID CERN GIIS is needed, however this approach can be optimised thanks to the addition of a new object class, the *Network Element*, thanks to which any Computing Element or Storage Element can be directly bound to the monitoring domain it belongs to. Both the SE and CE object class have to be extended by including the NetworkElement identifier they correspond to.