

31.08.05

Deliverable DS3.3.2: GÉANT2 Performance Enhancement and Response Team (PERT) User Guide and Best Practice Guide



Deliverable DS3.3.2

Contractual Date:	31/05/05
Actual Date:	31/08/05
Contract Number:	511082
Instrument type:	Integrated Infrastructure Initiative (I3)
Activity:	SA3
Work Item:	3
Nature of Deliverable:	R (Report)
Dissemination Level	PU (Public)
Lead Partner	SWITCH
Document Code	GN2-05-176v4

Authors: Francois Xavier Andreu (RENATER), Alex Gall (SWITCH), Ann Harding (HEAnet), Simon Leinen (SWITCH), Colm MacCárthaigh (HEAnet), Orla McGann (HEAnet), Simon Muyal (RENATER), Hank Nussbacher (IUCC), Toby Rodwell (DANTE), Ulrich Schmid (SWITCH), Robert Stoy (DFN), Chris Welti (SWITCH)

Abstract

Apart from investigating specific cases of poor network performance, the other main function of the GÉANT2 Performance Enhancement and Response Team (PERT) is to populate and manage the PERT Knowledgebase, an online resource for advice and information about all aspects of network performance issues. The content of the PERT Knowledgebase has been used to produce two reference documents,

called the 'Research Network User Guide to Performance' ('User Guide' for short) and 'The Best Practice Guide'. The former is of interest to everyone concerned with the network performance of end-systems, whilst the latter is more aimed at the campus network administrators responsible for LAN equipment and edge routers.

The User Guide begins with an introduction to the basics of network performance, relating what is experienced by the end-user with the principles of performance (one way delay, jitter, packet re-ordering and loss). There is then a section on how an end-user may investigate their own network performance problems (what to check, what tools to use). Subsequent sections look at the TCP protocol, hardware issues and enhancement, the tuning of different types of operating system, and considerations for application design. The final section presents a number of case studies, looking at some common problems (trans-Atlantic file transfer issues) and some not so common issues (a study of recent attempts to break the Internet land-speed record).

Being directed at network administrators, the Best Practice Guide differs assumes a certain level of network knowledge on the part of its reader (in the User Guide, only a basic appreciation of computing networking concepts is required). The Best Practice Guide re-visits, in greater depth, a few subjects introduced in the User Guide. It also looks monitoring and measurement methods, and the use of Differentiated Services, particularly in the GÉANT/GÉANT2 network.

Table of Contents

0	General Background and Context	1
0.1	Motivation and Purpose	1
0.2	Conclusions and Future Work	2
1	Acronyms	3
Appendix A	GÉANT2 PERT User Guide	5
0	Executive Summary	11
0.1	Goals and target readership	11
0.2	Related work	11
1	Performance Basics	12
1.1	User-perceived performance	12
1.1.1	Responsiveness	12
1.1.2	Capacity and throughput	12
1.1.3	Robustness	13
1.2	Network performance metrics	13
1.2.1	One-Way Delay (OWD)	13
1.2.2	Round-Trip Time (RTT)	15
1.2.3	Delay Variation (Jitter)	15
1.2.4	Packet Loss	15
1.2.5	Packet Reordering	16
1.2.6	Maximum Transmission Unit (MTU)	17
1.2.7	Bandwidth Delay Product (BDP)	18
1.3	Translating Performance Metrics	18
2	First steps at investigating performance problems	20
2.1	Problem isolation strategies	20
2.1.1	Defining the problem	20

2.1.2	Gathering facts	21
2.1.3	Considering possibilities	21
2.1.4	Reporting the problem	21
2.2	Measurement tools	22
2.2.1	Latency/Delay Measurement Tools	23
2.2.2	Bandwidth Utilisation Measurement Tools	29
2.2.3	Path Probe Tools	31
2.2.4	Network Simulation and Benchmarking Tools	33
2.2.5	Traffic Analysis Tools	35
3	TCP Performance Primer	36
3.1	Window-based transmission	37
3.2	Rate control	37
3.2.1	Slow Start	37
3.2.2	Congestion Avoidance	38
3.2.3	Fast Retransmit	38
3.2.4	Fast Recovery	38
3.3	TCP Performance enhancements	39
3.3.1	Window scaling & timestamps	39
3.3.2	SACK	40
3.3.3	Explicit Congestion Notification	41
3.4	High-performance TCP variations	42
3.4.1	HSTCP, H-TCP, BIC, FAST etc.	42
4	Application and protocol considerations	44
4.1	Designing tolerant applications	44
4.1.1	"Chatty" Protocols	44
4.1.2	Performance-friendly I/O interfaces	46
4.2	Choosing applications	47
4.2.1	Protocols	48
4.2.2	Applications	49
5	Performance Case Studies	51
5.1	Duplex Mismatch	51
5.2	Transatlantic File Transfer Performance Troubleshooting	52
5.2.1	Problem Statement	52
5.2.2	Results Summary	52

5.2.3	Improving Performance	53
5.3	High-Performance Web Server For Large Audiences	53
5.3.1	Apache	53
5.3.2	File System	55
5.3.3	Kernel	55
5.4	Internet2 Land Speed Record Hosts	57
5.4.1	SUNET/SPRINT	57
5.4.2	CALTECH/CERN/CENIC	58
5.4.3	University of Tokyo/WIDE/Chelsio	59
6	Conclusions and further work	60
7	References	61
Appendix B	GÉANT2 PERT Best Practice Guide	65
0	Executive Summary	70
0.1	Goals and target readership	70
0.2	Related work	70
1	Hosts, Adapters, OS Tuning	71
2	Hardware considerations	72
2.1	Network adapters	72
2.1.1	TCP Offload Engines (TOEs)	72
2.1.2	Large Send Offload (LSO)	73
2.1.3	Interrupt Coalescence	73
2.1.4	Checksum Offload	73
2.2	File systems and disks	74
2.2.1	Benchmarking	74
2.2.2	Tuning	74
3	Operating system considerations	76
3.1	Out-of-the box system settings and tuning	76
3.2	Operating-specific tuning tips and tools	76
3.2.1	Microsoft Windows	76
3.2.2	Linux	77

3.2.3	BSD Variants	79
3.2.4	MAC OS X	79
3.2.5	Solaris	79
4	Performance Pitfalls	80
4.1	Path MTU Discovery Issues	80
4.2	Middleboxes	81
4.3	Duplex modes and auto-negotiation	82
4.4	LAN Collisions	84
4.5	LAN Broadcast Domains	84
5	Monitoring and measurement	85
5.1	Proactive vs. Reactive Measurements	85
5.2	Proactive passive monitoring	86
5.2.1	SNMP monitoring frameworks	87
5.2.2	Netflow accounting	88
5.3	Proactive Active measurements	89
5.3.1	Traditional Methods and Tools	89
5.3.2	Measurement Boxes Frameworks	91
5.4	Reactive Passive Measurements	93
5.4.1	SNMP queries with small polling intervals	94
5.4.2	Traffic capture and analysis	94
5.5	Reactive Active Measurements	95
5.5.1	Reachability and Round Trip Times	95
5.5.2	Path Characteristics	96
5.5.3	Bandwidth Measurements	98
5.5.4	Advanced Diagnosis Tools	101
6	Differentiated Service over GEANT	102
6.1.1	Premium IP	102
6.1.2	LBE (Less Than Best Effort) Service	102
7	References	104

Table of Figures

Figure 1.1: End User and Network Metrics	19
Figure 2.1: Example IPv4 ping output	23
Figure 2.2: Example IPv4 traceroute output	24
Figure 2.3: Example MTR output	25
Figure 2.4: Example PingPlotter output	26
Figure 2.5: Example LFT output	27
Figure 2.6: Example Smokeping output 1	27
Figure 2.7: Example Smokeping output 2	27
Figure 2.8: Example pchar output	28
Figure 2.9: Example MRTG output	30
Figure 2.10: Example Cricket output	31
Figure 2.11: Example tracepath output	32
Figure 2.12: Example traceproto output	33
Figure 2.13: Example traceroute MTU Discovery	33
Figure 4.1: SSH Cipher Performance	50
Figure 5.1: Example Duplex Check	52
Figure 3.1: Microsoft Windows TCP Tuning 1	77
Figure 3.2: Microsoft Windows TCP Tuning 2	77
Figure 3.3: Linux TCP Tuning 1	78
Figure 3.4: Linux TCP Tuning 2	78
Figure 3.5: Linux TCP Tuning 3	78
Figure 3.6: Linux TCP Tuning 4	78
Figure 3.7: MAC OSX TCP Tuning	79
Figure 5.1: IPPM system example: Here one way delay and one way delay variation is shown graphically on a measurement path. Increased delay and variations have been caused by a saturated backbone access link.	92
Figure 5.2: RIPE TTM system example: Here the graphical output on a measurement path without any problems during a 24h timeframe is shown	93

0 General Background and Context

0.1 Motivation and Purpose

The GN2 Performance Enhancement and Response Team (PERT) is a virtual team committed to helping academic users get efficient network performance for their end-systems. There is an emerging need for this kind of support, due to the proliferation of systems which are dependent on Long Fat pipe Networks (LFNs), and whose requirements exceed the scope of the original TCP specification (TCP being the most common of the network transport protocols used on top of the Internet Protocol (IP)).

The PERT offers support in two ways. First, they respond to requests to investigate quality of service (QoS) issues submitted by the PERT Primary Customers. The PERT Primary Customers are the European NRENs, peer academic networks and certain international academic research projects (the full list of PERT Primary Customers is given in 'The PERT Policy' (GN2-05-18)). The purpose of specifying a relatively small number of Primary Customers is to ensure that the majority of potential users of the PERT (the PERT's 'End Customers') first contact their NREN, who will be able to help with minor problems, or those issues that are in fact outside the scope of the PERT (such as network hardware failure).

The second way in which the PERT helps its clients is by producing reference documentation that end users and network administrators can use for self-help. This documentation explains the concepts of network performance, highlights the most common causes of quality of service (QoS) problems, and offers general guidelines on how to configure systems to optimise performance. It also provides pointers to other resources for troubleshooting issues, such as the NREN network statistics and monitoring tools that will help end users and network administrators to determine quickly for themselves whether current problems are likely to be the result of changes in network conditions. This information is all held in the PERT Knowledgebase, an online resource that is accessible to all (see GEANT2 web site, under 'Users' -> 'PERT' for a link).

In order to showcase the work of the PERT in this area, a snapshot has been taken of the content of the PERT Knowledgebase and this has been used to create two self-contained guides to network quality of service.. The 'User Guide', as its name suggests, is targeted at end-users, and in particular those end-users who have demanding requirements (typically those who depend on LFNs), and the network administrators who support

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

them. The User Guide is at Annex A of this document. The second guide (Annex B) is called the 'Best Practice Guide'. This document is aimed more at campus network administrators, since it concentrates on issues which can not be controlled by re-configuring or upgrading end-systems. In fact the Best Practice Guide is really an addendum to the User Guide, in as much as its target audience (network administrators) is a subset of the User Guide's target audience.

0.2 Conclusions and Future Work

Whilst the online PERT Knowledgebase, which will always have the most up to date content, is expected to be the main reference source for PERT users seeking specific advice on a given subject, the two guides presented here will provide a useful and easy to read introduction to the main issues surrounding network performance issues today.

A significant amount of effort was put into building up the PERT Knowledgebase, prior to producing these two guides, and the effort is planned to continue, albeit at a slower, steadier pace, over the rest of GEANT2. The PERT Knowledge base will be kept under regular review, and as and when it has changed significantly from its present state the guides will be re-published.

1 Acronyms

ACK	TCP acknowledgement packet
ASIC	Application-Specific Integrated Circuit
AIMD	Additive Increase/Multiplicative Decrease
AQM	Active Queue Management
ATM	Asynchronous Transfer Mode
BDP	Bandwidth Delay Product
CE	Congestion Experienced
CPU	Central Processing Unit
CSMA/CD	Collision Sense Multiple Access/Collision Detection
Cwnd	Congestion Window
CWR	Congestion Window Reduced
ECE	ECN-Echo
ECN	Explicit Congestion Notification
ECT	ECN-Capable Transport
ECMP	Equal Cost Multipath
EGEE	Enabling Grids for E-Science in Europe
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPDV	IP Delay Variation Metric
IPPM	IP Performance Metrics
LFN	Long Fat Network
MTU	Maximum Transmission Unit
MDT	Multidata Transmit
MSS	maximum segment size
NREN	National Research and Education Network
OS	Operating System
PAWS	Protection Against Wrapped Sequence-number

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

PDU	Protocol Data Unit
PERT	Performance Enhancement & Response Team
Pmtud	Path MTU Discovery
POS	Packet Over Sonet
RCP	Remote Copy
RED	Random Early Detection
RFC	Request For Comment, Technical and organisational notes about the Internet
RTP	Real-time Transport Protocol
RTT	Round-trip time
SACK	Selective ACKnowledgments
SCP	Secure Copy
SCTP	Stream Control Transmission Protocol
SMDS	Switched Multimegabit Data Service
SSH	Secure Shell
Ssthresh	Slow Start Threshold
SYN	TCP synchronisation packet
TCP	Transmission Control Protocol
TOE	TCP Offload Engine
TTL	Time To Live
VOP	Velocity of Propagation
WIDE	Widely Integrated Distributed Environment
WLAN	Wireless Local Area Network

Appendix A **GEANT2 PERT User Guide**

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

31.08.05

Deliverable DS3.3.2 – Part 1 (App. A): Research Network User's Guide to Performance



Deliverable DS3.3.2 – Part 1

Contractual Date: 31/05/05
Actual Date: 31/08/05
Contract Number: 511082
Instrument type: Integrated Infrastructure Initiative (I3)
Activity: SA3
Work Item: 3
Nature of Deliverable: R (Report)
Dissemination Level: P (Public)
Lead Partner: SWITCH
Document Code: GN2-05-176v4

Authors: Francois Xavier Andreu (RENATER), Alex Gall (SWITCH), Ann Harding (HEAnet), Simon Leinen (SWITCH), Colm MacCárthaigh (HEAnet), Orla McGann (HEAnet), Simon Muyal (RENATER), Hank Nussbacher (IUC), Toby Rodwell (DANTE), Ulrich Schmid (SWITCH), Chris Welti (SWITCH)

Abstract

This paper is an end-user's guide to network performance issues. We present the fundamentals of network performance, and then look at ways of investigating and remedying problems on end-systems. We finish with several case studies which demonstrate the effects of the issues previously examined.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Table of Contents

0	Executive Summary	11
0.1	Goals and target readership	11
0.2	Related work	11
1	Performance Basics	12
1.1	User-perceived performance	12
1.1.1	Responsiveness	12
1.1.2	Capacity and throughput	12
1.1.3	Robustness	13
1.2	Network performance metrics	13
1.2.1	One-Way Delay (OWD)	13
1.2.2	Round-Trip Time (RTT)	15
1.2.3	Delay Variation (Jitter)	15
1.2.4	Packet Loss	15
1.2.5	Packet Reordering	16
1.2.6	Maximum Transmission Unit (MTU)	17
1.2.7	Bandwidth Delay Product (BDP)	18
1.3	Translating Performance Metrics	18
2	First steps at investigating performance problems	20
2.1	Problem isolation strategies	20
2.1.1	Defining the problem	20
2.1.2	Gathering facts	21
2.1.3	Considering possibilities	21
2.1.4	Reporting the problem	21
2.2	Measurement tools	22
2.2.1	Latency/Delay Measurement Tools	23
2.2.2	Bandwidth Utilisation Measurement Tools	29
2.2.3	Path Probe Tools	31
2.2.4	Network Simulation and Benchmarking Tools	33

2.2.5	Traffic Analysis Tools	35
3	TCP Performance Primer	36
3.1	Window-based transmission	37
3.2	Rate control	37
3.2.1	Slow Start	37
3.2.2	Congestion Avoidance	38
3.2.3	Fast Retransmit	38
3.2.4	Fast Recovery	38
3.3	TCP Performance enhancements	39
3.3.1	Window scaling & timestamps	39
3.3.2	SACK	40
3.3.3	Explicit Congestion Notification	41
3.4	High-performance TCP variations	42
3.4.1	HSTCP, H-TCP, BIC, FAST etc.	42
4	Application and protocol considerations	44
4.1	Designing tolerant applications	44
4.1.1	"Chatty" Protocols	44
4.1.2	Performance-friendly I/O interfaces	46
4.2	Choosing applications	47
4.2.1	Protocols	48
4.2.2	Applications	49
5	Performance Case Studies	51
5.1	Duplex Mismatch	51
5.2	Transatlantic File Transfer Performance Troubleshooting	52
5.2.1	Problem Statement	52
5.2.2	Results Summary	52
5.2.3	Improving Performance	53
5.3	High-Performance Web Server For Large Audiences	53
5.3.1	Apache	53
5.3.2	File System	55
5.3.3	Kernel	55
5.4	Internet2 Land Speed Record Hosts	57
5.4.1	SUNET/SPRINT	57
5.4.2	CALTECH/CERN/CENIC	58

5.4.3	University of Tokyo/WIDE/Chelsio	59
6	Conclusions and further work	60
7	References	61

Table of Figures

Figure 1.1: End User and Network Metrics	19
Figure 2.1: Example IPv4 ping output	23
Figure 2.2: Example IPv4 traceroute output	24
Figure 2.3: Example MTR output	25
Figure 2.4: Example PingPlotter output	26
Figure 2.5: Example LFT output	27
Figure 2.6: Example Smokeping output 1	27
Figure 2.7: Example Smokeping output 2	27
Figure 2.8: Example pchar output	28
Figure 2.9: Example MRTG output	30
Figure 2.10: Example Cricket output	31
Figure 2.11: Example tracepath output	32
Figure 2.12: Example traceproto output	33
Figure 2.13: Example traceroute MTU Discovery	33
Figure 4.1: SSH Cipher Performance	50
Figure 5.1: Example Duplex Check	52

0 Executive Summary

0.1 Goals and target readership

The aim of this document is to provide information to researchers and end-users that have networking requirements for their work, but are not necessarily networking specialists. It gives an overview of the issues that may be faced when trying to achieve the best network performance and Quality of Service possible. It will also provide tips and guidelines for the end-users themselves, in order to understand network performance metrics and to get the most from the network they are working on.

Due to constant research and development, new advances in this area occur frequently. This is therefore a snapshot document, based on information gathered to date in the PERT (Performance Enhancement and Response Team) Knowledge base as part of Géant2 Service Activity 3 (PACE (Performance and Allocated Capacity for End-users)) and may be subject to revision over the course of the Service Activity. For the most up to date information please refer to the PERT Knowledge base, which can be reached via the GEANT2 web site (www.geant2.net).

0.2 Related work

Delivery of high-quality end-to-end network performance is a co-operative effort. In addition to this Research Network User's Guide to Performance, DS3.3.2 Part 2, Best Practice Guide for Campus Networks provides advice for campus-sized networking organisations on maximising performance and monitoring of networking devices. These two guides, in conjunction with GN2 Deliverable D.S.3.9.1: Policy for allocation of Premium IP and the flexible hybrid network services offered by Géant2, facilitate delivery of high-quality end-to-end performance.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

1 Performance Basics

1.1 User-perceived performance

User-perceived performance of network applications is made up of a number of mainly qualitative metrics, some of which are in conflict with each other. In the case of some applications, a single metric will outweigh the others, such as responsiveness from video services or throughput for bulk transfer applications. More commonly, a combination of factors usually determines the experience of network performance for the end-user.

1.1.1 Responsiveness

One of the most important user experiences in networking applications is the perception of responsiveness. If end-users feel that an application is slow, it is often the case that it is slow to respond to them, rather than being directly related to network speed. This is a particular issue for real-time applications such as audio/video conferencing systems and must be prioritised in applications such as remote medical services and off-campus teaching facilities. It can be difficult to quantitatively define an acceptable figure for response times as the requirements may vary from application to application.

1.1.2 Capacity and throughput

An important user metric, in the case of network applications that rely on bulk transfer, is capacity. In the past, many applications were hindered by the lack of available high-bandwidth connections. A quantitative measurement term for this experience is “throughput”; defined as the rate at which a computer or network sends or receives data.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

1.1.3 Robustness

For networked applications to be viable replacements for the previous methods of bulk data transfer, and to facilitate new high-quality highly responsive applications, network performance must be robust. This includes not only planning for resilience and avoiding faults and outages, but also having predictable known performance metrics available for the effective design and use of these new applications. These two working definitions of robustness can conflict, as resilience techniques used to ensure availability could result in different performance metrics across different network paths.

1.2 Network performance metrics

There are many metrics that are commonly used to characterize the performance of networks and constituent parts of networks. We present the most important of these metrics, explain how they can be measured, what they indicate about end-to-end performance, and what can be done to improve them.

Traditionally, the metric of focus in networking has been bandwidth. As more and more parts of the Internet have their capacity upgraded, bandwidth is often no longer the main problem.

A framework for network performance metrics has been defined by the IETF's IP Performance Metrics (IPPM) Working Group in RFC 2330. The group also developed definitions for several specific performance metrics detailed in this section.

1.2.1 One-Way Delay (OWD)

One-way delay is the time it takes for a packet to reach its destination. It is considered a property of network links or paths. RFC 2679 contains the definition of the one-way delay metric of the IETF's IPPM Working Group. Network-induced latency often has noticeable impact on performance. While this can be measured in one-way delay, when studying end-to-end performance, it is usually more interesting to look at metrics that are derived from one-way delay such as Round Trip Time (RTT) and Delay Variation (DV).

One-way delay along a network path can be broken down into per-hop one-way delays, and these in turn into per-link and per-node delay components.

The per-link component of one-way delay consists of two sub-components: propagation delay and serialization delay. The per-node component of one-way delay also consists of two sub-components: forwarding delay and queuing delay.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

1.2.1.1 Propagation Delay

Propagation delay is the duration of time for signals to move from the transmitting to the receiving end of a link. On simple links, this is the product of the link's physical length and the characteristic propagation speed. The velocity of propagation (VoP) for copper and fibre optic are similar (the VoP of copper is slightly faster), being approximately 2/3 the speed of light in a vacuum.

On high-speed wide-area network (WAN) paths, delay is usually dominated by propagation times; i.e. the time required for transmission of the signal on a link. Therefore, the physical routing of network links on the campus and across the wide area network plays an important role in network performance, as well as the topology of the network and the selection of routing metrics.

1.2.1.2 Serialization Delay

Serialization delay is the time it takes for a packet to be separated into sequential link transmission units (typically bits). It is obtained by dividing the packet size (in bits) by the capacity of the link (in bits per second). Nowadays, as links increasingly have a higher bit rate, serialization delay is less relevant.

1.2.1.3 Forwarding Delay

Forwarding delay is the time it takes for the node to process a packet and send it to its destination. The processing of the packet involves reading the forwarding-relevant information (typically the destination address and other headers) from the packet and computing the forwarding decision (based on routing tables and other information), before actually forwarding the packet towards the destination. This may involve copying the packet to a different interface inside the node, rewriting parts of it such as the IP TTL and any media-specific headers and any other processing such as fragmentation, accounting, or checking access control lists.

1.2.1.4 Queuing Delay

Queuing delay is defined as the time a packet has to spend inside a node such as a router while waiting for availability of the output link. It depends on the amount of traffic competing to send packets towards the output link and on the priorities of the packet and those of the competing traffic. There can be causes for queuing delay other than contention on the outgoing link, such as contention on the node device's backplane.

1.2.2 Round-Trip Time (RTT)

Round-trip time (RTT) is the total time taken for a packet sent by a node A to reach a destination B and then for a response to be sent back by B to reach A. In other words, the round-trip time is the sum of the one-way delays from A to B and from B to A, and of the time it takes B to formulate the response to the original packet.

Large RTT values can cause problems for Transmission Control Protocol (TCP) and other window-based transport protocols. The round-trip time influences the achievable throughput, as there can only be a window's worth of unacknowledged data in the network. Section 3 explains the operation of TCP windows in more detail.

For interactive applications such as conversational audio/video, instrument control, or interactive games, the RTT represents a lower bound on response time, and thus impacts on perceived responsiveness directly.

1.2.3 Delay Variation (Jitter)

In an ideal network all packets travelling the same end-to-end path would experience exactly the same OWD. In a real network the OWD is not constant and the difference between a given packet's actual OWD and the average OWD is termed 'delay variation' or jitter. Jitter can be caused by competing traffic (i.e. queuing), or by contention on processing resources in the network and generally speaking a large value of jitter is symptomatic of a heavily loaded network.

RFC 3393 defines an IP Delay Variation Metric (IPDV). This particular metric only compares the delays experienced by packets of equal size, on the grounds that delay is naturally dependent on packet size, because of serialization delay.

Delay variation is related to Packet Reordering. However, the RFC 3393 IPDV of a network can be arbitrarily low, even zero, even though that network reorders packets, because the IPDV metric only compares delays of equal-sized packets.

1.2.4 Packet Loss

Packet loss is determined as the probability of a packet being lost in transit from a source A to a destination B. A One-way Packet Loss Metric for IPPM is defined in RFC 2680. RFC 3357 contains One-way Loss Pattern Sample Metrics.

Bulk data transfers may require reliable transmission i.e. small packet loss between A and B. In this situation, if any packets are lost, they must be retransmitted, reducing performance. In addition, congestion-sensitive protocols such as standard TCP assume that packet loss is due to congestion, and respond by reducing their transmission rate accordingly.

For real-time applications such as conversational audio/video, it usually does not make sense to retransmit lost packets as the retransmitted copy would arrive too late (see Delay Variation). The result of packet loss is usually degradation in sound or image quality. Some modern audio/video codecs provide a level of robustness to loss, so that the effect of occasional lost packets is benign. However, some of the most effective image compression methods are very sensitive to loss, in particular those that use "anchor frames", and represent the intermediate frames by compressed differences to these anchor frames. When such an anchor frame is lost, many other frames won't be able to be reconstructed.

Congestion and errors are the two main reasons for packet loss.

1.2.4.1 Congestion

When the offered load exceeds the capacity of a part of the network, packets are buffered in queues. Since these buffers are also of limited capacity, congestion can lead to queue overflows, which lead to packet losses. Congestion can be caused by moderate overload condition maintained for an extended amount of time or by the sudden arrival of a very large amount of traffic (traffic burst).

1.2.4.2 Errors

Another reason for loss of packets is corruption, where parts of the packet are modified in-transit. When such corruptions happen on a link (due to noisy lines etc.), this is usually detected by a link-layer checksum at the receiving end, which then discards the packet.

1.2.5 Packet Reordering

The Internet Protocol (IP) does not guarantee that packets are delivered in the order in which they were sent. This was a deliberate design choice that distinguishes IP from protocols such as, for instance, ATM and IEEE 802.3 Ethernet.

A network which reorders packets may do so because of some kind of parallelism, either because of a choice of alternative routes (Equal Cost Multipath, ECMP), or because of internal parallelism inside switching elements such as routers. One particular kind of packet reordering concerns packets of different sizes. A larger packet takes longer to transfer over a serial link or a limited-width backplane inside a router, so larger packets may be overtaken by smaller packets in transit. This is not usually a concern for high-speed bulk transfers where the segments tend to be equal-sized but may pose problems for simpler implementations of multi-media (Audio/Video) transport.

In principle, applications that use a transport protocol such as TCP or SCTP (Stream Control Transmission Protocol) don't have to worry about packet reordering, because the transport protocol is responsible for reassembling the data stream into the original ordering. However, reordering can have a severe performance

impact on some implementations of the Transmission Control Protocol. Recent TCP implementations, in particular those that support Selective Acknowledgements (SACK), may exhibit robust performance even in the face of packet reordering in the network but even in this case the transmission rate can be affected.

The measurements here can be done differently, depending on the measurement purpose:

- Measuring reordering for a particular application can be done by capturing the application traffic (e.g. using ethereal (???) tool), injecting the same traffic pattern via traffic generator and calculating the reordering.
- Measuring maximal reordering introduced by the network can be done by injecting relatively small amounts of traffic, shaped as a short burst of long packets immediately followed by a short burst of short packets, within the line rate. After capture and calculation on the other end of the path, the results will reflect the worst possible packet reordering situation which may occur on a particular path.

1.2.6 Maximum Transmission Unit (MTU)

The term MTU commonly refers to the 'protocol MTU' of an IP link and describes the maximum size of an IP packet that can be transferred over the link without fragmentation. Common MTU sizes are:

- 1500 bytes (Ethernet, 802.11 WLAN)
- 4470 bytes (FDDI, common default for POS and serial links)
- 9000 bytes (Internet2 and GÉANT convention, limit of some Gigabit Ethernet adapters)
- 9180 bytes (ATM, SMDS)

Note that a maximum size frame of bits travelling on an Ethernet link will actually be larger than 1500 bytes because the IP packet (1500 bytes) is wrapped in additional header and trailer bytes (which are part of the Ethernet specification). Because Ethernet works at Layer 2 of the standard OSI networking model an Ethernet frame is one type of Layer 2 Protocol Data Unit (PDU) (similarly IP operates at Layer 3 and an IP packet is one type of Layer 3 PDU), and the maximum sized Layer 2 PDU (Protocol Data Unit) that a given interface can support is termed the 'media MTU'. Clearly 'media MTU' must always be equal to or more than 'protocol MTU' plus the layer 2 (e.g. Ethernet) header and trailer bytes - otherwise after encapsulating the IP packet the Layer 2 PDU will exceed the media MTU and the PDU will be discarded.

1.2.6.1 Path MTU

The Path MTU is the Maximum Transfer Unit supported by a network path. It is the minimum of the MTUs of the links (segments) that make up the path. Sending regular packet sizes taking the size of the Path MTU into account reduces the risk of packet re-ordering. Larger Path MTUs generally allow for more efficient data transfers, because source and destination hosts, as well as the routing and switching devices along the network path have to process fewer packets. However, modern routers are typically designed to sustain very

high packet loads (so that they can resist denial-of-service attacks) so the packet processing rates caused by high-speed transfers are not normally an issue for today's high-speed networks. In addition, modern high-speed network adapters have mechanisms such as LSO (Large Segment Offload) and Interrupt Coalescence that mean increasing MTU sizes may no longer have as visible an impact on performance.

The prevalent Path MTU on the Internet is now 1500 bytes, the Ethernet MTU. There are some initiatives to support larger MTUs (Jumbo MTU) in networks, in particular on research networks, but their usability is hampered by last-mile issues, underlying vendor support and lack of robustness of RFC 1191 Path MTU Discovery. An IETF Working Group is currently defining a new mechanism for Path MTU Discovery which should solve these issues.

1.2.7 Bandwidth Delay Product (BDP)

The Bandwidth Delay Product (BDP) of an end-to-end path is the product of the bottleneck bandwidth and the delay of the path. Its dimension is "information", because bandwidth here expresses information per time, and delay is expressed as a time. Typically, one uses bytes as a unit, and it is often useful to think of BDP as the "memory capacity" of a path, i.e. the amount of data that fits entirely into the path between two end-systems. This relates to throughput, which is the rate at which data is sent and received.

Network paths with a large BDP are called Long Fat Networks or LFNs. In the research network environment, many end-to-end projects will transit such networks. BDP is an important parameter for the performance of window-based protocols such as TCP.

1.3 Translating Performance Metrics

One of the difficulties in identifying and solving end-to-end performance issues and ensuring Quality of Service is the difficulty in communicating metrics between all of those involved in diagnosis and operation of the application, from the end user, across the campus network, the national research networks and the Géant2 backbone. This table outlines some parallels in performance expectations of end users and network operators.

END USER METRIC	NETWORK METRIC
Responsiveness	One-Way Delay (OWD) Round-Trip Time (RTT) Delay Variation (Jitter)
Capacity and Throughput	Maximum Transfer Unit (MTU) Bandwidth Delay Product (BDP)
Reliability	Delay Variation (Jitter) Packet Loss

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Figure 1.1: End User and Network Metrics

2 First steps at investigating performance problems

2.1 Problem isolation strategies

End-user experience is the most important factor in problem isolation as the purpose of the troubleshooting is to resolve performance issues or to understand why expected performance is not achieved. Therefore the end user has a crucial role in defining, understanding and communicating problems experienced. This is all the more important in end-to-end performance situations due to the number of administrative domains involved.

2.1.1 Defining the problem

An important aspect of defining the problem is to know what the expected performance should be. If the application has been established and is suddenly experiencing performance problems, known good performance benchmarks should be available. If it is a new application and performance benchmarks have not been established, it is a good idea to do some basic benchmarking of network performance on the end-to-end path. The very simplest end-to-end test that can be done would be to use an application to transfer data memory-to-memory between hosts, bypassing any performance issues that disk IO, encryption or application overhead might introduce. This test is used in a Section 7 case study to identify performance bottlenecks.

It is best if these performance experiences are expressed in quantitative network metrics. Section 1.3 provides some information on mapping end-user experiences to network metrics. An initial performance observation that files are transferring 'slowly' could be expressed as a more formal problem statement of a data transfer rate in terms of Mb/s, compared against an expected transfer rate, expressed in the same way. Section 2.2 provides details of some tools that can help track these metrics.

2.1.2 Gathering facts

The next stage in investigating network problems is to gather facts. On an end-to-end path, it is useful to draw out the path to be taken and identify all the components involved in transporting the data (this will be a collaborative effort since it is unlikely that any one person or even organisation will have the full, detailed picture of the end to end path). It is important for end-users (or rather, end-system administrators, which may be the same thing) to be as detailed as possible as the 'network' aspect of the problem does not start at the Ethernet port on the end hosts. It includes applications, operating system protocol stacks and Network Interface Cards too.. Once all the elements have been identified and collated, appropriate measurements for the individual components on the path can be identified and gathered. It may be useful at this stage to introduce some more comparative metrics from other systems with similar paths.

2.1.3 Considering possibilities

The aim of the activity to date has been to localise the problem and pinpoint where action needs to be taken. By comparing expected performance against actual performance and identifying metrics along the path where possible, it ought to be clear to identify the points where performance is suffering. If information is not available for all components, it should still be possible to reduce the problem domain down to likely problem areas. The next step is to consider reasons for poor performance in those areas. Useful resources for this include vendor support, OS mailing lists, FAQs, and documentation and also the hints, tips and explanations provided in this document and by the wider community contributing to the Géant2 PERT Knowledge Base.

2.1.4 Reporting the problem

It is possible that initial troubleshooting may not identify the root cause of poor performance. In that case, end users may have to report the problem to local technical teams such as campus administrators, who in turn may report the issue on to NRENs and perhaps ultimately to the Géant2 PERT service which specialises in troubleshooting end-to-end performance problems. A general guideline is that too much information is never a bad thing, as long as it is clearly identified and preferably includes a time and date. However, in order to make this escalation path as smooth as possible, the following details are highlighted as particularly important in any problem report:

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

2.1.4.1 Problem Description

The current system behaviour that is unsatisfactory needs to be described in terms of the network metrics detailed in Section 1.3. If these metrics have not been taken, tools detailed in Section 2.2 can be used to benchmark performance and provide measurements. If possible, these metrics should be expressed in terms of expected or previously experienced good performance. If previous performance was acceptable, the time at when the performance degraded is very helpful. Any metrics taken, such as traceroute, tcpdump or the output from any other measurement tools used should be clearly identified and forwarded with the problem report.

2.1.4.2 Network Details

Although the network may not appear to be the responsibility of the end user, certain networking information is easiest collected by end users and should be detailed at this stage. The IP address of the host under the control of the end user is particularly important. In the case of an end-to-end problem, the IP address of the remote host is also necessary and a traceroute from the local host to the remote host to show the path. If possible, a return traceroute from the remote host to the local host is desirable. If the problem is not between two specific end points this needs to be stated and a sample traceroute from the local host to a typical destination is useful.

As well as the location, any protocol information relevant to the application will be needed to effectively troubleshoot the problem. This includes the traffic type (e.g. TCP, UDP, variations of TCP), IP Protocol (e.g. IPv4, IPv6), source port, destination port and any other relevant information. If possible, a Layer Four Traceroute (see Section 2.2) using the relevant ports should be taken and attached to the problem report.

2.1.4.3 Host details

It is necessary to provide details of the hardware, operating system and specific application in use on the local system. In particular, the version of the operating system, including the particular kernel version, and application versions are important. If any alterations or enhancement to the default systems have been applied, or any extra device drivers downloaded, it is important to include this information too. If it is possible to provide the same information about the remote host, ideally this should also be included.

2.2 Measurement tools

A range of test, measurement and troubleshooting tools exist for investigating factors contributing to performance issues which have been outlined in Section 1. A wide range of tools are also listed at <http://www.caida.org/tools/taxonomy/index.xml>. The following tools or tool-types have been recommended by members of the Géant2 PERT service and contributors to the Géant2 Service Activity PACE (Performance and Allocated Capacity for End-users) and identified as useful for end-users. Some tools provide multiple functions

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

and are listed under their most common use. Some tools are also more likely to be used by campus or NREN administrators.

2.2.1 Latency/Delay Measurement Tools

2.2.1.1 Ping

Ping is the simplest of all active measurement tools. It uses ICMP echo request and ICMP echo, and shows the RTT (Round Trip Time) between the host machine and the target. It is quite common these days for ICMP traffic to be blocked, so pings timing out does not necessarily mean a host is unavailable and you may not be able to check results for the whole path. Ping can be used to actively measure packet loss by sending a set of packets from a source to a destination and comparing the number of received packets against the number of packets sent. It can also be used to measure packet reordering by sending a numbered sequence of packets, which can then be compared to the received sequence numbers sequence using one packet reordering metrics.

Here is a sample of an IPv4 ping from a Linux system:

```
aharding@twilight:~$ ping www.geant2.net
PING newweb.dante.net (62.40.101.34) from 193.1.228.6 : 56(84) bytes of data.
64 bytes from www.dante.net (62.40.101.34): icmp_seq=1 ttl=58 time=14.1 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=2 ttl=58 time=14.0 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=3 ttl=58 time=14.0 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=4 ttl=58 time=14.1 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=5 ttl=58 time=14.1 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=6 ttl=58 time=14.0 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=7 ttl=58 time=14.2 ms
64 bytes from www.dante.net (62.40.101.34): icmp_seq=8 ttl=58 time=14.0 ms

--- newweb.dante.net ping statistics ---
8 packets transmitted, 8 received, 0% loss, time 7009ms
rtt min/avg/max/mdev = 14.057/14.113/14.261/0.146 ms
```

Figure 2.1: Example IPv4 ping output

2.2.1.2 Traceroute

The well known traceroute program was written by Van Jacobson in 1988. It sends "probe" packets with TTL values incrementing from one, and uses ICMP "Time Exceeded" messages to detect "hops" on the way to the specified destination. It also records "response" times for each hop, and displays losses and other types of failures in a compact way. It is important to note that nodes along the path may deprioritise this traffic compared to regular network traffic as a matter of policy or node engineering.

Traceroute is used to determine the route a packet takes through the Internet to reach its destination; i.e. the number of "hops" it takes. UDP packets are sent as probes to a high ephemeral port (usually in the range 33434--33525) with the Time-To-Live (TTL) field in the IP header increasing by one until the end host is reached. The originating host listens for ICMP Time Exceeded responses from each of the routers/hosts en-route. It knows that the packet's destination has been reached when it receives an ICMP Port Unreachable message; we expect a port unreachable message as no service should be listening for connections in this port range. The output of the traceroute program shows each host that the packet passes through on the way to its destination and the RTT to each gateway en-route. Occasionally, the maximum number of hops (specified by the TTL field, which defaults to 64 hops in *NIX implementations) is exceeded before the port unreachable is received. When this happens an "!" will be printed beside the RTT in the output. Note that *NIX implementations of traceroute send UDP probe packets whilst MS Windows traceroute sends ICMP echo probes. *NIX implementations of traceroute can be specified to use ICMP Echo messages instead of the default UDP probes, by using the "-I" flag. Note that either or both ICMP and UDP may be blocked by firewalls, so this must be taken into account when troubleshooting.

Here is a sample of a traceroute from a Linux system:

```
aharding@twilight:~$ /usr/sbin/traceroute pace.geant2.net
traceroute to cemp1.switch.ch (130.59.35.130), 30 hops max, 40 byte packets
 1 hsrp-vlan10.bh.access.hea.net (193.1.228.1)  0 ms  0 ms  0 ms
 2 mantova-po2.bh.access.hea.net (193.1.196.217)  0 ms  0 ms  0 ms
 3 hyperion-gige3-3-0.bh.core.hea.net (193.1.196.121)  0 ms  0 ms  0 ms
 4 deimos-gige5-2.cwt.core.hea.net (193.1.195.86)  1 ms  1 ms  1 ms
 5 heanet.iel.ie.geant.net (62.40.103.229)  1 ms  1 ms  1 ms
 6 ie.uk1.uk.geant.net (62.40.96.138)  14 ms  14 ms  113 ms
 7 uk.fr1.fr.geant.net (62.40.96.89)  21 ms  21 ms  21 ms
 8 fr.ch1.ch.geant.net (62.40.96.29)  84 ms  29 ms  29 ms
 9 swiCE2-P6-1.switch.ch (62.40.103.18)  30 ms  29 ms  30 ms
10 cemp1-eth1.switch.ch (130.59.35.130)  29 ms  29 ms  29 ms
```

Figure 2.2: Example IPv4 traceroute output

Since its inception, traceroute has been widely used for network diagnostics as well as for research in the widest sense, and there are now many variants of the original program.

2.2.1.3 MTR (Matt's Traceroute)

Mtr combines the functionality of the traceroute and ping programs in a single network diagnostic tool. More information is available from <http://www.bitwizard.nl/mtr/>. It is available as a package for several Linux distributions and for FreeBSD. Here is an example of the graphical output:

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Hostname 2,00

Hostname	Loss	Rcv	Snt	Last	Best	Avg	Worst	StDev
129.renater.fr	0,0%	9	9	10	1	6	32	10,32
gw1-renater.renater.fr	0,0%	9	9	1	0	2	5	1,72
nri-a-g13-0-50.cssi.renater.f	0,0%	9	9	1	1	1	2	0,39
193.51.185.1	0,0%	9	9	1	1	2	5	1,48
PO11-0.pascr1.Paris.opentr	0,0%	9	9	1	1	2	7	2,17
level3-1.GW.opentransit.net	0,0%	9	9	1	1	7	21	8,22
ae-0-17.mp1.Paris1.Level3.r	0,0%	9	9	2	1	11	64	20,45
so-1-0-0.bbr2.London2.Lev	0,0%	9	9	12	10	14	25	5,40
as-0-0.bbr1.NewYork1.Level	0,0%	9	9	101	69	79	101	8,94
as-0-0.bbr1.SanJose1.Level	0,0%	8	8	162	155	171	195	16,40
ge-7-0.ipcolo1.SanJose1.Le	0,0%	8	8	185	154	172	188	15,03
p1-0.cisco.bbnplanet.net	0,0%	8	8	163	155	165	180	8,76
sjck-dmzbb-gw1.cisco.com	0,0%	8	8	161	158	173	199	18,59
sjck-dmzdc-gw2.cisco.com	0,0%	8	8	163	158	175	210	17,15
www.cisco.com	12,5%	7	8	186	145	169	201	18,46

Figure 2.3: Example MTR output

2.2.1.4 Ping Plotter

Ping Plotter combines the traceroute, ping and whois utilities to collect detailed data on a path. It runs on the Windows platform. More information is available from <http://www.pingplotter.com/>. Here is an example of the output:

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

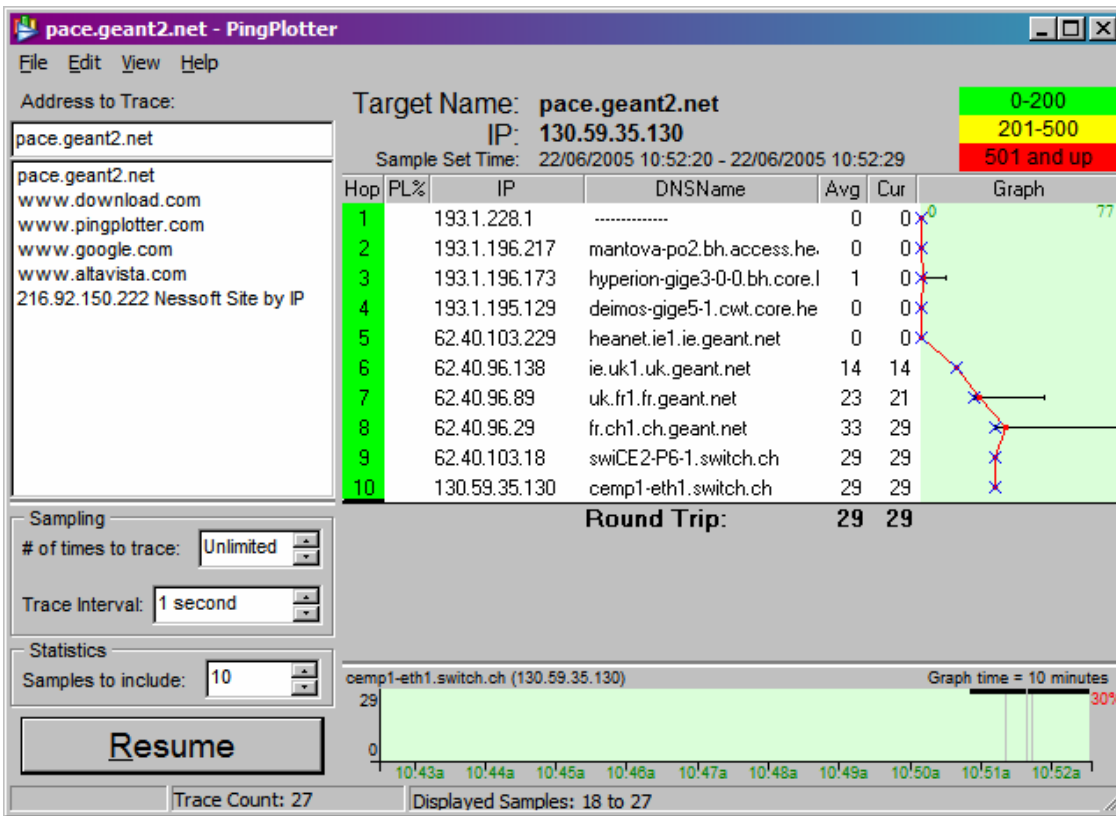


Figure 2.4: Example PingPlotter output

2.2.1.5 LFT (Layer Four Traceroute)

LFT is a sort of 'traceroute' but uses TCP port 80 to pass through packet-filter based firewalls. It can be tuned to use other ports for alternative tests and is available as a package in many Linux distributions. More information is available from <http://oppleman.com/lft/>

Here is an example of the output:

```
142:/home/andreu# lft -d 80 -m 1 -M 3 -a 5 -c 20 -t 1000 -H 30 -s 53 www.cisco.com
Tracing _____
TTL LFT trace to www.cisco.com (198.133.219.25):80/tcp
1 129.renater.fr (193.49.159.129) 0.5ms
2 gw1-renater.renater.fr (193.49.159.249) 0.4ms
3 nri-a-gl3-0-50.cssi.renater.fr (193.51.182.6) 1.0ms
4 193.51.185.1 0.6ms
5 PO11-0.pascri.Paris.opentransit.net (193.251.241.97) 7.0ms
6 level3-1.GW.opentransit.net (193.251.240.214) 0.8ms
7 ae-0-17.mp1.Paris1.Level3.net (212.73.240.97) 1.1ms
8 so-1-0-0.bbr2.London2.Level3.net (212.187.128.42) 10.6ms
9 as-0-0.bbr1.NewYork1.Level3.net (4.68.128.106) 72.1ms
10 as-0-0.bbr1.SanJose1.Level3.net (64.159.1.133) 158.7ms
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

```

11 ge-7-0.ipcolo1.SanJose1.Level3.net (4.68.123.9) 159.2ms
12 pl-0.cisco.bbnplanet.net (4.0.26.14) 159.4ms
13 sjck-dmzbb-gw1.cisco.com (128.107.239.9) 159.0ms
14 sjck-dmzdc-gw2.cisco.com (128.107.224.77) 159.1ms
15 [target] www.cisco.com (198.133.219.25):80 159.2ms
    
```

Figure 2.5: Example LFT output

2.2.1.6 Smokeping

Smokeping is a latency measurement tool that stores and displays latency, latency distribution and packet loss information over time in a graphical format. It is available for Unix/Linux systems. Although it is more likely to be deployed on the campus or WAN, it may be useful to run on end hosts or end systems in an equivalent network position. More information is available at <http://people.ee.ethz.ch/~oetiker/webtools/smokeping/>. Some examples of Smokeping graphs are below.

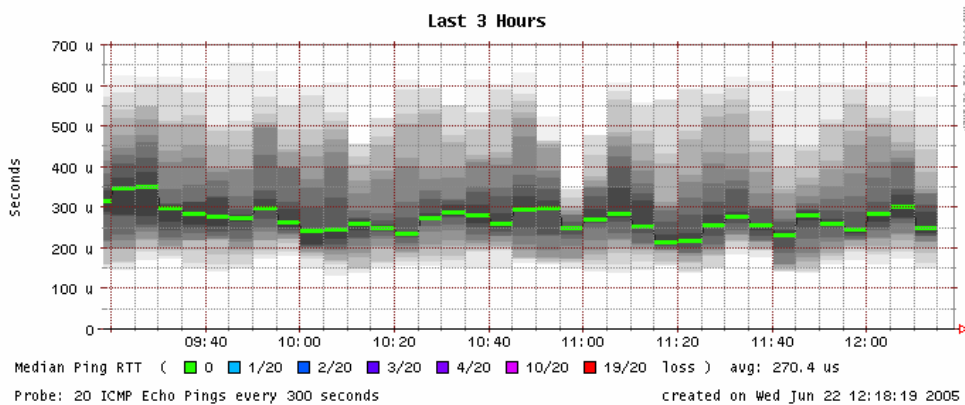


Figure 2.6: Example Smokeping output 1

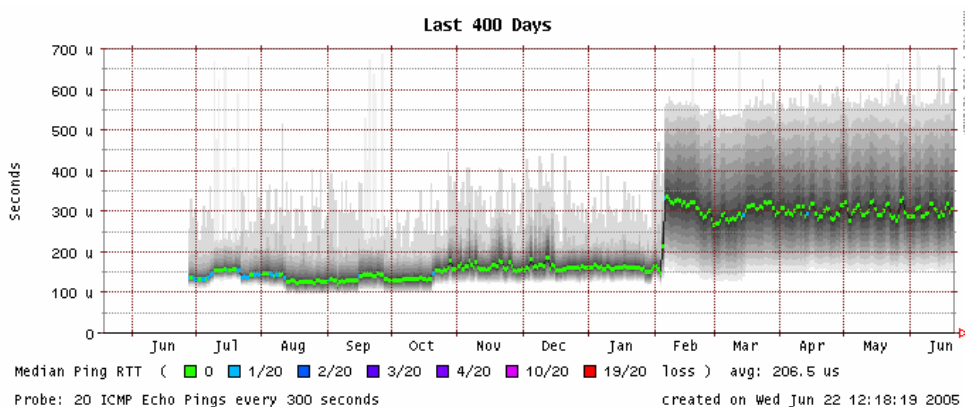


Figure 2.7: Example Smokeping output 2

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

2.2.1.7 Pchar

Pchar characterizes the bandwidth, latency and loss on network links. It is available as a package on Debian GNU/Linux from <http://www.kitchenlab.org/www/bmah/Software/pchar/>. An example output is below.

```
pchar to 193.51.180.221 (193.51.180.221) using UDP/IPv4
Using raw socket input
Packet size increments from 32 to 1500 by 32
46 test(s) per repetition
32 repetition(s) per hop
0: 193.51.183.185 (netflow-nri-a.cssi.renater.fr)
  Partial loss:      0 / 1472 (0%)
  Partial char:     rtt = 0.124246 ms, (b = 0.000206 ms/B), r2 = 0.997632
                   stddev rtt = 0.001224, stddev b = 0.000002
  Partial queueing: avg = 0.000158 ms (765 bytes)
  Hop char:         rtt = 0.124246 ms, bw = 38783.892367 Kbps
  Hop queueing:     avg = 0.000158 ms (765 bytes)
1: 193.51.183.186 (nri-a-g13-1-50.cssi.renater.fr)
  Partial loss:      0 / 1472 (0%)
  Partial char:     rtt = 1.087330 ms, (b = 0.000423 ms/B), r2 = 0.991169
                   stddev rtt = 0.004864, stddev b = 0.000006
  Partial queueing: avg = 0.005093 ms (23535 bytes)
  Hop char:         rtt = 0.963084 ms, bw = 36913.554996 Kbps
  Hop queueing:     avg = 0.004935 ms (22770 bytes)
2: 193.51.179.122 (nri-n3-a2-0-110.cssi.renater.fr)
  Partial loss:      5 / 1472 (0%)
  Partial char:     rtt = 697.145142 ms, (b = 0.032136 ms/B), r2 = 0.999991
                   stddev rtt = 0.011554, stddev b = 0.000014
  Partial queueing: avg = 0.009681 ms (23679 bytes)
  Hop char:         rtt = 696.057813 ms, bw = 252.261443 Kbps
  Hop queueing:     avg = 0.004589 ms (144 bytes)
3: 193.51.180.221 (caledonie-S1-0.cssi.renater.fr)
  Path length:      3 hops
  Path char:        rtt = 697.145142 ms r2 = 0.999991
  Path bottleneck: 252.261443 Kbps
  Path pipe:        21982 bytes
  Path queueing:    average = 0.009681 ms (23679 bytes)
  Start time:       Mon Jun  6 11:38:54 2005
  End time:         Mon Jun  6 12:15:28 2005
```

Figure 2.8: Example pchar output

Pathchar, the base for the pchar tool performs very well for slow, over-provisioned (speed < 10Mbit/s, network load < 10-20%) networks but has been observed to under-report bandwidth available by as much as 40% on faster networks and produce inconsistent results on networks with a high load.

2.2.1.8 *Iperf*

Iperf is a tool that measures maximum TCP bandwidth, allowing the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss. More information is available at <http://dast.nlanr.net/Projects/Iperf/>.

2.2.1.9 *BWCTL*

BWCTL (Bandwidth Control) is a command line client application and a scheduling and policy daemon that wraps Iperf. More information is available at <http://e2epi.internet2.edu/bwctl/>. An example is available at <http://e2epi.internet2.edu/pipes/pmp/pmp-switch.htm>.

2.2.1.10 *ip route show cache*

The Linux OS is able to apply specific conditions to specific routes, either on the fly in response to what it learns from TCP (such parameters include estimated rtt, cwnd and re-ordering) or manually. The learned info is stored in the route cache and thus can be shown with the 'ip route show cache' command. Note, this learning behaviour can actually limit TCP performance - if the last transfer was poor then the starting TCP parameters will be pessimistic. For this reason some tools, e.g. bwctl, always flush the route cache before starting a test.

2.2.1.11 *One-way delay measurement nodes.*

Measuring one-way-delay typically requires dedicated measurement equipment and is normally done by NRENs or campuses. These specialist devices measure one-way-delay by sending time stamped packets from A, and recording the reception times at B. The difficulty is that A and B need clocks that are synchronized to each other using techniques such as Global Positioning System (GPS)-derived time signals or the Network Time Protocol (NTP). Some devices of this kind include IPPM boxes (<http://www-win.rze.uni-erlangen.de/ippm/>), RIPE TTM boxes (<http://www.ttm.ripe.net/>) and QoS Metrics boxes (http://pasillo.renater.fr/metrologie/get_qosmetrics_results.php)

2.2.2 Bandwidth Utilisation Measurement Tools

Network elements such as routers and sometimes end hosts contain counters for events such as link and resource utilisation, checksum errors or queue drops. This information can be retrieved through protocols such as Simple Network Management Protocol (SNMP) and passed to a range of applications which store and display the information. Campus and NREN administrators frequently have this type of information available, though end-users may wish to set up some of these utilities on the end hosts.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

2.2.2.1 MRTG

The Multi Router Traffic Grapher (MRTG) is a tool designed to monitor the traffic load on network-links. In practice, it can be used to graph any trend for which continuous data is available either via SNMP or another external program. The campus network is more likely to need to record traffic levels but other host-specific resources such as file descriptor use etc. could be graphed using MRTG. More information is available from <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/>

An example graph based on 30 minute traffic averages over a week is below.

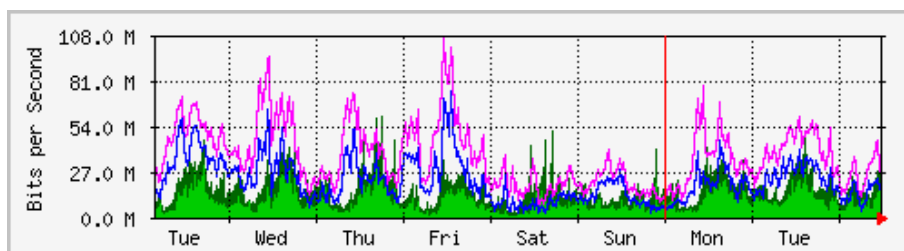


Figure 2.9: Example MRTG output

2.2.2.2 RRDTOol

RRDTOol is a data logging and graphing application. It is based on a Round Robin Database which stores and displays time-series data. It can be used via simple shell scripts or as a perl module and is frequently used as a back-end application for other measurement tools. Like MRTG, it is flexible in what it can measure and can be used by end users to monitor trends on the end systems or by campus administrators for network information. More information is available at <http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/>

2.2.2.3 Cricket

Cricket is a flexible tool developed to visualise and understand network traffic. It uses RRDTOol for data storage and can monitor any host or network based time-series data such as CPU usage, collisions and queue-drops. More information is available at <http://cricket.sourceforge.net/>. Here is a simple example graph based on daily usage:

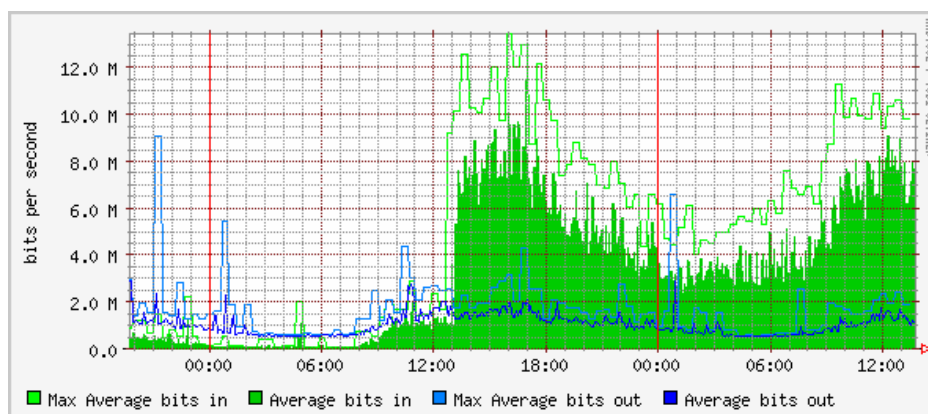


Figure 2.10: Example Cricket output

2.2.3 Path Probe Tools

A large and growing number of path-measurement tools are derived from the well-known traceroute tool. These tools all attempt to find the route a packet will take from the source (typically where the tool is running) to a given destination, and to find out some hop-wise performance parameters along the way

2.2.3.1 *tracpath*

Tracpath and tracpath6 trace the path to a network host, discovering the MTU along this path. Note that, as explained at <http://www.linuxmanpages.com/man8/tracpath.8.php>, if the MTU changes along the path, then the route can be incorrectly declared as asymmetric. Here is an example of the output:

```
142:/home/andreu# tracpath www.cisco.com
1: 142.renater.fr (193.49.159.142) 0.707ms pmtu 1500
1: 129.renater.fr (193.49.159.129) 0.480ms
2: gw1.renater.renater.fr (193.49.159.249) 0.480ms
3: nri-a-gl3-0-50.cssi.renater.fr (193.51.182.6) 0.767ms
4: 193.51.185.1 (193.51.185.1) 0.763ms
5: PO11-0.pascrl.Paris.opentransit.net (193.251.241.97) 0.643ms
6: level3-1.GW.opentransit.net (193.251.240.214) 0.930ms
7: ae-0-17.mp1.Paris1.Level3.net (212.73.240.97) asymm 8 1.162ms
8: so-1-0-0.bbr2.London2.Level3.net (212.187.128.42) asymm 10 16.205ms
9: as-0-0.bbr1.NewYork1.Level3.net (4.68.128.106) 100.452ms
10: ae-0-0.bbr2.SanJose1.Level3.net (64.159.1.130) 180.339ms
11: ge-11-2.ipcolol.SanJose1.Level3.net (4.68.123.169) asymm 10 158.088ms
12: pl-0.cisco.bbnplanet.net (4.0.26.14) asymm 11 166.252ms
13: sjck-dmzbb-gw1.cisco.com (128.107.239.9) asymm 12 159.842ms
14: sjck-dmzdc-gw2.cisco.com (128.107.224.77) asymm 13 158.445ms
15: no reply
```

Figure 2.11: Example tracepath output

2.2.3.2 traceproto

Traceproto is another traceroute variant which allows different protocols and ports to be used. It currently supports tcp, udp, and icmp traces. It comes with a wrapper script called HopWatcher, which can be used to quickly detect when a path has changed. More information is available at <http://traceproto.sourceforge.net/>. Here is an example of the output:

```
142:/home/andreu# traceproto www.cisco.com
traceproto: trace to www.cisco.com (198.133.219.25), port 80
ttl 1: ICMP Time Exceeded from 129.renater.fr (193.49.159.129)
      6.7040 ms      0.28100 ms      0.28600 ms
ttl 2: ICMP Time Exceeded from gw1-renater.renater.fr (193.49.159.249)
      0.16900 ms      6.0140 ms      0.25500 ms
ttl 3: ICMP Time Exceeded from nri-a-gl3-0-50.cssi.renater.fr (193.51.182.6)
      6.8280 ms      0.58200 ms      0.52100 ms
ttl 4: ICMP Time Exceeded from 193.51.185.1 (193.51.185.1)
      6.6400 ms      7.4230 ms      6.7690 ms
ttl 5: ICMP Time Exceeded from PO11-0.pascrl.Paris.opentransit.net
      (193.251.241.97)
      0.58100 ms      0.64100 ms      0.54700 ms
ttl 6: ICMP Time Exceeded from level3-1.GW.opentransit.net (193.251.240.214)
      6.9390 ms      0.62200 ms      6.8990 ms
ttl 7: ICMP Time Exceeded from ae-0-17.mp1.Paris1.Level3.net (212.73.240.97)
      7.0790 ms      7.0250 ms      0.79400 ms
ttl 8: ICMP Time Exceeded from so-1-0-0.bbr2.London2.Level3.net (212.187.128.42)
      10.362 ms      10.100 ms      16.384 ms
ttl 9: ICMP Time Exceeded from as-0-0.bbr1.NewYork1.Level3.net (4.68.128.106)
      109.93 ms      78.367 ms      80.352 ms
ttl 10: ICMP Time Exceeded from as-0-0.bbr1.SanJose1.Level3.net (64.159.1.133)
      156.61 ms      179.35 ms
      ICMP Time Exceeded from ae-0-0.bbr2.SanJose1.Level3.net (64.159.1.130)
      148.04 ms
ttl 11: ICMP Time Exceeded from ge-7-0.ipcolol1.SanJose1.Level3.net (4.68.123.9)
      153.59 ms
      ICMP Time Exceeded from ge-11-0.ipcolol1.SanJose1.Level3.net (4.68.123.41)
      142.50 ms
      ICMP Time Exceeded from ge-7-1.ipcolol1.SanJose1.Level3.net (4.68.123.73)
      133.66 ms
ttl 12: ICMP Time Exceeded from pl-0.cisco.bbnplanet.net (4.0.26.14)
      150.13 ms      191.24 ms      156.89 ms
ttl 13: ICMP Time Exceeded from sjck-dmzbb-gw1.cisco.com (128.107.239.9)
      141.47 ms      147.98 ms      158.12 ms
ttl 14: ICMP Time Exceeded from sjck-dmzdc-gw2.cisco.com (128.107.224.77)
      188.85 ms      148.17 ms      152.99 ms
ttl 15: no response      no response

hop : min / ave / max : # packets : # lost
-----
  1 : 0.28100 / 2.4237 / 6.7040 : 3 packets : 0 lost
  2 : 0.16900 / 2.1460 / 6.0140 : 3 packets : 0 lost
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

```
 3 : 0.52100 / 2.6437 / 6.8280 : 3 packets : 0 lost
 4 : 6.6400 / 6.9440 / 7.4230 : 3 packets : 0 lost
 5 : 0.54700 / 0.58967 / 0.64100 : 3 packets : 0 lost
 6 : 0.62200 / 4.8200 / 6.9390 : 3 packets : 0 lost
 7 : 0.79400 / 4.9660 / 7.0790 : 3 packets : 0 lost
 8 : 10.100 / 12.282 / 16.384 : 3 packets : 0 lost
 9 : 78.367 / 89.550 / 109.93 : 3 packets : 0 lost
10 : 148.04 / 161.33 / 179.35 : 3 packets : 0 lost
11 : 133.66 / 143.25 / 153.59 : 3 packets : 0 lost
12 : 150.13 / 166.09 / 191.24 : 3 packets : 0 lost
13 : 141.47 / 149.19 / 158.12 : 3 packets : 0 lost
14 : 148.17 / 163.34 / 188.85 : 3 packets : 0 lost
15 : 0.0000 / 0.0000 / 0.0000 : 0 packets : 2 lost
-----Total-----
total 0.0000 / 60.540 / 191.24 : 42 packets : 2 lost
```

Figure 2.12: Example traceproto output

2.2.3.3 Path MTU Discovery

Traceroute can be used to discover Path MTU sizes. However, the success of this is extremely variable as can be seen from this example:

```
aharding@twilight:~$ /usr/sbin/traceroute -M pace.geant2.net
traceroute to cempl.switch.ch (130.59.35.130), 30 hops max, 32000 byte packets
 1  MTU=17914 MTU=8166 MTU=4352 MTU=2002 MTU=1492 * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
```

Figure 2.13: Example traceroute MTU Discovery

An IETF Working Group (pmtud) is currently defining a new mechanism for Path MTU Discovery. Implementations are available for Linux 2.6 (<http://www.psc.edu/~jheffner/projects/mtup/>) and NetBSD (<http://www.patheticgeek.net/~kml/mmtu/>).

2.2.4 Network Simulation and Benchmarking Tools

2.2.4.1 Network Emulation

NISTnet and netem are network emulation software packages that can be run on Linux machines. In particular, they can be used to introduce delays to packets, thereby simulating a long(er) distance network. NISTnet is

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

known to work on 2.4.x Linux kernels. For the best support for recent Gigabit Ethernet cards, an alternative recommendation is Linux kernel 2.6.10 or .11 and netem (Network emulator).

You will need a recent iproute2 package that supports netem. More information is available at <http://developer.osdl.org/shemminger/netem/index.html>.

2.2.4.2 *Netperf*

Netperf is a benchmark that can be used to measure the performance of many different types of networking. It provides tests for both unidirectional throughput, and end-to-end latency. It is a client/server application. The Netperf organisation also maintains a public database where you can search for previous performance benchmarks. More information is available at <http://www.netperf.org/netperf/NetperfPage.html>.

2.2.4.3 *RUDE/CRUDE*

RUDE stands for Real-time UDP Data Emitter and CRUDE for Collector for RUDE. Together, they form a package to generate and measure UDP traffic between two points. The traffic pattern sent out by RUDE can be defined by the user. More information is available from <http://rude.sourceforge.net/>.

2.2.4.4 *TTCP*

TTCP (Test TCP) is a command-line utility for benchmarking UDP and TCP performance between two systems. More information is available at <http://www.pcausa.com/Utilities/pcattcp.htm>.

2.2.4.5 *Tweak Tools*

An online application at <http://www.dslreports.com/tweaks> is able to run a simple end-user test, checking such parameters as TCP options, receive window size, and data transfer rates. It is all done through the user's web-browser making it a simple test to perform. However, the tests are limited to that site.

2.2.4.6 *NDT*

NDT (Network Diagnostic Tester) is a client/server application that can be used to check network configuration and performance to an end host. The client tool is available either as a command line application or a Java applet. The server side includes a basic web server and the Web100 toolkit used to analyse the test results and return answers to the client. If an end user does not have access to an appropriate host on which to run the

server application, they should contact their campus administrators. More information on NDT is available from <http://e2epi.internet2.edu/ndt/>.

2.2.5 Traffic Analysis Tools

A range of tools exists for in-depth analysis of traffic. These tools can help detect protocol problems by detailed analysis of packet headers.

2.2.5.1 *TCPDump*

TCPDump is a free command-line utility that prints out the headers of packets on a network interface on a host. It can be used to dump all traffic for analysis or to match against particular types. The manual page provided with this package on Debian GNU/Linux systems provides detailed guidelines for use and interpretation.

2.2.5.2 *Ethereal*

Ethereal is a protocol analyser that analyses data collected live off the wire or provided by applications such as tcpdump. Information is displayed in a graphical or command line interface. It supports a wide range of protocols. More information is available at <http://www.ethereal.com/>. It is most likely that campus administrators would have access to this tool and training to use it.

2.2.5.3 *Jnettop*

Jnettop is a passive measurement tool that captures traffic coming across the host it is running on and displays streams sorted by the bandwidth they use. The result is a nice listing of communication on the network grouped by stream, which shows transported bytes and consumed bandwidth. More information is available from <http://jnettop.kubs.info/>.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

3 TCP Performance Primer

The Transmission Control Protocol (TCP) is the prevalent transport protocol used on the Internet today. It provides the service of a reliable byte stream, and adapts the rate of transfer to the state (of congestion) of the network and the receiver. Basic mechanisms include:

- Segments that fit into IP packets, into which the byte-stream is split by the sender,
- A checksum for each segment,
- A Window, which bounds the amount of data "in flight" between the sender and the receiver,
- Acknowledgements, by which the receiver tells the sender about segments received successfully.

Originally specified in September 1981 RFC 793, TCP was clarified, refined and extended in many documents, notably Van Jacobson's 1988 SIGCOMM article on "Congestion Avoidance and Control", later reissued as RFC 2581. It can be said that TCP's Congestion Control is what keeps the Internet working when links are overloaded.

The characteristics of Research Networks are not typical of the Internet in general. They can be characterised as Long Fat Networks (LFNs) due to a large Bandwidth-Delay Product (BDP) and arguably do not suffer from overload in the same degree. One of the issues with this type of network is that it can be challenging to achieve high throughput for individual data transfers with transport protocols such as TCP. A number of enhancements to TCP and tuning methodologies are available to help maximise use of such networks. However, although deployment of these methods is more likely to be supported on research networks, researchers should be aware that problems may be encountered and there can be practical difficulties in reaching maximum performance.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

3.1 Window-based transmission

TCP is a sliding-window protocol. The receiver tells the sender the available buffer space at the receiver (TCP header field "window"). The total window size is the minimum of sender buffer size, advertised receiver window size and congestion window size.

The sender can transmit up to this amount of data before having to wait for further buffer update from the receiver and should not have more than this amount of data in transit in the network. The sender must buffer the sent data until it has been ACKed by the receiver, so that the data can be retransmitted immediately if necessary. For each ACK the sent segment left the window and a new segment fills the window if it fits the (possibly updated) window buffer.

Due to TCP's flow control mechanism, TCP window size can limit the maximum theoretical throughput regardless of the bandwidth of the network path. Using too small a TCP window can degrade the network performance lower than expected and a too large window may have the same problems in case of error recovery.

The TCP window size is the most important parameter for achieving maximum throughput across high-performance networks. To reach the maximum transfer rate, the TCP window should be no smaller than the bandwidth-delay product.

Window size => Bandwidth (bytes/sec) x Round-trip time (sec)

Example:

```
window size: 8192 bytes  
round-trip time: 100ms  
maximum throughput: < 0.62 Mbit/sec.
```

3.2 Rate control

TCP flow control and window size adjustment is mainly based on two key mechanisms: Slow Start and Additive Increase/Multiplicative Decrease (AIMD), also known as Congestion Avoidance. (RFC 793 and RFC 2581)

3.2.1 Slow Start

To avoid that a starting TCP connection floods the network, a Slow Start mechanism was introduced in TCP. This mechanism effectively probes to find the available bandwidth.

In addition to the window advertised by the receiver, a Congestion Window (cwnd) value is used and the effective window size is the lesser of the two. The starting value of the cwnd window is set initially to the maximum segment size (MSS) of the connection (obtained during SYN handshake, discovered path MTU).

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

After each acknowledgment, the cwnd window is increased by one MSS. By this algorithm, the data rate of the sender doubles each round-trip time (RTT) interval. This increase continues until either the advertised window size is reached or congestion (packet loss) is detected on the connection. When congestion is detected, the TCP flow-control mode is changed from Slow Start to Congestion Avoidance.

3.2.2 Congestion Avoidance

Once congestion is detected (through timeout and/or duplicate ACKs), the data rate is reduced in order to let the network recover.

Slow Start uses an exponential increase in window size and thus also in data rate. Congestion Avoidance uses a linear growth function (additive increase). This is achieved by introducing - in addition to the cwnd window - a slow start threshold (ssthresh).

As long as cwnd is less than ssthresh, Slow Start applies. Once ssthresh is reached, cwnd is increased by at most one segment per RTT. The cwnd window continues to open with this linear rate until a congestion event is detected.

When congestion is detected, ssthresh is set to half the cwnd. cwnd is either set to 1 if congestion was signalled by a timeout, forcing the sender to enter Slow Start, or to ssthresh if congestion was signalled by duplicate ACKs and the Fast Recovery algorithm has terminated. In either case, once the sender enters Congestion Avoidance, its rate has been reduced to half the value at the time of congestion. This multiplicative decrease causes the cwnd to close exponentially with each detected loss event.

3.2.3 Fast Retransmit

In Fast Retransmit, the arrival of three duplicate ACKs is interpreted as packet loss, and retransmission starts before the retransmission timer expires. The missing segment will be retransmitted immediately without going through the normal retransmission queue processing. This improves performance by eliminating delays that would suspend effective data flow on the link.

3.2.4 Fast Recovery

Fast Recovery is used to react quickly to a single packet loss. In Fast recovery, the receipt of 3 duplicate ACKs, while being taken to mean a loss of a segment, does not result in a full Slow Start. This is because obviously later segments got through, and hence congestion is not stopping everything. In fast recovery, ssthresh is set to half of the current send window size, the missing segment is retransmitted (Fast Retransmit) and cwnd is set to ssthresh plus three segments. Each additional duplicate ACK indicates that one segment has left the network at the receiver and cwnd is increased by one segment to allow the transmission of another segment if allowed

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

by the new `cwnd`. When an ACK is received for new data, `cwnd` is reset to the `ssthresh`, and TCP enters congestion avoidance mode.

3.3 TCP Performance enhancements

RFC 1323 specifies a set of "TCP Extensions for High Performance", namely the Window Scaling Option, which provides for much larger windows than the original 64K, the Timestamp Option and the PAWS (Protection Against Wrapped Sequence numbers) mechanism. These extensions are supported by most contemporary TCP stacks, although they frequently must be activated explicitly or implicitly by configuring Large TCP Windows.

3.3.1 Window scaling & timestamps

In order to achieve high data rates with TCP over long fat networks, hosts receiving data transported by TCP (TCP sinks) must advertise a large TCP receive window.

The window is a 16 bit value (bytes 15 and 16 in the TCP header) and so is limited to a value of 65535 (64K). The receive window sets an upper limit on the sustained throughput achievable over a TCP connection since it represents the maximum amount of unacknowledged data (in bytes) there can be on the TCP path. Mathematically, achievable throughput can never be more than $\text{WINDOW_SIZE}/\text{RTT}$, so for a hypothetical trans-Atlantic link, with an RTT of 150ms, throughput is limited to a maximum of 3.4Mbps. With the emergence of long fat networks, the 64K limit was clearly insufficient and so RFC 1323 set out a way of scaling the advertised window, such that the 16-bit window value can represent numbers larger than 64K.

TCP window scaling option increases the maximum window size from 64KB to 1Gbyte by shifting the window field left by up to 14. The window scale option is used only during the TCP 3-way handshake (both sides send the window scale option in their SYN segments).

It is important to use TCP timestamps option with large TCP windows. With the TCP timestamps option, each segment contains a timestamp. The receiver returns that timestamp in each ACK and this allows the sender to estimate the RTT. On the other hand with the TCP timestamps option the problem of wrapped sequence number could be solved (PAWS - Protection Against Wrapped Sequences) which could occur with large windows.

There are several potential issues when TCP Windows are larger than necessary:

- When there are many active TCP connection endpoints (sockets) on a system - such as a popular Web or file server - then a large TCP window size will lead to high consumption of system (kernel) memory. This can have a number of negative consequences: The system may run out of buffer space so that no new connections can be opened, or the high occupation of kernel memory (which typically

must reside in actual RAM and cannot be paged out to disk) can starve other processes of access to fast memory (cache and RAM)

- Large windows can cause large bursts of consecutive segments/packets. When there is a bottleneck in the path, perhaps because of a slower link or because of cross-traffic, these bursts will fill up buffers in the network device (router or switch) in front of that bottleneck. The larger these bursts, the higher are the risks that this buffer overflows and causes multiple segments to be dropped. So a large window can lead to sawtooth behavior and worse link utilisation than with an optimal window size where TCP could operate at a steady rate.

Both these issues are arguments in favour of buffer auto-tuning, a promising but relatively new approach to better TCP performance in operating systems.

3.3.2 SACK

Another widely implemented performance enhancement to TCP is Selective Acknowledgements (SACK, RFC 2018). In TCP as originally specified, the acknowledgements (ACKs) sent by a receiver were always "cumulative", that is, they specified the last byte of the part of the stream that was completely received. Selective Acknowledgements are a refinement of TCP's traditional "cumulative" acknowledgements.

SACKs allow a receiver to acknowledge non-consecutive data, so that the sender can retransmit only what is missing at the receiver's end. This is particularly helpful on paths with a large bandwidth-delay product (BDP).

TCP may experience poor performance when multiple packets are lost from one window of data. With the limited information available from cumulative acknowledgments, a TCP sender can only learn about a single lost packet per round trip time. An aggressive sender could choose to retransmit packets early, but such retransmitted segments may have already been successfully received.

A Selective Acknowledgment (SACK) mechanism, combined with a selective repeat retransmission policy, can help to overcome these limitations. The receiving TCP sends back SACK packets to the sender informing the sender of data that has been received. The sender can then retransmit only the missing data segments.

Multiple packet losses from a window of data can have a catastrophic effect on TCP throughput. TCP uses a cumulative acknowledgment scheme in which received segments that are not at the left edge of the receive window are not acknowledged. This forces the sender to either wait a roundtrip time to find out about each lost packet, or to unnecessarily retransmit segments which have been correctly received. With the cumulative acknowledgment scheme, multiple dropped segments generally cause TCP to lose its ACK-based clock, reducing overall throughput. Selective Acknowledgment (SACK) is a strategy which corrects this behavior in the face of multiple dropped segments. With selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

The selective acknowledgment extension uses two TCP options. The first is an enabling option, SACK-permitted, which may be sent in a SYN segment to indicate that the SACK option can be used once the connection is established. The other is the SACK option itself, which may be sent over an established connection once permission has been given by SACK-permitted.

3.3.2.1 SACK blackholing issues

Enabling SACK globally used to be somewhat risky, because in some parts of the Internet, TCP SYN packets offering/requesting the SACK capability were filtered, causing connection attempts to fail. By now, it seems that the increased deployment of SACK has caused most of these filters to disappear but this behaviour may still be seen. If SACK blackholing is suspected, end users should contact campus administrators. If the problem is off-campus, the campus administrators will contact the NREN and the PERT as appropriate.

3.3.3 Explicit Congestion Notification

TCP traditionally has to rely on packet loss and queueing delay as the prime indicator of congestion. Both loss and delay are implicit signals of congestion. The alternative is to send explicit congestion signals.

The new Explicit Congestion Notification (ECN) mechanism consists of two components:

- Two new ECN bits in the former TOS field of the IP header:
 - The "ECN-Capable Transport" (ECT) bit must only be set for packets controlled by ECN-aware transports
 - The "Congestion Experienced" (CE) bit can be set by a router if
 - the router has detected congestion on the outgoing link
 - and the ECT bit is set.
- Transport-specific protocol extensions which communicate the ECN signal back from the receiver to the sender. For TCP, this takes the form of two new flags in the TCP header, ECN-Echo (ECE) and Congestion Window Reduced (CWR).

The basic idea is that when a transport supports ECN, it sends IP packets with ECT (ECN-Capable Transport) set. Then, when there is congestion, a router will set the CE (Congestion Experienced) bit in some of these packets. The receiver notices this, and sends a signal back to the sender (the CWR flag in the case of TCP). The sender then reduces its sending rate, as if it had detected packet loss.

(Note that the two-bit ECN field in the IP header has been redefined in the current ECN RFC (RFC3168), so that "ECT" and "CE" are no longer actual bits. But the old definition is somewhat easier to understand. If you want to know how these "conceptual" bits are encoded, please read RFC 3168.)

ECN provides two significant benefits:

- ECN-aware transports can properly adapt their rates to congestion without requiring packet loss

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

- Congestion feedback can be quicker with ECN, because detecting a dropped packet requires a timeout.

3.3.3.1 ECN blackholing issues

Attempts to use ECN can cause issues with certain devices such as firewalls or load balancers, which break connectivity when unexpected TCP flags (or, more rarely, unexpected IP TOS values) are encountered. The original ECN RFC (RFC 2481) didn't handle this gracefully, so activating ECN on hosts that implement this version caused much frustration because of "hanging" connections. RFC 3168 proposes a mechanism to deal with ECN-unfriendly networks, but that hasn't been widely implemented yet. If ECN blackholing is suspected, end users should contact their campus administrators. If the problem is off-campus, the campus administrators will contact the NREN and the PERT as appropriate.

3.3.3.2 ECN network support (or lack thereof)

ECN requires routers to use an Active Queue Management (AQM) mechanism such as Random Early Detection (RED). In addition, routers have to be able to mark eligible packets with the CE bit when the AQM mechanism notices congestion. Random Early Detection is widely implemented on routers today, although it is rarely activated in actual networks. The capability to ECN-mark packets can be added to CPU- or Network-Processor-based routing platforms relatively easily, Cisco's CPU-based routers such as the 7200/7500 routers support this with newer software, for example, but if queueing/forwarding is performed by specialized hardware (ASICs), this function has to be designed into the hardware from the start. Therefore, many of today's high-speed routers cannot easily support ECN.

3.4 High-performance TCP variations

There have been numerous ideas for improving TCP over the years. Some of those ideas have been adopted by mainstream operations (after thorough review). Recently there has been an uptake in work towards improving TCP's behaviour with Long Fat Networks. This is a reference list of some of the proposals and analysis.

3.4.1 HSTCP, H-TCP, BIC, FAST etc.

- HSTCP (HighSpeed TCP) by Sally Floyd. Information: <http://www.icir.org/floyd/hstcp.html>.
- H-TCP by Doug Leith et al. from the Hamilton Institute. Information: <http://www.hamilton.ie/net/htcp/>.
- TCP Westwood from UCLA. Information: <http://www.cs.ucla.edu/NRL/hpi/tcpw/>.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

- FAST from Caltech. Information: <http://netlab.caltech.edu/FAST/>
- BIC-TCP/CUBIC from North Carolina State University. Information: <http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/>.
- Scalable TCP by Tom Kelly. Information: <http://www-lce.eng.cam.ac.uk/~ctk21/scalable/>.
- LTCP from Texas A&M University. Information: <http://dropzone.tamu.edu/techpubs/2004/TAMU-ECE-2004-03.pdf>.
- SABUL from the University of Illinois. Information: <http://www.dataspaceweb.net/papers/sabul-hpdt-03.pdf>.
- SOBAS from Georgia Tech. Information: <http://www.cercs.gatech.edu/tech-reports/tr2004/git-cercs-04-03.pdf>

There have been several studies (a selection of which are listed below) that compare the performance of the various new TCP variants. A TCP variant is considered successful if it is able to both make good use of unused spare capacity (that is, it is “efficient”) and also it does not adversely affect co-existing TCP flows, including legacy Reno TCP flows (which is to say, it must be “friendly”).

- TCP Stack Measurements on Lightly Loaded Testbeds, Les Cottrell (SLAC), 2002-2003. Available: <http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/>
- Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks, H. Bulot, R. Les Cottrell, R. Hughes-Jones, J. Grid Comput. 1(4): 345-359 (2003).
- FAST TCP in High-Speed Networks: An Experimental Study, S. Hegde, D. Lapsley, B. Wydrowski, J. Lindheim, D. Wei, C. Jin, S. Low, and Harvey Newman, GridNets 2004. Available <http://netlab.caltech.edu/pub/papers/gridnets04.pdf>
- Protocols for long-distance networks, Guy Almes, TERENA Networking Conference 2004, PowerPoint presentation. Available: http://www.terena.nl/conferences/tnc2004/programme/presentations/show.php?pres_id=119
- Measured Comparative Performance of TCP Stacks, S. Jansen and A. McGregor, Proc. PAM 2005. Available: <http://www.pam2005.org/PDF/34310332.pdf>
- TCP Evaluation Discussion Forum, <http://www.hamilton.ie/net/eval/>

4 Application and protocol considerations

4.1 Designing tolerant applications

As part of Section 1, Performance Basics and Section 3, TCP Performance Primer, some performance issues were highlighted that can be dealt with by good application and protocol design.

The network components of RTT are the One Way Delays in both directions (which can use different paths), so read the information on One Way Delay in this document on how those can be improved. The speed of response generation can be improved through optimising the responding program. Delay variation is an issue for real-time applications such as audio/video conferencing systems. They usually employ a Jitter Buffer to eliminate the effects of delay variation.

One particular kind of packet reordering concerns packets of different sizes. A larger packet takes longer to transfer over a serial link (or a limited-width backplane inside a router), so larger packets may be "overtaken" by smaller packets that were sent subsequently. This is usually not a concern for high-speed bulk transfers - where the segments tend to be equal-sized (hopefully Path MTU-sized), but may pose problems for naive implementations of multi-media (Audio/Video) transport.

Real-time media applications such as audio/video conferencing tools often experience problems when operated on networks that reorder packets. This is somewhat remarkable in that all of these applications have Jitter Buffers to eliminate the effects of Delay Variation on the real-time media streams. Obviously, the code that manages these jitter buffers is often not written in a way to accommodate reordered packets sensibly, although this could be done with moderate effort.

4.1.1 "Chatty" Protocols

A common problem with naively designed application protocols is that they are too "chatty", i.e. they result in too many round-trip cycles where one party has to wait for a response from the other. It is an easy mistake to

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

make, because when testing such a protocol locally, these round-trips usually don't have much of an impact on overall performance. But when used over network paths with large RTTs, chattiness can dramatically impact perceived performance.

4.1.1.1 Example: SMTP (Simple Mail Transfer Protocol)

The Simple Mail Transfer Protocol (SMTP) is used to transport most e-mail messages over the Internet. In its original design (RFC 821, now superseded by RFC 2821), the protocol consisted of a strict sequence of request/response transactions, some of them very small. Taking an example from RFC 2920, a typical SMTP conversation between a client, "C" that wants to send a message, and a server "S" that receives it, would look like this:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 Innosoft.com SMTP service ready
C: HELO dbc.mtview.ca.us
S: 250 Innosoft.com
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
S: 250 sender <mrose@dbc.mtview.ca.us> OK
C: RCPT TO:<ned@innosoft.com>
S: 250 recipient <ned@innosoft.com> OK
C: RCPT TO:<dan@innosoft.com>
S: 250 recipient <dan@innosoft.com> OK
C: RCPT TO:<kvc@innosoft.com>
S: 250 recipient <kvc@innosoft.com> OK
C: DATA
S: 354 enter mail, end with line containing only "."
...
C: .
S: 250 message sent
C: QUIT
S: 221 goodbye
```

This simple conversation contains nine places where the client waits for a response from the server.

In order to improve this, the PIPELINING extension (RFC 2920) was later defined. When the server supports it - as signalled through the ESMTP extension mechanism in the response to an EHLO request - the client is allowed to send multiple requests in a row, and collect the responses later. The previous conversation becomes the following one with PIPELINING:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP service ready
C: EHLO dbc.mtview.ca.us
S: 250-innosoft.com
S: 250 PIPELINING
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
C: RCPT TO:<ned@innosoft.com>
C: RCPT TO:<dan@innosoft.com>
C: RCPT TO:<kvc@innosoft.com>
```

```
C: DATA
S: 250 sender <mrose@dbc.mtview.ca.us> OK
S: 250 recipient <ned@innosoft.com> OK
S: 250 recipient <dan@innosoft.com> OK
S: 250 recipient <kvc@innosoft.com> OK
S: 354 enter mail, end with line containing only "."
...
C: .
C: QUIT
S: 250 message sent
S: 221 goodbye
```

There are still a couple of places where the client has to wait for responses, notably during initial negotiation; but the number of these situations has been reduced to those where the response has an impact on further processing. The PIPELINING extension reduces the number of turn-arounds from nine to four. This speeds up the overall mail submission process when the RTT is high, reduces the number of packets that have to be sent (because several requests, or several responses, can be sent as a single TCP segment), and significantly decreases the risk of timeouts (and consequent loss of connection) when the connectivity between client and server is really bad.

The X Window System protocol (X11) is an example of a protocol that has been designed from the start to reduce turn-arounds.

4.1.2 Performance-friendly I/O interfaces

For applications with high input/output performance requirements (including network I/O), it is worthwhile to look at operating system support for efficient I/O routines.

4.1.2.1 *read()/write()*

As an example, here is simple pseudo-code that reads the contents of an open file in and writes them to an open socket out - this code could be part of a file server. A straightforward way of coding this uses the *read()/write()* system calls to copy the bytes through a memory buffer:

```
#define BUFSIZE 4096
long send_file (int in, int out) {
    unsigned char buffer[BUFSIZE];
    int result; long written = 0;
    while (result = read (in, buffer, BUFSIZE) > 0) {
        if (write (out, buffer, result) != result)
            return -1;
        written += result;
    }
    return (result == 0 ? written : result);
}
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Unfortunately, this common programming paradigm results in high memory traffic and inefficient use of a system's caches. Also, if a small buffer is used, the number of system operations and, in particular, of user/kernel context switches will be quite high.

4.1.2.2 *mmap()/write()*

On systems that support memory mapping of files using `mmap()`, the following is more efficient if the source is an actual file:

```
#define BUFSIZE 4096
long send_file (int in, int out) {
    unsigned char *b;
    struct stat st;
    if (fstat (in, &st) == -1) return -1;
    if ((b = mmap (0, st.st_size, PROT_READ, 0)) == -1)
        return -1;
    madvise (b, st.st_size, MADV_SEQUENTIAL);
    return write (out, b, st.st_size);
}
```

4.1.2.3 *sendfile()*

An even more efficient - and also more concise - variant is the `sendfile()` call, which directly copies the bits from the file to the network.

```
#define BUFSIZE 4096
long send_file (int in, int out) {
    off_t offset = 0;
    return sendfile (out, in, &offset, 1);
}
```

Note that an operating system could optimise this internally up to the point where data blocks are copied directly from the disk controller to the network controller without any involvement of the CPU.

For more complex situations, the `sendfilev()` interface can be used to send data from multiple files and memory buffers to construct complex protocol units with a single call.

4.2 Choosing applications

A common problem for many applications is the replication of data sets (often large) from one system to another, or to several others. This can require reliable transfer such as that provided by TCP, access control based on some sort of authentication, and encryption. This section provides an overview of some common

applications and protocols. As part of Section 2, 'First Steps at Investigating Performance Problems' and Section 3, 'TCP Performance Primer', some high-performance variants of standard applications and protocols were introduced. It is often worth investigating if optimised versions of standard software packages are available or if packages can be tuned for performance.

4.2.1 Protocols

4.2.1.1 TCP

HS-TCP, H-TCP, BIC-TCP and FAST are all TCP variations optimised for performance. More detail is available in the final references and in Section 3. It is worth checking with the campus network provider and NREN when choosing to use a TCP variant as these variants can have unexpected effects on shared links.

4.2.1.2 RTP

Real-Time Transport Protocol (RTP) is a generic transport protocol for real-time media streams such as audio or video. RTP is typically run over the User Datagram Protocol (UDP). RTP's services include timestamps and identification of media types. The User Datagram Protocol (UDP) is a very simple layer over the host-to-host protocol. It only adds 16-bit source and destination port numbers for multiplexing between different applications on the pair of hosts, and 16-bit length and checksum fields. UDP can perform badly in congested networks.

4.2.1.3 FTP

File Transfer Protocol (FTP), was one of the earliest protocols used on the ARPAnet and the Internet, and predates both TCP and IP. It supports simple file operations over a variety of operating systems and file abstractions, and has both a text and a binary mode. FTP uses separate TCP connections for control and data transfer.

4.2.1.4 HTTP

Hypertext Transfer Protocol (HTTP) is the basic protocol used by the World Wide Web. It is quite efficient for transferring files, but is typically used to transfer from a server to a client only.

4.2.1.5 SSH

Secure Shell (SSH) is a widely used protocol for remote terminal access with secure authentication and data encryption. It is also used for file transfers, using tools such as scp (Secure Copy), sftp (Secure FTP), or rsync-over-ssh.

When users use SSH to transfer large files, they often think that performance is limited by the processing power required for encryption and decryption. While this can indeed be an issue in a LAN context, the bottleneck over the full network path is most likely a window limitation. Even when TCP parameters have been tuned to allow sufficiently large TCP Windows, the most common SSH implementation (OpenSSH) has a hardwired window size at the application level.

This limitation is removed in a modification of the OpenSSH software provided by the Pittsburgh Supercomputing Centre.

4.2.1.6 BitTorrent

BitTorrent is an example of a peer-to-peer file-sharing protocol. It employs local control mechanisms to optimise the global problem of replicating a large file to many recipients, by allowing peers to share partial copies as they receive them. BitTorrent has become a focus of attention of media interest groups such as the Motion Picture Artists of America (MPAA) but is also used to distribute large software archives under "Free Software" or similar legal-redistribution regimes. More information is available at <http://www.bittorrent.com/>.

4.2.2 Applications

4.2.2.1 RCP

Remote Copy (RCP) from Berkeley, is a convenient application for transferring files between Unix systems, but lacks real security beyond address-based authentication and clear-text passwords and has mostly fallen out of use.

4.2.2.2 SCP

Secure Copy (SCP) is a file-transfer protocol using SSH. It provides various modern methods of authentication and encryption, but its current implementations shares performance limitations with SSH.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

4.2.2.3 OpenSSH

When the window-size limitation for SSH as detailed in the previous section is removed, encryption/decryption performance may become the bottleneck again. Therefore it is useful to choose a encryption/decryption cipher that performs well, while still being regarded as sufficiently secure to protect the data in question. Here is a table that displays the performance of several ciphers supported by OpenSSH in a reference setting:

cipher	throughput
3des-cbc	2.8MB/s
arcfour	24.4MB/s
aes192-cbc	13.3MB/s
aes256-cbc	11.7MB/s
aes128-ctr	12.7MB/s
aes192-ctr	11.7MB/s
aes256-ctr	11.3MB/s
blowfish-cbc	16.3MB/s
cast128-cbc	7.9MB/s
rijndael-cbc@lysator.liu.se	12.2MB/s

Figure 4.1: SSH Cipher Performance

The High Performance Enabled SSH/SCP version also supports an option to the SCP program that supports use of the "none" cipher, when confidentiality protection of the transferred data is not required.

4.2.2.4 Apache

The Apache web server is a freely available application and can be tuned for performance. Performance tuning tips for Apache HTTP Server Version 1.3 are available at <http://httpd.apache.org/docs/misc/perf-tuning.html> and for Apache 2.0 at <http://httpd.apache.org/docs-2.0/misc/perf-tuning.html>. Section 7.2 describes a case study in which Apache 2.x is tuned to cope with many simultaneous connections.

5 Performance Case Studies

5.1 Duplex Mismatch

A point-to-point Ethernet segment (typically between a switch and an end-node, or between two directly connected end-nodes) can operate in one of two duplex modes: half duplex means that only one station can send at a time, and full duplex means that both stations can send at the same time. Full-duplex mode is preferable for performance reasons if both stations support it.

Duplex mismatch describes the situation where one station on a point-to-point Ethernet link uses full-duplex mode, and the other uses half-duplex mode. A link with duplex mismatch will seem to work fine as long as there is little traffic. But when there is traffic in both directions, it will experience packet loss and severely decreased performance to the point where performance is worse than when both stations operate in half-duplex mode. Work in the Internet2 "End-to-End Performance Initiative" suggests that duplex mismatch is one of the most common causes of bad bulk throughput.

Duplex mismatch can be avoided by either using auto-negotiation or manually setting the duplex mode. In older devices, the common recommendation was to manually configure the desired duplex mode - typically full duplex by hand. Nowadays, it is generally preferable to rely on auto-negotiation of duplex mode. Recent equipment handles auto-negotiation in a reliable and interoperable way, with very few exceptions.

Most Network Interface Cards automatically do auto-negotiation but settings should be verified. On Linux systems, the `ethtool` utility displays and allows changes to card settings. This example shows a NIC set to auto-negotiate that has negotiated a full duplex connection:

```
twilight:/home/aharding# /usr/sbin/ethtool eth0
Settings for eth0:

    Supported ports: [ MII ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Half 1000baseT/Full
    Supports auto-negotiation: Yes
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

```
Speed: 1000Mb/s
Duplex: Full
Port: Twisted Pair
PHYAD: 1
Transceiver: internal
Auto-negotiation: on
Supports Wake-on: g
Wake-on: d
```

Figure 5.1: Example Duplex Check

For more complex testing, NDT uses heuristics to try to determine whether the path to a remote host suffers from duplex mismatch. More information on this tool is available in Section 2.2.

5.2 Transatlantic File Transfer Performance Troubleshooting

This example of performance troubleshooting focuses on end-host issues and is from a case handled by the pilot PERT in 2004. It shows how a problem was identified and how tuning of end-host systems, without intervention on the network path delivered significant improvements.

5.2.1 Problem Statement

Initially, a problem was observed transferring files from a host in FermiLab (US) to a host in Strasbourg. The data consisted of many 15MByte files, totalling a few hundred gigabytes and the transferring application was rsync. Although the bottleneck links on the network were 100Mbps, the achieved transfer rate was typically only 5Mbps.

5.2.2 Results Summary

Test machines in similar locations were set up and web100 tools installed. These tests showed that while memory-memory routinely achieved 90+Mbps, using nttcp, disk-disk only achieved 20Mbps. This narrowed the problem down to something on the host systems, in particular, limited system & disk i/o capability on the receiving machine.

An alternative receiving test machine was set up, giving a scenario of a long path, with fast machines on both ends. In this set of tests, data transfer via ssh over TCP was slower than accountable by cryptographic overhead. This highlighted the ssh/ssl buffer limitations on both the sender-side and receiver-side end hosts.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

5.2.3 Improving Performance

On Linux, auto-buffer-tuning could be used on the send and receive hosts. It was calculated that the hosts needed to have at least 8MBytes of buffer space available for 1xGigE across an ocean. Once these parameters were changed on the end hosts, final throughput reached 429Mbits/sec for memory to memory transfer and ~30Mbytes/sec (~240Mbits/sec) disk to disk.

5.3 High-Performance Web Server For Large Audiences

HEAnet's National Mirror Server for Ireland, ftp.heanet.ie, currently mirrors over 50,000 software projects and is a popular source of content on the Internet. It serves mostly static content via HTTP, FTP and RSYNC, all available via IPv4 and IPv6. It regularly sustains over 20,000 concurrent connections on a single Apache instance and has served as many as 27,000 with about 3.5 Terabytes of content per day. The front-end system is a Dell 2650, with two 2.4 Ghz Xeon processors, 12Gb of memory, the usual two system disks and 15k RPM SCSI disks, running Debian GNU/Linux and Apache 2.x.

Considerable system and application tuning enabled this system to achieve these performance rates. Apachebench was used for web server benchmarking, bonnie++ and iofzone for file system benchmarking and an in-house script to measure buffering, virtual memory management and scheduling.

Some of the steps taken to tune this system are highlighted below:

5.3.1 Apache

5.3.1.1 MPM Tuning

Apache 2.x has a choice of multi-processing modules. For this system, the prefork MPM was chosen, tuned to have 10 spare servers, the number of spare servers calculated such that there are enough child processes available to handle new requests when the rate of new connections exceeds the rate at which Apache can manage to create new processes.

5.3.1.2 Module Compilation

Apache modules can be compiled directly into one binary, or as dynamically-loaded shared objects which are then loaded by a smaller binary. For our load, a small performance gain (measurable as about 0.2%) was found by compiling the modules in directly.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

5.3.1.3 *htaccess*

As the `highperformance.conf` sample provided with Apache suggests, turning off the use of `.htaccess` files, if appropriate can give significant performance improvements.

5.3.1.4 *sendfile*

Sendfile is a system call that enables programs to hand off the job of sending files out of network sockets to the kernel, improving performance and efficiency. It is enabled by default at compile-time if Apache detects that the system supports the call. However, the Linux implementation of sendfile corrupted IPv6 sessions so this was not implemented on `ftp.heanet.ie` for policy reasons.

5.3.1.5 *Mmap*

Mmap (memory map) support allows Apache to treat a file as if it were a contiguous region of memory, greatly speeding up the I/O by dispensing with unnecessary read operations. This allowed serving of files roughly 3 times quicker.

5.3.1.6 *mod_cache*

`mod_disk_cache` is an experimental feature in Apache 2.x that caches files in a defined area as they are being served for the first time. Repeated requests are served from this cache, avoiding the slower file systems. The default was further tuned to increase the `CacheDirLevel4` to 5 to facilitate more files in the cache.

5.3.1.7 *Configure options*

The following configure options were used:

```
CFLAGS="-D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE"; export CFLAGS
"./configure" \
"--with-mpm=prefork" \
"--prefix=/usr/local/web" \
"--enable-cgid" \
"--enable-rewrite" \
"--enable-expires" \
"--enable-cache" \
"--enable-disk-cache" \
"--without-sendfile"
```

As little as possible was compiled into the `httpd` binary to reduce the amount of memory used. The `CFLAGS` exported enabled serving of files over 2Gb in size.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

5.3.2 File System

Choosing a fast and efficient filesystem is very important for a web server. Tests at the time showed XFS gave better performance than ext2 and ext3, up to a margin of 20%. As a caveat, as the number of used inodes in the filesystem grows, XFS becomes very slow at directory traversal. This resulted in a migration to ext3, despite the reduced performance.

5.3.2.1 *noatime*

One significant mount option, *noatime*, was set as knowing the access time was not critical for a busy mirror server.

5.3.2.2 *logbufs*

For XFS, the *logbufs* mount option allows the administrator to specify how many in-memory log buffers are used. While it was not clear what these log buffers do, increasing this number to its maximum increased performance. This performance increase comes at the expense of memory, which was acceptable for the overall design.

5.3.2.3 *dir_index*

For ext3, *dir_index* option is an option whereby ext3 uses hashed binary-trees to speed up lookup in directories. This has proved much faster for directory traversal.

5.3.3 Kernel

The system originally ran the SGI Linux 2.4 kernel, giving about 12,000 sessions as a maximum. However after simply upgrading to the 2.6 kernel the server hit the compiled-in 20,000 limit of Apache without any additional effort, so the scheduler in the 2.6 kernel appears to have markedly improved.

5.3.3.1 *File Descriptors*

One of the most important options to tune for a large-scale web server is the maximum number of file descriptors the system is allowed to have opened at once. The default is not sufficient when serving thousands of clients. It is important to remember that regular files, sockets, pipes and the standard streams for every running process are all classed as file descriptors and that it is easy to run out.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

This figure was set to 5049800.

5.3.3.2 Virtual Memory Manager

In Linux, the VM manages the memory allocated to processes and the kernel and also manages the in-memory cache of files. By far the easiest way to “tune” the VM in this regard is to increase the amount of memory available to it. This is probably the most reliable and easy way of speeding up a web server - add as much memory as you can afford.

The VM takes a similar approach to `mod_disk_cache` for freeing up space - it assigns programs memory as they request it and then periodically prunes back what can be made free. If a lot of files are being read very quickly, the rate of increase of memory usage will be very high. If this rate is so high that memory is exhausted before the VM has had a chance to free any, there will be severe system instability. To correct for this 5 `sysctl` options were set:

```
vm/min_free_kbytes = 204800
vm/lower_zone_protection = 1024
vm/page-cluster = 20
vm/swappiness = 200
vm/vm_vfs_scan_ratio = 2
```

- The first `sysctl` sets the VM to aim for at least 200 Megabytes of memory to be free.
- The second `sysctl` sets the amount of “lower zone” memory directly addressable by the CPU that should be kept free.
- The third `sysctl`, “`vm/page-cluster`” tells Linux how many pages to free at a time when freeing pages.
- The fourth `sysctl`, “`swappiness`,” is a very vague `sysctl` which seems to boil down to how much Linux “prefers” swap, or how “swappy” it should be.
- The final `sysctl`, the “`vm vfs scan ratio`,” sets what proportion of the filesystem-data caches should be scanned when freeing memory. By setting this to 20 we mean that 1/20th of them should be scanned - this means that some cached data is kept longer than it otherwise would, leading to increased opportunity for re-use.

5.3.3.3 Network Stack

Six `sysctl` options were set relating to the network stack:

```
net/ipv4/tcp_rfc1337=1
net/ipv4/tcp_syncookies=1
net/ipv4/tcp_keepalive_time = 300
net/ipv4/tcp_max_orphans=1000
sys/net/core/rmem_default=262144
sys/net/core/rmem_max=262144
```

- TCP `syncookies` and the `RFC1337` options were enabled for security reasons.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

- The default tcp keepalive time was set to 5 minutes to avoid the situation where httpd children handling connections which have not been responsive for 5 minutes are not needlessly waiting in the queue. This has the minor impact that if the client does try to continue with the TCP session at a later time it will disconnect.
- The max orphans option ensures that even despite the 5 minute timeout there are never more than 1,000 processes held in such a state, and will instead start closing the sockets of the longest waiting processes. This prevents process starvation due to many broken connections.
- The final two options increase the amount of memory generally available to the networking stack for queueing packets.

5.3.3.4 Hyperthreading

Hyperthreading is a technology that makes one processor show up as two with the aim of improving resource usage efficiency within the processor. The web server was benchmarked with hyperthreading enabled and hyperthreading disabled. Hyperthreading enabled resulted in a 37% performance increase (from 721 requests per second to 989 requests per second, with the same test). It was therefore enabled.

5.4 Internet2 Land Speed Record Hosts

The Internet2 Land Speed Record is an open-ended competition for the highest-bandwidth, end-to-end transfer of data. More information on the rules and requirements for the competition is available from <http://lsr.internet2.edu/>. One significant requirement for entries is that all hardware units and software modules used to transfer contest data on the source node, the destination node, the links, and the routers must be offered for commercial sale or as open source software to all U.S. members of the Internet2 community. This means that the lessons learned from these demonstration events can be learned and applied to production research projects. The following are some examples of hosts that have set records.

5.4.1 SUNET/SPRINT

In September 2004, SUNET and Sprint set a Land Speed Record of 124,935 terabit-meters per second for multiple and single streams in IPv4. Significantly, this result was achieved on the normal GigaSunet and Sprintlink production infrastructures, shared by millions of other users of those networks. Detailed information about this record is available at <http://proj.sunet.se/LSR3-s/>.

The sender system was a Dell 2650, with one single Intel Xeon 2.0 GHz CPU and 1024 Mbytes of RAM. The receiver was a Dell Precision 650, with one single Intel Xeon 2.8 GHz CPU and 512 Mbytes of RAM. Both hosts used the Intel® PRO/10GbE LR Network Interface Card. While setting the record, it was noted that the PCI-X bus and the memory bandwidth in the end hosts are currently the bottlenecks.

The transferring application was ttcp and the operating system was NetBSD with the following tuning applied:

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

5.4.1.1 Kernel compile-time parameters:

```
options NMBCLUSTERS=8192 # Increase number of network buffers.
options MAX_KMAPENT=3000 # Need more kmap entries due to extensive use of kernel
virtual memory
DGE_BUFFER_SIZE=8192 # Size of NIC received pages used in private pool
dge* at pci? dev ? function ? # Intel PRO/10GbE network adapter
```

5.4.1.2 Sysctl parameters:

```
net.inet.tcp.init_win=131000 # Tune TCP start up time
kern.sbmax=300000000 # Max memory a socket can use, 300MB
kern.somaxkva=300000000 # Max memory for all sockets together, 300MB
net.inet.tcp.sendspace=250000000 # Size of transmit window, 250MB
net.inet.tcp.recvspace=250000000 # Size of receive window, 250MB
net.inet.ip.ifq.maxlen=20000 # Max length of interface queue
```

5.4.1.3 Ifconfig settings:

```
ifconfig dge0 10.0.0.1/30 ip4csum tcp4csum udp4csum link0 link1 mtu 4470 up
ip4csum, tcp4csum, udp4csum # Enable hardware checksums
link0, link1 # Set PCI-X burst size to 4k.
```

5.4.2 CALTECH/CERN/CENIC

In November 2004, CALTECH, CERN and CENIC set a Land Speed Record of 184,877 terabit-meters per second for multiple IPv4 streams. The path for the record was between Geneva and Los Angeles over the LHCnet, NLR, Abilene and CENIC backbones. More information about the record is available at <http://dnae.home.cern.ch/dnae/lsr4-nov04/>.

The end systems were Dual Opteron 250 machines (2.4 Ghz) with S2io Xframe 10 Gigabit Ethernet Adapter Network Interface Cards. The operating system was Linux kernel 2.6.9 with FAST TCP enabled and the transferring application was lperf.

The system was tuned to increase socket and TCP buffers, disable TCP timestamps and SACK, set txqueuelen and the MTU and other alterations. The details are available from http://dnae.home.cern.ch/dnae/lsr4-nov04/s2io_perf.sh-GE and http://dnae.home.cern.ch/dnae/lsr4-nov04/sysctl_s2io.conf.huge-GE.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

5.4.3 University of Tokyo/WIDE/Chelsio

In December 2004, the University of Tokyo, Chelsio and the WIDE (Widely Integrated Distributed Environment) Project set a Land Speed Record of 216,300 terabit-meters per second. The path for the record was from Tokyo to Tokyo via Amsterdam and New York over the IEEAF, CA*net, Surfnet, Abilene and JGN2 networks. Detailed information is available at <http://data-reservoir.adm.s.u-tokyo.ac.jp/lsr-20041225/>.

The end systems were Dual AMD Opteron 248 machines (2.2GHz) with 1G byte of RAM each and the Chelsio T110 (10GBASE-SR) Network Interface Card with TCP offload engine (TOE) support enabled. The operating system was Linux kernel 2.6.6 and the transferring application Iperf.

As well as using the TCP offload engine, the system was further tuned to alter the congestion window size, buffer size, size of queue, and Ethernet frame size. Flow control was also used.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

6 Conclusions and further work

As stated in the introduction, this document is part of a wider group of resources designed to get the best performance and quality of service available from end-to-end services. The information provided is currently of use to end-system users, administrators and designers but it is important that the process of information gathering and dissemination continues. A key part of the further work introduced in this document is the ongoing maintenance and growth of the PERT Knowledge Base, available at <http://pace.geant2.net/cgi-bin/twiki/view/PERTKB/WebHome>. This collection of information forms the basis of this guide, and its companion document, DS3.3.2 Current Good Practice for Campus Networks and is a resource for future revisions during the life of this Service Activity. Feedback from end-users and ongoing experience in the Géant2 PERT contribute to the body of knowledge available.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

7 References

M. Allman, V. Paxson, W. Stevens

RFC 2581, *TCP Congestion Control*, [ftp://ftp.rfc-editor.org/in-notes/rfc2581.txt](http://ftp.rfc-editor.org/in-notes/rfc2581.txt), April 1999

G. Almes, S. Kalidindi, M. Zekauskas

RFC 2679: *A One-way Delay Metric for IPPM* [ftp://ftp.rfc-editor.org/in-notes/rfc2679.txt](http://ftp.rfc-editor.org/in-notes/rfc2679.txt),
September 1999

CERN

Internet2 Land Speed Record: 6.86 Gbps Geneva - US – Geneva,
<http://dnae.home.cern.ch/dnae/lsr4-nov04/>, November 2004

S. Cheshire

It's the Latency, Stupid, <http://www.stuartcheshire.org/rants/Latency.html>, May 1996

Cisco Systems

Understanding Delay in Packet Voice Networks
<http://www.cisco.com/warp/public/788/voip/delay-details.html#serializationdelay>

Cisco Systems

MTU Tuning for L2TP
http://www.cisco.com/en/US/tech/tk801/tk703/technologies_tech_note09186a0080094c4f.shtml#mtu/

Cisco Systems

Troubleshooting Cisco Catalyst Switches to NIC Compatibility Issues, Cisco Tech Note 17053,
<http://www.cisco.com/warp/public/473/46.html>

A. Currid

TCP Offload to the Rescue, ACM Queue vol. 2, no. 3,
<http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=154>, May 2004

EGEE

EGEE Network Performance Metrics, <https://edms.cern.ch/document/475908/1>

J. Eggers, S. Hodnett

Ethernet Autonegotiation Best Practices, <http://www.sun.com/blueprints/0704/817-7526.pdf>,
July 2004

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

- FastCompany.com** *Why the Long Wait?* <http://www.fastcompany.com/online/38/ifaqs.html>, September 2000
- S. Floyd** *ECN (Explicit Congestion Notification) in TCP/IP*, <http://www.icir.org/floyd/ecn.html>
N. Freed *RFC 2920/STD 60 SMTP Service Extension for Command Pipelining*, <ftp://ftp.rfc-editor.org/in-notes/rfc2920.txt>, September 2000.
- Global Grid Forum's Network Measurements Working Group**
A Hierarchy of Network Performance Characteristics for Grid Applications and Services,
<http://www.didc.lbl.gov/NMWG/docs/measurements.pdf>
- GN2 PERT** *Géant2 PERT KnowledgeBase*, <http://pace.geant2.net/cgi-bin/twiki/view/PERTKB/WebHome>
- Carl Harris** *Windows 2000 TCP Performance Tuning Tips*
<http://rdweb.cns.vt.edu/public/notes/win2k-tcpip.htm>
- G. Huston** *It's Latency* <http://www.potaroo.net/papers/isoc/2004-01/latency.html>, January 2004
- G. Huston** *Faster*, <http://www.potaroo.net/ispcol/2005-06/faster.html>, June 2005.
- IETF** IETF IPPM Working Group <http://www.ietf.org/html.charters/ippm-charter.html>
- Intel Corporation** *Interrupt Moderation Using Intel Gigabit Ethernet Controllers*, Intel Application Note AP-450,
<http://www.intel.com/design/network/applnots/ap450.pdf>, September 2003
- V. Jacobson, R. Braden, D. Borman,**
RFC 1323, TCP Extensions for High Performance, <ftp://ftp.rfc-editor.org/in-notes/rfc1323.txt>,
May 1992
- Juniper Networks** *Supporting Differentiated Service Classes: TCP Congestion Control Mechanisms*
http://www.juniper.net/solutions/literature/white_papers/200022.pdf
- J. Klensin (Ed.),** *RFC 2821 Simple Mail Transfer Protocol*, <ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt>, April 2001
- Lawrence Berkeley National Laboratory**
TCP Tuning Guide - FreeBSD, <http://www.didc.lbl.gov/TCP-tuning/FreeBSD.html>
- Lawrence Berkeley National Laboratory**
TCP Tuning Guide - Linux, <http://www.didc.lbl.gov/TCP-tuning/linux.html>
- Lawrence Berkeley National Laboratory**
TCP Tuning Guide - Solaris, <http://www.didc.lbl.gov/TCP-tuning/Solaris.html>
- C. MacCárthaigh** *Scaling Apache 2.x Beyond 20,000 Concurrent Connections*,
<http://www.stdlib.net/~colmmacc/Apachecon-EU2005/scaling-apache-handout.pdf>, July 2005

M. Mathis, J. Mahdavi, S. Floyd, A. Romanow

RFC 2018, *TCP Selective Acknowledgment Options*, <ftp://ftp.rfc-editor.org/in-notes/rfc2018.txt>,
October 1996

M. Mathis R. Reddy *Enabling High Performance Data Transfers*. <http://www.psc.edu/networking/projects/tcptune/>

Microsoft Corporation *Performance Tuning Guidelines for Microsoft Services for Network File System*.
<http://www.microsoft.com/technet/interopmigration/unix/sfu/perfnfs.msp#EEAA/>

Microsoft Corporation *TCP/IP and NBT configuration parameters for Windows XP (KB314053)*
<http://support.microsoft.com/default.aspx?scid=kb:en-us:314053>,

Microsoft Corporation *Microsoft Windows Server 2003 TCP/IP Implementation Detail*
<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/networking/tcpip03.msp>,

Microsoft Corporation *TechNet Support WebCast: TCP/IP Stack Improvements in Windows Server 2003 and Windows Server 2003 Service Pack 1*, <http://support.microsoft.com/default.aspx?kbid=900937>

Microsoft Corporation *Monitoring Scripts*
<http://www.microsoft.com/technet/scriptcenter/scripts/network/monitor/default.msp>

Microsoft Corporation *Windows Network Task Offload, Windows Hardware and Driver Central*,
<http://www.microsoft.com/whdc/device/network/taskoffload.msp>

Netperf *Public Netperf benchmark database* <http://www.netperf.org>

K. Packard, J. Gettys, *X Window System Network Performance*,
<http://keithp.com/~keithp/talks/usenix2003/html/net.html>, June 2003

V. Paxson, G. Almes, J. Mahdavi, M. Mathis

RFC 2330: *Framework for IP Performance Metrics* <ftp://ftp.rfc-editor.org/in-notes/rfc2330.txt>,
May 1998

Pittsburgh Supercomputing Center

High Performance Enabled SSH/SCP, <http://www.psc.edu/networking/projects/hpn-ssh/>

R. Prasad, M. Jain, and C. Dovrolis

Effects of Interrupt Coalescence on Network Measurements,
<http://www.pam2004.org/papers/265.pdf> , April 2004.

K. Ramakrishnan, S. Floyd, D. Black

RFC 3168 *The Addition of Explicit Congestion Notification (ECN) to IP*,
<ftp://ftp.ietf.org/rfc/rfc3168.txt>, September 2001

S. Shalunov, R. Carlson *Detecting Duplex Mismatch on Ethernet*, <http://www.pam2005.org/PDF/34310138.pdf>

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

- SUNET** *SUNET Internet2 Land Speed Record: 124.935 Pbmps (single stream),*
<http://proj.sunet.se/LSR3-s/>, September 2004
- Sun Microsystems** *Solaris OS Network Performance, BigAdmin System Administration Portal,*
<http://www.sun.com/bigadmin/content/networkperf/>
- SysKonnnect,** *SK-NET GE Gigabit Ethernet Server Adapter,*
http://www.syskonnnect.com/syskonnnect/technology/SK-NET_GE.PDF, 2003
- R. Thomas** *Unix IP Stack Tuning Guide, <http://www.cymru.com/Documents/ip-stack-tuning.html>*
- B. L. Tierney** *TCP Tuning Guide, <http://www-didc.lbl.gov/TCP-tuning/>*
- S. Tripathi** *FireEngine - A New Networking Architecture for the Solaris Operating System ,*
http://www.sun.com/bigadmin/content/networkperf/FireEngine_WP.pdf, November 2004
- S. Ubik, P. Cimbali** *Achieving Reliable High Performance in LFNs,*
<http://staff.cesnet.cz/~ubik/publications/2003/terena2003.pdf>, May 2003
- S. Ubik, P. Cimbali** *Debugging end-to-end performance in commodity operating systems,*
<http://staff.cesnet.cz/~ubik/publications/2003/pfldnet2003.pdf>, February 2003
- University of Tokyo** *Internet2 Land Speed Record in single and multiple TCP stream, <http://data-reservoir.adm.s.u-tokyo.ac.jp/lsr-20041225/>,* December 2004
- J. S. Vöckler** *Solaris - Tuning your TCP/IP stack <http://www.sean.de/Solaris/soltune.html>*
- Yee-Ting Li** *The Effect of TxqueueLen on High Bandwidth Delay Product Network*
<http://www.hep.ucl.ac.uk/~ytl/tcpip/linux/txqueueLen/datatag-tcp/>

Appendix B **GÉANT2 PERT Best Practice Guide**

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

31.08.05

Deliverable DS3.3.2 – Part 2 (App. B): GÉANT2 Performance Enhancement and Response Team (PERT) User Guide and Best Practice Guide



Deliverable DS3.3.2 – Part 2

Contractual Date: 31/05/05
Actual Date: 31/08/05
Contract Number: 511082
Instrument type: Integrated Infrastructure Initiative (I3)
Activity: SA3
Work Item: 3
Nature of Deliverable: R (Report)
Dissemination Level: PU (Public)
Lead Partner: SWITCH
Document Code: GN2-05-176v4

Authors: Francois Xavier Andreu (RENATER), Alex Gall (SWITCH), Ann Harding (HEAnet), Simon Leinen (SWITCH), Colm MacCárthaigh (HEAnet), Orla McGann (HEAnet), Simon Muyal (RENATER), Hank Nussbacher (IUCC), Toby Rodwell (DANTE), Ulrich Schmid (SWITCH), Robert Stoy (DFN), Chris Welti (SWITCH)

Abstract: This document is an information source for network administrators and experts dealing with end-to-end network performance problems. This guide is based on the currently documented knowledge of the PERT group as documented in the PERT knowledge base [PERT-KB]. The process of information provision around end-to-end performance problems will produce more input in the future and enhance the PERT knowledge base.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Table of Contents

0	Executive Summary	70
0.1	Goals and target readership	70
0.2	Related work	70
1	Hosts, Adapters, OS Tuning	71
2	Hardware considerations	72
2.1	Network adapters	72
2.1.1	TCP Offload Engines (TOEs)	72
2.1.2	Large Send Offload (LSO)	73
2.1.3	Interrupt Coalescence	73
2.1.4	Checksum Offload	73
2.2	File systems and disks	74
2.2.1	Benchmarking	74
2.2.2	Tuning	74
3	Operating system considerations	76
3.1	Out-of-the box system settings and tuning	76
3.2	Operating-specific tuning tips and tools	76
3.2.1	Microsoft Windows	76
3.2.2	Linux	77
3.2.3	BSD Variants	79
3.2.4	MAC OS X	79
3.2.5	Solaris	79
4	Performance Pitfalls	80
4.1	Path MTU Discovery Issues	80
4.2	Middleboxes	81
4.3	Duplex modes and auto-negotiation	82
4.4	LAN Collisions	84

4.5	LAN Broadcast Domains	84
5	Monitoring and measurement	85
5.1	Proactive vs. Reactive Measurements	85
5.2	Proactive passive monitoring	86
5.2.1	SNMP monitoring frameworks	87
5.2.2	Netflow accounting	88
5.3	Proactive Active measurements	89
5.3.1	Traditional Methods and Tools	89
5.3.2	Measurement Boxes Frameworks	91
5.4	Reactive Passive Measurements	93
5.4.1	SNMP queries with small polling intervals	94
5.4.2	Traffic capture and analysis	94
5.5	Reactive Active Measurements	95
5.5.1	Reachability and Round Trip Times	95
5.5.2	Path Characteristics	96
5.5.3	Bandwidth Measurements	98
5.5.4	Advanced Diagnosis Tools	101
6	Differentiated Service over GEANT	102
6.1.1	Premium IP	102
6.1.2	LBE (Less Than Best Effort) Service	102
7	References	104

Table of Figures

Figure 3.1: Microsoft Windows TCP Tuning 1	77
Figure 3.2: Microsoft Windows TCP Tuning 2	77
Figure 3.3: Linux TCP Tuning 1	78
Figure 3.4: Linux TCP Tuning 2	78
Figure 3.5: Linux TCP Tuning 3	78
Figure 3.6: Linux TCP Tuning 4	78
Figure 3.7: MAC OSX TCP Tuning	79
Figure 5.1: IPPM system example: Here one way delay and one way delay variation is shown graphically on a measurement path. Increased delay and variations have been caused by a saturated backbone access link.	92
Figure 5.2: RIPE TTM system example: Here the graphical output on a measurement path without any problems during a 24h timeframe is shown	93

0 **Executive Summary**

0.1 **Goals and target readership**

This document is a good practice guide and information source for network administrators and experts dealing with end-to-end network performance problems. Hence those problems comprise the whole path between the network connected hosts including the Campus LANs and the multi domain WAN this document serves for the Campus LAN network administrators as well as for the networking experts working at the NRENs and the GEANT2 network.

0.2 **Related work**

Whereas this guide assumes some already available knowledge around network performance problems, the complementary user guide document [user guide] provides information especially for the end user, who may not so experienced with performance issues within the network.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

1 Hosts, Adapters, OS Tuning

When campus network administrators are investigating performance problems of their users they should first consider (and discuss with their users) the following issues:

- Operating system, host and network interface hardware must be optimally coordinated. Software issues on the adapter drivers sometimes prevent optimal adapter performance. Especially with new adapters on the market, the driver software is often in an early stage of development, so it is recommended to watch out for the latest driver software and bug announcements.
- Before looking for causes of poor network performance at the network one should ensure that the end system is behaving well. The best way to do that is a lab performance test with traffic generators and analysers. Often there is no full lab environment available in which case it is useful to connect the end system under test back to back to a known, well behaving end system test for performance measurements.
- Once an end system has satisfied all performance expectations in a local network environment this does not mean that the end system it is fault free with respect to poor performance over the wide area network. If TCP is used as transport protocol over the WAN, then usually the TCP settings on the operating systems on both end systems must be tuned in order to adapt to the specific WAN environment.

2 Hardware considerations

2.1 Network adapters

One aspect that causes many performance problems is adapter and NIC compatibility issues. The following link from Cisco covers many vendor NICs:

http://www.cisco.com/en/US/products/hw/switches/ps700/products_tech_note09186a00800a7af0.shtml

2.1.1 TCP Offload Engines (TOEs)

The idea of a TOE is to put the TCP implementation onto the network adapter itself. This relieves the computer's CPUs of handling TCP packet processing.

The drawbacks of TOEs are that they require driver support in the operating system, as well as additional kernel/driver interfaces for TCP-relevant operations. Also, when the operating system implements improvements to TCP over time, those normally have to be implemented on the TOE as well. And additional instrumentation such as the Web100 kernel instrumentation set would also need to be implemented separately.

For these and other reasons, TOEs (which are a relatively old idea) have never become a mainstream technology. In contrast, some more generic performance enhancements such as Large Send Offload (LSO), interrupt coalescence, or checksum offload, are now part of many "commodity" network adapter chip-sets, and enjoy increasing support in operating systems.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

2.1.2 Large Send Offload (LSO)

TCP Large Send Offload is typically assumed to be a subset of TOE functionality and is a feature available in some network adapters. With TCP LSO (aka Segmentation Offload), TCP can pass a buffer to be transmitted that is bigger than the MTU supported by the medium. Intelligent adapters implement large sends by using the prototype TCP and IP headers of the incoming send buffer to carve out segments of required size. Copying the prototype header and options, then calculating the sequence number and checksum fields creates TCP segment headers. All other information, such as options and flag values, are preserved except in a few special instances. More information on LSO is available at <http://www.microsoft.com/whdc/device/network/taskoffload.aspx>.

2.1.3 Interrupt Coalescence

A common bottleneck for high-speed data transfers is the high rate of interrupts that the receiving system has to process - traditionally, a network adapter generates an interrupt for each frame that it receives. These interrupts consume signalling resources on the system bus(es) and introduce significant CPU overhead as the system transitions back and forth between productive work and interrupt handling many thousands of times a second.

To alleviate this load, some high-speed network adapters support interrupt coalescence or interrupt moderation. When multiple frames are received in a short timeframe ("back-to-back"), these adapters buffer those frames locally and only interrupt the system once.

While this scheme lowers interrupt-related system load significantly, it can have adverse effects on timing and can make TCP traffic more bursty. Therefore it would make sense to combine interrupt coalescence with on-board time stamping functionality. Unfortunately that doesn't seem to be implemented in commodity hardware/driver combinations yet.

On Linux systems with additional driver support, the `ethtool -C` command can be used to modify the interrupt coalescence settings of network devices on the fly.

2.1.4 Checksum Offload

A large part of the processing costs related to TCP is the generation and verification of the TCP checksum. Many Gigabit Ethernet chipsets include on-board hardware that can verify and/or generate these checksums. This significantly reduces the amount of work that has to be done by the system kernel on a CPU, especially when combined with other adapter/driver enhancements such as Large-Send Offload. Checksum Offload is also part of TCP Offload Engines (TOEs), which move the entire TCP processing from the CPU(s) to the adapter. Checksum Offload requires special driver support and a kernel infrastructure that supports such drivers.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

2.2 File systems and disks

Different file systems can offer different levels of performance under varying conditions. Depending on the use of the system, an end-host administrator may wish to tune the file system to improve read or write performance.

2.2.1 Benchmarking

Benchmarking is an important part of checking performance. Two commonly used applications for benchmarking disk and/or file system performance are *bonnie++* and *iozone*.

2.2.1.1 *Bonnie++*

Bonnie++ is an application that provides a number of checks for file system performance. The basic tests are for the types of file system activity that have been observed to be bottlenecks in I/O-intensive applications. For each of these tests, *Bonnie++* reports the number of Kilo-bytes processed per elapsed second, and the % CPU usage. Further tests available are file create/stat/unlink tests to simulate some operations that are common bottlenecks on large Squid and INN servers, and machines with tens of thousands of mail files in `/var/spool/mail`.

Bonnie++ is available from <http://sourceforge.net/projects/bonnie/>.

2.2.1.2 *iozone*

The *iozone* benchmark tests file I/O performance for read, write, re-read, re-write, read backwards, read strided, *fread*, *fwrite*, random read, *pread*, *mmap*, *aio_read* and *aio_write* operations. It produces MS Excel compatible output for graph generation and is available for AIX, BSDI, HP-UX, IRIX, FreeBSD, Linux, OpenBSD, NetBSD, OSFV3, OSFV4, OSFV5, SCO OpenServer, Solaris, Windows95/98/NT.

iozone is available from <http://www.iozone.org/>. This site also has a handy comparison of the performance of various file systems at http://www.iozone.org/src/current/DL580_multi.xls.

2.2.2 Tuning

Different file systems have different features available for tuning. Care must be taken when enabling or disabling features that the overall impact on the system is not negative compared to performance gain. Here are a few specific tuning tips that may be appropriate for some systems.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

2.2.2.1 *noatime*

Setting *noatime* as a mount option is the easiest way to dramatically increase filesystem performance for read operations. Normally, when a file is read, Unix-like systems update the inode for the file with this access time so that the time of last access is known. This operation means that read operations also involve writing to the filesystem - a severe performance bottleneck in most cases. If knowing this access time is not critical, it may be appropriate to use this option.

2.2.2.2 *dir_index*

For ext3, *dir_index* option is an option whereby ext3 uses hashed binary-trees to speed up lookup in directories. This significantly speeds up directory traversal.

3 Operating system considerations

3.1 Out-of-the box system settings and tuning

Operating Systems (OSs) can sometimes reconfigure network interface settings back to their default, even when the correct values have been written in a specific configuration file. This is the result of bugs, and they appear in almost all OSs. Sometimes they get fixed in a given release but then get broken again in a later release. It is not known why this is the case but it may be partly that driver programmers don't test their products under conditions of large latency. It is worth noting experience shows that 'ifconfig' works well for tuning txqueuelen and MTU sizes.

3.2 Operating-specific tuning tips and tools

Most operating systems require manual tuning to use large TCP windows and other performance enhancements. This section contains information on this and other tuning tips and resources available which have been gathered by contributors to the Géant2 PERT and to Géant2 Service Activity PACE (Performance and Allocated Capacity for End-users).

3.2.1 Microsoft Windows

It appears that, by default, not only does Microsoft Windows not support TCP 1323 scalable windows, but the required key is not even in the Windows registry. The key (Tcp1323Opts) can be added to at least 2 places, and it is not clear if either location has an advantage over the other.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP]
```

OR

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters]
Value Name: Tcp1323Opts
Data Type: REG_DWORD (DWORD Value)
Value Data: 0, 1, 2 or 3
    * 0 = disable RFC 1323 options
    * 1 = window scale enabled only
    * 2 = time stamps enabled only
    * 3 = both options enabled
```

Figure 3.1: Microsoft Windows TCP Tuning 1

(Note: the key need only be added to one of the locations above, not both)

Inquiry at Microsoft has revealed that the default send window is 8KB and that there is no official support for configuring a system-wide default. However, the current Winsock implementation uses the following undocumented registry key for this purpose

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\AFD\Parameters]
Value Name: DefaultSendWindow
Data Type: REG_DWORD (DWORD Value)
Value Data: The window size in bytes.
```

Figure 3.2: Microsoft Windows TCP Tuning 2

The maximum window size value is unknown.

According to Microsoft, this parameter may not be supported in future Winsock releases.

3.2.2 Linux

A comprehensive guide to TCP Tuning Guide on Linux is available at <http://www-didc.lbl.gov/TCP-tuning/linux.html>. Some particularly useful aspects are detailed below.

Linux has its own implementation of the TCP/IP Stack. With recent kernel versions, the TCP/IP implementation contains many useful performance features. Parameters can be controlled via the /proc interface or using the sysctl mechanism.

A typical configuration for high Transmission Control Protocol throughput over Long Fat Networks would include the following in /etc/sysctl.conf:

```
# setting some tcp tuning values effective for long distance high speed networks
net/core/rmem_default = 65536
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

```
net/core/wmem_default = 65536
net/core/rmem_max = 8388608
net/core/wmem_max = 8388608
net/ipv4/tcp_sack = 1
net/ipv4/tcp_mem = 1048576 2097152 4194304
net/ipv4/tcp_rmem = 8192 65536 8388608
net/ipv4/tcp_wmem = 8192 87380 8388608
net/core/netdev_max_backlog = 2500
```

Figure 3.3: Linux TCP Tuning 1

Note if you have a server with hundreds of connections, you might not want to use a large default value for TCP buffers, as memory will quickly run out.

If you are using a web100 kernel, the following parameters seem to improve networking performance even further:

```
# web100 tuning
# turn off caching of ssthresh
net/ipv4/web100_no_metrics_save = 1
# turn off using txqueuelen as part of congestion window computation
net/ipv4/WAD_IFQ = 1
# turn on HSTCP
net/ipv4/tcp_altAIMD = 1
```

Figure 3.4: Linux TCP Tuning 2

Note that although some of these parameters have ipv4 in their names, they apply equally to TCP over IPv6.

Another important parameter to note is txqueuelen, the transmission queue length, which limits the number of packets in the transmission queue in the interface's device driver. The default value is often not suitable for high-speed interfaces. For Gigabit Ethernet interfaces, it is suggested to use at least a txqueuelen of 1000. Values of up to 8000 have been used successfully to further improve performance, e.g.

```
ifconfig eth0 txqueuelen 1000
```

Figure 3.5: Linux TCP Tuning 3

Older versions of Linux have a TCP/IP weakness in that their interface buffers' max window size is based on the experience of previous connections - if you have loss at any point (or a bad end host at the same route) you limit your future TCP connections. So, you have to flush the route cache to improve performance.

```
sysctl -w net.ipv4.route.flush=1
```

Figure 3.6: Linux TCP Tuning 4

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

3.2.3 BSD Variants

A comprehensive guide to TCP Tuning Guide on FreeBSD is available at <http://www.didc.lbl.gov/TCP-tuning/FreeBSD.html>. Information on tuning OpenBSD is available at <http://www.openbsd.org/faq/faq11.html>.

3.2.4 MAC OS X

As Mac OS X is mainly a BSD derivative, you can use similar mechanisms to tune the TCP stack. For testing temporary improvements, you can directly use sysctl in a terminal window: (you have to be root to do that)

```
sysctl -w kern.ipc.maxsockbuf=8388608
sysctl -w net.inet.tcp.rfc1323=1
sysctl -w net.inet.tcp.sendspace=1048576
sysctl -w net.inet.tcp.recvspace=1048576
sysctl -w kern.maxfiles=65536
sysctl -w net.inet.udp.recvspace=147456
sysctl -w net.inet.udp.maxdgram=57344
sysctl -w net.local.stream.recvspace=65535
sysctl -w net.local.stream.sendspace=65535
```

Figure 3.7: MAC OSX TCP Tuning

Users that are unfamiliar with terminal windows can also use the GUI tool "TinkerTool System" and use its Network Tuning option to set the TCP buffers. The TinkerTool System is available from <http://www.bresink.de/osx/TinkerToolSys.html>.

3.2.5 Solaris

Solaris 10, and Solaris 9 with patches, supports TCP Multidata Transmit (MDT), which is Sun's name for Large Send Offload (LSO). In Solaris 10, this is enabled by default, but in Solaris 9 (with the required patches for MDT support), the kernel and driver have to be reconfigured to be able to use MDT. More information is available from <http://docs.sun.com/app/docs/doc/817-0493/6mg9pruab?a=view> for Solaris 9 and <http://docs.sun.com/app/docs/doc/817-0547/6mgbdbsmn?a=view#whatsnew-updates-98> for Solaris 10.

The TCP/IP stack in Solaris 10 has been largely rewritten from previous versions, mostly to improve performance. This improved version is known as FireEngine. More information is available from FireEngine - A New Networking Architecture for the Solaris Operating System, S. Tripathi, November 2004, http://www.sun.com/bigadmin/content/networkperf/FireEngine_WP.pdf

Other useful resources for tuning Solaris include the Solaris OS Network Performance, BigAdmin System Administration Portal at <http://www.sun.com/bigadmin/content/networkperf/>, a TCP Tuning Guide for Solaris at <http://www.didc.lbl.gov/TCP-tuning/Solaris.html> and Solaris - Tuning your TCP/IP stack, <http://www.sean.de/Solaris/soltune.html>

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

4 Performance Pitfalls

4.1 Path MTU Discovery Issues

The Path Maximum Transfer Unit (MTU) is defined as the minimum of the MTUs of the links (segments) that make up the end to end path in question. Larger Path MTUs generally allow for more efficient data transfers, since they represent a better ratio of packet content bytes to packet header bytes

RFC 1191 describes a method for a sender to detect the Path MTU to a given receiver. This method is widely implemented, but is not robust in today's Internet because it relies on 'ICMP Destination Unreachable - Fragmentation Needed' messages that must be sent by the routers back to the source host that issues the Path MTU discovery. However, such packets do often not receive the source host because either routers do not generate them, or they get lost on the way back to the source, because of firewalls and other packet filters or rate limitations. Improperly working Path MTU discovery results in a variety of problems, ranging from TCP performance degradation to TCP connection loss, all of which are described in detail in [RFC 2923].

In order to overcome these issues the IETF PMTUD working group has specified a robust method for determining the IP Maximum Transmission Unit supported over an end-to-end path. The proposed new method does not rely on ICMP or other messages from the network, but on the packet exchange between the end hosts.

In the meanwhile, there are two main ways to work around the current issues:

- In the network:
On the LAN to WAN borders administrators should carefully configure network filters that passes 'ICMP -fragmentation needed' packets, so that those hosts with the current PMTU Discovery enabled at least are not hampered by the local network. Drawback: On a multi domain, path there is usually no control over the entire path. When configuring tunnels or offering dial in services network administrators should

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

calculate with the additional overhead and thereby reduced IP MTU on those connections. Good practise is to prevent fragmentation where possible. This means that an agreement with the upstream network provider on the maximum possible MTU should be done and to configure that value within the campus LAN consistently. The IP MTU on the end hosts network segments should be less or equal to the MTU value configured with in the Campus LAN.

- On the end host:
Host administrators are advised to disable the Path MTU discovery used in the TCP stacks, and configure a static MTU value. A good summary of how this can be done on various operation systems can be found in [Cisco Note 13708]. Note that the hosts' adaptation to Path MTU changes caused by other network domains is then not possible. If the Path MTU falls below the statically configured host MTU fragmentation will occur in the network. In such cases performance degradation could be possible, but severe effects such as path MTU discovery black holes are prevented.

See also [MTU-Netheaven] and [PMTUD-Cisco] for more information.

4.2 Middleboxes

A middle box is a device that sits in between an end-to-end connection, mostly on the Campus LAN at the backbone border or within the LAN. From the Campus LAN point of view they provide important functions - typically middle boxes are:

- Firewalls
- Traffic Shapers
- Network Address Translators (NATs)
- Proxies

However, these functions have the potential to decrease end-to-end performance and often complicate performance-troubleshooting procedures.

The most common middleboxes are firewalls. [Firewall-Tuning] provides a guide that helps in Performance Tuning of firewalls.

From the PERT point of view we call middlebox devices that disturb the normal end-to-end traffic in some way "evil middleboxes". As you can often not see these devices (they usually work at layer 2), it is difficult to debug issues that involve them. Examples are HTTP proxies and Gateway proxies (all protocols). Normally, these devices are installed for security reasons to filter out "bad" traffic. Bad traffic may be viruses, trojans, "evil javascript", or anything that is not known to the device. Another type of middlebox is the rate shaper. Whilst rate shapers do not change the contents of the traffic, they do drop packets according to rules only known by

themselves. Bugs in such middleboxes can have fatal consequences for "legitimate" Internet traffic which may lead to performance or even worse connection issues.

One example of such a performance issue occurred at the beginning of 2005 in the SWITCH network:

Title: Http Proxy: very slow response from a web server only for a specific circle of people

Symptom: Accessing a specific web-site which contains javascript, is very slow (around 30 seconds for one page)

Analysis Summary: HTTP traffic is split between the webserver and a transparent HTTP proxy on the customer site and the HTTP proxy server and the end-hosts. The transparent HTTP proxy fakes the end-points; to the HTTP web server it pretends to be the customer accessing it and to the customer the HTTP proxy appears to be the web server (faked IP addresses). Accordingly there are 2 TCP connections to be considered here. The proxy receives a HTTP request from the customer to the webserver. It then forwards this request to the webserver and WAITS until it has received the whole reply (this is essential, as it needs to analyze the whole reply to decide if it is bad or not). If the content of that HTTP reply is dynamic, the length is not known. With HTTP1.1 a TCP session is not built for every object but remains intact until a timeout has occurred. This means the proxy has to wait until the TCP session gets torn down, to be sure there is not more content coming. When it has received the whole reply it will forward that reply to the customer who asked for it. Of course the customer will suffer from a major delay.

A future possible evil middle box could be implementation trials of DNS based Global Server Load Balancing (GSLB). [GSLB] provides an in depth explanation of the potential problem.

4.3 Duplex modes and auto-negotiation

A point-to-point Ethernet segment (typically between a switch and an end-node, or between two directly connected end-nodes) can operate in one of two duplex modes: *half duplex* means that only one station can send at a time, and *full duplex* means that both stations can send at the same time. Of course full-duplex mode is preferable for performance reasons if both stations support it.

Duplex Mismatch

Duplex mismatch describes the situation where one station on a point-to-point Ethernet link uses full-duplex mode, and the other uses half-duplex mode. A link with duplex mismatch will seem to work fine as long as there is little traffic. But when there is traffic in both directions, it will experience packet loss and severely decreased performance. Note that the performance in the duplex mismatch case will be much worse than when both stations operate in half-duplex mode.

Work in the Internet2 "End-to-End Performance Initiative" suggests that duplex mismatch is one of the most common causes of bad bulk throughput. Rich Carlson's NDT (Network Diagnostic Tester) uses heuristics to try to determine whether the path to a remote host suffers from duplex mismatch.

Duplex Auto-Negotiation

In early versions of Ethernet, only half-duplex mode existed, mostly because point-to-point Ethernet segments weren't all that common - typically an Ethernet would be shared by many stations, with the CSMA/CD (Collision Sense Multiple Access/Collision Detection) protocol used to arbitrate the sending channel.

When "Fast Ethernet" (100 Mb/s Ethernet) over twisted pair cable (100BaseT) was introduced, an *auto-negotiation* procedure was added to allow two stations and the ends of an Ethernet cable to agree on the duplex mode (and also to detect whether the stations support 100 Mb/s at all - otherwise communication would fall back to traditional 10 Mb/s Ethernet). Gigabit Ethernet over twisted pair (1000BaseTX) had speed, duplex, and even "crossed-cable" (MDX) auto-negotiation from the start.

Why people turn off auto-negotiation

Unfortunately, some early products supporting Fast Ethernet didn't include the auto-negotiation mechanism, and those that did sometimes failed to interoperate with each other. So many knowledgeable people recommended avoiding the use of duplex-auto-negotiation, because it introduced more problems than it solved. The common recommendation was thus to manually configure the desired duplex mode - typically full duplex by hand.

Problems with turning auto-negotiation off

There are two main problems with turning off auto-negotiation

- *You have to remember to configure both ends consistently.* Even when the initial configuration is consistent on both ends, it often turns into an inconsistent one as devices and connectinos are moved around.
- *Hard-coding one side to full duplex when the other does auto-configuration causes duplex mismatch.* In situations where one side must use auto-negotiation (maybe because it is a non-manageable switch), it is *never* right to manually configure full-duplex mode on the other. This is because the auto-negotiation mechanism requires that, when the other side doesn't perform auto-negotiation, the local side *must* set itself to half-duplex mode.

Both situations result in duplex mismatches, with the associated performance issues.

Recommendation: Leave auto-negotiation on

In the light of these problems with hard-coded duplex modes, it is generally preferable to rely on auto-negotiation of duplex mode. Recent equipment handles auto-negotiation in a reliable and interoperable way, with very few exceptions.

4.4 LAN Collisions

In some legacy networks, workstations or other devices may still be connected as into a LAN segment using hubs. All incoming and outgoing traffic is propagated throughout the entire hub, often resulting in a collision when two or more devices attempt to send data at the same time. For each collision, the original information will need to be resent, reducing performance.

Operationally, this can lead to up to 100% of 5000-byte packets being lost when sending traffic off network and 31%-60% packet loss within a single subnet. It should be noted that common applications (e.g. email, FTP, WWW) on LANs produce packets close to 1500 bytes in size, and that packet loss rates >1% render applications such as video conferencing unusable.

To prevent collisions from traveling to every workstation in the entire network, bridges or switches should be installed. These devices will not forward collisions, but will permit broadcasts to all users and multicasts to specific groups to pass through.

When only a single system is connected to a single switch port, each collision domain is made up of only one system, and full-duplex communication also becomes possible.

4.5 LAN Broadcast Domains

While switches help network performance by reducing collision domains, they will permit broadcasts to all users and multicasts to specific groups to pass through. In a switched network with a lot of broadcast traffic, network congestion can occur despite high speed backbones. As universities and colleges were often early adopters of Internet technologies, they may have large address allocations, perhaps even deployed as big flat networks which generate a lot of broadcast traffic. In some cases, these networks can be as large as a /16, and having the potential to put up to sixty five thousand hosts on a single network segment could be disastrous for performance.

The main purpose of sub-netting is to help relieve network congestion caused by broadcast traffic. A successful sub-netting plan is one where most of the network traffic will be isolated to the subnet in which it originated and broadcast domains are of a manageable size. This may be possible based on physical location, or it may be better to use VLANs. VLANs allow you to segment a LAN into different broadcast domains regardless of physical location. Users and devices on different floors or buildings have the ability to belong to the same LAN, since the segmentation is handled virtually and not via the physical layout.

5 Monitoring and measurement

Perhaps the main way of ensuring optimal end to end performance is the validation (by monitoring) of the service quality within the network, coupled with the quick detection of abnormal situations and prompt starting of the appropriate troubleshooting procedures.

5.1 Proactive vs. Reactive Measurements

Proactive measurements comprise the regular monitoring of vital functions on the network nodes as well as on the interconnecting links. The load and failure indicators on network nodes depend on the specific hardware architecture and it's the task of the network operators to ensure that those indicators are continuously monitored. Abnormal situations should be detected and alarms generated in order to start immediate troubleshooting procedures.

Once a performance problem has been detected, either by permanent monitoring measures or by indications from end users, the troubleshooting process includes mainly two types of actions:

- Going through all the available historic monitoring data along the end to end path and correlate them.
- Start additional measurements.

5.2 Proactive passive monitoring

Network nodes including their interfaces provide lots of counters that could be monitored passively, usually through SNMP. Generally, the network nodes operating systems provide node resources consumption counters that could be queried using SNMP. The following network resources on a network node should be monitored:

- Load on all CPUs on a node
- Traffic load on internal bus or switching systems

Network links could be monitored on network nodes using the counters provided by the interfaces. These are especially

- Byte counters
- Packet counters
- Packet drop counters
- CRC error counters

Depending on the link type, the network node interfaces could also provide also layer2 or layer1 counters:

- On optical level receive and transmit powers on the transponder lasers
- bit interleave parity (BIP) errors at the SDH level

However the above described network monitoring procedures have some shortcomings in detecting problems that affects the performance on the end-to-end path.

The counter update interval of network nodes is dependent on the network nodes hardware and software and often is in the range of tens of seconds. Thus by monitoring the nodes counters one cannot detect short overload situations, like traffic bursts or high peaks on CPU load. Another problem is that full monitoring access to the network nodes is usually possible only within an administrative domain, like a campus LAN or a NREN, and access to monitoring data from all the nodes and links along a path is therefore often not feasible (at least today).

Packet loss or short overload situations caused by traffic bursts (for example) are often undetectable, either because of insufficient update intervals of the available counters, or because of missing regular monitoring setups in one of the administrative domains. In advanced traffic conditioning and prioritisation environments there is the danger of adverse side effects leading to unwanted service degradation for specific traffic flows. Counter support in the network nodes in these complex environments is often not sufficient for detecting problems on the node itself.

Therefore it is often useful to consider the end to end path or parts of the end to end path as a black box, and to measure the service quality by external active measurement devices.

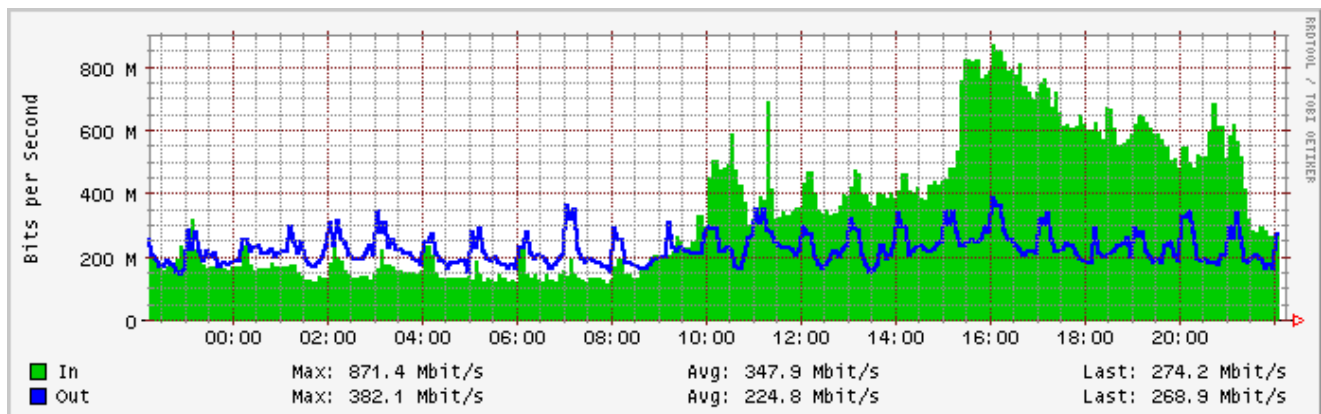
5.2.1 SNMP monitoring frameworks

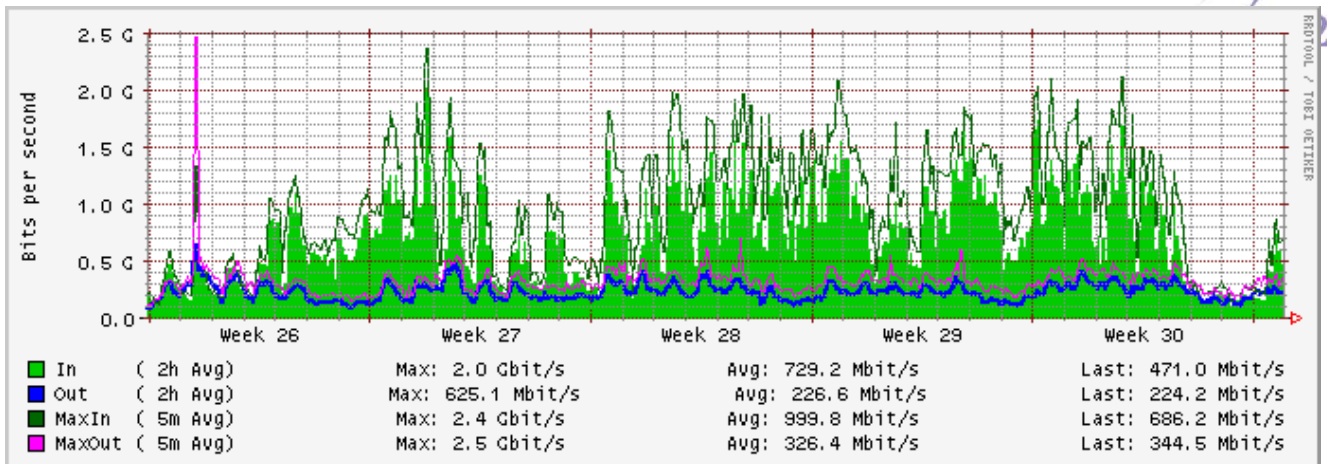
SNMP based monitoring tools that are used for permanent performance monitoring, problem detection and post problem analysis on historic data, should have the following characteristics:

- Polling interval of 5 min at least, but more than 30s.
- Alarming feature
- Storing of historic data without compression over time at least for 24 hours (no compression over time would be ideal, but perhaps impractical).
- Node auto discovery
- Public and well defined format of raw data

The tools MRTG and CRICKET were originally designed for bandwidth utilisation measurements and generic long term monitoring of SNMP counters, although nowadays a lot of feature extensions enable both tools to be used for monitoring and graphically representing almost all relevant counters. The main advantage of these tools is the immediate graphical presentation on the web. Both tools store the measured data in Round Robin Database files using the RRDtool system, and are using that system to create on demand graphs which show time series data in daily, monthly, weekly and yearly graphs. Whereas the daily graphs displays the data averaged over the polling interval of normally 5 minutes, the data polled during the last seven days, the last five weeks and the last twelve months are averaged over 30 minutes, 2 hours and 24 hours respectively. Ideally the maximum value observed during each average interval should also be kept.

The following figure shows a typical MRTG daily graph of link utilisation data and as an example of an aggregated data graph the accompanying monthly graph:





The particular advantage of MRTG and Cricket is the on demand graphing display on the web, thus enabling network administrators to publish utilisation of selected links to selected user groups like project end users as well as network administrators in other domains. Following this idea it would be even possible to calculate the available bandwidth on a given path by correlating all utilisation graphs on all links along the path.

Conversely, because of its excellent graphical representations (especially graph navigation capabilities) the commercially available tools on the market, such as HP Openview, do have advantages compared to the publicly available tools. However, the main disadvantage of the commercial tools is the missing interface to the public because the raw data are usually stored in proprietary vendor formats and a graph presentation on the web is often not possible.

5.2.2 Netflow accounting

Netflow accounting allows for an in depth view into data streams on a link, although real-time (and even near-real time) analysis and user-friendly graphical displays are in their infancy. In principle it is possible to use Netflow accounting for monitoring specific traffic flows through the network, such as detecting abnormal flows during denial of service attacks. However, the current Netflow accounting schemes implemented in router hardware requires the export of sampled Netflows because exporting the full amount of all generated netflow data would consume too much CPU and link resources. Studies have shown that under normal traffic conditions sampling intervals of as little as 1 in 100 produce a good approximation to the real traffic flows, and even 1 in 1000 samples produces a reasonable view of the traffic.

5.3 Proactive Active measurements

5.3.1 Traditional Methods and Tools

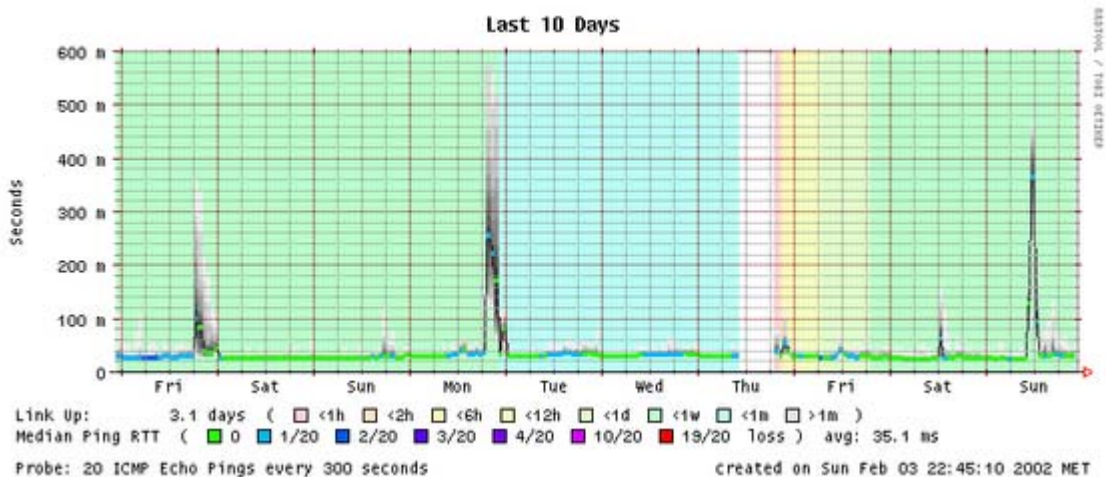
5.3.1.1 Smokeping

Smokeping is a software based measurement framework that uses various software modules (called probes) to measure round trip times, packet losses and availability at layer 3 (IPv4 and IPv6) and even applications' latencies. Layer 3 measurements are based on the ping tool and for analysis of applications there are such probes as for measuring DNS lookup and RADIUS authentication latencies. The measurements are centrally controlled on a single host from which the software probes are started, which in turn emit active measurement flows. The load impact to the network by these streams is usually negligible.

As with MRTG the results are stored in RRD databases in the original polling time intervals for 24 hours and then aggregated over time, and the peak values and mean values on a 24h interval are stored for more than one year. Like MRTG, the results are usually displayed in a web browser, using html embedded graphs in daily, monthly, weekly and yearly timeframes. A particular strength of smoke ping is the graphical manner in which it displays the statistical distribution of latency values over time.

The tool has also an alarm feature that, based on flexible threshold rules, either sends out emails or runs external scripts.

The following picture shows an example output of a weekly graph. The background colour indicates the link availability, and the foreground lines display the mean round trip times. The shadows around the lines indicate graphically about the statistical distribution of the measured round-trip times.



Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

For detailed information on the tool and its software download see [Smokeping]

5.3.1.2 Multicast Beacon

The Multicast Beacon framework has been proven to be a good method for IP multicast performance monitoring. The method relies on software clients that are sending and receiving small artificial RTP packet streams to a definable dedicated multicast group. The receiving clients are using the timestamp and packet sequence information carried within the RTP packets for measuring packet loss, latency and jitter, and reports those values to a central server. The Multicast Beacon server displays all measured values as html code in a sender/receiver matrix. Network administrators as well as end users can use the server's output in order to assess the current situation.

The beacon method enables network administrators to detect end to end multicast performance problems without interaction to the end users, and much more importantly, allows for proactive IP multicast problem detection and isolation.

Beacon software clients should be installed on strategically located hosts. Because of scalability issues it is recommended to restrict the number of clients participating in a beacon multicast session to less than 20. The multicast service monitoring may be divided into sessions within the Campus LAN, national and international sessions. A good approach is to install a beacon server within the campus network, and to ask the WAN providers for a running beacon server to which one software beacon client could join.

Beacon implementations are available for IPv4 and IPv6. The original IPv4 beacon has been derived from DAST/NLANR (see [Beacon-NLANR] for details information and software downloads). An extension including a history feature and ipv6 support, but relying on an older java coded DAST/NLANR version, can be found at PSNC:[Beacon-PSNC]. Dbeacon is another IPv4 and IPv6 capable multicast beacon, which is written in C++ and used within the IPv6 community. More information can be found at [Dbeacon].

5.3.1.3 Permanent Bandwidth measurements

In principle, there are two classes of bandwidth measurement tools. The first class comprises tools that use full data transfers i.e. TCP or UDP streams between the end hosts memory by using the same transport protocols as the applications. Those tools do have the intrinsic danger of affecting production traffic, but they do produce good results.

The second class of tools use artificial packet trains with well-defined packet sizes and packet gaps. Those tools estimate the available bandwidth on the network by calculations based on the changes of the packet inter arrival times at the receiver. The accuracy of the measured estimates is often too coarse; in the best case, those tools can only provide indications to the real available bandwidth.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

The conclusion is that bandwidth measurements using full data transfers should in principle not be done permanently, but only as reactive measurements during isolation of performance problems. Please refer to section 5.5.3 for a more in depth description of using full data transfers.

With regard to bandwidth estimation tools, the PChar tool can be mentioned. This tool has the goal to measure the performance characteristics on each hop along an end-to-end path, please refer to section 5.5.3.2 for a more in depth description.

5.3.2 Measurement Boxes Frameworks

Because of the above described shortcomings of the passive monitoring methods, it is useful to complement the passive network node and link monitoring with active measurements between strategic measurement points within and at the border of the network. However, it is important to note that the amount of measurement traffic emitted to the network by these boxes should be very restricted. The indicators of network performance problems at layer 3 are the metrics:

- Packet loss
- One way delay
- One way delay variation

And additionally

- Packet reordering

It is also important to note, that the used active measurement systems should have an alarm feature implemented in order to be useful in a large-scale deployment with lots of measurement paths.

5.3.2.1 Example Advanced Systems: IPPM and RIPE-TTM

Systems that are capable of measuring the first three of those metrics are the RIPE TTM or the IPPM systems, the latter system is able to measure also packet reordering. Both systems are using off the shelf HW together with means for synchronising the HW clocks on each box. For as most as possible accurate measurements, GPS receiver cards are being used. The systems are working by sending centrally scheduled short packet probes between the measurement boxes. The results from each measurement box are centrally collected, and graphical presentations are presented on the web. Both systems send alarm messages via email.

The following two figures shows example presentations of the graphical tool output on a measurement path, for measurement data in a 24h timeframe.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

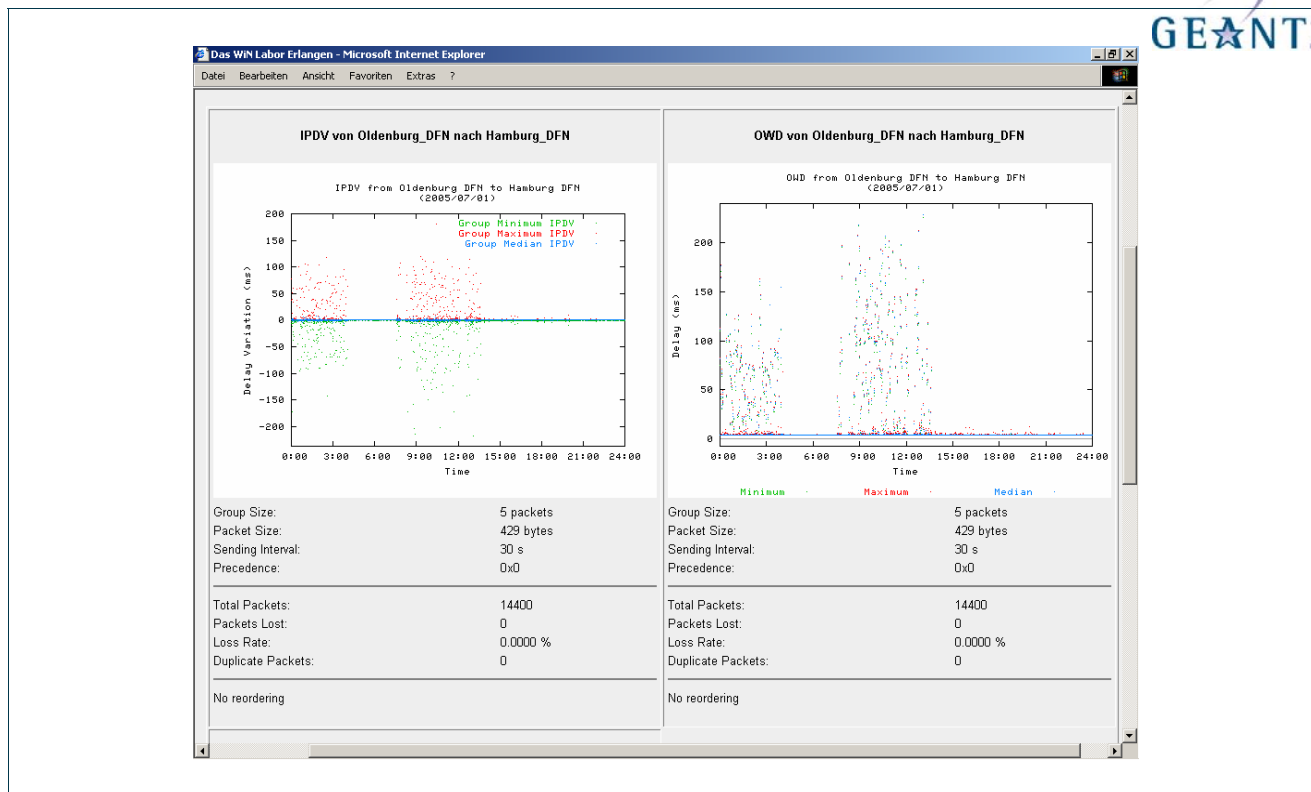


Figure 5.1: IPPM system example: Here one way delay and one way delay variation is shown graphically on a measurement path. Increased delay and variations have been caused by a saturated backbone access link.

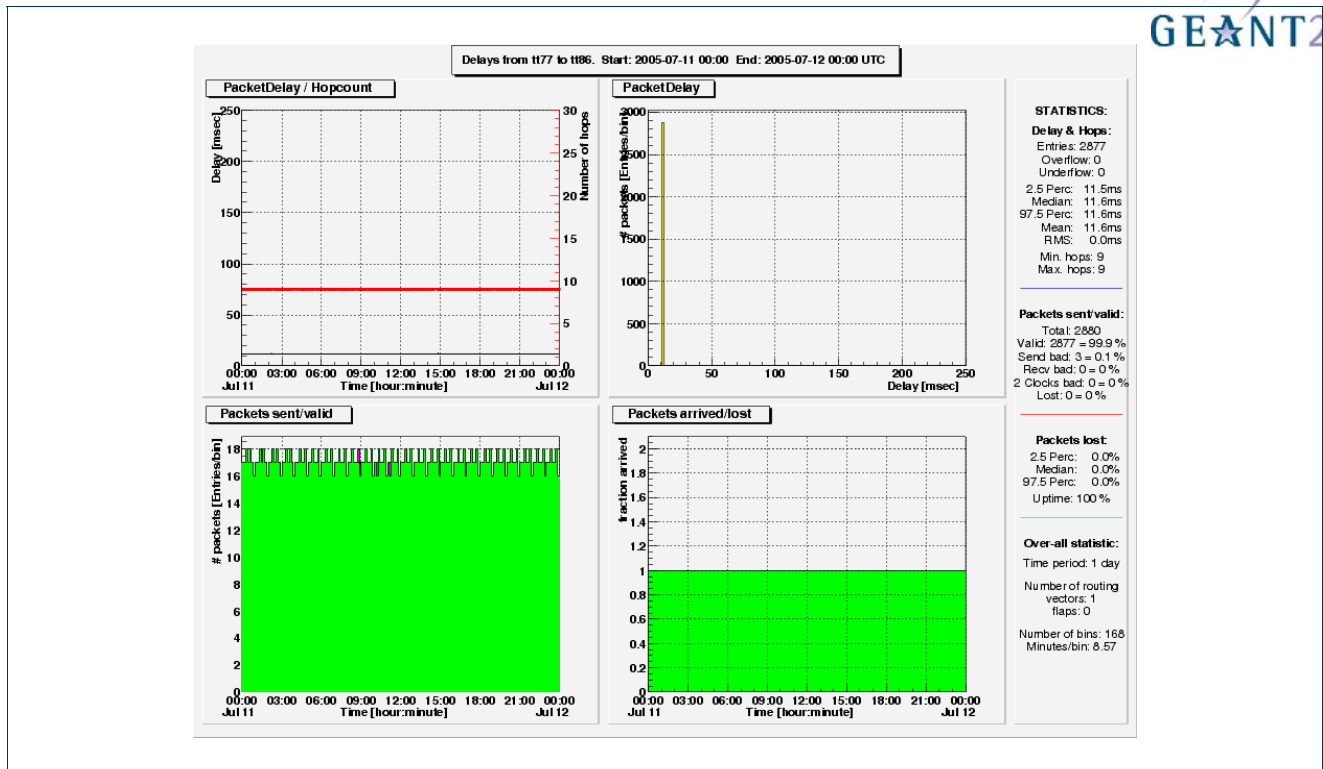


Figure 5.2: RIPE TTM system example: Here the graphical output on a measurement path without any problems during a 24h timeframe is shown

For an in depth description of both systems please refer to [DFN-IPPM] and [RIPE-TTM]. It should be noted that the IPPM system is being developed for the NREN community, hence its feature developments focus on that specific community needs; for example, it implements measurement of out of order packets as well as metric measurements for different IP Precedence values.

5.4 Reactive Passive Measurements

This chapter concentrates on the recommended methods and usable tools to isolate network performance problems and to produce meaningful information that could be provided to the PERT. Furthermore, it's the aim of this section to provide to the campus network administrator staff information on tools enabling them to troubleshoot performance problems in co-operation with the PERT team.

5.4.1 SNMP queries with small polling intervals

Under some circumstances, problem isolation strategies require SNMP polling intervals in order to detect short events like CPU load increases, or traffic bursts. SNMP queries can be used usually with intervals from 20s onwards. This polling interval bound is restricted by the update intervals on today's available router HW.

Lower Polling intervals are usually required to monitor traffic profiles in the second or millisecond timeframe, to detect some sorts of short but severe DoS Attacks. In order to reach those lower polling intervals special passive monitoring HW is required.

5.4.2 Traffic capture and analysis

Capturing specific traffic flows and its analysis is sometimes required for an in depth inspection to performance problems. This section describes two SW based tools that could be used either on an end system or on sniffer hosts within a campus network.

5.4.2.1 *Tcpdump*

Tcpdump is a very useful software based packet capturing tool which is implemented in various Unix environments, thus it is predominantly usable in end systems. Filters allow reducing the captured traffic to specific flows, and the verbosity can be varied largely from interesting packet header fields up to full packet dumps with decoded headers. The tool limitation are bound to the end system HW it runs on especially the host CPU and file storage capacity. Capture sessions with speeds below 100 Mbit/s and periods of tens of minutes are the typical applications on end systems.

5.4.2.2 *Etherreal*

Etherreal is a similar public available tool as Tcpdump, but it has advanced decoding capabilities. On the Microsoft Windows operating systems, it provides an impressive GUI that has features similar to well known hardware sniffers. The limitations on speed and amount of captured data depend on the host hardware and are similar to Tcpdump.

5.5 Reactive Active Measurements

Once a performance Problem has been detected, the campus network administrators are usually involved in the problem isolation process. This section describes the methods and various tools that help them during that process.

5.5.1 Reachability and Round Trip Times

5.5.1.1 *Ping*

Ping sends ICMP echo request messages to a remote host and waits for ICMP echo replies to determine the latency between those hosts. The output shows the Round Trip Time (RTT) between the host machine and the remote target. Ping is often used to determine whether a remote host is reachable. Unfortunately, it is quite common these days for ICMP traffic to be blocked by packet filters / firewalls, so a ping timing out does not necessarily mean that a host is unreachable. There are various flags that control ping's operation and allowing other use cases than simple reachability checks. The following three flags are the common ones used by network administrators. (Note that the flags characters may differ in various operating systems, the ones shown here are implemented with Debian Linux).

The "-f" flag may be specified to send ping packets as fast as they come back, or 100 times per second - whichever is more frequent. As this option can be very hard on the network only a super-user (the "root" account on *NIX machines) is allowed to specified this flag in a ping command.

The "-c" flag specifies the number of Echo request messages sent to the remote host by ping. If this flag isn't used, then the ping continues to send echo request messages until the user types "CTRL C". If the ping is cancelled after only a few messages have been sent, the RTT summary statistics that are displayed at the end of the ping output may not have finished being calculated and won't be completely accurate. To gain an accurate representation of the RTT, it is recommended to set a count of 100 pings. The MS Windows implementation of ping just sends 4 echo request messages by default.

The "-s" flag is followed by the packet size of the ICMP packet payload to, allowing for connectivity checks with large packets. This flag is useful for troubleshooting MTU issues.

5.5.1.2 *Ping6*

Ping6 is the IPv6 implementation of the ping program. It works in the same way, but sends ICMPv6 Echo Request packets and waits for ICMPv6 Echo Reply packets to determine the RTT between two hosts. There are no discernable differences between the *NIX implementations and the MS Windows implementation.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

5.5.1.3 telnet

Another method for checking remote host availability is to telnet to a port that you know to be accessible; such as port 80 (HTTP) or 25 (SMTP). If the connection is still timing out, then the host is probably not reachable; of course it is also possible that there is no service listening on that port. If a connection is made to the remote host, then it can be ended by typing the escape character (usually 'CTRL ^ J') then 'quit'.

5.5.2 Path Characteristics

5.5.2.1 Traditional Traceroute

Traceroute is used to determine the route a packet takes through the Internet to reach its destination; i.e. the number of "hops" it takes. UDP packets are sent as probes to a high ephemeral port (usually in the range 33434--33525) with the Time-To-Live (TTL) field in the IP header increasing by one for each probe sent until the end host is reached. The originating host listens for ICMP Time Exceeded responses from each of the routers/hosts en-route. It knows that the packet's destination has been reached when it receives an ICMP Port Unreachable message; we expect a port unreachable message as no service should be listening for connections in this port range. If there is no response to the probe within a certain time period (typically 5ms), then a "*" is displayed.

The output of the traceroute program shows each host that the packet passes through on it's way to its destination and the RTT to each gateway en-route. Occasionally, the maximum number of hops (specified by the TTL field, which defaults to 64 hops in *NIX implementations) is exceeded before the port unreachable is received. When this happens an "!" will be printed beside the RTT in the output.

Other error messages that may appear after the RTT in the output of a traceroute are:

Error message (Debian implementation)	meaning
"!H"	Host unreachable
"!N"	Network unreachable
"!P"	Protocol unreachable
"!S"	Source route failed
"!F [pmtu]"	Fragmentation needed. [pmtu] displays the Path MTU Discovery value
"!X"	Administratively prohibited. The gateway prohibits these packets, but sends an ICMP message back to the source of the traceroute to inform them of this.
"!V"	Host precedence violation
"!C"	Precedence cut-off
"! [num]"	displays the ICMP unreachable code, as defined in RFC 1812 Requirements for IP Version 4 Routers

Note that *NIX implementations of traceroute send UDP probe packets by default, whilst MS Windows *tracert* sends ICMP echo probes. *NIX implementations of traceroute can be specified to use ICMP Echo messages instead of the default UDP probes, by using the "-I" flag. Note that either or both of ICMP and UDP may be blocked by firewalls, so this must be taken into account when troubleshooting.

5.5.2.2 Traceroute6

Traceroute6 uses the Hop-Limit field of the IPv6 protocol to elicit an ICMPv6 Time Exceeded ICMPv6 message from each gateway ("hop") along the path to some host. Just as with *traceroute*, it prints the route to the given destination and the RTT to each gateway/router.

The following are a list of possible errors that may appear after the RTT for a gateway (especially for OSes that use the KAME IPv6 network stack, such as the BSDs):

Error message	
"!N"	No route to host
"!P"	Administratively prohibited (i.e. Blocked by a firewall, but the firewall issues an ICMPv6 message to the originating host to inform them of this)
"!S"	Not a Neighbour
"!A"	Address unreachable
"!"	The hop limit is ≤ 1 on a Port Unreachable ICMPv6 message. This means that the packet got to it's destination, but that the reply only had a hop limit large enough that was just large enough to allow it to get back to the source of the <i>traceroute6</i> . This option was more interesting in IPv4, where bugs in some implementations of the IP stack could be indentified by this behaviour.

Traceroute6 can also be specified to use ICMPv6 Echo messages to send the probe packets, instead of the default UDP probes, by specifying the "-I" flag when running the program. This may be useful in situations where UDP packets are blocked by a packet filter / firewall.

5.5.2.3 TCP Traceroute

TCP Traceroute is a traceroute implementation that uses TCP packets instead of UDP or ICMP packets to send its probes. TCP traceroute can be used in situations where a firewall blocks ICMP and UDP traffic. It is based on the "half-open scanning" technique that is used by NMAP, sending a TCP with the SYN flag set and waiting for a SYN/ACK (which indicates that something is listening on this port for connections). When it receives a response, the TCP traceroute program sends a packet with a RST flag to close the connection. There are numerous traceroute servers located throughout the world. Good overviews are shown at:

- <http://www.traceroute.org/>
- http://www.bgp4.net/wiki/doku.php?id=tools:ipv4_traceroute_ping

5.5.2.4 Traceroute like tools

There are a number of other traceroute like tools available for diagnosing network problems, see also [User-Guide]

5.5.3 Bandwidth Measurements

Many performance problems reported by end users are about insufficient network throughput. The problem isolation process requires here to distinguish between issues with applications, the used transport protocol, or problems in the network. Campus network administrators can use active measurement software on well located workstations in order to detect bottleneck situations along an end to end path.

5.5.3.1 Iperf

Iperf is an active measurement tool designed for measuring network characteristics as seen at the transport layer, either TCP or UDP. The tool uses memory-to-memory data transfers from a client instance on a sending host to a server instance on a receiving host, and it is using the protocol stacks of the end system up to and including the transport protocols. After a measurement has been taken the receiver (server) reports the results to the sender (client), where it is displayed to the user. The iperf receiver instance may be started as a daemon, listening on a user specified TCP or UDP port, thus allowing for a permanent measurement point to which clients could connect. IP based authorisation schemes (e.g. hosts access lists on the end systems) may be used to restrict access to well defined senders. In its default operation, iperf transfers data between the hosts' memories, but it allows also for transfers between end systems disks. The tool uses IPv4 by default, but on suitably enabled hosts, it can use IPv6 as well.

Caveats

As Iperf sends real full data streams it can reduce the available bandwidth on a given path. In TCP mode, the effect to the co-existing production flows should be negligible assuming the number of production flows is much greater than the number of test data flows, which is normally a valid assumption on paths through a WAN. However, in UDP mode iperf has the potential to disturb production traffic, and in particular TCP streams, if the sender's data rate exceeds the available bandwidth on a path. Therefore, one should take particular care whenever running iperf tests in UDP mode.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Control of measurements

There are two typical deployment scenarios which differ in the kind of access the operator has to the sender and receiver instances. A measurement between well-located measurement workstations within an administrative domain e.g. a campus network allow network administrators full control on the server and client configurations (including test schedules), and allows them to retrieve full measurement results. Measurements on paths that extend beyond the administrative domain borders require access or collaboration with administrators of the far-end systems. Iperf has two features implemented that simplify its use in this scenario, such that the operator does not need to have an interactive login account on the far-end system:

- The server instance may run as a daemon (option `-D`) listening on a configurable transport protocol port, and.
- It is possible to bi-directional tests, either one after the other (option `-r`) , or simultaneously (option `-d`).

Problem isolation procedures using iperf

TCP Throughput measurements

Typically end users are reporting throughput problems as they see on with their applications, like unexpected slow file transfer times. Some users may already report TCP throughput results as measured with iperf. In any case, network administrators should validate the throughput problem. It is recommended this be done using iperf end-to-end measurements in TCP mode between the end systems' memory. The window size of the TCP measurement should follow the bandwidth*delay product rule, and should therefore be set to at least the measured round trip time multiplied by the path's bottle-neck speed. If the actual bottleneck is not known (because of lack on knowledge of the end-to-end path) then it should be assumed the bottleneck is the slowest of the two end-systems' network interface cards.

For instance if one system is connected with Gigabit Ethernet, but the other one with Fast Ethernet and the measured round trip time is 150ms, then the window size should be set to $100 \text{ Mbit/s} * 0.150\text{s} / 8 = 1875000$ bytes, so setting the TCP window to a value of 2 MBytes would be a good choice.

In theory the TCP throughput could reach, but not exceed, the available bandwidth on an end-to-end path. The knowledge of that network metric is therefore important for distinguishing between issues with the end system's TCP stacks, or network related problems.

Available bandwidth measurements

Iperf could be used in UDP mode for measuring the available bandwidth. Only short duration measurements in the range of 10 seconds should be done so as not to disturb other production flows. The goal of UDP measurements is to find the maximum UDP sending rate that results in almost no packet loss on the end-to-end path, in good practice the packet loss threshold is 1%. UDP data transfers that results in higher packet losses are likely to disturb TCP production flows and therefore should be avoided. A practicable procedure to find the available bandwidth value is to start with UDP data transfers with a 10s duration and with interim result



reports at one second intervals. The data rate to start with should be slightly below the reported TCP throughput. If the measured packet loss values are below the threshold then a new measurement with slightly increased data rate could be started. This procedure of small UDP data transfers with increasing data rate should be repeated until the packet loss threshold is exceeded. Depending on the required result's accuracy further tests can be started beginning with the maximum data rate causing packet losses below the threshold and with smaller data rate increasing intervals. At the end the maximum data rate that caused packet losses below the threshold could be seen as a good measurement of the available bandwidth on the end to end path.

By comparing the reported applications throughput with the measured TCP throughput and the measured available bandwidth, it is possible to distinguish between applications problems, TCP stack problems, or network issues. Note however that differing nature of UDP and TCP flows means that their measurements should not be directly compared. Iperf sends UDP datagrams at a constant steady rate, whereas TCP tends to send packet trains. This means that TCP is likely to suffer from congestion effects at a lower data rate than UDP.

In case of unexpected low available bandwidth measurements on the end-to-end path, network administrators are interested on the bandwidth bottleneck. The best way to get this value is to retrieve it from passively measured link utilisations and provided capacities on all links along the path. However, if the path is crossing multiple administrative domains this is often not possible because of restrictions in getting those values from other domains. Therefore, it is common practice to use measurement workstations along the end-to-end path, and thus separate the end-to-end path in segments on which available bandwidth measurements are done. This way it is possible to identify the segment on which the bottleneck occurs and to concentrate on that during further troubleshooting procedures.

Other iperf use cases

Besides the capability of measuring TCP throughput and available bandwidth, in UDP mode iperf can report on packet reordering and delay jitter.

Other use cases for measurements using iperf are IPv6 bandwidth measurements and IP multicast performance measurements. More information of the iperf features, its source and binary code for different UNIXes and Microsoft Windows operating systems can be retrieved from [iperf]. Iperf version 2.0 is also available as Debian Linux package.

An example of a public iperf server or client that could be started remotely using a web interface is be found at [iperf-gpn].

5.5.3.2 Pchar

Pchar is based on Van Jacobson's pathchar tool, and can be used for bandwidth estimation on all hops on the end-to-end path as well as on the end-to-end connection. Because it uses packet trains and relies on measured losses and inter packet gaps of the responses produced by the routers on the path the results are only estimates and their accuracy depend heavily on the control plane response behaviour of the routers in question.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

Therefore, the measured estimates can indicate problems, but one should not use it to measure performance bottlenecks. The tool is available at the Pchar Home page [Pchar].

5.5.4 Advanced Diagnosis Tools

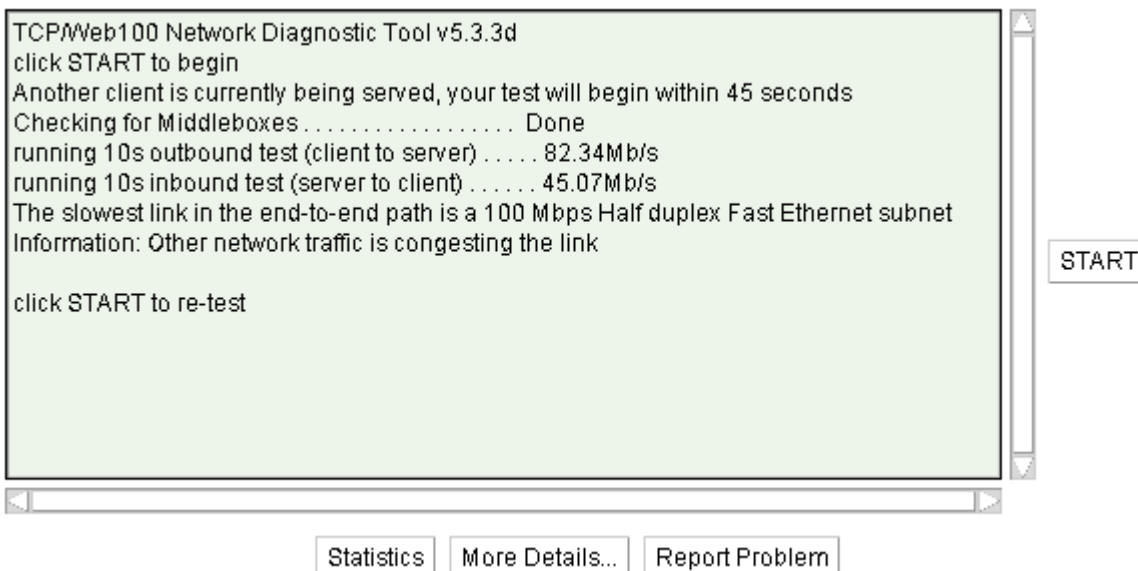
5.5.4.1 NDT

NDT can be used to check the TCP configuration of any host that can run Java applets. The client connects to a Web page containing a special applet. The Web server that serves the applet must run a kernel with the Web100 extensions for TCP measurement instrumentation. The applet performs TCP memory-to-memory tests between the client and the Web100-instrumented server, and then uses the measurements from the Web100 instrumentation to find out about the TCP configuration of the client. NDT also diagnoses common configuration errors such as duplex mismatch.

The NDT server application is an interesting tool for campus administrators who wish to provide an automated diagnostic service to their end users. An overview of NDT is given at [NDT-overview] and a more in depth description is at [NDT-cookbook]

The following example shows the results overview of an NDT measurement between a client in Switzerland with a Fast Ethernet interface and an NDT server in Argonne, IL, USA.

A test takes about 20 seconds. Click on "start" to begin.



```
TCP/Web100 Network Diagnostic Tool v5.3.3d
click START to begin
Another client is currently being served, your test will begin within 45 seconds
Checking for Middleboxes . . . . . Done
running 10s outbound test (client to server) . . . . 82.34Mb/s
running 10s inbound test (server to client) . . . . 45.07Mb/s
The slowest link in the end-to-end path is a 100 Mbps Half duplex Fast Ethernet subnet
Information: Other network traffic is congesting the link

click START to re-test
```

START

Statistics More Details... Report Problem

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

6 Differentiated Service over GEANT

The differentiated services approach to providing quality of service in networks employs a small, well-defined set of building blocks from which a variety of aggregate behaviours may be built. A small bit-pattern in each packet (which in IPv4 is the TOS octet and in IPv6, the Traffic Class octet), is used to mark a packet to receive a particular forwarding treatment, or per-hop behaviour, at each network node.

Examples of Diffserv-based services include the Premium IP and Less Than Best Effort services, which are available on GEANT/GN2 and some NRENs.

6.1.1 Premium IP

Premium IP uses Differentiated Services, and in particular the EF (Expedited Forwarding) PHB (per-hop behaviour) to protect a given aggregate of IP traffic against disturbances by other traffic, so that One Way Delay, Delay Variation and Packet Loss are assured to be low. The aggregate is specified by source and destination IP address ranges, or by ingress/egress AS (Autonomous System) numbers when in the core domain. In addition, a Premium IP aggregate must conform to strict rate limits. Unlike earlier proposals for similar "Premium" services, GEANT/GN2's Premium IP service downgrades excess traffic (over the contractual rate) to "best effort" (DSCP=0), rather than dropping it.

6.1.2 LBE (Less Than Best Effort) Service

Less Than Best Effort (LBE) uses Differentiated Services to mark traffic that should be forwarded with less than the default "best effort" from the network.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	31/08/05
EC Contract No.:	511082
Document Code:	GN2-05-176v4

LBE defines a traffic class that will be able to make use of bandwidth not employed for other traffic on the GÉANT2 network. It allows high-volume, low-priority applications to run in the available bandwidth without adversely affecting best-effort and premium IP traffic (LBE traffic is allocated any bandwidth not used by these services).

If the network becomes congested, Premium IP and Best Effort traffic will take precedence – LBE packets will always be the first to be dropped. LBE traffic is therefore subject to a relatively high risk of packet loss. Despite this, it is particularly useful for researchers who need to transfer large amounts of data, where the speed and completeness of transfer and the order of arrival of the packets is not so important. Examples include:

- Data mirroring, where the mirror is updated continuously rather than once every night
- Non-invasive test traffic
- Network backups that need to be performed continuously throughout the day.

No performance guarantees are offered for LBE traffic.

7 References

- [PERT-KB]** PERT Knowledge Base
<http://kb.pert.switch.ch/cgi-bin/twiki/view/PERTKB/WebHome>
- [User-Guide]** Research Network User's Guide to Performance ("User Practice Guide"), GN2 Deliverable DS 3.2.2 part 1.
- [Firewall-Tuning]** Check Point VPN-1 & FireWall-1 NG Performance Tuning Guide
http://www.checkpoint.com/techsupport/documentation/FW-1_VPN-1_performance.html
- [GSLB]** Why DNS Based Global Server Load Balancing (GSLB) Doesn't Work, Pete Tenereillo
<http://www.tenereillo.com/GSLBPageOfShame.htm>
- [Duplex Mismatch]** Detecting Duplex Mismatch on Ethernet, S. Shalunov and R. Carlson, PAM 2005,
<http://www.pam2005.org/PDF/34310138.pdf>
- [Ethernet Autoneg]** Ethernet Autonegotiation Best Practices, J. Eggers and S. Hodnett, Sun BluePrints™ OnLine, July 2004, <http://www.pam2005.org/PDF/34310138.pdf>
- [Cisco Note 17053]** Troubleshooting Cisco Catalyst Switches to NIC Compatibility Issues, Cisco Tech Note 17053,
<http://www.cisco.com/warp/public/473/46.html>
- [Cisco Note 13708]** Adjusting IP MTU, TCP MSS, and PMTUD on Windows and Sun Systems, Cisco Tech Note 13708, (<http://www.cisco.com/warp/public/105/38.shtml/>)
- [RFC 2923]** TCP Problems with Path MTU Discovery, K. Lahey, September 2000
- [MTU-Netheaven]** Case Study about Path MTU problem "PMTU (Path MTU) Discovery - Some servers are unusable for many internet users" <http://www.netheaven.com/pmtu.html>
- [PMTUD-Cisco]** Cisco White paper: "IP Fragmentation and PMTUD"
http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800d6979.shtml/
- [NDT]** Rich Carlson, Network Diagnostic Tester,
<http://e2epi.internet2.edu/ndt>
- [DFN-IPPM]** DFN IPPM active measurement system
<http://www.win.rrze.uni-erlangen.de/ippm/index.html.en>
- [RIPE-TTM]** RIPE TTM active measurement system
<http://www.ripe.net/test-traffic/>
- [SmokePing]** Smoking Home Page:
<http://people.ee.ethz.ch/~oetiker/webtools/smokeping/index.en.html>

- [Beacon-NLANR]** Multicast Beacon Homepage of the Original at DAST/NLANR
<http://dast.nlanr.net/Projects/Beacon/>
- [Beacon-PSNC]** Multicast Beacon with extensions from the original, done at PSNC
<http://beacon.man.poznan.pl/>
- [Dbeacon]** Dbeacon Homepage
<http://artemis.av.it.pt/~hsantos/dbeacon/>
- [MTR]** Matt's Traceroute home page:
<http://www.bitwizard.nl/mtr/>
- [Iperf]** Iperf Home Page
<http://dast.nlanr.net/Projects/Iperf/>
- [iperf-gpn]** Great Plains iperf server
<http://noc.greatplains.net/measurement/iperf.php>
- [Pchar]** Pchar Homepage, Bruce A. Mah
<http://www.kitchenlab.org/www/bmah/Software/pchar/>
- [NDT-overview]** NDT: Desktop Troubleshooting for All Users, Internet2 End-to-End Performance Initiative, 2005,
<http://e2epi.internet2.edu/ndt/overview.pdf>
- [NDT-cookbook]** Network Diagnostic Tool (NDT): An Internet2 Cookbook, Internet2 End-to-End Performance Initiative, 2005
<http://e2epi.internet2.edu/network-perf-wk/ndt-cookbook.pdf>