23.08.06

Deliverable DS3.3.3: PERT Performance Guides

Deliverable DS3.3.3

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

In addition to investigating specific cases of poor network performance, the GÉANT2 Performance Enhancement and Response Team (PERT) also populate and maintain the PERT Knowledgebase, an online resource for advice and information about all aspects of network performance. Information in this document has been extracted from the PERT Knowledgebase. It serves as a two-part (basic/advanced) 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.

Table of Contents

1 Executive Summary

Until recently the data rates of Wide Area Network (WAN) links were such that the end components of an end system (application, operating system, hardware) could easily match any demands placed upon them by the attached network. With the advent of multi gigabit backbone networks (such as GÉANT) and (more importantly) Gigabit Ethernet Network Interface Cards (NICs) for workstations, it is no longer the case that the network is always the bottleneck.

In order to investigate a network performance issue it is necessary to appreciate the key parameters that describe a network flow. The most common metric is bandwidth, which is the data flow rate measured in bits per second. Of increasing importance is the metric One Way Delay (OWD) which is the time taken for a packet to travel from end-host A to end-host B. A derived metric, jitter, measures the variation in OWD of packets (in fact it is more properly called One Way Delay Variation (OWDV)). The metric Packet loss indicates, as a percentage, how many packets are lost on a given circuit or path – even packet losses in the order of 0.01% can seriously affect the network performance for high-speed applications.

Network performance issues are difficult to troubleshoot for two reasons. First, because when the system is not actually broken (just degraded) there are rarely obvious alarms to indicate the location of the problem. Secondly, the possible locations of the deficient component could be in a number of differently administered domains (campus, national backbone network, international network etc).

When end systems connected to Long, Fat pipe Networks (LFNs) do not live up to expectations it is most likely to be as a result of old protocols which, by default, are not capable of making full use of modern high-speed networks. The best example of this is TCP, which is the prevalent transport protocol in use on the Internet.

Although TCP has received many modifications since it was introduced in the early 1980s, more often than not default installations of end system operating systems will not enable the features needed to transmit data at high speed over long distance. This last point is important since for reliable transmission (such as that provided by TCP) there is a significant difference between short distance and long distance transfers. For reliable transmission, the sender must keep a copy of all data until the receiver has acknowledged its receipt. This means that the sender must keep a copy of all packets in flight (PIF), plus all those packets which have been received by the receiver but which have not yet been acknowledged (because the Acknowledgment packets themselves are 'in flight' back to the data sender). The amount of data the sender has to keep buffered can be simply calculated as the rate at which it is transmitting, multiplied by the round trip time (of packet making its

way from sender to receiver and back again). Clearly as the distance and/or data rate increase, so does the memory required to store the unacknowledged sent data.

However, it is not the consumption of the sending host's memory which causes legacy TCP to be such a potential performance bottleneck. By design, the TCP algorithm seeks to share available bandwidth with other traffic. It does this by starting with a very low transmission rate and increasing it until it experiences packet loss. TCP assumes (quite reasonably, but not always accurately) that packet loss indicates congestion and that it must therefore reduce its transmission rate. The way in which legacy TCP executes this flow control means that it can not take full advantage of the bandwidth offered by LFNs. The first advance was a set of new extensions to TCP (specifically, Window scaling) which gives legacy TCP the possibility of achieving Gigabit rates over international distances. However, the conservative manner in which legacy TCP reacts to perceived congestion (i.e. packet loss) is exacerbated over long distance paths, such that even very small packet loss (perhaps not even caused by congestion) dramatically reduces a TCP flow, which then increases at only a very low rate.

New versions of TCP, generically called "high speed TCP", overcome legacy TCP's inability to make proper use of LFNs, but their development is more complex than might be assumed. A new version of TCP cannot simply be made more aggressive when recovering from packet loss, since it would then not be "friendly" (that is to say, it would not share available bandwidth with other flows). Without co-operation between traffic flows there could be massive instability on the Internet.

An alternative approach to the TCP flow problem (which can be used instead of or as well as a high speed TCP) is to establish multiple parallel flows. This is an effective solution, but introduces extra complexity for the application itself.

The shortcomings of TCP have also been addressed by a new breed of NIC, called TCP Offload Engines (TCP). These devices take over all the computationally heavy jobs of TCP and so reduce the load on the main CPU. However, these devices currently need to non-standard drivers and/or patches to be made to the host operating system, so they are not a "plug-and-play" solution.

Another legacy protocol which can be problematic is the ubiquitous Ethernet. When Ethernet was upgraded to support duplex operation it needed to be backwardly compatible with the original half-duplex equipment. New Ethernet NICs were designed to auto-negotiate (with their neighbour) both their duplex setting and also their speed. Early implementations of auto-negotiation were often incompatible with one another and network administrators often manually configured the settings. Now, auto-negotiation is more reliable, and should in most cases be sued. However, if an auto-negotiate port is linked to a manually configured port then the auto port will default to its most basic setting, which would now be 100Mpbs half-duplex. Connecting a duplex port to a half duplex port will result in significant packet loss and as such wrongly configured Ethernet ports are one of the most common causes of network performance problems.

The two parts of this document give information, guidance and advice on a wide range of topics affecting network performance. The first modern high-speed data respectively a basic and advanced 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 network specialists. It gives an overview of the issues that may be faced when trying to achieve the best possible 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 continued research and development, new advances in this area occur frequently. This is therefore a snapshot document, based on information in the PERT (Performance Enhancement and Response Team) Knowledge Base, and may be subject to revision. For the most up to date information please refer to the PERT Knowledge Base, which can be reached via the GÉANT2 web site (www.geant2.net).

2   General Background and Context

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 users.

The second way in which the PERT helps its users 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 them selves whether current problems are likely to be the result of changes in network conditions. This information is all held in the PERT Knowledge Base[1], an online resource that is accessible to all (see GÉANT2 web site[2], 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 Knowledge Base and this has been used to create two guides to network quality of service. The 'Basic Guide', as its name suggests, contains the basics of end-to-end network performance, as well as information about "basic" performance measurement and analysis tools. It is targeted at end-users, and in particular those end-

---

[1]   The PERT Knowledge Base is currently implemented as a Wiki reachable under http://kb.pert.switch.ch/
[2]   http://www.geant2.net/

users who have demanding requirements (typically those who depend on LFNs), and the network administrators who support them. The Basic Guide is at Annex A of this document. The second, "Advanced" guide (Annex B), builds on top of the Basic Guide. It is aimed at people with a greater knowledge of data networking, typically campus and wide-area network administrators   It concentrates on issues which can not be controlled by re-configuring or upgrading end-systems.

Differences to DS3.3.2

An earlier version of these guides was published as GN2 deliverable DS3.3.2. That older deliverable also consisted of two main parts, but the separation was different, namely into a *User Guide,* directed to end users, and a *Best Practice Guide* for campus system and network administrators. In addition to the different partitioning, this version of the guides contains material that has been added to the knowledge base over the past year.

Project:	GN2	
Deliverable Number:	DS3.3.3	
Date of Issue:	23/08/06	
EC Contract No.:	511082	
Document Code:	GN2-06-135v2	

2

3 Conclusions and Future Work

Whilst the online PERT Knowledge Base, 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 Knowledge Base, prior to producing these two guides, and the effort is planned to continue over the rest of GÉANT2.

4 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
ETTR	Estimated Time To Repair
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, a network (path) with a large BDP
MTU	Maximum Transmission Unit
MDT	Multidata Transmit
MSS	maximum segment size
NREN	National Research and Education Network
OS	Operating System

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

4

PAWS	Protection Against Wrapped Sequence-numbers
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
rwnd	Receiver's window
SACK	Selective ACKnowledgments
SCP	Secure Copy
SCTP	Stream Control Transmission Protocol
SMDS	Switched Multimegabit Data Service
SMSS	Sender's maximum Segment Size
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
WLAN	Wireless Local Area Network

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

5

Appendix A Deliverable DS3.3.3 – Part 1 (App. A): Basic Network Performance Guide

Deliverable DS3.3.2 – Part 1

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), Chris Welti (SWITCH)

Abstract

This paper is an basic guide to network performance issues. We present the fundamentals of network performance, explain how to report performance problems in an effective manner, and give information on performance diagnostic tools that can be used without privileged access to network components.

Table of Contents

Table of Figures

1   Performance Basics

User-perceived performance

User-perceived performance of network applications is made up of a number of mainly qualitative criteria, some of which are in conflict with each other. For some applications, a single criterium will outweigh the others, such as *responsiveness* from conversational audio/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.

Responsiveness

One of the most important user experiences in networking applications is the perception of responsiveness, which is to say how quickly an application appears to respond to the user's input. This is a particular issue for real-time applications such as audio/video conferencing systems and must be prioritised in applications such as remote instrument control 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.

However, some applications have relatively well-defined physiological bounds beyond which the feeling of quick response vanishes. For example, for voice conversations, a (round-trip) delay of 150ms is practically unnoticeable, but an increase of just a few more milliseconds can be felt as very intrusive.

Capacity and throughput

Throughput per se is not directly perceived by the user, although a lack of throughput will increase waiting times and reduce the impression of responsiveness. However, "bandwidth" is widely used as a marketing metric to differentiate "fast" connections from "slow" ones, and many applications display throughput during long data transfers. Therefore users often have specific performance expectations in terms of bandwidth, and are disappointed when the actual throughput figures they see are significantly lower than the advertised capacity of their network connection.

Reliability

Reliability is often the most important performance criterion for a user: The application must be available when the user requires it. Note that this doesn't necessarily mean that it is *always* available, although there are some applications - such as the public Web presence of a global corporation - that have "24x7" availability requirements.

The impression of reliability will also be heavily influenced by what happens (or is expected to happen) in cases of unavailability: Does the user have the possibility of a workaround? In case of provider problems: Does the user have someone competent to call - or can they even be sure that the provider will notice the problem themselves, and fix it in due time? How is the user informed during the outage, in particular with regard to the estimated time to repair (ETTR)?

Another aspect of reliability is the *predictability* of performance. It can be profoundly disturbing to a user to see large performance variations over time, even if the varying performance is still within the required performance range - who can guarantee that variations won't increase beyond the tolerable during some other time when the application is needed? A 10 Mbps throughput that remains rock-stable over time can feel more reliable than throughput figures that vary between 200 and 600 Mbps.

Network performance metrics

Another perspective for looking at performance is to focus on measurable properties of the network. There are many metrics that are commonly used to characterize the performance of networks and their constituent parts. Described below are the most important of these metrics, and details on 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 greatest interest in networking has been data rate (commonly referred to as 'bandwidth' or, incorrectly, 'speed'). However, as more and more parts of the Internet have their capacity upgraded, data rate is often no longer the limiting factor.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

10

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

One-Way Delay (OWD)

One-way delay is the time it takes for a packet to reach its destination, and is 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 mainly 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.

Per-link delay components

Propagation delay

Propagation delay is the length of time it takes for signals to move from the transmitting end 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 fibre optic is about two thirds the speed of light in a vacuum (0.66c). The VoP for copper very much depends on the transmission line structure, but gerneally ranges between 0.66c and 0.95c. Note, these values are for the propagation of the signal, not the electrons, which themselves move only very slowly (millimetres per second)

Propagation delay, along with serialization delay and processing delays in nodes such as routers, is a component of overall delay/RTT. For uncongested long-distance network paths, it is usually the dominant component. Therefore, the physical routing of network links 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.

Here are a few examples for *propagation delay* components of one-way and round-trip delays over selected distances in fibre. Clearly for symmetric routing the propagation-related RTT is simply twice the OWD.

Fibre length	One-way delay	Round-trip time
1m	5 ns	10 ns

1km	5 µs	10 µs
10km	50 µs	100 µs
100km	500 µs	1 ms
1000km	5 ms	10 ms
10000km	50 ms	100 ms

Serialization delay

Serialization delay is the time it takes for a unit of data, such as a packet, to be serialized for transmission on a narrow (e.g. serial) channel such as a cable. Serialization delay is dependent on size, which means that longer packets experience longer delays over a given network path. Serialization delay is also dependent on channel capacity ("bandwidth"), which means that for equal-size packets, the faster the link, the lower the serialization delay.

Serialization delays are incurred at processing nodes, when packets are stored-and-copied between links and (router/switch) buffers. This includes the copying over internal links in processing nodes, such as router backplanes/switching fabrics.

In the core of the Internet, serialization delay has largely become a non-issue, because link speeds have increased much faster over the past years than packets sizes. Therefore, the "hopcount" (as shown by e.g. traceroute) is a bad predictor for delay today.

To illustrate the effects of link rates and packet sizes on serialization delay, here is a table of some representative values. Note that the maximum packet size for most computers is 1500 bytes today, but 9000-byte "jumbo frames" are already supported by many research networks.

Link Rate / Packet Size	64 kbps	1 Mbps	10 Mbps	100 Mbps	1 Gbps
64 bytes	8 ms	0.512 ms	51.2 µs	5.12 µs	0.512 µs
512 bytes	64 ms	4.096 ms	409.6 µs	40.96 µs	4.096 µs
1500 bytes	187.5 ms	12 ms	1.2 ms	120 µs	12 µs
9000 bytes	1125 ms	72 ms	7.2 ms	720 µs	72 µs

Other sources of per-link delay

In addition to the propagation and serialization delays, some types of links may introduce additional delays, for example to avoid collisions on a shared link, or when the link-layer transparently retransmits packets that were damaged during transmission.

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.

Depending on router architecture, forwarding can compete for resources with other activities of the router. In this case, packets can be held up until the router's forwarding resource is available, which can take many milliseconds on a router with CPU-based forwarding. Routers with dedicated hardware for forwarding don't have this problem, although for some routers there may be delays when a packet arrives as the hardware forwarding table is being reprogrammed due to a routing change.

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. The amount of queuing that a given packet may encounter is hard to predict, but it is bounded by the buffer size available for queuing (once the queue has grown to consume the whole buffer then any other arriving packets will be "tail dropped").

There can be causes for queuing delay other than contention on the outgoing link, such as contention on the node device's backplane.

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 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 one 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.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

13

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. 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.

Jitter is usually introduced in network nodes (routers), as an effect of queuing or contention for forwarding resources, especially on CPU-based router architectures. Some types of links can also introduce jitter, for example through collision avoidance (shared Ethernet) or link-level retransmission (802.11 wireless LANs).

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.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

14

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).

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.

Packet Reordering

By deliberate design choice the Internet Protocol (IP) does not guarantee that packets are delivered in the order in which they were sent.

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) do not 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 TCP. 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 the 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.

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), 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.

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.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

16

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.

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	Availability Delay Variation (Jitter) Packet Loss Packet Reordering

F**igure .1:** End User and Network Metrics

GE✩NT2

2 Reporting Performance Problems

Leaving aside for the moment the problem of *whom* to turn to, this section gives guidance on *how* to report a performance issue.

The recommendations in this section are derived from experience trying to help users with performance problems, notably things that are easily forgotten and lead to e-mail round-trips. While supporters (such as PERT teams) will simply ask iteratively for information that is missing during an investigation, it is almost always better to include as much information as possible up front, in order to speed up the process and to make documentation of the issue easier. If all relevant information is included in a few e-mails, it also becomes easier to ask a third party for help.

Describe the performance experienced

Try to describe clearly, and if possible quantify, how the application performs. Be clear about metrics and units, in particular when stating bandwidth and throughput measurements (bits vs. bytes per second). State how you did the measurements. Cut-and-pasted text from a terminal session with the actual application run and/or measurements is always appreciated. If such text cannot be provided, application screen-dumps are also useful.

Describe your performance expectations

It is very important to state how your observed performance differed from your expectations, and how far performance should improve until you would consider the problem solved. If performance has regressed from

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

18

past performance, it is very useful to state past performance numbers, if possible in the same manner as described above for current performance. Note that past performance is not sufficient; you should always state your performance requirements.

Describe your endpoints (hosts etc.)

You should provide as much information as possible about the hosts and network attachments participating in the application. Often you will only control one side of the application (e.g. when you complain about downloading performance from a popular Web server), but when you know the people at the other end(s), it is extremely helpful if you can ask them for such information too.

Vital information about an endpoint includes:

Its IP address. On a Unix system, `ifconfig -a` is a good way to provide address information for all interfaces. On a Windows system, `ipconfig` can be used to generate similar information.

The hostname. While less important than the IP address, it makes it easier to refer to systems when communicating.

Operating System information. On Unix systems, `uname -a` provides summary information about the system version. Other systems typically have an information box that will provide OS version information.

Application(s) used, including version information.

The exact *hardware configuration* used is often hard to retrieve, and usually only little of this is relevant. Include what you know and what you think might be important. One aspect that is often important is the type of *Network Interface Card (NIC).*

This brings us back to the *network attachment* (the IP address is part of that, too.) Cut-and-paste output from `traceroute` is always very welcome. If possible, provide traceroutes in both directions, from your host towards the other relevant host(s), and if available, traceroutes back from the other host(s) to yours. In the campus network environment, try to find out about the structure of your local area network (switched network), and the use of devices such as firewalls, bandwidth management/rate shaping tools and similar equipment at the border between the site and the external network. You will typically have to ask your local networking support team, but this kind of information has proven extremely important in the past, and it usually cannot be derived from `traceroute` output.

3 Packet Capture and Analysis

When investigating performance issues, it is often necessary to look at the exact patterns of the traffic exchanged. In the following sub-sections, advice is given on how to best collect packet traces to support this kind of analysis, and what tools can be used to do this.

1.1 General Hints for Taking Packet Traces

1.1.1 Capture enough data

You can always throw away stuff later. Note that tcpdump's default capture length is small (96 bytes), so use `-s 0` (or something like `-s 1540`) if you are interested in payloads (see para 1.2.2). Ethereal and snoop capture entire packets by default. Seemingly unrelated traffic can impact performance. For example, Web pages from `foo.example.com` may load slowly because of the images from `adserver.example.net`. In situations of high background traffic, it may however be necessary to filter out unrelated traffic.

1.1.2 Traces from interesting observation points

It can be extremely useful to collect packet traces from multiple points in the network

- On the endpoints of the communication
- Near "suspicious" intermediate points such as firewalls

Synchronise clocks when collecting packets

Synchronised clocks (e.g. by NTP) are very useful for matching traces. Remember to take into account time zone differences.

1.1.3 Try to reduce clutter in the trace

Address-to-name resolution can slow down display and causes additional traffic which pollutes the trace. With tcpdump, consider using `-n` or tracing to file (`-w file`).

1.1.4 Request remote packet traces

When a packet trace from a remote site is required, this often means having to ask someone at that site to provide it. When requesting such a trace, consider making this as easy as possible for the person having to do it. Try to use a packet tracing tool that is already available - tcpdump for most BSD or Linux systems, snoop for Solaris machines. Windows doesn't seem to come bundled with a packet capturing program, but you can direct the user to ethereal, which is reasonably easy to install and use under Windows. Try to give clean indications on how to call the packet capture program. It is usually best to ask the user to capture to a file, and then send you the capture file as an e-mail attachment or so.

1.2 tcpdump

One of the early diagnostic tools for TCP/IP, `tcpdump` can be used to capture and decode packets in real time, or to capture packets to files (in "libpcap" format, see below), and analyze (decode) them later.

There are now more elaborate and, in some ways, more user-friendly packet capturing programs, such as Ethereal, but `tcpdump` is widely available and widely used, so it is very useful to know how to use it.

Tcpdump/libpcap is still actively being maintained, although not by its original authors.

1.2.1 `libpcap`

Libpcap is a library that is used by `tcpdump`, and also names a *file format* for packet traces. This file format - usually used in files with the extension `.pcap` - is widely supported by packet capture and analysis tools.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

21

1.2.2 Selected options

Some useful options to `tcpdump` include:

- `-s` *snaplen* capture *snaplen* bytes of each frame. By default, `tcpdump` captures only the first 68 bytes, which is sufficient to capture IP/UDP/TCP/ICMP headers, but usually not payload or higher-level protocols. If you are interested in more than just headers, use `-s 0` to capture packets without truncation.
- `-r` *filename* read from an previously created capture file (see `-w`)
- `-w` *filename* dump to a file instead of analyzing on-the-fly
- `-i` *interface* capture on an interface other than the default (first "up" non-loopback interface). Under Linux, `-i any` can be used to capture on *all* interfaces, albeit with some restrictions.
- `-c` *count* stop the capture after *count* packets
- `-n` don't resolve addresses, port numbers, etc. to symbolic names - this avoids additional DNS traffic when analyzing a live capture.
- `-v` verbose output
- `-vv` more verbose output
- `-vvv` even more verbose output

Usage examples

Capture a single (`-c 1`) udp packet to file `test.snoop`:

```
: root@diotima[tmp]; tcpdump -c 1 -w test.pcap udp
tcpdump: listening on bge0, link-type EN10MB (Ethernet), capture size 96 bytes
1 packets captured
3 packets received by filter
0 packets dropped by kernel
```

This produces a binary file containing the captured packet as well as a small file header and a timestamp:

```
: root@diotima[tmp]; ls -l test.pcap
-rw-r--r-- 1 root root 114 2006-04-09 18:57 test.pcap
: root@diotima[tmp]; file test.pcap
test.pcap: tcpdump capture file (big-endian) - version 2.4 (Ethernet, capture
length 96)
```

Analyze the contents of the previously created capture file:

```
: root@diotima[tmp]; tcpdump -r test.pcap
reading from file test.pcap, link-type EN10MB (Ethernet)
18:57:28.732789 2001:630:241:204:211:43ff:fee1:9fe0.32832 > ff3e::beac.10000: UDP,
length: 12
```

Display the same capture file in verbose mode:

```
: root@diotima[tmp]; tcpdump -v -r test.pcap
reading from file test.pcap, link-type EN10MB (Ethernet)
18:57:28.732789 2001:630:241:204:211:43ff:fee1:9fe0.32832 > ff3e::beac.10000: [udp
sum ok] UDP, length: 12 (len 20, hlim 118)
```

1.2.3 References

- tcpdump/libpcap Web site - http://www.tcpdump.org/

1.3   snoop

Sun's Solaris operating system includes a utility called `snoop` for taking and analyzing packet traces. While `snoop` is similar in spirit to tcpdump, its options, filter syntax, and stored file formats are all slightly different. The file format is described in the Informational RFC 1761, and supported by some other tools, notably Ethereal.

1.3.1 Selected options

Useful options in connection with `snoop` include:

- `-i` *filename* read from capture file. Corresponds to tcpdump's `-r` option.
- `-o` *filename* dump to file. Corresponds to tcpdump's `-w` option.
- `-d` *device* capture on *device* rather than on the default (first non-loopback) interface. Corresponds to tcpdump's `-i` option.
- `-c` *count* stop capture after capturing *count* packets (same as for tcpdump)
- `-r` do not resolve addresses to names. Corresponds to tcpdump's `-n` option. This is useful for suppressing DNS traffic when looking at a live trace.
- `-V` verbose summary mode - between (default) summary mode and verbose mode
- `-v` (full) verbose mode

Usage examples

Capture a single (`-c 1`) udp packet to file `test.snoop`:

```
: root@diotima[tmp]; snoop -o test.snoop -c 1 udp
Using device /dev/bge0 (promiscuous mode)
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

23

```
1 1 packets captured
```

This produces a binary file containing the captured packet as well as a small file header and a timestamp:

```
: root@diotima[tmp]; ls -l test.snoop
-rw-r--r-- 1 root root 120 2006-04-09 18:27 test.snoop
: root@diotima[tmp]; file test.snoop
test.snoop:     Snoop capture file - version 2
```

Analyze the contents of the previously created capture file:

```
: root@diotima[tmp]; snoop -i test.snoop
  1     0.00000  2001:690:1fff:1661::3  -> ff1e::1:f00d:beac  UDP  D=10000  S=32820
LEN=20
```

Display the same capture file in "verbose summary" mode:

```
: root@diotima[tmp]; snoop -V -i test.snoop
_____
  1    0.00000 2001:690:1fff:1661::3 -> ff1e::1:f00d:beac ETHER Type=86DD (IPv6),
size = 74 bytes
  1          0.00000   2001:690:1fff:1661::3   ->   ff1e::1:f00d:beac   IPv6
S=2001:690:1fff:1661::3 D=ff1e::1:f00d:beac LEN=20 HOPS=116 CLASS=0x0 FLOW=0x0
  1     0.00000  2001:690:1fff:1661::3 -> ff1e::1:f00d:beac  UDP  D=10000  S=32820
LEN=20
```

Finally, in full verbose mode:

```
: root@diotima[tmp]; snoop -v -i test.snoop
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 1 arrived at 18:27:16.81233
ETHER:  Packet size = 74 bytes
ETHER:  Destination = 33:33:f0:d:be:ac, (multicast)
ETHER:  Source      = 0:a:f3:32:56:0,
ETHER:  Ethertype = 86DD (IPv6)
ETHER:
IPv6:   ----- IPv6 Header -----
IPv6:
IPv6:   Version = 6
IPv6:   Traffic Class = 0
IPv6:   Flow label = 0x0
IPv6:   Payload length = 20
IPv6:   Next Header = 17 (UDP)
IPv6:   Hop Limit = 116
IPv6:   Source address = 2001:690:1fff:1661::3
IPv6:   Destination address = ff1e::1:f00d:beac
IPv6:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 32820
UDP:  Destination port = 10000
UDP:  Length = 20
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

24

```
UDP:  Checksum = 0635
UDP:
```

Note: the captured packet is an IPv6 multicast packet from a "beacon" system. Such traffic forms the majority of the background load on our office LAN at the moment.

1.3.2 References

- [RFC 1761](#), *Snoop Version 2 Packet Capture File Format*, B. Callaghan, R. Gilligan, February 1995
- [snoop(1) manual page](#) *for Solaris 10*, April 2004, `docs.sun.com`

1.4   Ethereal

Ethereal is a packet capture/analysis tool, similar to [tcpdump](#) but much more elaborate. It has a graphical user interface (GUI) which allows "drilling down" into the header structure of captured packets. In addition, it has a "plugin" architecture that allows decoders ("dissectors" in Ethereal terminology) to be written with relative ease. This and the general user-friendliness of the tool has resulted in Ethereal supporting an abundance of network protocols. Lastly, Ethereal includes some useful graphical analysis/statictics tools such as `tcptrace` and `xplot`.

One of the main attractions of Ethereal is that it works well under Microsoft Windows (although it requires a third-party library to implement the equivalent of the `libpcap` packet capture library).

1.4.1 Usage examples

The following screenshot shows Ethereal 0.10.14 under Linux/Gnome when called as `ethereal -r test.pcap`, reading the single-packet example trace generated in the [tcpdump](#) example. The "data" portion of the UDP part of the packet has been selected by clicking on it in the middle pane, and the corresponding bytes are highlighted in the lower pane.

File Edit View Go Capture Analyze Statistics Help

Filter: ▼   ✚ Expression...   ✖ Clear   ✔ Apply
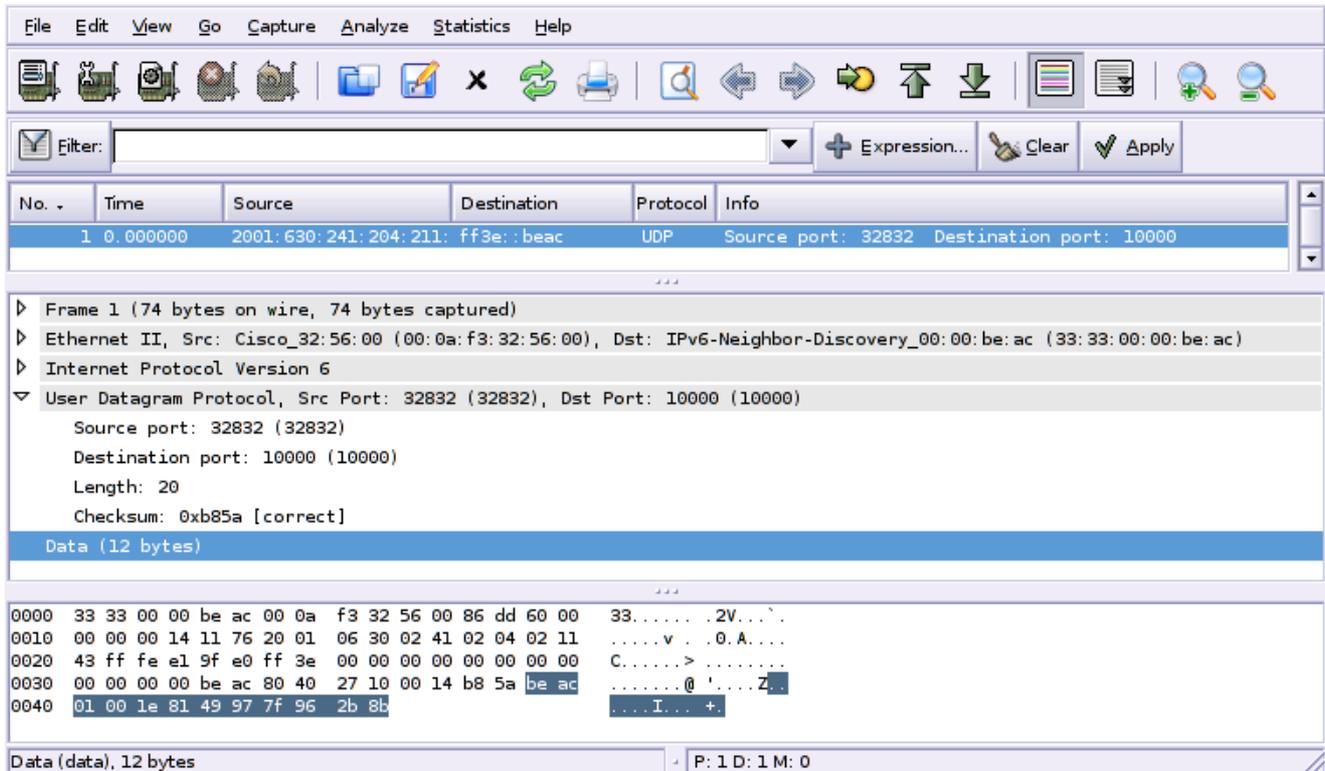
No. ▾	Time	Source	Destination	Protocol	Info
1	0.000000	2001:630:241:204:211:ff3e::beac		UDP	Source port: 32832 Destination port: 10000

▷ Frame 1 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: Cisco_32:56:00 (00:0a:f3:32:56:00), Dst: IPv6-Neighbor-Discovery_00:00:be:ac (33:33:00:00:be:ac)
▷ Internet Protocol Version 6
▽ User Datagram Protocol, Src Port: 32832 (32832), Dst Port: 10000 (10000)
    Source port: 32832 (32832)
    Destination port: 10000 (10000)
    Length: 20
    Checksum: 0xb85a [correct]
  Data (12 bytes)

```
0000  33 33 00 00 be ac 00 0a  f3 32 56 00 86 dd 60 00   33...... .2V...`.
0010  00 00 00 14 11 76 20 01  06 30 02 41 02 04 02 11   .....v . .0.A....
0020  43 ff fe e1 9f e0 ff 3e  00 00 00 00 00 00 00 00   C......> ........
0030  00 00 00 00 be ac 80 40  27 10 00 14 b8 5a be ac   .......@ '....Z..
0040  01 00 1e 81 49 97 7f 96  2b 8b                     ....I... +.
```

Data (data), 12 bytes   P: 1 D: 1 M: 0

1.4.2 References

- Ethereal Web page - http://www.ethereal.com/

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

26

4   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 to 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.

Window-based transmission

TCP is a sliding-window protocol. The size of the TCP window at any given time is effectively the minimum of three other separate 'windows' – the sender's buffer, the advertised receiver's window (declared in TCP header field "window" and in TCP terminology called 'rwnd'), and the sender's congestion window (called 'cwnd').

The sender can transmit an amount of date up to the size of the window before having to wait for an acknowledgement from the receiver – put another way, the amount of data in transit in the network (that is to say, unacknowledged by their receiver) should be no greater than the size of the window. 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 received the associated segment(s) are "deleted" from the window and the freed up space used for the next segment(s) to be sent.

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 restrict the network performance to less than expected and a too large window may lead to problems when recovering from error (quite apart from adversely affecting neighbouring flows).

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.
```

1.5   Congestion Control

One of TCP's important tasks is to regulate the rate at which data flows according to the state of both the endpoints and of the network. In particular, TCP senses congestion in the network – typically by observing packet loss, but also queuing-induced delay – and reacts by lowering its data rate. When congestion subsides, TCP will increase the rate again.

The details of this *congestion control* mechanism are important for understanding TCP performance, in particular in the presence of congestion and other kinds of network impediments such as packet loss due to transmission errors or delay variations. However, this is such a vast area that we defer the discussion to the Advanced Guide.

5 Measurement tools

A range of test, measurement and troubleshooting tools exist for investigating factors contributing to the performance issues which have been described 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 and identified as useful for end-users. Some tools provide multiple functions and are listed under their most common use. Some tools are also more likely to be used by campus or NREN administrators.

Latency/Delay Measurement Tools

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
```

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

29

```
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: Example IPv4 ping output

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 de-prioritise 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 of successive packets 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 most Unix implementations) is exceeded before the port unreachable is received. When this happens an "!" will be printed beside the RTT in the output.
Note that Unix implementations of traceroute send UDP probe packets whilst MS Windows tracert sends ICMP echo probes. Unix 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.ie1.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 .3: Example IPv4 traceroute output

Why UDP?

It would seem natural to use ICMP ECHO requests as probe packets, but Van Jacobson chose UDP packets instead. It is believed that this was because at that time, some gateways (as routers were called then) refused to send ICMP (TTL exceeded) messages in response to ICMP messages, as specified in the introduction of RFC 792, "Internet Control Message Protocol". Therefore the UDP variant was more robust.

These days, all gateways (routers) send ICMP TTL Exceeded messages about ICMP ECHO request packets (as specified in RFC1122, "Requirements for Internet Hosts -- Communication Layers", so more recent traceroute versions (such as Windows `tracert`) do indeed use ICMP probes, and newer Unix traceroute versions allow ICMP probes to be selected with the `-I` option.

Why increment the UDP port number?

Traceroute varies (increments) the UDP destination port number for each probe sent out, in order to reliably match ICMP TTL Exceeded messages to individual probes. Because the UDP ports occur right after the IP header, they can be relied on to be included in the "original packet" portion of the ICMP TTL Exceeded messages, since the ICMP standards require that at least the first eight octets following the IP header of the original packet must be included in ICMP messages.

When ICMP ECHO requests are used, the responses can be distinguished from one another by using the sequence number field, which also happens to be located just before that 8-octet boundary.
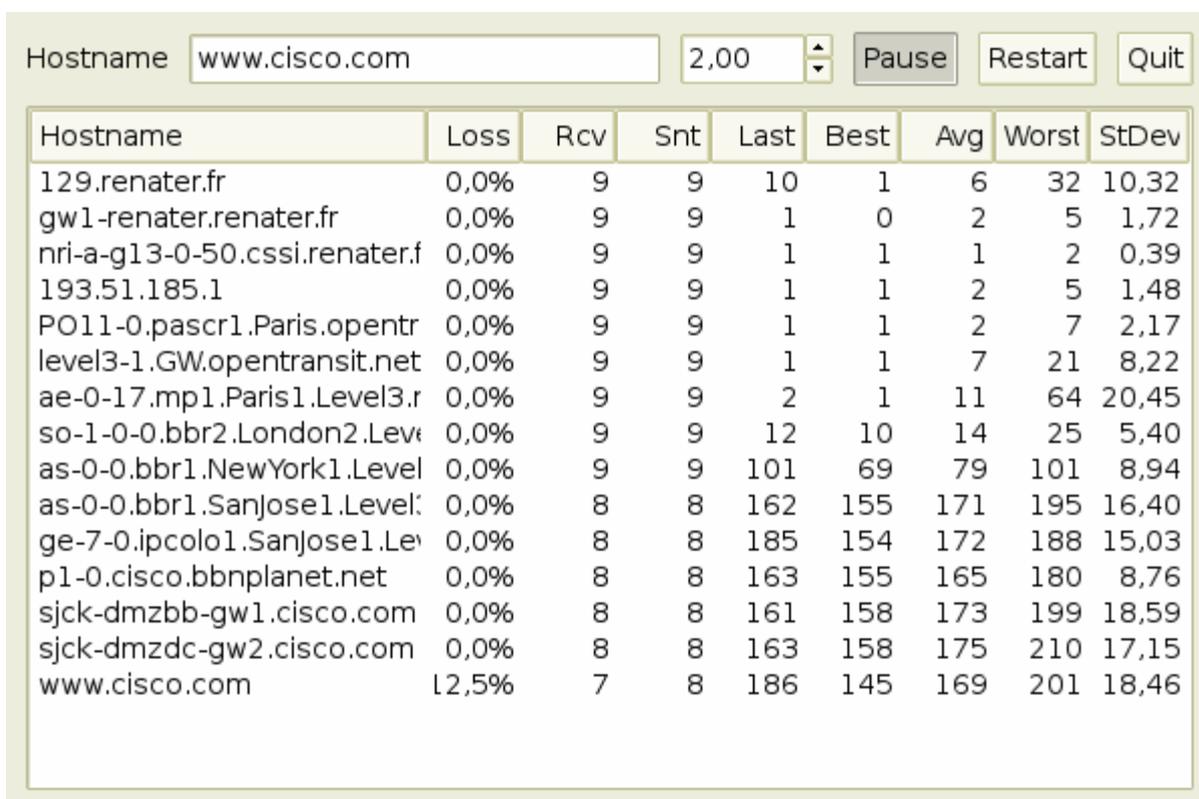
Traceroute Variants

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. Some of these variants try to improve on traceroute's results presentation, others measure different aspects of the path, while others still use different "stimulus" traffic, in order to be more useful in today's Internet where filters prevail.

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

31

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:

Figure .4: Example MTR output

Hostname	Loss	Rcv	Snt	Last	Best	Avg	Worst	StDev
Hostname	www.cisco.com	2,00	Pause	Restart	Quit			
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.Leve	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

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-g13-0-50.cssi.renater.fr (193.51.182.6) 1.0ms
4   193.51.185.1 0.6ms
5   PO11-0.pascr1.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
11   ge-7-0.ipcolo1.SanJose1.Level3.net (4.68.123.9) 159.2ms
12   p1-0.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 .5: Example LFT output

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

tracepath

Tracepath and tracepath6 trace the path to a network host, discovering the MTU along this path. Note that, as explained at http://www.linuxmanpages.com/man8/tracepath.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# tracepath 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-g13-0-50.cssi.renater.fr (193.51.182.6)     0.767ms
4:  193.51.185.1 (193.51.185.1)                       0.763ms
5:  PO11-0.pascr1.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.ipcolo1.SanJose1.Level3.net (4.68.123.169)  asymm 10 158.088ms
12:  p1-0.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 .6: Example tracepath output

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

33

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.

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/.

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.

Example NDT output

A test takes about 20 seconds. Click on "start" to begin.
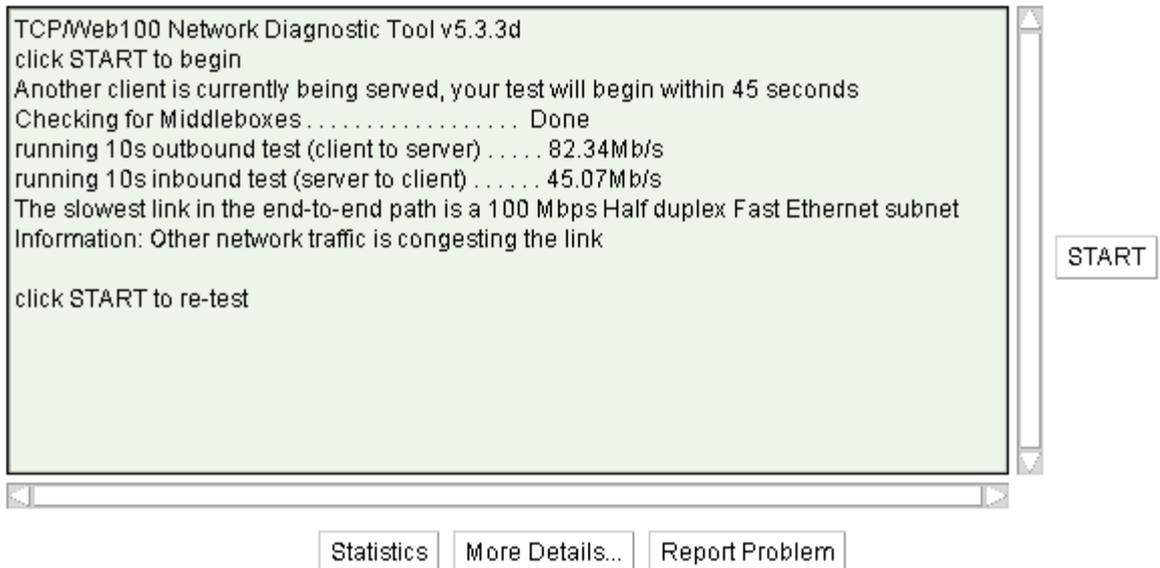
```
TCP/Web100 Network Diagnostic Tool v5.3.3d
click START to begin
Another client is currently being served, your test will begin within 45 seconds
Checking for Middleboxes . . . . . . . . . . . . . . . . . . Done
running 10s outbound test (client to server) . . . . . 82.34Mb/s
running 10s inbound test (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

6 Differentiated Service over GÉANT

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 GÉANT/GN2 and some NRENs.

Premium IP

Premium IP (PIP) 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. By restricting the total amount of allowed, concurrent PIP traffic on any given link (by requiring reservations for PIP to be made in advance) One Way Delay, Delay Variation and Packet Loss are assured to be low. The aggregate is specified by source and destination IP address ranges. In addition, a Premium IP aggregate must conform to strict rate limits. If a PIP flow exceeds its contracted rate then the excess is dropped.

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.

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).

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

35

If the network becomes congested, Premium IP and Best Effort traffic will take precedence – LBE packets will always 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

M. Allman, V. Paxson, W. Stevens

RFC 2581, *TCP Congestion Control*, ftp://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, 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:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

37

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

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

38

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.mspx#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.mspx,

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.mspx

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

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

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

39

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

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/

SysKonnect,   *SK-NET GE Gigabit Ethernet Server Adapter,* http://www.syskonnect.com/syskonnect/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. Cimbal   *Achieving Reliable High Performance in LFNs*,
http://staff.cesnet.cz/~ubik/publications/2003/terena2003.pdf, May 2003

S. Ubik, P. Cimbal   *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/

Project:	GN2
Deliverable Number:	DS3.3.2
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2

40

Appendix B Deliverable DS3.3.3 – Part 2 (App. B): Advanced Network Performance Guide

Table of Contents

Table of Figures

1 First steps at investigating performance problems

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. For information on network metrics and basic troubleshooting tools refer to the Basic Network Performance Guide.

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.

It is best if these performance experiences are expressed in quantitative 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 Mbps, compared against an expected transfer rate, expressed in the same way.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

44

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 (NICs) 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 make some comparative measurements from other systems with similar paths. In particular the following should be considered:

- 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 check for the latest driver software and bug announcements.

- Before looking for causes of poor network performance on the network one should ensure that the end system is behaving well. The best way to do this is a lab performance test with traffic generators and analysers. Of course, normally such an environment is not available in which case it is useful to connect the end system under test back to back with 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.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

45

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 possible 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.

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 of particular importance in any problem report:

Problem Description

The current system behaviour that is unsatisfactory needs to be described in terms of network metrics. 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.

Network Details

Although the network may not appear to be the responsibility of the end user, certain networking information is most easily 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 at all possible, a return traceroute from the remote host to the local host should also be included. 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 should be included.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

46

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 using the relevant ports should be taken and attached to the problem report.

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, this should also be included.

2   Advanced TCP Topics

The Basic Network Performance Guide gives an introduction to how TCP works, and describes some of the performance limitations caused by its window-based flow control. The following sections delve deeper into TCP's performance properties, and looks at some of the many enhancements that have been introduced over the years to make TCP work in a wide range of environments, in particular those that are relevant to the typical research networking environment of "LFNs" (Long Fat Networks).

1.6   Rate control

TCP flow control and window size adjustment is mainly based on two key mechanisms: Slow Start and Congestion Avoidance. (RFC 793 and RFC 2581)

Slow Start

To prevent a new TCP connection from flooding a network with traffic, a Slow Start mechanism was introduced in TCP. This mechanism effectively probes to find (and use) the network's available bandwidth.

In addition to the window advertised by the receiver, a Congestion Window (cwnd) value is considered 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). 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. When congestion is detected, the TCP flow-control mode is changed from Slow Start to Congestion Avoidance.

Congestion Avoidance

Once congestion is detected (through timeout and/or duplicate ACKs), the data rate is reduced in order to let the network recover. This is initially done by reducing the size of cwnd. The rate at which cwnd then grows depends on whether TCP falls back to Slow Start mode or whether it goes into Congestion Avoidance mode. Slow Start uses an exponential increase in window size and thus also in data rate. Congestion Avoidance uses a linear growth function (additive increase). To determine whether Slow Start or Congestion Avoidance applies, the slow start threshold (ssthresh) value is checked.

As long as the new cwnd value (post packet loss) is less than ssthresh, TCP operates in Slow Start. Once ssthresh is reached, TCP enters Congestion Avoidance mode and 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 current (pre packet loss) cwnd value. 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.

Fast Retransmit and Fast Recovery

Fast Recovery and Fast Retransmit (which are normally implemented together) are used to react quickly to a single packet loss. 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 Retransmit, the arrival of three duplicate ACKs is assumed to mean there has been packet loss, and retransmission starts before the retransmission timer expires. This improves performance by eliminating delays that would suspend effective data flow on the link. Because TCP assumes that any segment loss (even just one segment) indicates network congestion, ssthresh is set to half the current cwnd value.

Fast Recovery immediately follows Fast Retransmit. cwnd is set to ssthresh plus three SMSS (sender maximum segment size). Each additional duplicate ACK indicates that one segment has left the network at the receiver and so cwnd is increased by one segment to allow the transmission of another segment (if allowed by the new cwnd). When an ACK is received for new data, cwmd is reset to the ssthresh, and TCP enters congestion avoidance mode. This second reduction in cwnd is termed "deflating the window".

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

49

PERT Performance Guides
Appendix B – Advanced Network Performance Guide

TCP Performance enhancements

Over the years TCP has been enhanced with a number of extensions so that it can be used effectively over modern networks. These extensions are specified in various RFCs and are supported by most contemporary TCP stacks, although they frequently must be activated explicitly or implicitly by configuring Large TCP Windows.

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 WINDOW_SIZE/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.

The TCP window scaling option increases the maximum window size from 64KB to 1GB by shifting the window field left by up to 14 binary places (64KB * 2^14 = 1GB). The window scale option is used only during the TCP 3-way handshake (both sides send the window scale option in their SYN segments).

When Window Scaling is used, it is important to use also the TCP timestamps option. With the TCP timestamps option, each segment contains a timestamp. The receiver returns that timestamp in each ACK and this provides two benefits. First, it allows Round Trip Time Measurement (RTTM) which means the sender can choose an accurate Retransmit Timeout (RTO) value. Second, it enables Protection Against Wrapped Sequences (PAWS) which avoids the risk that re-ordered packets with the same sequence number are confused with one another..

Although large TCP windows are required for efficient use of LFNs, 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 needs to reside in actual RAM and cannot be paged out to disk) can starve other processes of access to fast memory (cache and RAM)

Project: GN2
Deliverable Number: DS3.3.3
Date of Issue: 23/08/06
EC Contract No.:   511082
Document Code: GN2-06-135v2v2

50

- 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 behaviour 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.

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.

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`.

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 should contact the NREN and the PERT as appropriate.

Explicit Congestion Notification

A TCP sender has traditionally had to infer network congestion from packet loss and slow responses (ACKs) from the receiver. A new alternative to these implicit congestion signals is Explicit Congestion Notification. (ECN). ECN makes use of two new flags in the TCP header and two unused bits in the old TCP TOS field.

The two new ECN bits[3] in the former TOS field of the IP header are the "ECN-Capable Transport" (ECT) bit (must only be set for packets controlled by ECN-aware transports) and 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).

The two new single bit flags in the TCP header are called ECN-Echo (ECE) and Congestion Window Reduced (CWR). They are used by the TCP receiver to communicate the ECN signal back to the sender.

ECN works as follows. 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 ECE flag in the case of TCP). The sender then reduces its sending rate, as if it had detected the loss of a single packet, and sets the CWR flag so as to inform the receiver of this action.

ECN provides two significant benefits. First, ECN-aware transports can properly adapt their rates to congestion without requiring packet loss. Second, congestion feedback can be quicker with ECN, because detecting a dropped packet requires a timeout.

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 this has not 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 should contact the NREN and the PERT as appropriate.

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

---

[3] 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. The original definition as presented here is somewhat easier to understand but I you want to know how these "conceptual" bits are encoded, please read RFC 3168.

mechanism sees 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 queuing/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.

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.

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/.

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

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

53

GE⋆NT2

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. Bullot, 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/

3   Hardware considerations

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

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.

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), the operating system's TCP can give to the NIC a segment that is bigger than the MTU supported by the medium. Intelligent adapters implement 'large sends' by carving up the over-sized segment in to smaller segments that meet the media's MTU. The headers for these TCP packets are based on the original over-sized segments, but use newly calculated sequence numbers and checksums   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.mspx.

Interrupt Coalescence

Traditionally, a network adapter generates an interrupt for each frame that it receives so a common bottleneck for high-speed data transfers is the high rate of interrupts that the receiving system has to process. 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 then issue one interrupt for the whole group.

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 this does not 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.

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. The Checksum Offload function is included TOEs,, and like TOEs, adapters with Checksum Offload require special driver support and a kernel infrastructure that supports such drivers.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

56

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.

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.

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/.

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.

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:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

57

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.

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.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

58

4   Operating system considerations

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 reappear 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.

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.

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:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

59

```
[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 .1: Microsoft Windows TCP Tuning 1

 Note that the key need only be added to one of the locations above, not both.

Inquiries made to Microsoft have 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 .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.

Linux

A comprehensive guide to TCP Tuning Guide on Linux is available at http://www-didc.lbl.gov/TCP-tuning/linux.html. The most 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
```

```
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: Linux TCP Tuning 1

Note if you have a server with hundreds of connections, you might not want to use such 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 .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. To change txqueuelen use the ifconfig command as follows.

```
ifconfig eth0 txqueuelen 1000
```

Figure .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. This can be addressed by flushing the route cache.

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

Figure .6: Linux TCP Tuning 4

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.

MAC OS X

As Mac OS X is mainly a BSD derivative, you can use similar mechanisms to tune the TCP stack. Like other UNIX derived OSes temporary changes, can be made using `sysctl` at the command line (root privileges are required to do this). For Mac OS X the commands would be as follows:

```
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 .7: MAC OSX TCP Tuning

Users who are unfamiliar with the command line interface 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.

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:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

62

5   Application and protocol considerations

Designing tolerant applications

A popular use of modern high performance networks is for real-time media applications such as audio/video conferencing tools. Delay variation is often an issue for such applications and most employ a Jitter Buffer to eliminate these effects. Real-time applications often experience problems when operated on networks that reorder packets, even when they use jitter buffers. It appears that in these cases the code that manages these jitter buffers is often not written in a way to accommodate reordered packets sensibly. It is to be hoped that future applications improve upon this should not be a complex problem to solve.

One particular kind of packet reordering concerns packets of different sizes. Larger packets may be "overtaken" by smaller packets and so Audio/Video applications should wherever possible avoid sending a mix of large and small packets.

"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 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.

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 "."
```

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

64

```
   ...
  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-around.

Performance-friendly I/O interfaces

For applications with high input/output performance requirements (including network I/O),developers should the operating system support available for efficient I/O routines.

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);
}
```

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.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

65

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);
}
```

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.

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. Note that it is often worth investigating if optimised versions of standard software packages are available or if packages can be tuned for performance.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

66

Protocols

TCP

HS-TCP, H-TCP, BIC-TCP and FAST are all TCP variations optimised for performance (see above – "TCP Performance Enhancements"). 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.

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. Note that UDP can perform badly in congested networks.

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.

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.

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. http://www.psc.edu/networking/projects/hpn-ssh/

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/.

Applications

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.

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.

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:

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

68

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

Figure .8: 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.

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. [PERT-KB-APACHE] is a case study in which Apache 2.x is tuned to cope with many simultaneous connections.

6   Performance Pitfalls

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 Mbps 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

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

70

duplex mode (and also to detect whether the stations support 100 Mbps at all - otherwise communication would fall back to traditional 10 Mbps Ethernet). Gigabit Ethernet over twisted pair (1000BaseTX) had speed, duplex, and even "crossed-cable" (MDX) auto-negotiation from the start.

Reasons for disabling auto-negotiation

Unfortunately, some early products supporting Fast Ethernet did not include the auto-negotiation mechanism, and those that did sometimes failed to interoperate with each other. As a result it was often recommended to disable duplex-auto-negotiation, as 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 disabling auto-negotiation

There are two main problems with disabling 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 connections 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 enabled

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.

LAN Collisions

In some legacy networks, workstations or other devices may still be connected using a network of hubs. All incoming and outgoing traffic is propagated throughout the network of hubs, 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.

To prevent collisions from traveling to every workstation in the entire network, bridges or switches should be installed. 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, such that there are never any

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

71

collisions. This is the normal mode of operation for modern networks and is required for any high speed application/

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. This can 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 to do using physically separate switches, or it may be done using 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.

Path MTU Discovery Issues

The Path Maximum Transfer Unit (MTU) is defined as the minimum of the MTUs of the links that make up the path in question. Large 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 intermediate routers back to the source host that initiated the Path MTU discovery. However, such messages often do not get back to the source host because either routers do not generate them, or they get dropped on the way back to the source, because of either 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 rather on the packet exchange between the end hosts. Implementations are available for Linux 2.6 (http://www.psc.edu/~jheffner/projects/mtup/) and NetBSD (http://www.patheticgeek.net/~kml/mmtu/).

Until the new mechanism is widely deployed, network administrators should be careful not to block 'ICMP - fragmentation needed' packets at LAN to WAN borders, otherwise hosts with the current PMTU Discovery enabled would be hampered by the local network. To avoid the risk of PMTUD leading to failures network administrators should endeavour to make sure that the MTU of a given end-system is going to be no greater than the Path MTU. Normally setting a n MTU of 1500 bytes is safe but care should be taken when configuring tunnels or offering dial in services. In such cases network administrators should calculate what the additional overhead will be and then reduce the MTU on those connections. Good practise is to prevent fragmentation where possible. This means here should be an agreement with the upstream network provider as to what the access link MTU will be and then ensure that campus systems do not exceed this limit.

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

Middleboxes

A middle box is a device, other than a router or switch, through which traffic passes. Typically they are found on the Campus LAN (at the backbone border) or within a LAN. From the Campus LAN point of view they can provide several important functions. Middle boxes include:

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 middlebox devices that disturb the normal end-to-end traffic in some way are termed "evil middlboxes". As they are often hard to detect (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 known only to 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)

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

73

Analysis Summary: HTTP traffic is split between the webserver and a transparent HTTP proxy on the user 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 user accessing it and to the user 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 user 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 user who asked for it. Of course the user 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.

7   Measurement Tools

7.1   Reachability and Round Trip Times

7.1.1 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.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

75

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.

7.1.2 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.

7.1.3 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 ^]') then 'quit'.

7.2   Path Characteristics

7.2.1 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:

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

76

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.

7.2.2 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 traceroute6. This option was more interesting in IPv4, where bugs in some implementations of the IP stack could be identified 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.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

77

7.2.3 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

7.2.4 Traceroute like tools

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

7.3   iperf – A Bandwidth Measurement Tool

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.

Iperf is an active measurement tool designed for measuring network characteristics as seen at the transport layer, either TCP or UDP. Iperf reports bandwidth (throughput), delay, jitter, and datagram loss.

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.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

78

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

7.3.1 Issues with iperf

7.3.1.1 *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.

7.3.1.2 *TCP buffer allocation*

On Linux systems, if you request a specific TCP buffer size with the "-w" option, the kernel will always try to allocate double as much bytes as you specified.

Example: when you request 2MB window size you'll receive 4MB:

```
welti@mamp1:~$ iperf -c ezmp3 -w 2M -i 1
------------------------------------------------------------
Client connecting to ezmp3, TCP port 5001
TCP window size: 4.00 MByte (WARNING: requested 2.00 MByte)    <<<<<<
------------------------------------------------------------
```

7.3.1.3 *Counter overflow*

Some versions seem to suffer from a 32-bit integer overflow which will lead to wrong results.

e.g.:

```
[ 14]  0.0-10.0 sec   953315416 Bytes   762652333 bits/sec
[ 14] 10.0-20.0 sec  1173758936 Bytes   939007149 bits/sec
[ 14] 20.0-30.0 sec  1173783552 Bytes   939026842 bits/sec
[ 14] 30.0-40.0 sec  1173769072 Bytes   939015258 bits/sec
[ 14] 40.0-50.0 sec  1173783552 Bytes   939026842 bits/sec
[ 14] 50.0-60.0 sec  1173751696 Bytes   939001357 bits/sec
[ 14]  0.0-60.0 sec  2531115008 Bytes   337294201 bits/sec
[ 14] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
```

As you can see the summary 0-60 seconds doesn't match the average that one would expect. This is due to the fact that the total number of Bytes is not correct as a result of a counter wrap.

If you're experiencing this kind of effects, upgrade to the latest version of iperf, which should have this bug fixed.

7.3.2 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 to need of 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).

7.3.3 Examples

The following examples are from an iperf run between two machines with 10GE (10 Gigabit Ethernet) adapters, connected over a 10 Gbps WAN.

7.3.3.1 *TCP Throughput Test*

The following shows a TCP throughput test, which is `iperf`'s default action. The following options are given:

- `-s` - *server* mode. In `iperf`, the server will *receive* the test data stream.
- `-c` *server* - *client* mode. The name (or IP address) of the server should be given. The client will *transmit* the test stream.
- `-i` *interval* - display interval. Without this option, `iperf` will run the test silently, and only write a summary after the test has finished. With `-i`, the program will report intermediate results at given intervals (in seconds).
- `-w` *windowsize* - select a non-default TCP window size. To achieve high rates over paths with a large bandwidth-delay product, it is often necessary to select a larger TCP window size than the (operating system) default.

The "-i 1" option was given to obtain intermediate reports every second, in addition to the final report at the end of the ten-second test. The TCP buffer size was set to 2 Megabytes (4 Megabytes effective, see below) in order

to permit close to line-rate transfers. The systems haven't been fully tuned, otherwise up to 7 Gbps of TCP throughput should be possible. Normal background traffic on the 10 Gbps backbone is on the order of 30-100 Mbps. Note that in iperf, by default it is the *client* that transmits to the *server*.

Server Side:

```
welti@ezmp3:~$ iperf -s -w 2M -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 4.00 MByte (WARNING: requested 2.00 MByte)
------------------------------------------------------------
[  4] local 130.59.35.106 port 5001 connected with 130.59.35.82 port 41143
[  4]  0.0- 1.0 sec    405 MBytes  3.40 Gbits/sec
[  4]  1.0- 2.0 sec    424 MBytes  3.56 Gbits/sec
[  4]  2.0- 3.0 sec    425 MBytes  3.56 Gbits/sec
[  4]  3.0- 4.0 sec    422 MBytes  3.54 Gbits/sec
[  4]  4.0- 5.0 sec    424 MBytes  3.56 Gbits/sec
[  4]  5.0- 6.0 sec    422 MBytes  3.54 Gbits/sec
[  4]  6.0- 7.0 sec    424 MBytes  3.56 Gbits/sec
[  4]  7.0- 8.0 sec    423 MBytes  3.55 Gbits/sec
[  4]  8.0- 9.0 sec    424 MBytes  3.56 Gbits/sec
[  4]  9.0-10.0 sec    413 MBytes  3.47 Gbits/sec
[  4]  0.0-10.0 sec  4.11 GBytes  3.53 Gbits/sec
```

Client Side:

```
welti@mamp1:~$ iperf -c ezmp3 -w 2M -i 1
------------------------------------------------------------
Client connecting to ezmp3, TCP port 5001
TCP window size: 4.00 MByte (WARNING: requested 2.00 MByte)
------------------------------------------------------------
[  3] local 130.59.35.82 port 41143 connected with 130.59.35.106 port 5001
[  3]  0.0- 1.0 sec    405 MBytes  3.40 Gbits/sec
[  3]  1.0- 2.0 sec    424 MBytes  3.56 Gbits/sec
[  3]  2.0- 3.0 sec    425 MBytes  3.56 Gbits/sec
[  3]  3.0- 4.0 sec    422 MBytes  3.54 Gbits/sec
[  3]  4.0- 5.0 sec    424 MBytes  3.56 Gbits/sec
[  3]  5.0- 6.0 sec    422 MBytes  3.54 Gbits/sec
[  3]  6.0- 7.0 sec    424 MBytes  3.56 Gbits/sec
[  3]  7.0- 8.0 sec    423 MBytes  3.55 Gbits/sec
[  3]  8.0- 9.0 sec    424 MBytes  3.56 Gbits/sec
[  3]  0.0-10.0 sec  4.11 GBytes  3.53 Gbits/sec
```

7.3.3.2 *UDP Test*

In the following example, we send a 300 Mbps UDP test stream. No packets were lost along the path, although one arrived out-of-order. Another interesting result is jitter, which is displayed as 27 or 28 microseconds (apparently there is some rounding error or other impreciseness that prevents the client and server from agreeing on the value). According to the documentation, "Jitter is the smoothed mean of differences between consecutive transit times."

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

81

Server Side

```
: leinen@mamp1[leinen]; iperf -s -u
-------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
-------------------------------------------------------------
[  3] local 130.59.35.82 port 5001 connected with 130.59.35.106 port 38750
[  3]  0.0-10.0 sec    359 MBytes    302 Mbits/sec  0.028 ms    0/256410 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

Client Side

```
: leinen@ezmp3[leinen]; iperf -c mamp1-eth0 -u -b 300M
-------------------------------------------------------------
Client connecting to mamp1-eth0, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
-------------------------------------------------------------
[  3] local 130.59.35.106 port 38750 connected with 130.59.35.82 port 5001
[  3]  0.0-10.0 sec    359 MBytes    302 Mbits/sec
[  3] Sent 256411 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec    359 MBytes    302 Mbits/sec  0.027 ms    0/256410 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

7.3.4 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 Mbit/s * 0.150s / 8 = 1875000 bytes, so setting the TCP window to a value of 2 MBytes would be a good choice.

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

82

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 can be used in UDP mode for inferring 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 no packet loss on the end-to-end path, (for non-high speed applications packet loss of up to 1% can be tolerated). 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 it their measurements should not be directly compared. Iperf sends UDP datagrams are a constant steady rate, whereas TPC 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.

7.3.5 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/.

7.3.5.1   *Example*

A typical BWCTL result looks like one of the two print outs below. The first print out shows a test run **from** the localhost **to** 193.136.3.155 ('-c' stands for 'collector'). The second print out shows a test run **to** the localhost **from** 193.136.3.155 ('-s' stands for 'sender').

```
[user@ws4 user]# bwctl -c 193.136.3.155
bwctl: 17 seconds until test results available
RECEIVER START
3339497760.433479: iperf -B 193.136.3.155 -P 1 -s -f b -m -p 5001 -t 10
-------------------------------------------------------------
Server listening on TCP port 5001
Binding to local address 193.136.3.155
TCP window size: 65536 Byte (default)
-------------------------------------------------------------
[ 14] local 193.136.3.155 port 5001 connected with 62.40.108.82 port 35360
[ 14]  0.0-10.0 sec  16965632 Bytes  13547803 bits/sec
[ 14] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
RECEIVER END


[user@ws4 user]# bwctl -s 193.136.3.155
bwctl: 17 seconds until test results available
RECEIVER START
3339497801.610690: iperf -B 62.40.108.82 -P 1 -s -f b -m -p 5004 -t 10
-------------------------------------------------------------
Server listening on TCP port 5004
Binding to local address 62.40.108.82
TCP window size: 87380 Byte (default)
-------------------------------------------------------------
[ 16] local 62.40.108.82 port 5004 connected with 193.136.3.155 port 5004
[ ID] Interval        Transfer      Bandwidth
[ 16]  0.0-10.0 sec  8298496 Bytes  6622281 bits/sec
[ 16] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
[ 16] Read lengths occurring in more than 5% of reads:
[ 16]  1448 bytes read  5708 times (99.8%)
RECEIVER END
```

The read lengths are shown when the test is for received traffic. Stanislav Shalunov of Internet2 writes:

> "The values are produced by Iperf and reproduced by BWCTL.

> An Iperf server gives you several most frequent values that the read() system call returned during a TCP test, along with the percentages of all read() calls for which the given read() lengths accounted.

> This can let you judge the efficiency of interrupt coalescence implementations, kernel scheduling strategies and other such esoteric applesauce. Basically, reads shorter than 8kB can put quite a bit of load on the CPU (each read() call involves several microseconds of context switching). If the target rate is beyond a few hundred megabits per second, one would need to pay attention to read() lengths."

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

84

7.3.6 Pchar – An alternative to iperf

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. Therefore, the measured estimates can indicate problems, but one should it to measure performance bottlenecks. The tool is available at the Pchar Home page [Pchar].

Example

```
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
```

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

85

7.4   Advanced Measurement Tools

7.4.1 NDT Server

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. Use of the NDT client is described in the Basic Network Performance Guide.

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]

8 References

[PERT-KB]   PERT Knowledge Base

http:// pace.geant2.net /cgi-bin/twiki/view/PERTKB/WebHome

[PERT-KB-APACHE]   PERT Knowledge Base Apache case study

http://pace.geant2.net/cgi-bin/twiki/view/PERTKB/ApacheScaling

[User-Guide]   Research Network User 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.sh
tml/

[NDT]   Rich Carlson, Network Diagnostic Tester,

http://e2epi.internet2.edu/ndt

[DFN-IPPM]   DFN IPPM active measurment 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]   Smokeping Home Page:

http://people.ee.ethz.ch/~oetiker/webtools/smokeping/index.en.html

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

87

[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

Project:	GN2
Deliverable Number:	DS3.3.3
Date of Issue:	23/08/06
EC Contract No.:	511082
Document Code:	GN2-06-135v2v2

88