



An Introduction to Computer Network Monitoring and Performance

A.C. Davenhall & M.J. Lease

December 2005
Version 1

Contact Details

A.C. Davenhall, National e-Science Centre,
M.J. Leese, CCLRC Daresbury Laboratory.

Correspondence about this document should be sent to:

A.C. Davenhall, National e-Science Centre, 15, South College Street, Edinburgh, EH8 9AA,
UK. E-mail: clive@nesc.ac.uk

Purpose and Scope

This document is an introduction to computer networks and to ways of obtaining good performance from them. It covers:

- the basics of computer networks,
- how to monitor their performance,
- simple techniques to improve their performance.

It also describes some of the common protocols used to transfer files across networks. These topics are all large and complex subjects, and the document is only an introduction to them. However, references to more advanced and specialised sources of information are included.

Target Audience

The document is aimed at people who are not network experts, but who need to use networks and to obtain good performance from them. No previous experience of computer networking is assumed. People who are already knowledgeable about computer networks are less likely to find the introductory sections useful, but might still obtain some benefit from the later sections which discuss how to improve network performance. Some very basic knowledge of Unix is assumed in places, but is not necessary for most of the document.

Revision History

1. 22 December 2005: Version 1. First released version. Expanded from material in the report *Computer Networks for the AVO* by A.C. Davenhall, A. Lawrence, E.T.W. Sutorius and R.G. Mann (Version 1.1, 11 February 2005). See URL:

<http://www.euro-vo.org/twiki/bin/view/Avo/WorkAreaThree>

This report was compiled as part of the programme of work for the Astrophysical Virtual Observatory (AVO) Project. For further details of the AVO see URL:

<http://www.euro-vo.org/>

Contents

1	Introduction	1
1.1	Further Reading	2
1.2	Internet Standards or RFCs	3
1.3	Jargon	3
2	A Network Primer	5
2.1	Types of Network	7
2.2	Network Protocols	11
2.2.1	Protocol stack	11
2.2.2	Connectionless and connection-orientated services	14
2.2.3	Datagrams, packets and PDUs	15
2.3	TCP/IP	15
2.4	Internetworks and the Internet	18
2.5	Addressing	18
2.5.1	Physical addressing	19
2.5.2	Logical addressing	19
2.5.3	IP subnets	21
2.6	IPv4 and IPv6	22
2.7	National Research and Education Networks	22
2.8	Pan-European Networking	24
2.8.1	Experimental circuit-switched networks	25
3	Performance Monitoring	27
3.1	Types of Measurement	28
3.2	Common Metrics	29
3.3	Simple Tools	32
3.3.1	Firewall configuration for ping, traceroute and pchar	34
3.3.2	Hints on using pchar	37
3.4	Modern Tools	38
3.5	Network Monitoring and Weather Maps	38
3.6	Additional Tools	40

4	Obtaining Good Network Performance	41
4.1	TCP Congestion Control	41
4.2	Bandwidth Delay Product and Network Performance	43
4.3	TCP Tuning Parameters	44
4.3.1	Calculating the buffer size	44
4.3.2	Examining and setting the parameters	45
4.3.3	Checklist	46
4.4	Parallel Streams	47
4.5	TCP Implementations	47
4.5.1	Selective acknowledgements	47
4.5.2	Fast TCP implementations	47
4.5.3	Packet size	48
4.6	Choosing Hardware	48
4.6.1	Host computer	49
4.6.2	Disks	49
4.6.3	LAN	49
4.7	Further Information	49
5	File Transfer Protocols	51
5.1	Desiderata for a File Transfer Protocol	51
5.2	FTP	52
5.3	SCP	53
5.3.1	Hints on using the encryption and compression options	53
5.3.2	High Performance Network SSH	55
5.4	GridFTP	55
5.4.1	GridFTP and firewalls	56
5.5	BbFTP	56
5.5.1	Hints on installing and using bbFTP	57
5.6	FTP Accelerators	58
5.7	Peer-to-Peer Protocols and BitTorrent	58
6	Discussion	60
6.1	Ten Easy Questions	62
A	The Internet Standardisation Process	65
A.1	Standardisation Bodies	65
A.2	RFCs	66
A.2.1	Obtaining RFCs	66

Acknowledgements	66
Bibliography	68

List of Tables

2.1	Common effects causing signal degradation along a transmission line	6
2.2	Connectionless and connection-oriented services	15
2.3	Class A, B and C licences	20
2.4	The principal European NREs	23
2.5	Capacities of some European academic networks	23
3.1	Common network performance metrics	29
4.1	Typical round trip times	43
4.2	Default and suggested TCP parameters	44
4.3	Optimal TCP send and receive buffer sizes	45
5.1	File transfer protocols	52
5.2	Encryption techniques available in OpenSSH on Linux	54

List of Figures

2.1	A computer network	5
2.2	Point-to-point links	8
2.3	LAN topologies	9
2.4	A typical WAN	10
2.5	Layers, protocols and interfaces in network software	13
2.6	The layers of the OSI and TCP/IP models	14
2.7	TCP/IP protocols	16
2.8	The GÉANT network	26
3.1	Unix ping output	33
3.2	Example output from pchar	35
3.3	SURFNet Detective	39
4.1	Congestion control in TCP/IP	42
4.2	Setting TCP parameters in a <code>/etc/sysctl.conf</code> file	46
5.1	Example bbFTP control file	57

Chapter 1

Introduction

A very important concession has been made to the Smithsonian Institution by the Directors of the Associated Transatlantic Cable Companies, who have agreed to transmit gratuitously between Europe and the United States, a limited number of short messages on astronomical subjects. Under this arrangement two telegrams have already been received from the United States by the Astronomer Royal, who on his part has undertaken, at the request of Dr. Henry, Secretary of the Smithsonian Institution, to forward from Europe any message announcing an important astronomical discovery. The Directors of the Associated Companies have consented that ten messages, of ten words each, may be sent free over the cables annually. This liberal concession on the part of the Directors cannot be too highly appreciated by astronomers generally, and especially by the Fellows of this Society.

In conformity with this agreement the Astronomer Royal will be prepared to forward any important astronomical message, limited to *ten* words, which may be sent to him for this purpose from the principal European astronomers.

*Royal Observatory, Greenwich,
April 8, 1873.*

The above note appeared in the *Monthly Notices of the Royal Astronomical Society* for 1873, only a few years after the inauguration of the first reliable transatlantic cable in 1866. Though anonymous, it was presumably inserted on the authority of the Astronomer Royal, Sir George Airy. The first message transmitted under the agreement reported the discovery of the 129th minor planet, subsequently named Antigoné, by Dr. H.C.F. Peters of Clinton, New York[1].

Telecommunications capacity has increased enormously since the 1870s. Computer networks are now commonplace and the ubiquitous Internet spans the globe. Nonetheless computer networking remains an arcane subject, often appearing both mysterious and problematic. Further the performance of the networks can often appear poor and idiosyncratic.

Many areas of science, engineering and medicine now routinely move large volumes of data between distant locations. Examples might include the routine transfer of the previous night's observations made with a telescope located on a remote mountain site to its home institution for processing and analysis, video conferencing and interactive remote imaging. The first example simply involves copying the remote data in a timely fashion without errors. The other examples are interactive and so it is also important that the data are transferred in close to real time. These requirements have been apparent for some time, but have been

exacerbated by the recent e-Science and Grid initiatives, which are largely about distributed resources and high-throughput applications communicating over large distances.

A consequence of applications which generate substantial network traffic becoming more common is that computer networks are no longer the sole preserve of network experts. Rather, the developers and users of these applications often need to have some understanding of the networks over which their applications communicate. This document is aimed at such people. It is an introduction to computer networks and to ways of obtaining good performance from them. It covers:

- the basics of computer networks,
- how to monitor their performance,
- simple techniques to improve their performance.

It also describes some of the common protocols used to transfer files across networks. These topics are all large and complex subjects, and the document is only an introduction to them. However, references to more advanced and specialised sources of information are included. No previous experience of computer networking is assumed. The structure of the document is as follows:

Chapter 2 introduces the technologies which underlie computer networks,

Chapter 3 introduces techniques for monitoring network performance,

Chapter 4 describes some techniques for obtaining good network performance,

Chapter 5 discusses some of the common file transfer protocols,

Chapter 6 concludes the document with a discussion.

1.1 Further Reading

This report is not a textbook on computer networks, though Chapter 2 is a primer. If more detail is required numerous text books are available. Some which we have found useful are:

Computer Networks by A.S. Tanenbaum, 2003[2],

Computer Networks by P.J. Irving, 2003[3],

TCP/IP Illustrated Volume 1 by R.W. Stevens, 1994[4],

TCP/IP Network Administration by C. Hunt, 2002[5].

Tanenbaum's and Irving's books are both introductions to networking, though they have contrasting strengths. Tanenbaum's is a standard text: thorough, authoritative, comprehensive and recently revised. Irving's is similarly recent but much shorter: it is short enough to read completely, which Tanenbaum's, in practice, is not. One way to use them is to read either Irving or the first chapter of Tanenbaum for an introduction and then refer to subsequent chapters of Tanenbaum for areas that are particularly relevant or interesting.

Stevens' book specifically covers the TCP/IP protocol used by the Internet. Hunt's is a practical guide to setting up and running a TCP/IP network and is aimed at system administrators. Both contain much useful material and again their first chapters are good introductions.

A more idiosyncratic choice is *Where Wizards Stay Up Late* by K. Hafner and M. Lyon[6]. It is a history of the early development of the Internet written in an accessible, non-technical style. Though not directly relevant to the capabilities and performance of modern networks it makes fascinating reading.

The first *Networks for Non-Networkers* workshop was held at University College London during July 2004 and a second was held at the National e-Science Centre (NeSC) in Edinburgh a year later. The Web site associated with this series of workshops¹ contains much relevant information.

1.2 Internet Standards or RFCs

Various Internet Standards documents will be mentioned in this report. These standards are usually issued in the RFC ('Request For Comments') series (the name is something of a misnomer and the RFCs are actually standards documents which are largely immutable). Appendix A explains how to obtain copies of these documents and gives a brief description of the bodies responsible for developing Internet standards. You should be aware, however, that RFCs are standard documents; they usually contain detailed specifications of technical standards and are often not for the faint-hearted.

1.3 Jargon

Computer networking is a field particularly replete with impenetrable jargon and acronyms. Fortunately a number of useful glossaries are available, including:

Wikipedia	http://www.wikipedia.org
Webopedia	http://www.webopedia.com
whatIs	http://whatis.com

You should beware that much networking jargon is ambiguous: a given term will often have different meanings in different circumstances. Also, often several different terms are used to mean the same thing.

One point of nomenclature is worth mentioning because it presents a trap for the unwary newcomer. The size of both physical memory and storage is usually quoted as some multiple of bytes: Megabytes, Gigabytes or whatever. Conversely, network transmission speeds are often given in multiples of bits per second: Megabits per second, Gigabits per second *etc.* Unfortunately the same abbreviations are used for multiples of both bytes and bits. For example, both 'MB/sec' and 'Mb/sec' can both be used for either 'Megabyte per second' or 'Megabits per second'. 'B' is perhaps more common for 'bytes' and 'b' for 'bits', but neither usage is a *de jure* or *de facto* standard, and both are common. Another common usage is 'Mbps' which is always interpreted as 'Megabits per second'. In any event, you must exercise care to extract the required meaning.

¹See URL: <http://gridmon.dl.ac.uk/nfnm/>

In this document the words ‘byte’ and ‘bit’ will always be included in transfer speeds, for example, ‘Mbyte/sec’, in order to avoid any ambiguity. However, other documents use different conventions. Finally, and at the risk of stating the obvious, because a byte comprises eight bits, for the purposes of ‘back of the envelope’ approximations one unit can be converted to the other by shifting the decimal point one place in the appropriate direction.

In older literature transmission speeds were often specified as a number of ‘baud’ or a ‘baud rate’ given. The baud is a unit inherited from telegraphy, where it has been used since its introduction at the International Telegraph Conference of 1927. It is named after the early French pioneer of telegraphy J.M.E. Baudot (1845-1903). Strictly speaking the baud rate is the number of electronic state changes per second. It does not correspond to the number of bits per second because each state change can correspond to more than one bit of data. Nonetheless it was often used incorrectly to mean the number of bits per second. The term is now obsolete and should be avoided in favour of explicitly specifying the number of bits or bytes per second, which is unambiguous.

In some older network documents the term ‘octet’ is used instead of ‘byte’. This usage was adopted because some of the early network software was developed on computers that did not have eight-bit bytes, such as the DEC-10. The eight-bit byte is now ubiquitous, so ‘octet’ is superfluous and ‘byte’ should be used. However, the meaning can be different in other (human) languages. For example, ‘octet’ is simply the French word for ‘byte’.

A final caution concerns the use of SI prefixes such as ‘Mega’ or ‘Giga’ *etc.* The proper SI usage of ‘Mega’ is, of course, to denote a factor of 1,000,000 or 10^6 . However, when being used to describe the storage capacity of memory or disk it is often used (incorrectly) to denote a factor of 1,048,576 or 2^{20} , which is a convenient power of two close to the correct SI factor. Conversely, when specifying transmission speeds the prefixes are usually used with their correct SI meaning.

Chapter 2

A Network Primer

Net. Any thing reticulated or decussated at equal distances, with interstices between the intersections.

Dictionary of the English Language,
Samuel Johnson, 1755.

This chapter is a primer which introduces some of the basic features of computer networks. It assumes no prior knowledge of the subject. It is not, however, a textbook; for more detailed discussions see, *inter alia*, the books by Tanenbaum[2] and Irving[3]. Various definitions of the term ‘computer network’ are possible, but here we shall use: ‘a set of transmission paths, interconnected at nodes’ which link a group of *autonomous* computers (a definition which is not so far removed from Dr Johnson’s definition of a net, above). That is, the group of computers, though able to function on their own, are also capable of exchanging messages and information through the network linking them. Further, this network comprises not just simple links between machines, down which messages pass, but also nodes where two or more links meet (see Figure 2.1). Further the nodes can control down which outgoing path an incoming message will be forwarded. Such networks are now ubiquitous. However, it was not always so: early computers existed largely in isolation and it was only possible to communicate with them via a directly-connected console or printer. Attempts to inter-connect them came later.

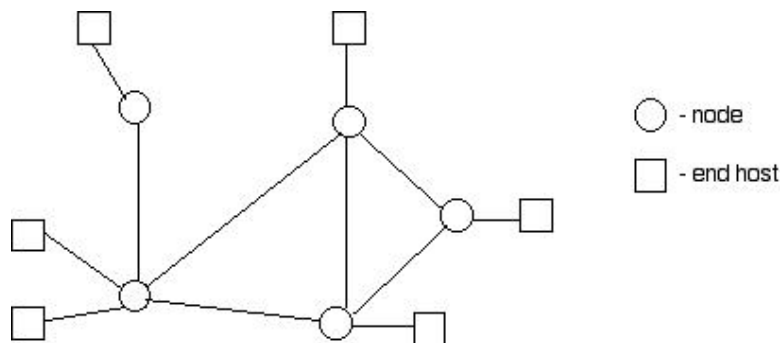


Figure 2.1: The definition of a computer network: a set of transmission paths, interconnected at nodes

Computer networks exist on various scales, from links between machines in the same room up through wiring connecting the machines in a building or campus to regional, national and global networks. Various media are used to carry the communications signals: copper wire, fibre-optic cables and wireless or radio transmissions *etc.* Similarly, the network connecting an organisation's computers might be owned and managed by the organisation itself (typically in small-scale networks linking machines in a room or building) or capacity can be rented from a firm providing telecommunications services (typically in wider area networks). Also the signals can be carried in bespoke cables designed for computer communication, or they can utilise conventional telephone lines designed to carry speech, though this is now much less common than it used to be. Indeed the opposite is true; nearly all long-distance telephone traffic is now digitised.

Attenuation
 Impulse noise
 Thermal noise
 Cross-talk
 Inter-modulation noise
 Radiation
 Radio frequency interference
 Signal reflection

Table 2.1: Common effects causing signal degradation along a transmission line (after Irving[3], pp35-36)

Irrespective of the medium used, any signal transmitted is subject to various types of degradation, simply because of the underlying physics. Table 2.1 lists some common effects, though the details are not germane here. The books by Tanenbaum and Irving give details and Strand *et al.*[7] discuss the effects present in optical fibres. Further, the properties of the different media vary enormously, and fibre-optic cables, in particular, allow signals to be transmitted over long distances with remarkably little loss. Even so, over long distances analogue signals need to be amplified. Digital signals can also be amplified, but alternatively they can be periodically decoded and regenerated, which is preferable because the process can be made essentially lossless, whereas simple amplification will necessarily also amplify the noise.

Similarly the machine receiving the signal needs to check for transmission errors. Such checks are performed by including redundant, derived, information in the transmission which can then be re-derived from the signal received and the values compared. Conceptually these checks are similar to simple 'parity checks', though in practice more sophisticated techniques such as **cyclic redundancy checks** (CRC) are used. Also, long transmissions are split into manage-ably-sized chunks or **packets**, each of which can be sent, checked, acknowledged and, if necessary, resent separately. If transmissions were not split up in this way then a single error could cause an entire, potentially long, transmission to be resent. Note that in this brief discussion there is already emerging the notion of a **protocol** between the two computers; they must agree on: the way in which bit patterns are to be encoded as signals, the signals for the start and end of a packet, the way that error checking information is to be included in the packet *etc.*

So far we have only considered communication along a single wire linking two machines. Real networks, of course, contain multiple machines, any of which can require to communicate with each other. Conceptually there are (at least) two types of mechanisms by which any

two machines in a multiple-machine network can communicate with each other: **circuit switching** and **packet switching**. In circuit switching a dedicated circuit or ‘connection’ is created (or **nailed-up**) when communication between the two machines is initiated, remains in place while the message is passed and is relinquished (or **torn down**) when the transmission has finished. Whilst the message is being transmitted the two machines have sole use of this circuit. Circuit switching was used in early telephone systems.

In packet switching every machine on the network has an **address** which identifies it uniquely. When a message is transmitted the address of the machine to which the message is directed is included in each of its packets. The sending machine emits the addressed packets onto the network, where they are passed and forwarded until they reach their intended recipient. The postal service is a reasonable analogy for this process. Packet switching is now ubiquitous in computer networks and some of its ramifications are discussed in the following section. In the early days of computer networks packets in packet-switched networks were also called **datagrams**, by analogy with telegrams, though this term is now little used.

A final concept which is worth mentioning is **multiplexing**, which entails combining signals for transmission over a shared medium. Two variants are common: **frequency domain multiplexing** (FDM) and **time domain multiplexing** (TDM). In FDM several independent signals are transmitted at different wavelengths. FDM is often used where the medium is fibre-optic cables or radio transmission. Ultimately the physics of the detectors, receivers and transmission medium put limitations on the number of independent signals which can be used. In TDM the various inputs simply take turns to transmit down the medium. TDM is often used when the medium is copper cable.

The capacity of a network is usually expressed in terms of the **data transfer rate**. The data transfer rate between two points on a network is simply the number of bits or bytes which can be transferred from the first point to the second in a given time interval. It is usually measured in (Kilo, Mega or Giga) bits or bytes per second. Synonyms are the **data rate**, **throughput** or **bandwidth**. However, the term ‘bandwidth’ will be avoided in this document because it also has an alternative meaning concerning the range of frequencies which can be used to transit a signal along a channel (see above). This alternative meaning is also, of course, closer to the term’s normal use in the physical sciences.

2.1 Types of Network

There is no generally agreed taxonomy for classifying computer networks. However, two distinct properties are important: transmission technology and scale. Transmission technology can be broadly divided into two types: **broadcast** and **switched**.

In a broadcast network all the machines on the network are connected by a single, continuous communications channel (sometimes called a **bus** or **backbone**). A packet emitted by any of the machines propagates through the bus and is received by every machine on the network. The receiving machines will ignore the packet unless its destination address matches their own address. (Broadcast networks usually have special arrangements for sending simultaneous messages to either multiple machines or all machines. These features are respectively called **multicasting** and **broadcasting**.)

Broadcast networks are essentially a shared channel, meaning that only one device can transmit at a time. Consequently, broadcast networks are only suitable for situations in which traffic loads are expected to be light. The advantage of broadcast networks however is that their wiring costs are very low and their routing very simple.

Conversely, in **switched networks** data are routed to their destination via a sequence of **point-to-point**, that is node-to-node, links (see Figure 2.2). Unless the source and destination of traffic happen to be adjacent, network traffic will pass through one or more intermediate nodes during its journey. By definition, circuit and packet switched networks are both also point-to-point networks.

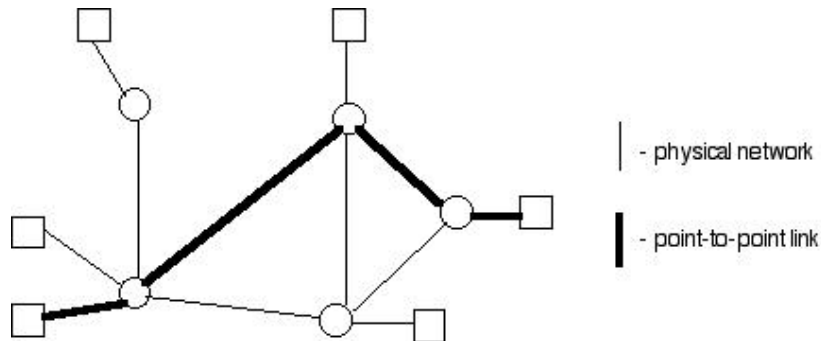


Figure 2.2: Point-to-point links

Many topologies are possible for point-to-point networks, though usually they will be linked in an incomplete mesh where each node is linked to some, but not all, of the other nodes in the network. Thus there will often be multiple routes from the source to destination, and whilst techniques for routing traffic expeditiously are important, multiple routes provide a level of redundancy. This property is a key advantage of switched networks.

The unit of currency in packet switched networks is, of course, the packet. Packets arriving at one point are **stored** temporarily before being **forwarded** to the next point in the route, hence the name **store and forward networks** (or ‘hot potato’ networks).

Packet switching is used in computer networks because it works well in networks with heavy traffic flow: nodes can interleave packets from end-hosts and other nodes to achieve better utilisation of network capacity than dedicated circuit-switched connections. However, there are, of course, also disadvantages. The process of storing packets upon arrival, selecting an outgoing link based on routing information, and retrieving the packet for retransmission on the selected link adds delays to the overall transmission time. This delay is exacerbated by the need to place packets in queues during periods of heavy loading. **Node delay** and **queueing delay** both have severe implications for network performance. Additionally the required networking software is complex and the provision of multiple routes significantly increases the importance of routing strategy.

Though there are many exceptions, small, local networks are often broadcast, whereas larger, more geographically dispersed ones are often switched. An alternative classification of networks is by scale. In this case they are usually divided into LANs, MANs and WANs.

LANs (Local Area Networks) typically link the computers in a single building or campus. They are usually privately owned. That is, the network cabling and kit is owned and managed by the same organisation which owns the the computers which the network is linking and which occupies the building that they are in. **Ethernet** is a popular technology for LANs. Early versions of Ethernet broadcast on a bus, though modern versions are usually switched (through a central switch). However, conceptually Ethernet is still a bus rather than a collection of point-to-point links.

Various topologies are possible and two are shown in Figure 2.3.

MANs (Metropolitan Area Networks) are larger versions of LANs. Typically they link several distinct sites dispersed around a city. They may be privately owned or they may be implemented using network capacity rented from a telecommunications service provider. Early MANs used technologies similar to LANs, but now they often use technologies more akin to WANs (below).

WANs (Wide Area Networks) span a wide geographical area, typically a country or continent. They are almost invariably operated by a telecommunications service provider rather than being owned by an end user. Rather, an end user's individual machine, or more likely his LAN or MAN, will have a connection to the WAN. The backbone of the familiar Internet is basically a collection of linked WANs. WANs are usually point-to-point networks.

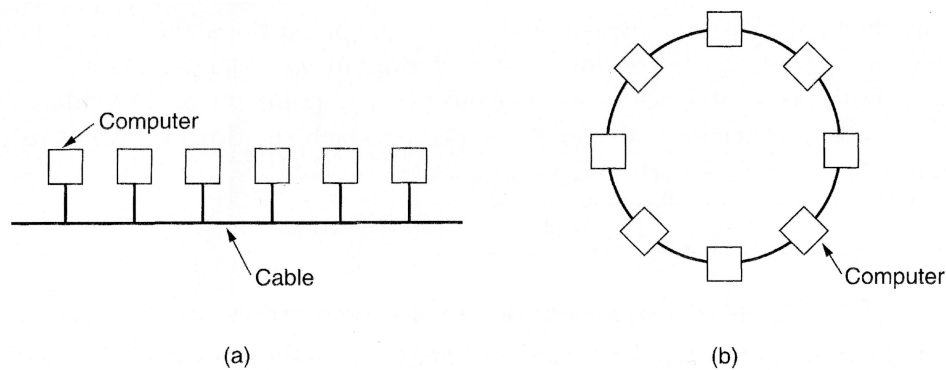


Figure 2.3: LAN topologies: (a) is a simple bus and (b) a ring (from Tanenbaum[2] p17)

In the terminology of WANs, an end-user's computer which is connected to the WAN is called a **host** or **end-system**. The hosts on a WAN are connected by a **subnet** or **communication subnet** (this usage of 'subnet' is different from, and not be confused with, an 'IP subnet,' which is described in Section 2.5.3, below). The subnet, in turn, usually consists of two types of components: **links** and **routers**. The links (also called **transmission lines**, **circuits**, **channels** or **trunks**) are simply the cables or other media connecting the routers. The routers (also called **packet switching nodes**) are specialised computers and each will be connected to two or more transmission links. When an incoming packet arrives at a router from a transmission link the router will choose one of its other transmission links onto which to dispatch the packet.

A typical topology is shown in Figure 2.4. Each of the hosts is connected to a local LAN, which is also connected to one of the routers on the WAN. A router which is also connected to an external network, such as a LAN or another WAN is also called a **gateway** (though sometimes this term is reserved for a link between two networks which use different protocols and hence which is translating between these protocols) or a **bridge** (though this term too is also used in other contexts). A packet emitted by one of the hosts in Figure 2.4 traverses the local LAN to its gateway, passes on to the subnet and is passed between routers until it reaches the gateway of the destination LAN, where it passes into the LAN and on to the desired destination host. Note that the packets comprising a message will not necessarily all

travel by the same route. Nor will they necessarily arrive in the correct order; the destination host must accept them as they arrive and reassemble the message correctly.

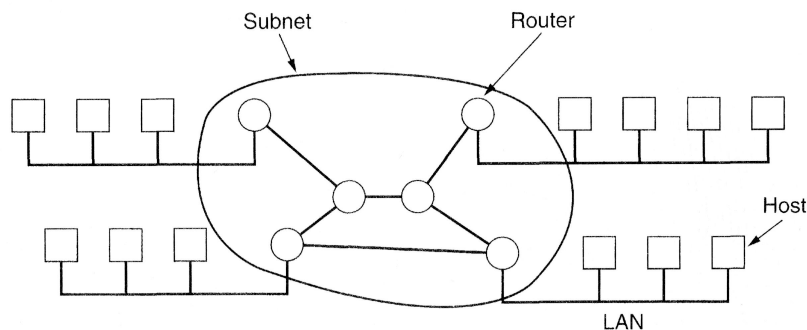


Figure 2.4: A typical WAN (from Tanenbaum[2] p19)

Subnets usually have irregular topologies, similar to (though, in practice more complicated than) the one in Figure 2.4. Though Figure 2.4 shows only a single subnet, WANs will often have gateways to other WANs, forming a network which spans the globe. Also there is often an informal, irregular hierarchy of more local routers connecting to higher-capacity regional routers for transmitting longer-distance traffic and these regional routers in turn connect to national and international ones.

If the network is lightly used then when a packet arrives at a router it can immediately be processed (by checking its destination address) and dispatched down the appropriate transmission line (though this processing will take a finite time, of course). However, if the network is busy then packets might arrive faster than the router can process them. Thus, incoming packets must be placed in a queue and await their turn to be processed. An important consequence of this arrangement is that the transmission time for a packet to travel from one host to another is not merely the sum of the time that it spends propagating through the transmission lines that it traverses, but also includes the time that it spends sitting in the queues of the routers which it visits *en route*. In practice, in busy networks packets usually spend more time waiting in router queues or being processed by routers than they do in transit along fibres. Similarly routing information is important in order to ensure that routers direct packets on an efficient route through the network rather than sending them on a circuitous detour. Indeed, it is conceivable that not all the packets in a given message will travel by the same route.

In modern networks, particularly in the context of the Internet, a gateway connecting a LAN to a MAN or WAN will often be protected by a **firewall**. The usual purpose of a firewall is to protect the LAN from unwelcome external intrusions, such as hackers trying to probe the system prior to attempting unauthorised access. Tanenbaum[2] (p776) memorably describes firewalls as ‘just a modern adaptation of that old medieval security standby: digging a deep moat around your castle. This design forced everyone entering or leaving the castle to pass over a single drawbridge, where they could be inspected by the I/O police.’ In this analogy the gateway is the drawbridge and the firewall is the detachment of guards who stand on it, checking the authorisation of everyone attempting to pass in (or out). Firewalls are configurable. A firewall might examine every packet in detail, though often they do not. Many will inspect only packets which are attempting to initiate a connection. Other packets

may be allowed through without check, or simply be checked to establish that they belong to a previously established connection. The important point here is that firewalls are not ‘magic bullets’ which protect against all intrusions. Rather, they will give the protection that they are designed and configured to give.

Firewalls can also be used to police outgoing traffic. This function can help to prevent unauthorised access by your users to banned services or machines, can force certain types of traffic through another machine (for example, preventing out-bound HTTP access on port 80, thus forcing users to go through a proxy server machine) and can help stop the spread of propagating infections such as viruses.

There is likely to be a firewall superintending the traffic passing through the gateway to your site. For extra protection, each machine at your site could also operate its own firewall, such as `iptables` in Linux[8]. Firewalls are notorious amongst users for preventing services, particularly uncommon ones, from working (though arguably the obloquy should be directed at the vendors of operating systems for releasing systems whose default state leaves them wide open to external attack). Firewalls will often block whole ranges of port numbers which are not usually used. Thus, a service which uses such a blocked port will fail. (A **port** in this context is simply a ‘doorway’ or address which a program uses to read from or write to a hardware network interface. Typically such ports are identified by a sequence number of **port number**.)

Finally, a note about the addresses used to identify hosts. Clearly it would be impossible to maintain a comprehensive list of all the addresses of all the (millions) of hosts available on all the WANs spanning the globe. Rather, a hierarchical scheme is used. Telephone numbers provide a good analogy. In order to ring the National e-Science Centre in Edinburgh from overseas the number to dial is:

00 44 131 650 9833

The initial 00 directs the local telephone exchange to route the call to an international exchange; 44 routes the call to the UK; the 131 to Edinburgh; 650 to the local exchange and 9833 to the NeSC’s reception desk. With this scheme a local exchange in, for example, rural New Zealand, can handle the call by simply knowing that 00 is the code for international calls. Though the technical details are different, network addresses work on a similar principle. Network addressing is discussed in further detail in Section 2.5, below.

2.2 Network Protocols

In order to communicate across a network a source and destination host, and any intermediate routers, must agree on a common interpretation of signals (or **protocol**). Such protocols are both complex and fundamental to the operation of computer networks.

2.2.1 Protocol stack

Hosts attached to a network usually have utility programs which provide a user with the familiar network applications, such as e-mail, file transfer and access to the Web. These application programs, in turn, invoke libraries of standard network software in order to access the network. At its lowest level (that is, closest to the hardware) this network software is creating and receiving traffic on the network: constructing and emitting packets *etc*, and at

the hardware level, even setting voltages. At its highest level it interacts with the end-user programs, such as a file transfer application, in terms of the ‘high level’ operations which the program understands, such as opening, writing and closing files. The software between the highest and lowest levels is complex and in order to manage this complexity it is conventionally divided into a number of layers. The higher layers deal with the abstractions and concepts that are familiar to users (such as reading and writing files) and the lower layers become successively less abstract and closer to manipulating individual packets on the network. Each layer communicates only with the layers immediately above and below it (with the obvious exception of the top and bottom layers which respectively talk to the end-user application and the physical network) and they do so through a well-defined interface or **protocol**. The collection of protocols defining the various interfaces between all the layers is called the **protocol stack**. The advantages of this approach are fairly obvious. Components of distributed systems can be developed independently, for various hardware platforms, perhaps using different programming languages. Also the decomposition into separate layers allows the layers themselves to be developed, tested and, indeed replaced, individually.

When a network application (such as file transfer) runs on a host, information starts in the application program, passes ‘down’ through the layers of the network software, crosses the network and ascends through the corresponding layers of network software on the destination computer (see Figure 2.5). However, each layer in the protocol is an abstraction, which gives the impression that the layer is communicating with the corresponding layer on the destination machine (these corresponding layers are called **peers**). As an analogy of the way that the stacks work, imagine that a Portuguese businessman in Lisbon wishes to exchange a letter with an Estonian colleague in Tallinn and neither speaks the other’s language. They cannot find a Portuguese to Estonian translator, so each employs an English translator instead. The Portuguese businessman dictates his letter which is duly translated into English. A secretary working for the translator posts the translated letter to Tallinn where the reverse process occurs: a secretary dealing with the incoming post arranges for the letter to be translated so it can then be read by the Estonian businessman. Note how the lower layers can be substituting without affecting the higher ones: the letter might be e-mailed or faxed rather than being sent by post, or it might be translated into French rather than English.

Most network systems adopt some-such layered approach (though the lower layers might be implemented in hardware rather than software). However, two sets of protocol standards are particularly important: The OSI Seven Layer Model and TCP/IP. The OSI (Open Systems Interconnection) Seven Layer Model was developed during the 1980s by the International Standards Organisation (ISO) and so is also known as the ISO Seven Layer Model. Unsurprisingly, it consists of seven layers, each with a defined function (see Figure 2.6). The details are not germane here, other than to note that layers one to three deal with the physical network, layers five to seven are network independent, and layer four acts as an interface between the two. The OSI model is intended as a reference model. It specifies the function of each layer, but not the precise services and protocols to be used by each layer. ISO has produced additional standards for all the layers, but these are not part of the reference model itself.

The OSI reference model was intended as a ‘blueprint’ against which real network protocols would be developed, and by the end of the 1980s it looked likely to assume this central role. However, subsequently its importance has dwindled, though many of the concepts which it introduced remain important. Rather, what actually happened was that the TCP/IP protocol, which is used by the Internet, has become ubiquitous. TCP/IP is described briefly in Section 2.3, below (see also Figure 2.6). Both the OSI and TCP/IP models are complex topics which the discussion here can merely adumbrate; at least one book has been written

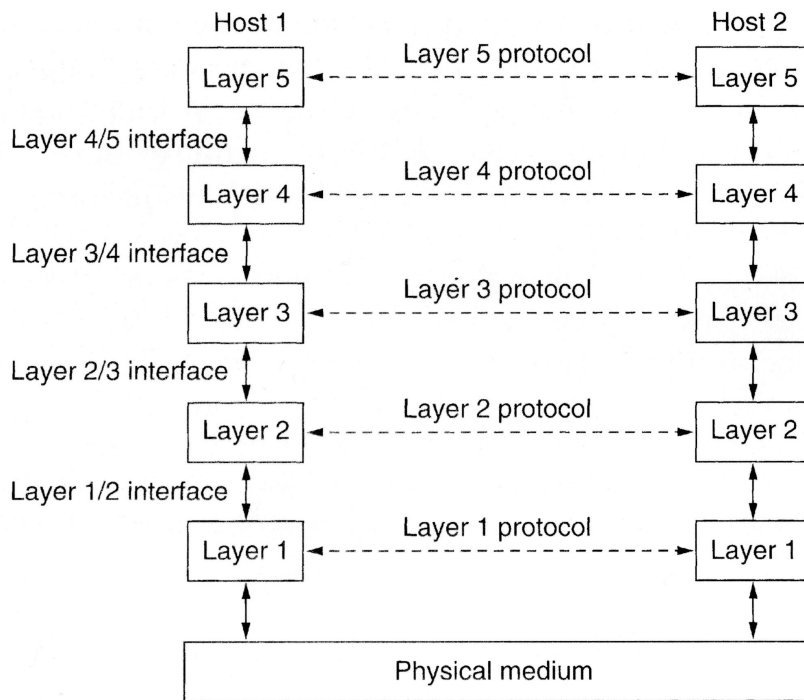


Figure 2.5: Layers, protocols and interfaces in network software (from Tanenbaum[2], p27)

which merely compares them[9]. For further information Sections 1.3 and 1.4 of Tanenbaum[2] (pp26-49) are a good place to start.

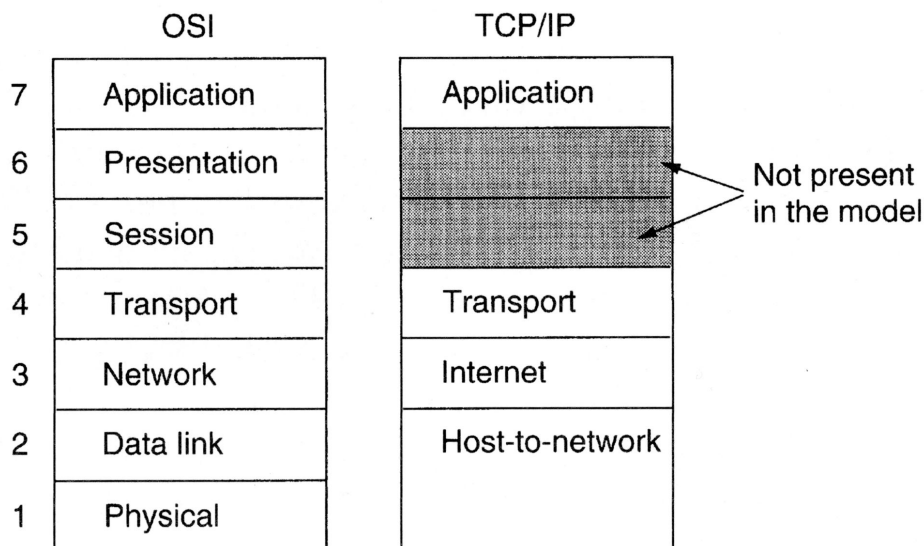


Figure 2.6: The layers of the OSI (left) and TCP/IP (right) models (from Tanenbaum[2], p43). The corresponding layers in the two models are horizontally aligned. The correspondence is not perfect, and the layers themselves are, in a sense, abstractions. In the TCP/IP model the internet layer is also called the network layer and the host-to-network layer is also variously known as the link, data-link or network interface layer

2.2.2 Connectionless and connection-orientated services

Packet-switched networks are inherently **connectionless** in that there is no direct, dedicated connection between the source and destination hosts, and each packet travels independently from one to the other. If the protocol by which a layer in the network stack is accessed reflects this behaviour it is said to offer a **connectionless** or **datagram service**. A higher layer accessing a connectionless service must typically ensure that incoming packets are re-assembled in the correct order and perform its own checks to ensure that all the packets have arrived.

Conversely, a layer can offer a **connection-oriented** or **virtual circuit** service. Here the layer is mimicking a direct connection, such as a telephone line, and includes code which automatically checks that dispatched packets have arrived, resends lost packets *etc.* The OSI and TCP/IP protocols provide connectionless and connection-oriented services at different layers in their stacks. The OSI transport layer protocol is only connection-oriented whereas TCP/IP transport offers both connection-oriented and connectionless alternatives. Conversely, OSI offers both options in the network layer whereas TCP/IP has only a connectionless network layer (see Table 2.2).

Layer	OSI		TCP/IP	
	Connection-less	Connection-oriented	Connection-less	Connection-oriented
Transport		•	•	•
Network	•	•	•	

Table 2.2: Connectionless and connection-oriented services in various layers of the OSI and TCP/IP models

2.2.3 Datagrams, packets and PDUs

A **datagram** is defined in RFC 1594[10] as ‘a self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network.’ Datagrams need to be self-contained and able to propagate without reliance on earlier exchanges because they are used by connectionless services.

A **packet** is simply the unit of data transmitted from a source to a destination on a packet-switched network, such as the Internet.

The ubiquity of the Internet means that ‘datagram’ and ‘packet’ are now largely interchangeable terms for the message units handled by the IP protocol (layer three) in the TCP/IP Internet protocol stack (see Section 2.3, below).

A **PDU** (Protocol Data Unit) is simply a unit of data passed across a network. Although the term can be used to refer to a unit of data at any of the OSI model’s seven layers, its use implies a specific layer of the model and a specific protocol, for example ‘Ethernet PDU (layer two).’

2.3 TCP/IP

TCP/IP is the protocol used by the global Internet. It originated in the US ARPANET in the 1970s, though it is now much evolved and revised. Ultimately TCP/IP is controlled by the Internet Society (ISOC), though much of the work is delegated to subsidiary Boards and Task Forces (see Appendix A). Like the OSI reference model, TCP/IP is a layered protocol and it has four layers (see Figure 2.6). Briefly, the function of each layer is as follows.

Link or **Host-to-network** handles the hardware-specific details of interfacing to the physical communications channel,

Network or **Internet** handles the movement of packets around the network. For example, routing is handled here,

Transport provides a flow of data from the source to the destination host, for use by the application layer above,

Application comprises the specific functionality of each application, such as file transfer, remote login, electronic mail, *etc.*

On Unix systems the application layer usually executes as a user process whereas the lower layers are part of the operating system kernel. The TCP/IP protocol does not prescribe this structure but it is a common arrangement.

The acronym TCP/IP is constructed from ‘Transmission Control Protocol’ (corresponding to the transport layer) and ‘Internet Protocol’ (corresponding to the Internet layer). An alternative name is the **Internet Protocol Suite**. Indeed, TCP/IP is something of a misnomer because there are rather more protocols involved than just TCP and IP. Figure 2.7 shows some of them, arranged by layer. Some brief details are given below, but first, some important points to note about TCP/IP are:

- The TCP/IP protocols (which occur at OSI layers three, four and seven) layer on top of ‘something else’ (OSI layers one and two). A wide variety of technologies can be used to provide layers one and two without affecting the TCP/IP superstructure.
- The IP protocol (see below) at OSI layer three handles inter-networking, that is communication between networks. Routing is central to IP and routers can sit between networks, which might employ different technologies (that is, implementations for OSI layers one and two). It is the resulting flexibility to inter-connect disparate networks which makes the global Internet possible.

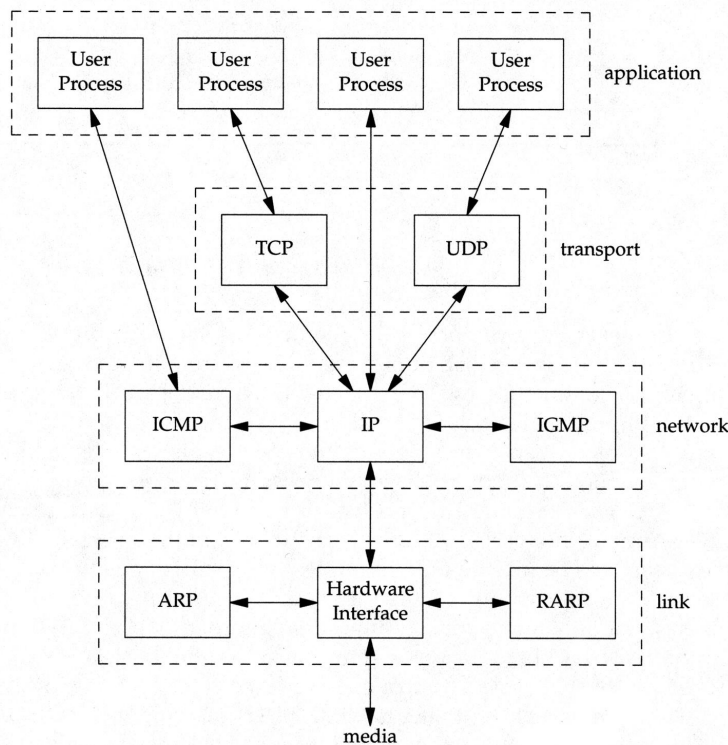


Figure 2.7: Some of the TCP/IP protocols, arranged by layer (from Stevens[4], p6)

Transport layer There are two important, but very different protocols for the transport layer: TCP and UDP.

TCP (Transmission Control Protocol) provides a reliable connection-oriented service for the complete transfer of messages from the source to the destination host. It handles not just dividing the outgoing messages into packets and transmitting them but also ensures that received packets are acknowledged, lost packets resent *etc.* TCP is used in applications where it is important that a message arrives complete and intact, such as file transfer, electronic mail or accessing Web pages. Such examples are, of course, the common applications for which networks are used. See Stevens[4], Chapter 17, pp223-228 and subsequent chapters for further details. TCP was originally defined in RFC 793[11] in 1981 and has been tweaked endlessly ever since.

UDP (User Datagram Protocol) is a much simpler protocol than TCP. The message is simply split up into packets and the packets dispatched from a source to a destination host. No acknowledgement that packets have been received is returned and there is no guarantee that all (or any) of the packets arrived. If an application using UDP requires any reliability then it must be included in the code in the application layer. On first consideration UDP might appear much less useful than TCP. However, there are applications, such as streaming video (as used in video conferences), where it is either inappropriate or impossible to re-send lost packets and it more important to keep on transmitting a steady stream of new packets. Further, the absence of acknowledgements and checks permits packets to be transmitted at a higher rate. See Stevens[4], Chapter 11, pp143-168 for further details. UDP is defined in RFC 768[12].

Note that the packets in a UDP stream are less obviously related to each other than those in a TCP stream. Consequently it is more difficult for firewalls to properly handle UDP than TCP and hence it is often preferable to use the latter for transfer over a WAN.

Though a considerable over-simplification, it might be useful to think of the difference between TCP and UDP as being that TCP is suitable for applications where it is more important that all the packets arrive than that they arrive quickly, whereas UDP is suitable for cases where it is more important that (most of the) packets arrive quickly than they all arrive.

Network or internet layer IP is the main protocol of the network layer, though the ICMP and IGMP protocols are also important.

IP (Internet Protocol) is the principal protocol of the network layer. It is used by both TCP and UDP (above) and hence by most network applications. Further, ICMP and IGMP messages (below) are transmitted as IP packets.

IP defines a datagram delivery service which is connectionless and unreliable. A datagram service is always connectionless because it maintains no state information about successive packets in a message: each is handled entirely independently of its peers. Further, the IP protocol is unreliable in that though it makes best efforts to deliver packets it gives no guarantees. It does, however, include a simple error handling mechanism: if a router detects that it has lost a packet due to some misadventure, such as its input queue filling up, it will attempt to send an ICMP packet (below) back to the source host. If a connection-oriented and reliable service is required then this must be implemented above IP (and this, of course, is the function of TCP, above). See Stevens[4], Chapter 3, pp33-51 for further details. IP is defined in RFC 791[13].

ICMP (Internet Control Message Protocol) communicates error messages and similar information between routers and hosts. ICMP messages are usually processed in the network or transport layers, though some will cause error conditions to be reported to user processes. See Stevens[4], Chapter 6, pp69-83 for further details. ICMP is defined in RFC 792[14].

IGMP (Internet Group Management Protocol) is a rather more specialised protocol which is used for multicasting a UDP datagram to multiple hosts (recall that multicasting is the process of sending the same packet simultaneously to multiple recipients). See Stevens[4], Chapter 13, pp179-186 for further details. IGMP is defined in RFC 1112[15].

Link or host-to-network layer ARP (Address Resolution Protocol) and RARP (Reverse Address Resolution Protocol) are specialised protocols used to interface to some types of network. See Stevens[4], Chapters 4, pp53-64 and 5, pp65-68 respectively.

2.4 Internetworks and the Internet

Multiple networks connected via gateways or bridges are said to form an **internetwork**. An **internet** (note the lower-case ‘i’) is any network that communicates using TCP/IP, or more precisely the Internet Protocol Suite. Such a network might be; simply two linked machines, an isolated LAN or a more complex network with routers. The **Internet** (with an upper-case ‘I’) is the collection of all the linked hosts and networks which communicate using TCP/IP. TCP/IP is now ubiquitous and millions of hosts are linked via a collection of networks which span the globe.

2.5 Addressing

As mentioned earlier, every computer attached to a network has a unique address, that is, a sequence of bits, which differs from the address of every other computer and which identifies it. In a broadcast network, such as Ethernet, the computer monitors the network bus and, if well-behaved, should only ingest packets which match its own address. In a routed network, routers examine packets and use the packets’ address to direct them on the next stage of their journey, rather in the manner of a sorting office in a postal service. It is commonplace to speak of a computer’s address. However, strictly speaking, it is a computer’s **interface** to the network which has a unique address. If a computer has two network interfaces (for example because it is connected to two networks), each interface will have its own unique address. There are two types of addressing mechanism:

- physical addressing,
- logical addressing,

which are briefly discussed separately below.

2.5.1 Physical addressing

A physical address is part of the computer's hardware and will not normally change. It is usually programmed into firmware on a Media Access Control (MAC) unit (and hence is referred to as the **MAC address**), which itself is usually part of a Network Interface Card (NIC). Consequently if you swap the NIC on a computer its MAC address will change. Early versions of Ethernet could have either 16 or 48 bit addresses, though now only 48 bit versions are likely to be encountered. Indeed, recent enhancements to the Ethernet standard are defined solely in terms of 48 bit addresses. Physical addressing works for LANs, but does not scale to larger networks with millions of hosts. In this latter case logical addressing is used.

2.5.2 Logical addressing

Logical addressing offers the following advantages:

- it allows address to be retained when the hardware changes,
- it allows the demarcation of the network into subsidiary networks,
- it provides a structure or hierarchy to the addresses.

The last two points allow addresses to be grouped hierarchically, in the manner of telephone networks or postal systems, as described above, and provides the flexibility necessary in the global Internet with its millions of hosts in a state of constant flux.

Internet addresses are managed and assigned by the Internet Corporation for Assigned Names and Numbers (ICANN).¹ The ICANN delegates given address ranges to regional and national authorities which, in turn, allocate ranges to ISPs and other institutions. In this document such ranges of addresses will be called **licences** (because the recipient organisation is licenced to allocate the addresses within its given range).

All IP addresses are 32 bits (4 bytes) long. They are usually represented (for human consumption) using a **dotted decimal notation** where each of the bytes is shown as a decimal number (which, necessarily, must be in the range 0 to 255). An example might be:

195.194.14.17

Over the years the way that address ranges have been allocated within the IP address space has evolved.

Classful addressing

For several decades after the introduction of the IP protocol (before, say, the early 1990s), licences were awarded in one of five classes A to E. This scheme is no longer used, but is often still referred to and so is worth mentioning. Classes D and E were special and are not discussed further here. Classes A to C were used for different sizes of organisation. The classes differed in:

- the number of licences which could be issued in each class,

¹See URL: <http://www.icann.org/>

- the number of individual addresses available in each licence.

The first byte of the IP address contains information to identify the class (and also part of the common base address for the licence). briefly:

Class A The licence address is contained in the first byte of the IP address, thus leaving three bytes for individual addresses. For historical reasons this type of licence was only issued to some American universities.

Class B The licence address is contained in the first two bytes of the IP address, thus leaving two bytes for individual addresses. This type of licence was used for larger organisations.

Class C The licence address is contained in the first three bytes of the IP address, thus leaving one byte for individual addresses. This type of licence was used for smaller organisations.

Table 2.3 shows the number of licences which can be issued in each class and the number of individual addresses available with each type of licence.

Class	Licences	Addresses
A	128	16.3×10^6
B	16,384	64×10^3
C	2×10^6	256

Table 2.3: Details of class A, B and C licences. The ‘Licences’ column lists the total number of licences which can be issued in each class. The ‘Addresses’ column gives the number of individual addresses available within each licence.

Classless range allocation

The Internet has grown beyond the wildest expectations of its original designers. As the number of hosts increases free addresses are becoming increasingly scarce. Classful addressing wastes many addresses. Many organisations need a class B licence rather than a class C one because they have (or think they will soon have) more than 256 computers, but they do not have anything like 64,000 (see Table 2.3), and hence will not use most of the addresses allocated to them.

Classless interdomain routing (CIDR) is an interim solution to this problem which has been used since about 1993. The basic idea is to allocate addresses in variable-sized blocks, suitable for the expected needs of the licensee and without regard to classes. Consequently the licence addresses no longer lie on byte boundaries. Basically CIDR is a trade-off which makes better use of the available address space, but at the expense of additional complexity and processing time when computing routing information.

2.5.3 IP subnets

IP subnets are a way of further subdividing the addresses allocated to a licence in classful addressing (above). In class A and B licences there are many addresses available within each licence (see Table 2.3). An organisation with a class A or B licence will usually run its own LAN, and LANs often become inefficient when the number of hosts that they contain becomes large (in the case of Ethernet this is true of both the original simple bus topologies and modern switched Ethernet).

What is required is a way of sub-dividing the addresses within a licence, so that each packet need be broadcast only on the appropriate sub-network. Consider a university with a class A or B licence. Typically it will have distinct Ethernet LANs for each department or faculty. When an external packet arrives at the university's gateway a mechanism is needed to route it to the appropriate departmental Ethernet. IP subnets provide this facility.

Importantly, though individual machines are externally visible, any IP subnet hierarchy is invisible outside the organisation operating the licence. External routers do not need to know anything about any IP subnets which the destination organisation might be using. Organisations are free to set up and use their own IP subnets without fearing consequences for external users. Consequently IP subnets do not require any external licencing or policing.

In basic classful addressing (above) the 32 bits of an IP address are divided into:

- the base address for the licence (16 bits for class B),
- the individual host address (16 bits for class B).

IP subnets modify this scheme slightly by allocating a specified number of bits, immediately following the base address, to identify the subnet. Following the subnet address, the remaining bits identify the individual host within the subnet. Thus, in basic classful addressing, an address comprises:

licence base address + individual host address

and in an IP subnet an address comprises:

licence base address + subnet address + individual host address

To return to the earlier example, when a packet enters the university gateway, the IP subnet address is examined and the packet routed to the appropriate departmental Ethernet.

In basic classful addressing, the number of bits allocated for the base address is fixed for each type of licence. Conversely, in an IP subnet the number of bits allocated for the subnet address can vary. Consequently, a **subnet mask** is used to indicate which bits belong to the base or subnet address and which to the host address. The subnet mask is 4 bytes long, like the IP address. The values of bits in a subnet mask are interpreted as follows:

- a '1' indicates that the corresponding bit in the IP address (to which the mask is being applied) is part of the base or IP subnet address,
- a '0' indicates that the corresponding bit in the IP address is part of the individual host address.

Two notations are in use for subnet masks. One is the same dotted decimal notation used for IP addresses, for example 255.255.252.0. The other is ‘/x’ where *x* is the number of bits in the base and subnet addresses, for example if a total of 22 bits were so used, the subnet mask would be written ‘/22.’

Consider the example (adapted from Tanenbaum) where a university was using a mask 22 bits long. The first subnet might start at 130.50.4.1 and with this length of mask can contain 1024 addresses (minus a couple that are reserved for special purposes). The second will start at 130.50.8.1, the third at 130.50.12.1 *etc.* Writing out the binary addresses makes it clear why this is the case:

Mask	11111111	11111111	111111 00	00000000	or /22
Subnet 1	10000010	00110010	000001 00	00000001	or 130.50.4.1
Subnet 2	10000010	00110010	000010 00	00000001	or 130.50.8.1
Subnet 3	10000010	00110010	000011 00	00000001	or 130.50.12.1

Here the vertical bar (‘|’) is indicating the boundary between the subnet address and the individual host address. For further details of IP subnets see Tanenbaum[2] pp438-441.

2.6 IPv4 and IPv6

The current version of the IP protocol is version 4, so-called IPv4. IPv4 is in widespread use and is implied in the rest of this document. However, it has a number of problems, most notably, though not only, the increasing scarcity of free addresses. A new version of the IP protocol which addresses these problems, IPv6, has been developed and is coming into use. (IPv5 was an experimental protocol for real-time streaming which was not widely used.)

Work on IPv6 started in 1990. Though the protocols are now defined, take-up has been patchy. The transition seems likely to continue for some years and might never be complete. Though IPv6 is usually spoken of as tackling the shortage of free addresses (which problem it solves), it also addressed a number of other issues, including, *inter alia*, reducing the size of routing tables, simplifying the protocol, providing better security and aiding multicasting.

In general IPv6 is not compatible with IPv4. However, it is compatible with many of the other TCP/IP protocols, including TCP, UDP, ICMP and IGMP, albeit sometimes with small modifications, usually to accommodate larger addresses. For further information see Tanenbaum[2] (pp464-473) or Loshin[16]. IPv6 is defined in RFCs 2460 to 2466[17].

2.7 National Research and Education Networks

Most countries have a **National Research and Education Network** (NREN) to carry traffic between their universities, higher education establishments and research institutes. Such academic networks are provided to support research and other academic activities. Typically they will be linked to both the NRENs of other countries and the wider Internet. In the UK the NREN is SuperJANET and it is operated by UKERNA. The principal European NRENs are listed in Table 2.4 and the capacities of some of their backbones are shown in Table 2.5. There is also a pan-European network linking the individual national networks which is described briefly below.

NREN	Country	URL http://
ARIADNET (GRNET)	Greece	www.grnet.gr/index.php?language=en
ARNES	Slovenia	www.arnes.si/english/
CESNET	Czech Republic	www.ces.net/
DFN	Germany	www.dfn.de/content/en/enhome/
FCCN	Portugal	www.fccn.pt/
GARR	Italy	www.garr.net/
HEAnet	Ireland	www.heanet.ie/
HUNGARNET	Hungary	www.hungarnet.hu/
NORDUnet	Nordic Countries	www.nordu.net/
RedIRIS	Spain	www.rediris.es/
RENATER	France	www.renater.fr/
RESTENA	Luxembourg	www.restena.lu/restena/
SURFnet	Netherlands	www.surfnet.nl/en/
SWITCH	Switzerland	www.switch.ch/
UKERNA/JANET	United Kingdom	www.ukerna.ac.uk/

Table 2.4: The principal European NRENs. Strictly speaking the NRENs listed here are the shareholders in DANTE (except that for legal reasons the shareholder in the United Kingdom is the Higher Education Funding Council for England). The partners in NORDUnet are Denmark, Finland, Iceland, Norway and Sweden. For a more extensive list see URL: <http://www.garr.it/garreuropa/nrn-engl.shtml>. Also, TERENA publishes an annual compendium of NRENs; see URL: <http://www.terena.nl/compendium/>

NREN	Country	Capacity (Gbit/sec)	Expected Increase
GÉANT	pan-European	10	-
RENATER	France	2.4	4
DFN	Germany	2.4	2
GARR	Italy	2.5	4
UKERNA/JANET	United Kingdom	1-10	1

Table 2.5: Capacities of some European academic networks. The ‘Expected Increase’ column shows the factor by which the capacity is expected to have increased by 2006 (and is taken from the TERENA annual compendium of NRENs)

2.8 Pan-European Networking

Since the early 1990s there has been a series of pan-European networks. These networks have been funded jointly by the individual NRENs and the European Commission. In chronological order they were:

EuropaNET
TEN-34
TEN-155²
GÉANT

GÉANT is the most recent and has superseded the earlier ones. GÉANT is managed by a company called DANTE. In addition the TERENA association promotes European academic networking. GÉANT, DANTE and TERENA are discussed briefly below.

DANTE (Delivery of Advanced Network Technology to Europe)³ is a small firm based in Cambridge whose purpose is to plan, build and operate pan-European research networks. It was founded in 1993 by a consortium of NRENs (the current list is shown in Table 2.4) and was established as a limited liability company and a ‘not for profit’ organisation. DANTE currently manages GÉANT and previously ran the earlier pan-European networks. It is also involved in a number of similar activities.

GÉANT is the international network connecting the national research networks of European countries⁴. It is funded jointly by twenty-six NRENs and the European Commission and is managed by DANTE. GÉANT is a five-year project which became fully operational on 1 December 2001. GÉANT serves some 3500 higher education institutions in 32 countries. There are nine circuits at the core of GÉANT which operate at speeds of 10 Gbit/sec, with a further eleven others which run at 2.5 Gbit/sec (see Figure 2.8). It also provides a 12 Gbit/sec connection to North America and 2.5 Gbit/sec to Japan.

GÉANT was originally due to finish in October 2004. However, because of its success and in order to permit a smooth succession to its successor, GÉANT-2, it was extended until the end of June 2005. GÉANT-2 was funded by the European Commission in September 2004 and has now superseded GÉANT. It will run until September 2008. In addition to a conventional packet switched network it will also provide some circuit switched connections of the sort discussed in Section 2.8.1, below.

TERENA (Trans-European Research and Education Networking Association)⁵ exists in order ‘to promote and participate in the development of a high quality international information and telecommunications infrastructure for the benefit of research and education’, as laid down in its statutes. Its members include most European NRENs and a few from further afield. It was founded in 1994 from a merger of the earlier organisations Réseaux Associés pour la Recherche Européenne (RARE) and European Academic and Research Network (EARN). It has a secretariat based in Amsterdam.

TERENA promotes the development of academic networks by pursuing activities in four categories: encouraging new activities, a technical programme, knowledge transfer

²See URL: <http://www.dante.net/ten-155/>

³See URL: <http://www.dante.net/>

⁴See URL: <http://www.dante.net/geant/>

⁵See URL: <http://www.terena.nl/>

and promoting the interests of its members. In particular, it coordinates a number of Special Interest Area groups and projects, organises conferences and workshops and publishes reports and other documents.

2.8.1 Experimental circuit-switched networks

Tanenbaum[2] (p415) has remarked that ‘Despite the fact that many people in the Internet community have an intense dislike for connection-oriented networking, the idea seems to keep coming back.’ A case in point is that many areas of research in science, medicine and engineering now require high data transfer rates between widely separated sites. Often substantial volumes of data will need to be moved between fixed end hosts over periods of hours or days. These requirements are difficult to meet using conventional packet-switched networks. Rather, they seem better suited to circuit-switched networks. Here a dedicated connection or circuit is established between two end hosts for the duration of the transfer. Briefly, all the packets then pass along the same, pre-configured path, thus avoiding the overheads of routing each packet individually. Circuit-switched networks can offer both higher transfer rates and more predictable behaviour.

Several NRENs have collaborated to create the Global Lambda Infrastructure facility (GLIF),⁶ an experimental testbed to pioneer the use of circuit-switched networks. The founding partners were StarLight⁷ in the US, NetherLight⁸ in The Netherlands and CANARIE⁹ in Canada. The UK has joined this collaboration through its new optical research network UKLight.¹⁰ The ESLEA (Exploitation of Switched Light Paths for e-Science Applications) Project¹¹ is pioneering the use of UKLight for real scientific work with applications in particle physics, radio astronomy, high-performance computing and medicine.

GLIF and its constituent networks are intended for experimental work rather than production systems. However, it is expected that production circuit-switched services will become available in due course. They are, for example, likely to be available in SuperJANET5, GÉANT-2, the next generation of the US back-bone and other networks.

⁶See URL: <http://www.glif.is>

⁷See URL: <http://www.startap.net/starlight/>

⁸See URL: <http://www.netherlight.net/info/home.jsp>

⁹See URL: <http://www.canarie.ca/>

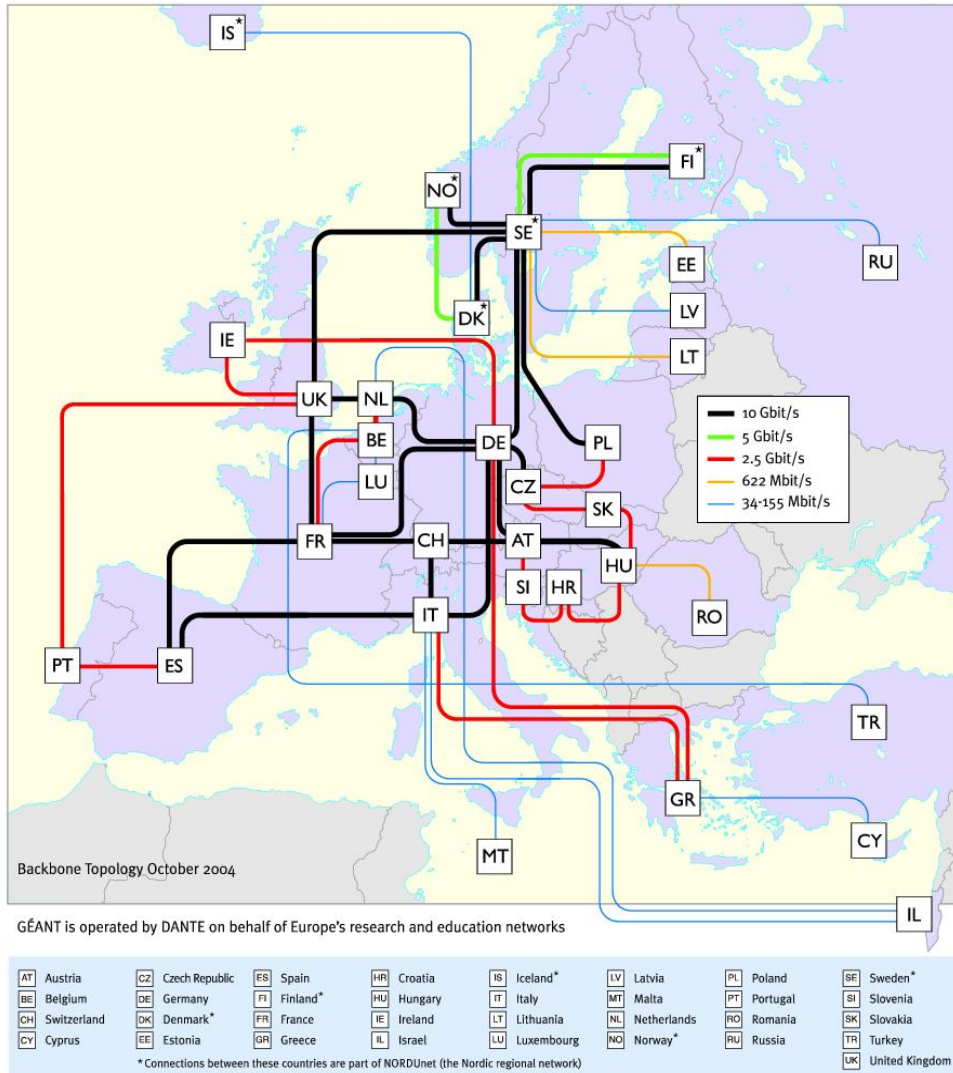
¹⁰See URL: <http://www.uklight.ac.uk/>

¹¹See URL: <http://www.eslea.uklight.ac.uk/>

GEANT

The world's most advanced international research network

Providing pan-European and international connectivity for research and education



GEANT is co-funded by The European Commission within its 5th R&D Framework programme

Information Society
Technologies
Contract No.
IST-2000-26417

Figure 2.8: The topology of the GEANT network in April 2004

Chapter 3

Performance Monitoring

In her *The Secrets of Rue St. Roche*[18] Janet Morgan tells the remarkable tale of a British spy-ring which operated in occupied Luxembourg during the closing months of the Great War. The agents recruited to the ring mostly comprised station masters and other railway employees who reported on the movement of German troop trains as they passed through the marshalling yards of the Grand Duchy *en route* to the front. The agents were trained to recognise unit insignia, differentiate different types of field-piece and count ammunition wagons. The information that they returned proved invaluable. What the agents were doing, apart from risking their lives, was monitoring the traffic on the Luxembourg railway network. Such monitoring of networks is commonplace, and is limited to neither railways nor wartime. It can have a number of purposes, ranging from determining the type of traffic and its destination (as here) to estimating the volume of traffic and predicting the performance of the network. Computer networks are no exception and there are various techniques and tools for monitoring their performance. This chapter is a brief introduction to these topics.

Before discussing network monitoring a few general remarks about network performance are probably in order. At first sight packet switched networks might seem a contrapted, ramshackle way to organise a high-speed network. The packets comprising a message are injected into the network and then meander through it, possibly by differing routes, and perhaps tarrying in the input queues of the various routers that they pass through along the way, before finally arriving at their destination where they are re-assembled into the original message. Compare this arrangement to circuit switching, where a direct circuit is established between the host and destination, through which packets simply pass unhindered and in order. In fact, a high data transfer rate was not one of the prime requirements which drove the development of packet switched networks. The idea of packet switching was developed by Paul Baran of the Rand Corporation in the early 1960s. He was working on communications systems for military bases which could survive a nuclear war. The prime requirement of such systems was that they should continue to function when parts of the network were not operational (having, literally, been blown up). This work led to the US ARPANET in the 1970s, which ultimately evolved into the present Internet. The idea of packet switching was also developed independently by Donald Davies of the National Physical Laboratory in the UK, but the modern Internet descends from ARPANET (see Hafner and Lyon[6] for a very readable account of these developments). Packet switching's robustness, flexibility and lack of a requirement for a centralised control are the features which made it suitable for use in rugged military communications systems, and these same features have led to its subsequent civilian success. However, it is important to remember that the technique was not originally designed to optimise the data transfer rate.

There is an apparent contradiction that WANs are often perceived to be slower than LANs, even though a local building or campus LAN usually has a lower capacity than a major trunk transmission line of a regional, national or inter-national network WAN. Part of the problem, of course, is that the WAN is shared with many more users than the LAN. However, this explanation is not the whole story: often traffic on the WAN will be light and the bottleneck will occur somewhere between the router terminating the trunk line and the user's end host. That is, the delay is in the 'last mile' to the user's machine. Finally, any firewall between two end hosts which are performing a transfer can affect the transfer rate. A firewall must examine every packet which it receives and check that the packet satisfies a potentially complex set of criteria before it allows the packet to pass. A firewall will usually run on a fast machine, but it can still potentially affect the transfer rate achieved.

Much of the rest of this chapter is based on *Fundamentals of Internet Measurement: A Tutorial* by N. Brownlee and C. Loosely[19]. This useful document can be consulted for further details of the techniques and tools described.

Perhaps surprisingly, there are some RFCs specifying metrics for network monitoring. These standards are developed and over-seen by the IETF IP Performance Metrics (IPPM) working group.¹ The basic standard is RFC 2330 *Framework for IP Performance Metrics*, but there are also a number of others[20].

For specifically GRID network monitoring there is an analogous working group, the GGF (Global GRID Forum) Network Measurements Working Group (NMWG).² This group has recently produced *A Hierarchy of Network Performance Characteristics for Grid Applications and Services*[21] which provides information to the GRID community about the current practice for measuring network performance. The NMWG work is at an earlier stage than that of the IPPM.

3.1 Types of Measurement

This section discusses the types of measurement which can be made in order to assess network performance. It also introduces several common metrics which have been developed for measuring network performance. Some of the different ways in which network performance might be measured are as follows.

Passive or active monitoring? In passive monitoring the traffic at some point in the network is simply monitored; perhaps the rate at which packets flow through a router or gateway is measured. The pertinent feature of passive monitoring is that it does not alter the operation or state of the network in any way. Ideally every packet flowing past the observation post should be monitored, though in practice it might be necessary to use a subset, perhaps every n^{th} . In active monitoring, by comparison, test packets are injected into the network in order to measure some aspect of its performance. For example, a test packet might be dispatched to a remote host in order to measure the time taken until a return package is received acknowledging that it has arrived. Such test packets necessarily perturb the network and if injected in sufficient numbers might alter its performance (and degrade it for other users).

Measure at one point or many? In order to get an overall view of the flow of data through a network it is tempting to measure it at a set of representative points. However, this approach is often a mistake: it is can be difficult to correlate or visualise the

¹See URL: <http://www.ietf.org/html.charters/ippm-charter.html>

²See URL: <http://nmwg.internet2.edu>

results. Rather, it is better to measure the traffic between two hosts by measuring the flow at points of ingress to, and egress from, the network, rather than internally within it.

Network or application performance? There are (at least) two approaches to measuring the performance of a network. One measures the performance of the network directly, either passively or by injecting test packets. The other involves running a test application program to perform some specified task (the obvious example is to transfer a file) and timing how long it takes to complete. These two approaches are measuring quite different things: one is measuring the behaviour of packets on the network, the other the behaviour of an application which involves using the network. Further, the two types of measurement are not equivalent: the performance of applications (such as file transfer) cannot easily be predicted by measuring the network performance and nor can the network performance be deduced by measuring the performance of applications. The IPPM's performance metrics[20] were developed, in part, to clarify the difference between measurements of network and application performance.

3.2 Common Metrics

A number of metrics are used to measure network performance. Some of the more common ones are listed in Table 3.1 and briefly described below. The discussion here follows the usage adopted by the IPPM; other interpretations may be encountered in the literature.

- latency
- jitter
- packet loss
- throughput
- link utilisation
- availability and reliability

Table 3.1: Common network performance metrics

Latency In common usage latency is the time that something takes to happen. In many kinds of network communication, once a packet has been dispatched nothing seems to happen until the sending host receives an acknowledgement packet indicating that the original packet reached its destination. Thus, a common measure of latency is the **round trip time**, the time between dispatch of a packet and receipt of an acknowledgement that it has reached its destination. It is also possible to make more complex latency measurements, perhaps timing the outward and return journeys separately or timing the individual hops between routers, though this is less common.

The common round trip time measure of latency comprises not just the time that the original packet and its acknowledgement spend travelling down transmission lines and waiting in the queues of routers, but also the time that the remote host takes to process the incoming packet and generate the acknowledgement. Because the time which elapses whilst the remote host performs this processing is not necessarily negligible the latency is a somewhat crude measure of network performance. Nonetheless it is widely used. In

order to minimise the processing time on the remote host a method implemented within the TCP/IP stack should be used (but note that this behaviour is already different to that of an application program using the network, which would require the processing to ascend all the way through the stack). Ping (see Section 3.3) is often used for this process.

Latency can vary for a number of reasons, including:

- a lightly loaded destination host will respond more quickly than a heavily loaded one,
- if the network is lightly loaded then the packets will spend less time in router queues,
- the route by which the packets travel between the source and destination hosts might change if the network configuration changes.

Variations can occur on a wide range of time-scales. Short-term variations are called **jitter** (below).

Jitter Short-term variation in the rate at which packets travel across a network is called jitter. Variation in the time it takes packets to reach their destination is **delay jitter**. The corresponding variation in the latency is **latency jitter**.

Packet loss Packet loss is the fraction (usually expressed as a percentage) of packets dispatched to a given destination host during some time interval for which an acknowledgement is never received. Such packets are referred to as being **lost**. There are a number of causes of lost packets. There may be hardware faults *en route*, or the packets might become corrupted by accumulating noise. However, such occurrences are relatively rare. A more common cause is packets being deliberately dropped from the queues of busy routers. Router queues are necessarily of finite size. If such a queue is full and packets are still arriving then some packets must be dropped. There are two alternatives. Packets can be dropped from the end of the queue, which favours applications which already have packets queued, or packets can be dropped from random locations in the queue, which distributes the losses more equitably amongst the various applications with packets queued. The latter technique is known as **random early detection (RED)**.

For applications which rely on all the packets in a message being transmitted correctly (such as file transfer) the higher levels in the protocol stack must check that all the packets dispatched have arrived correctly and re-send any which were lost. The TCP protocol provides these features. Conversely, there are other sorts of applications, such as streaming video or video-conferencing, which can tolerate some packet loss and where there is little point in re-sending lost packets (indeed they may no longer be available). Such applications will usually use the UDP protocol (see Section 2.3, above).

An additional important point is that a small fraction of lost (and resent) packets is part of the normal operation of TCP. It uses the fraction of lost packets to gauge its transmission rate: if the fraction becomes large then the transmitting host will reduce the rate at which it dispatches packets (on the assumption that they are being lost because the queue of some router in the path is filling up). Further, the default behaviour is ‘conservative’: if the loss rate goes up the dispatch rate will be substantially reduced and a long time allowed to elapse before any attempt it made to increase it again. This behaviour is ‘benign’ in the sense that it mitigates against one application hogging the network to the detriment of others, but it has implications for performance

(see Section 4.1 for a more detailed discussion). Finally, whilst a small packet loss rate is part of the normal operation of the network there are limits to the fraction of packet loss which is sustainable. As a rule of thumb, a network with a packet loss of 5-15% is severely congested, and one with a higher rate is likely to be unusable for most practical purposes.

Throughput Throughput is the rate at which data flow past some measurement point in the network. It can be measured in bits/sec, bytes/sec or packets/sec. It usually refers to all the traffic flowing past the measurement point, though it is possible to measure just a subset, perhaps just the Web traffic or the packets bound for a given destination. There are, however, a number of subtleties to throughput.

Firstly, the number of bytes transmitted across the network by an application, for example during a file transfer, is not the same as the number of bytes ultimately received by the application. There are overheads involved in splitting the file into packets: each packet must include its destination address and other control information. Also any lost packets will generate extra traffic when they are resent. The actual transfer rate across the network, including header overheads and resent packets, is sometimes called the **wire rate**.

A given link will have a maximum throughput or capacity which it can provide and usually this capacity must be shared between several users. It is useful to distinguish between:

capacity the throughput of which the link is capable,

utilised capacity the current traffic load excluding your traffic,

available capacity the remaining capacity, which is available to you. That is:

$$\text{available capacity} = \text{capacity} - \text{utilised capacity} \quad (3.1)$$

In a route with multiple hops the hop with the smallest available capacity is called the **tight link**,

achievable capacity is the fraction of the available capacity which you can utilise.

Usually it is not possible to utilise the entire available capacity because of limitations with the transmission protocol and end hosts.

Finally, throughput is measured by counting the traffic over an interval and care must be exercised to ensure that the interval chosen is appropriate. A long interval will average out transient bursts and lulls in traffic. A shorter interval will record these temporary effects, even if they are not important in the context of the measurement.

Link utilisation Link utilisation is used in the context of a link to an external network which is supplied by a telecommunications service provider. The link connecting the remote network to the gateway on the local LAN or MAN will have a maximum data rate, the **access rate**. The **link utilisation** is then simply the throughput (above) divided by the access rate and expressed as a percentage.

For some types of link the service provider might quote a **committed information rate (CIR)** rather than an access rate. In this case some traffic in excess of the CIR is permitted, but only for short periods. Here the link utilisation should be computed from the CIR rather than the access rate.

Availability and reliability Availability and reliability are not strictly IPPM metrics, though they are closely related. The availability is the fraction of time during a given period when the network is unavailable. Such periods of unavailability are called **out-ages**. Though this definition is unambiguous it is an over-simplification. Depending on the use to which the network is being put, frequent short outages might be more (or less) disruptive than one long one. The **mean time between failures** and **mean time to repair**, which have the obvious meanings, help to distinguish these cases. Also, the availability of ‘the network’ is not the same as the availability of a given remote host accessed through the network.

Reliability is related to both availability and packet loss. It is the frequency with which packets get corrupted, as distinct from being lost (the former happens because of either a malfunction in the network or transmission noise, the latter typically because a router queue fills up). But note that when calculating the packet loss (above) it is conventional to include corrupted packets as well as lost ones.

3.3 Simple Tools

Numerous tools are available for measuring network performance, and they vary greatly in approach, scope and applicability. A few simple ones, specifically ping, traceroute, pathchar, pchar and iperf, will be briefly described in this section and the following section will introduce the more modern ones. Usually the simple tools are run by a user on his own machine.³ They measure the latency and packet loss (see above) to a specified destination host and work by sending test packets to that host.

The tools usually emit a special type of packet, an **ICMP echo request packet** (see Section 2.3), often colloquially referred to as a **ping packet** (after the simplest of the tools, below). In order for the tools to work the destination host must be configured to reply to ping packets. In the early days of the Internet, when security was less of a concern, most sites would pass and respond to ping packets, so you could simply run the tools. However, nowadays system administrators tend to regard ping packets as intrusive and unwelcome, and configure their hosts not to acknowledge them. Similarly, firewalls are often configured not to permit their passage. In case of difficulty you will need to contact the system administrator of your destination host and attempt to persuade him to enable acknowledgement of ping packets. Section 3.3.1, below, gives some hints on how a firewall should be configured to pass ping packets.

A further caveat is that these tools establish that the destination host is reachable and willing to acknowledge ping packets. They do not establish that the host offers other services, such as file transfer. Also, it is tempting to select a router as the destination because routers should always acknowledge ping packets. However, routers will often process ping packets at low priority, leading to high round trip times. A final note of caution is that the tools should be used sparingly and with discretion. If used unwisely they can flood the network with packets, to the detriment of other users.

Because of firewall, and other, problems, if all you want to do is to confirm that a remote TCP server is available the simplest approach might be to **telnet** to it on its own port, for example:

```
telnet www.roe.ac.uk 80
```

³Alternatively, versions of ping, traceroute and other useful tools can be conveniently run from URL: <http://www.all-nettools.com/toolbox>

If all is well a message such as the following will appear:

```
Trying 192.108.120.192...
Connected to spider.roe.ac.uk (192.108.120.192).
Escape character is '^]'.

```

and you can quit by hitting the return key.

```
PING clapton.nesc.ed.ac.uk (129.215.30.12): 56 data bytes
64 bytes from 129.215.30.12: icmp_seq=0 ttl=249 time=1.5 ms
64 bytes from 129.215.30.12: icmp_seq=1 ttl=249 time=1.3 ms
64 bytes from 129.215.30.12: icmp_seq=2 ttl=249 time=1.4 ms
64 bytes from 129.215.30.12: icmp_seq=3 ttl=249 time=2.2 ms
64 bytes from 129.215.30.12: icmp_seq=4 ttl=249 time=1.3 ms
64 bytes from 129.215.30.12: icmp_seq=5 ttl=249 time=1.2 ms
64 bytes from 129.215.30.12: icmp_seq=6 ttl=249 time=2.1 ms
64 bytes from 129.215.30.12: icmp_seq=7 ttl=249 time=1.9 ms
64 bytes from 129.215.30.12: icmp_seq=8 ttl=249 time=2.1 ms

--- clapton.nesc.ed.ac.uk ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 1.2/1.6/2.2 ms

```

Figure 3.1: Unix ping output. Ping continues to run until it is interrupted, typically by `<ctrl-c>` on a Unix system. The final summary statistics are then displayed

Ping Ping is a utility intended for determining the availability of a destination host. It runs on the user's own machine and sends ping packets (above) to the specified destination host. The destination host acknowledges by returning a ping packet and ping displays the round trip time. Ping is the simplest of the tools and allows you to determine whether a destination host is reachable. Ping is available for a variety of platforms. On Unix it is usually supplied as part of the operating system. The Unix version reports the following information (see the summary at the bottom of Figure 3.1):

- the name of the destination host (`clapton.nesc.ed.ac.uk` in the example),
- a count of the packets transmitted,
- a count of the packets received back,
- the packet loss, expressed as a percentage,
- the minimum, average and maximum round trip time.

There are important caveats to using ping. Specifically, there are circumstances where ping will respond when the remote server is hanging or processes on the server are not

running. Conversely, as mentioned above, ping might fail to obtain a response when all is well because ping traffic is being blocked by a firewall *en route*.

Ping was written by Michael Muuss in 1983. The name comes from the sound of a returned sonar pulse. An acronym (Packet InterNet Groper) was coined later. For further information see the appropriate `man` page or Stevens[4] Chapter 7, pp85-96.

Traceroute Traceroute is similar to ping in that the user specifies a destination host and it sends packets to that host, receives acknowledgements back and displays the results. Also like ping it is usually supplied as part of the operating system. However, unlike ping it produces a hop-by-hop listing of each router that the packets pass through *en route* to their destination and displays the latency for each hop. It only shows the hops on the outward leg of the journey, not the return hops. Since it is inherent to the way that packet-switching works that each router knows only the address of the next router to which it should direct a packet, rather than the full path, it is surprising that traceroute can work at all. The algorithm that it uses has famously been described as a ‘cool hack’ and is beyond the scope of this document. Traceroute is a common diagnostic tool which is particularly useful for determining why a remote host is not responding to ping. It was written by Van Jacobsen around 1987. For further information see the appropriate `man` page or Stevens[4] Chapter 8, pp97-110.

Pathchar, pchar and iperf There are number of utilities broadly similar to traceroute but which are more concerned with measuring throughput than establishing a route. Three useful ones are pathchar, pchar and iperf. Pathchar⁴ was developed to assist in diagnosing network congestion problems. It measures the data transfer rate, delay, average queue and loss rate for every hop from the user’s machine to the chosen destination host. It was written by Van Jacobsen, the author of traceroute. Pchar⁵ provides similar functionality and is based on the same algorithms. It was written by Bruce Mah. Some example output from pchar is shown in Figure 3.2.

Iperf⁶ is also a tool for measuring the available TCP bandwidth to a destination host. It allows various TCP parameters and UDP datagrams to be tuned. It reports the available bandwidth, delay jitter and datagram loss. Unlike the other tools it requires software to be installed on the destination as well as the source host. It was developed by Ajay Tirumala and colleagues at the National Laboratory for Applied Network Research, San Diego Supercomputer Center.

3.3.1 Firewall configuration for ping, traceroute and pchar

Often firewalls will block the traffic generated by ping and traceroute. The traffic could be blocked by either a local firewall protecting your source host (which is swallowing outgoing packets) and/or by a remote firewall protecting the destination host (which is swallowing incoming packets). If you find that traffic is being blocked you will need to persuade whoever is responsible for the appropriate firewall to change its rules to allow access. *Building Internet Firewalls* by E.D. Zwicky, S. Cooper and D.B. Chapman[22] gives the following advice about allowing ping and traceroute traffic:

⁴See URL: <http://www.caida.org/tools/utilities/others/pathchar/>

⁵See URL: <http://www.kitchenlab.org/www/bmah/Software/pchar/>

⁶See URL: <http://dast.nlanr.net/Projects/iperf/>

```

pchar to grus.jb.man.ac.uk (130.88.24.231) 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: 192.108.120.198 (grendel12.roe.ac.uk)
  Partial loss:      292 / 1472 (19%)
  Partial char:      rtt = 0.079436 ms, (b = 0.000165 ms/B), r2 = 0.999861
                    stddev rtt = 0.000361, stddev b = 0.000000
  Partial queueing:  avg = 0.000038 ms (232 bytes)
  Hop char:          rtt = 0.079436 ms, bw = 48517.401423 Kbps
  Hop queueing:      avg = 0.000038 ms (232 bytes)
1: 192.108.120.254 (teine.roe.ac.uk)
  Partial loss:      0 / 1472 (0%)
  Partial char:      rtt = 0.554800 ms, (b = 0.000272 ms/B), r2 = 0.999015
                    stddev rtt = 0.001042, stddev b = 0.000001
  Partial queueing:  avg = 0.000116 ms (954 bytes)
  Hop char:          rtt = 0.475364 ms, bw = 74436.428087 Kbps
  Hop queueing:      avg = 0.000078 ms (722 bytes)
2: 192.108.120.14 (eastman.roe.ac.uk)
  Partial loss:      0 / 1472 (0%)
  Partial char:      rtt = 0.374863 ms, (b = 0.000359 ms/B), r2 = 0.998957
                    stddev rtt = 0.001413, stddev b = 0.000002
  Partial queueing:  avg = 0.000733 ms (8078 bytes)
  Hop char:          rtt = --.--- ms, bw = 92310.976695 Kbps
  Hop queueing:      avg = 0.000617 ms (7124 bytes)
...
11: 194.66.21.241 (gw-uom.mcc.ac.uk)
  Partial loss:      0 / 1472 (0%)
  Partial char:      rtt = 7.060438 ms, (b = 0.000502 ms/B), r2 = 0.998086
                    stddev rtt = 0.002677, stddev b = 0.000003
  Partial queueing:  avg = 0.000182 ms (50091 bytes)
  Hop char:          rtt = 0.743205 ms, bw = 97985.081673 Kbps
  Hop queueing:      avg = -0.000049 ms (0 bytes)
12: 194.66.25.38 (gw-jodrell.netnw.net.uk)
  Partial loss:      0 / 1472 (0%)
  Partial char:      rtt = 7.256724 ms, (b = 0.000772 ms/B), r2 = 0.997328
                    stddev rtt = 0.005193, stddev b = 0.000006
  Partial queueing:  avg = 0.000164 ms (50091 bytes)
  Hop char:          rtt = 0.196286 ms, bw = 29658.348766 Kbps
  Hop queueing:      avg = -0.000018 ms (0 bytes)
13: 130.88.24.231 (grus.jb.man.ac.uk)
  Path length:      13 hops
  Path char:        rtt = 7.256724 ms r2 = 0.997328
  Path bottleneck:  29658.348766 Kbps
  Path pipe:        26902 bytes
  Path queueing:    average = 0.000164 ms (50091 bytes)
  Start time:       Fri Nov 12 13:00:01 2004
  End time:         Fri Nov 12 14:33:17 2004

```

Figure 3.2: Example output showing a pchar probe from the Royal Observatory Edinburgh to the Jodrell Bank Observatory in Cheshire. Statistics are shown for each hop along the route and final statistics for the entire route are given. Some intermediate hops have been removed for clarity

There are few risks for outbound *ping* or *traceroute*, and those risks can be avoided by using them without hostname resolution. Inbound *ping* and *traceroute*, however, pose significant risks. *ping*, in particular, is a frequent basis for denial of service attacks. *ping* and *traceroute* can both be used to determine which hosts at your site exist, as a preliminary step to attacking them. For this reason, many sites either prevent or limit the relevant packets inbound (p57).

Below are some hints, mostly also from Zwicky, Cooper and Chapman[22] (pp647-652) about the traffic which should be allowed.

ping

outbound (i.e. to ping a remote host) allow ICMP echo request packets outbound and ICMP echo response packets inbound.

inbound (i.e. to allow a local host to be pinged from without) allow ICMP echo request packets inbound and ICMP echo response packets outbound.

If possible a maximum size of accepted ICMP echo request packets should be specified. Limiting the inbound size is a defence against some kinds of denial service attacks; limiting the outbound size is a politeness to remote systems.

traceroute

outbound (i.e. to traceroute to a remote host) allow the UDP and ICMP packets constructed by traceroute outbound and the appropriate ICMP packets (particularly ‘time to live exceeded’ and ‘service unavailable’) inbound.

inbound (i.e. to allow a local host to be the destination of a remote traceroute) allow the UDP and ICMP packets constructed by traceroute inbound and the appropriate ICMP packets (as above) outbound.

The ability to emit and receive these packets might be restricted to the computers on which traceroute needs to be run, in order to limit the variety of UDP packets allowed through the firewall. UDP packets can be used in various sorts of denial of service attacks. The considerations for ICMP packets are as for ping, above.

Traceroute uses a number of UDP port numbers in its probes. The range used in a given probe is:

$$base + hops - 1$$

Where *base* is the lowest port number used. The default is 33434, but this can be overridden using the `-p` option. *hops* is the number of hops from the source to the destination. The default maximum number of hops permitted in a given run is 30, though this can be overridden with the `-m` option. It is possible to get from the Royal Observatory Edinburgh to the Joint Astronomy Centre in Hawaii in 20-odd hops. The range of ports used must be open for outbound traffic from the source host. However, traceroute requires that nothing on the destination host is listening on these ports, so that an ICMP ‘port unreachable’ packet will be returned to terminate the probe.

pchar The requirements for pchar are similar to those for traceroute, above. However, the default base for the UDP port range is 32768, and the option to override the default is `-P`. Like traceroute, the default maximum number of hops is 30, but the option to override it is `-H`.

3.3.2 Hints on using pchar

The following list gives a few hints on using pchar.

1. pchar has a ‘tiny traceroute’ mode in which it emulates traceroute and traces out the route to the destination host. This mode is useful to check that you can establish a route to the destination. It is invoked by:

```
pchar -M trout destination-host
```

The `-M` flag specifies pchar’s mode of operation and the `trout` option corresponds to the ‘tiny traceroute’ mode.

2. In order to generate measurements which are statistically reliable, pchar makes multiple probes at each hop along its route. Consequently it can take a significant time to complete; perhaps tens of minutes, or even longer for intercontinental probes. When making an initial test it is often useful to override some of the defaults in order to generate less reliable results more quickly. This procedure is useful to check that you have the required access along the entire route and are not, for example, encountering firewall problems at one end or the other (it is possible for pchar to complete successfully in its ‘tiny traceroute’ mode, above, but still generate errors in its normal mode). Two alternative invocations of this sort are:

```
pchar -R 2 destination-host  
pchar -m 93 destination-host
```

Pchar sends probes using packets of different sizes and for each size of packet the probes are repeated a number of times. The `-R` flag specifies the number of times probes using a given packet size are repeated. The default if the flag is omitted is 32, so specifying a value of 2 should speed up operation by a factor of 16 (but will yield results that are much less statistically significant). The `-m` flag specifies the maximum size of probe packet sent, in bytes. Again, setting a low value should speed up the operation of pchar. Note however, that setting the value of this flag will sometimes result in error conditions being returned by remote hosts, particularly from sites overseas.

3. Thus, often a series of probes to a given destination host will start with a probe using the `-M trout` option to establish that the route is available, followed by a quick probe using, for example, the `-R 2` option to check that none of the necessary traffic is blocked. Finally there will be a series of probes to measure the network capacity to the chosen destination. Often for these latter probes it will be adequate to run pchar without overriding any of its defaults.
4. Sometimes some of the quantities listed in the pchar output will be shown as zero or a row of dashes (or in early versions of pchar they may contain negative values). Such values do not necessarily indicate that anything is amiss. The pchar FAQ advises that these values are artefacts of the analysis that pchar performs on the measurements that it gathers. They can be caused by:
 - excessive packet loss along a link,
 - variations in the amount of time taken for routers to process packets,
 - transient changes in network conditions for hops that have been previously measured.

3.4 Modern Tools

The traditional tools can be somewhat difficult to use, with arcane command-line options and obscure output. In recent years a number of new tools have been developed which are more user-friendly and comprehensible to people who are not network experts. Typically these modern tools will be driven from a GUI, though this GUI may be invoking standard tools ‘behind the scenes.’ The tools attempt to conduct a dialogue with the user in terms which he understands, such as which sites are available, what services they offer and what throughput can be expected. A typical example is the SURFNet Detective⁷ (see Figure 3.3). It attempts to test:

- connectivity,
- multicasting,
- TCP throughput.

Beneath the GUI it is actually using ping, traceroute and iperf. SURFNet Detective communicates with servers in the Dutch SURFNet, so it is of limited usefulness outside Holland. Nonetheless, it is typical of the more modern tools.

3.5 Network Monitoring and Weather Maps

Most organisations responsible for running wide-area networks routinely monitor network performance and collect and analyse the resulting data. Often these data are made publicly available, so that users can check network performance. An example is the Netsight⁸ facility provided by UKERNA to give access to performance data about JANET.

Rather than (or as well as) presenting the results as tables and charts a more recent approach is to display network performance data as a ‘weather map.’ Such maps typically show a more-or-less schematic map of the sites in a network, with a graphical representation of the performance between them[23]. An example is the Internet Traffic Report.⁹

A number of projects are developing suites of software for network monitoring. Often the approach will be to write scripts which run simple tools such as ping, traceroute and iperf (see Section 3.3 above) and collect and analyse the results. Perhaps the most relevant of these projects is GridMon,¹⁰ which is based at the Daresbury Laboratory. It is funded by the UK DTI (Department of Trade and Industry) and is associated with the European DataGrid Project. GridMon is developing a toolkit of applications for monitoring network performance. In practice these applications are mostly about measuring generic network performance rather than being specifically tailored for GRID applications. The first version of the toolkit has been deployed at most of the UK e-Science centres¹¹ and it is hoped to deploy subsequent versions more widely. The GridMon software is installed at the host sites, from where it makes a series of probes every half hour. The results are collected, analysed and presented as a series of graphs and statistics available from a public Web page.

⁷See URL: http://detective.surfnet.nl/en/index_en.html

⁸See URL: <http://www.ja.net/services/network-services/netsight/>

⁹See URL: <http://www.internettrafficreport.com/main.htm>

¹⁰See URL: <http://gridmon.dl.ac.uk/>

¹¹See URL: <http://www.nesc.ac.uk/centres>

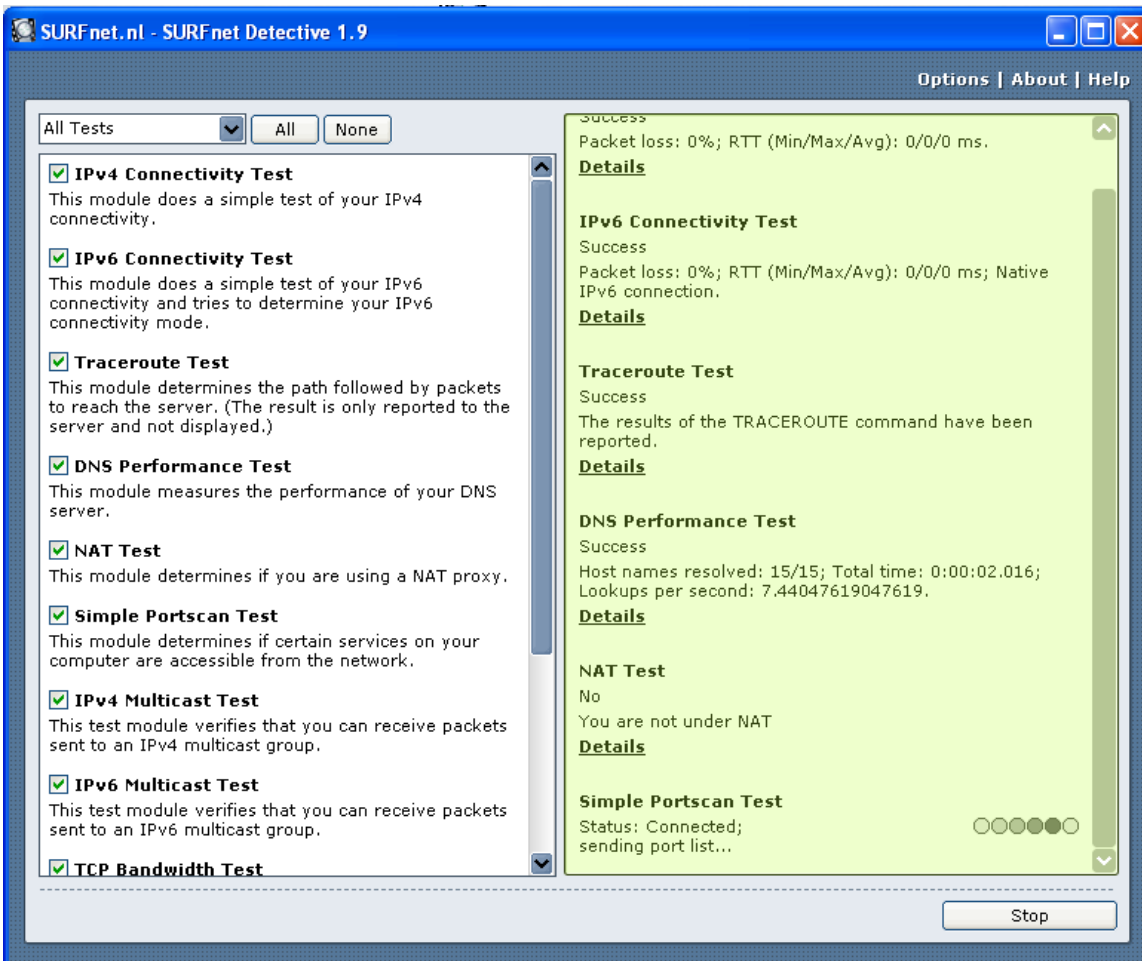


Figure 3.3: SURFNet Detective

3.6 Additional Tools

Many more tools are available than the few mentioned here. Inevitably some are more useful than others. Extensive lists are maintained by:

CAIDA (Co-operative Association for Internet Data Analysis):

<http://www.caida.org/tools/>

SLAC (Stanford Linear ACcelerator):

<http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html>

Chapter 4

Obtaining Good Network Performance

Harpists spend 90 percent of their lives tuning their harps and 10 percent playing out of tune.

Igor Stravinsky.

This chapter introduces some techniques which can be adopted in order to improve the performance of applications that transfer data across the Internet. Much of the material is based on the talk by Brian Tierney at the second *Networks for Non-Networkers* workshop (see Section 1.1). Further details are given in his recent article *TCP Tuning and Network Troubleshooting*[24].

A wide range of approaches to improving performance are possible, from adjusting TCP/IP tuning parameters to considerations to be borne in mind when buying the hardware on which the applications are to run. Obviously, these various approaches will be applicable in different circumstances. However, before describing them it is sensible to outline the TCP congestion control algorithm because it affects the behaviour of the network and hence the steps to improve network performance. As a rough benchmark of the performance which can be achieved with conventional, but modern hardware, and some network tuning, but without resorting to special techniques, transfer rates of 20 - 50 Mbyte/sec should be possible across JANET within the UK[25].

4.1 TCP Congestion Control

As mentioned briefly in Chapter 2, TCP automatically adjusts the rate at which it emits packets in order to avoid congestion in the network. Furthermore, its behaviour is deliberately ‘conservative’ or ‘benign’: if an application detects that it is exceeding the capacity of the network it will substantially reduce its transmission rate in order to avoid affecting the performance of other users. The complete congestion control behaviour of TCP is complex, but it is useful to have at least an outline understanding of it.

TCP uses lost packets to control its transmission rate. If packets are not being lost it assumes that there is spare capacity and attempts to increase the transmission rate. If they are lost it assumes that this is because of congestion and decreases the rate. The underlying assumption here is that the sole cause of losing packets is packets being discarded from the queues of

busy routers and none are lost due to being corrupted during transmission. This assumption is reasonable for modern fibre-optic networks (but certainly is not true for wireless networks).

In TCP an acknowledgement is returned for every packet sent. However, clearly the source host does not wait for every packet to be acknowledged before emitting the next (which really would lead to really poor performance). Rather it has a **window** of n packets which are allowed to be in transit simultaneously. It will not emit packet $n + 1$ until at least one acknowledgement is received, nor emit $n + 2$ until there is a second acknowledgement *etc.* An important consequence of this approach is that the speed with which TCP can respond to changes in networks depends on the round trip time.

Ideally the window size should be set to correspond to the smaller of the rates at which the source can emit and the destination receive packets. If it is smaller than either then both machines will spend some time idle while packets propagate through the network. The size of the window is set to the minimum of:

- a **receive buffer** specified by the destination host,
- the source host's estimate of the network capacity, the so-called **congestion window**.

The receive buffer is simply related to the rate at which the destination host thinks that it can process incoming packets. When a transfer starts the congestion window is initially set deliberately small. As the transfer proceeds it is modified:

- if no congestion is detected (that is, no packets are lost) it is increased by a constant amount,
- if congestion is detected (that is, one or more packets are lost) it is halved.

This algorithm is known as **AIMD** (Additive Increase, Multiplicative Decrease) and is illustrated in Figure 4.1. This description is something of a simplification; see Tanenbaum[2] pp547-549. for further details.

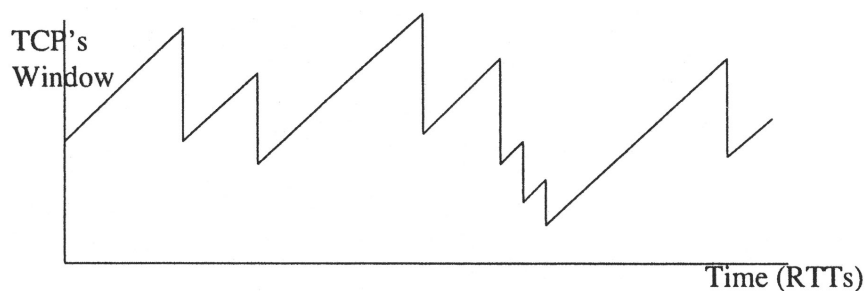


Figure 4.1: Congestion control in TCP/IP using the AIMD (Additive Increase, Multiplicative Decrease) algorithm

Some of the consequences are immediately clear. As soon as a packet is lost the window size is halved. The rate at which it will increase depends on the round trip time (nothing

will happen until acknowledgements are received). This behaviour is particularly acute for high-capacity, long-distance WANs and is exacerbated because the rate at which the window is increased will often be unnecessarily low. The optimal size for the congestion window is the bandwidth delay product (BDP) for the link. BDP is described in the following section.

4.2 Bandwidth Delay Product and Network Performance

The **bandwidth delay product** (BDP) is defined as:

$$\text{BDP} = \text{bandwidth} \times \text{delay} \tag{4.1}$$

where the delay is simply the travel time for a packet to travel across the link from the source to destination host. For a symmetric path the delay will be half the round trip time (RTT). In order to fully use capacity of the link the number of bytes which must be kept simultaneously in transit across the link is equal to the BDP.

The BDP becomes a large number for so-called ‘long fat pipes’: high-capacity, long distance links such as those found in an inter-continental WAN. In this case both the terms of which the BDP is a product are large numbers. High BDP links have two consequences for the performance of TCP.

- If the congestion window or receive buffer are smaller than the BDP then the full capacity of the link will not be used.
- If packets are lost then TCP will take a long time to recover. TCP can only respond on a time-scale of the round trip time (typically twice the delay) and because of the AIMD algorithm each increase can only be a small step. Round trip times of less than 10 msec in the UK and 150 msec across the Atlantic are typical (see also Table 4.1). The practical upshot is that it can take tens of minutes for a transatlantic transfer to recover from a single instance of congestion.

Path	Round trip time (msec)	Light travel time (msec)
LAN	1	-
Within the UK	< 10	3.6
UK to Europe	15-30	6.1
UK to North America	≈ 150	42.4

Table 4.1: Typical round trip times. The rightmost column shows the corresponding round trip time for light *in vacuo* (the internal UK entry corresponds to the distance from London to Edinburgh, the European entry to London to Berlin and the transatlantic one to London to Chicago)

4.3 TCP Tuning Parameters

The standard TCP implementations come with a set of ‘tuning parameters’, such as the receive buffer size mentioned in Section 4.1. The default values of these parameters with which TCP implementations are usually shipped have not changed for some years and are usually un-necessarily small for modern networks and computers. Many TCP installations continue to use these inappropriate default settings, and significant performance improvements can often be obtained simply by setting the parameters to appropriate values.

Table 4.2 lists the various parameters for a Linux system and shows the default and some suggested values. There will be equivalent parameters for other variants of Unix. Simply adopting the improved values will probably yield some improvement in performance. However, ideally you should calculate and set values appropriate for the capacity of the link that you are using. The individual parameters are described in the documentation usually included with Linux systems[26].

Parameter	Default	Suggested
<code>/proc/sys/net/ipv4/tcp_timestamps</code>	1	1
<code>/proc/sys/net/ipv4/tcp_window_scaling</code>	1	1
<code>/proc/sys/net/ipv4/tcp_sack</code>	1	1
<code>/proc/sys/net/core/wmem_max</code>	65535	8388608
<code>/proc/sys/net/core/rmem_max</code>	65535	8388608
<code>/proc/sys/net/ipv4/tcp_rmem</code>	4096	4096
	87380	87380
	174760	4194304
<code>/proc/sys/net/ipv4/tcp_wmem</code>	4096	4096
	16384	65536
	131072	4194304
<code>/proc/sys/net/ipv4/tcp_mem</code>	48128	4194304
	48640	4194304
	49152	4194304

Table 4.2: Default and suggested TCP parameters

Finally, if you are using the SCP file transfer utility (see Section 5.3) then the transfer rate may be limited by SCP/SSH’s own static internal buffers, which are unaffected by changing the TCP tuning parameters. A patch for SCP/SSH is available to overcome this limitation; see Section 5.3.2 for details.

4.3.1 Calculating the buffer size

The TCP socket send and receive buffer sizes should both be set to:

$$2 \times \text{BDP} \tag{4.2}$$

or equivalently (because the RTT is in practice twice the delay):

$$\text{bandwidth} \times \text{RTT} \tag{4.3}$$

It is important to use a reasonably correct value. If the buffers are too small the TCP window will never fully open. If they are too large the source can swamp the destination causing TCP to decrease its transfer rate. Thus values should ideally be calculated specifically for each link and, in particular, different values are likely to be required for a WAN and the local LAN. However, the chosen value does not need to be precise: approximate values within a factor of two or three are usually adequate. Table 4.3 lists some buffer sizes for typical links.

Path	Round trip time (msec)	Throughput (Mbit/sec)	Buffer size (Mbyte)
Within the UK	5	1000	0.625
UK to Europe	25	500	1.56
UK to North America	150	500	9.4
UK to Japan	260	150	4.9

Table 4.3: Optimal TCP send and receive buffer sizes for various paths

4.3.2 Examining and setting the parameters

On Linux systems the parameters look like the names of files but are actually operating system end points. Ordinary users can usually display them, but setting them normally requires a system administrator's privileges (and is best done by someone with a system administrator's expertise). The values of individual parameters can be displayed either with `cat`, for example:

```
cat /proc/sys/net/core/wmem_max
```

or with the `sysctl` command:

```
/sbin/sysctl net/core/wmem_max
```

Their values can also be set by using `sysctl`:

```
/sbin/sysctl -w net/core/wmem_max=8388608
```

For further details see the `sysctl` man page. The parameters which should be set to your calculated value for the TCP buffer size are:

```
/proc/sys/net/core/wmem_max  
/proc/sys/net/core/rmem_max
```

and the final values of:

```
/proc/sys/net/ipv4/tcp_rmem  
/proc/sys/net/ipv4/tcp_wmem  
/proc/sys/net/ipv4/tcp_mem
```

An alternative way to change the values is to add entries of the form shown in Figure 4.2 to file:

```
/etc/sysctl.conf
```

and type:

```
/sbin/sysctl -p

# increase TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

Figure 4.2: Example fragment of `/etc/sysctl.conf` file for setting TCP tuning parameters

4.3.3 Checklist

This section gives a brief checklist of the steps involved in resetting the TCP tuning buffers to values suitable for your transfers.

1. Run and time a representative set of transfers in order to estimate the basic transfer rate that you are getting before attempting to tune the system. You will probably want to run several transfers at different times of the day *etc.*
2. Use `pchar` (see Section 3.3) to determine the bandwidth and round trip time (RTT) of your route. Again you will probably want to run several `pchar` probes at different times of the day to get representative results. The bandwidth and RTT can be simply determined from the `pchar` output:
 - RTT is read from the summary statistics at the end of the `pchar` output,
 - the bandwidth might be the ‘path bottleneck’ given at the end of the `pchar` output. However, if the value corresponds to the first or last hops you might want to ignore these values and choose the next smallest bandwidth from amongst the remaining hops.
3. Calculate the required buffer size, which is the bandwidth, RTT product (see Equation 4.3).
4. Set the appropriate tuning parameters to this size, as described in Section 4.3.2.
5. If you are using SCP ensure that you are using the high-performance patch (see Section 5.3.2).
6. Repeat the set of tests run in step 1 and review the improvement in performance.

4.4 Parallel Streams

TCP simulates a connection-oriented virtual circuit (or **stream**) between the source and destination hosts on top of the connectionless IP layer. Parallel streams are a way of obtaining improved performance by simply simultaneously opening several separate streams between the end hosts and sharing the transfer between these streams.

The advantage of parallel streams is that, unlike setting tuning parameters, which require system administrator privilege, they require no special privileges, and so are available to ordinary users. FTP Accelerators (see Section 5.6) use parallel streams. Running several SCP sessions simultaneously will also create a corresponding number of parallel streams.

Each extra stream puts an additional load on the CPU, and ultimately all the available CPU time will be being used. Adding further streams will not improve the transfer rate. You will need to experiment to determine the number of streams which give the best compromise between CPU loading and overall throughput for your route and end hosts. Usually this number will be less than ten, and adding more streams yields diminishing returns.

4.5 TCP Implementations

There are various implementations of the standard TCP protocol:

- TCP Tahoe
- TCP Reno
- TCP NewReno
- TCP Sack

Tahoe and Reno are now obsolete and should be avoided. NewReno and Sack are still current and are to be preferred.

4.5.1 Selective acknowledgements

Selective acknowledgements (or ‘sacks’) are an optimisation of TCP which make it much more robust to multiple packet losses within a single window. They are available in the standard implementation Sack (as its name implies). Note, however, that on an uncongested Gigabit network they can actually degrade performance. On Linux systems they can be disabled by:

```
/sbin/sysctl -w net.ipv4.tcp_sack=0
```

4.5.2 Fast TCP implementations

There are several ‘fast’ implementations of TCP which aim to give improved performance. They work by modifying or replacing the AIMD algorithm and all are currently to some degree experimental. However, they still operate on standard TCP packets and so can usually inter-operate with standard TCP implementations. These fast implementations can often give significantly improved performance, though they are usually less stable and predictable than standard TCP. Also, they can sometimes have a deleterious effect on other traffic on the

network. *Before experimenting with them you should ensure that you have permission to use them over your entire network.* The fast implementations currently available are:¹

Scalable TCP (STCP)²
High Speed TCP³
Fast TCP⁴
TCP Westwood+

TCP Westwood+ is available as part of the Linux kernel 2.6.6. To invoke it set:

```
/sbin/sysctl -w /proc/sys/net/ipv4/tcp_westwood=1
```

In addition Falk *et al.*[27] have described the Explicit Control Protocol (XCP). Unlike the above fast TCP implementation, XCP requires modifications to any routers through which its traffic passes, thus making it more difficult to deploy. Falk *et al.* also briefly, but usefully, review the other fast implementations and He *et al.*[28] have recently reviewed the protocols and various related applications (such as GridFTP; see Section 5.4).

4.5.3 Packet size

In modern, reliable networks it is feasible to use packets larger than the standard 64 Kbytes. Such packets are called **jumboframes** or **jumbograms**. Larger packets include more data per packet, so fewer packets need be transmitted, thus reducing the overheads associated with handling packets and improving performance.

4.6 Choosing Hardware

Obtaining good network performance is not simply a matter of tuning an existing setup or choosing suitable software. Clearly, the hardware used will also determine the performance available. Hardware considerations are mostly beyond the scope of this document, though a few brief remarks follow.

When planning new systems and projects the network throughput required should be considered from the outset and appropriate hardware bought. The performance of the overall system will be determined not just by the computers acting as the end hosts but also by the disks attached to these computers and the LANs connecting them. Users, even ones running their own projects, are unlikely to be able to affect the purchase of MANs and WANs. They are, however, likely to be able to choose their own end hosts and might be able to influence the provision of the local LAN. Better performance is usually obtained by using modern hardware.

¹For a comparison see URL:

<http://www-iepm.sslac.stanford.edu/monitoring/bulk/tcpstacks/summary.html>

²See URL: <http://www-lce.eng.cam.ac.uk/~ctk21/scalable/>

³See URL: <http://www.icir.org/floyd/hstcp.html>

⁴See URL: <http://netlab.caltech.edu/FAST/>

4.6.1 Host computer

When choosing a host computer the straightforward CPU speed, though important, is not the only quantity to consider. In order to achieve high data transfer rates some desirable features are as follows.

- The Network Interface Cards (NICs) should be well designed. Ones which use the advanced PCI (Peripheral Component Interconnect) commands will allow efficient use of memory. Also the NIC drivers should be well written.
- Data crosses the memory bus at least 3 times, so the CPU-Memory bandwidth is important, as well as PCI bandwidth.
- Use motherboards with multiple 64 bit PCI-X buses to separate the data transfers. Also, a 32 bit 33 MHz bus is too slow for Gigabit transfer rates; a 64 bit 33 MHz bus will be more than 80% used.
- To sustain Gigabit transfer rates fast CPU rates are needed. Typically, for a 1 Gbit/sec transfer the processor speed should be no less than 3 GHz.

4.6.2 Disks

Disk I/O speeds are much slower than CPU speeds, and in some applications that are reading or writing large quantities of data to or from disk they can be the limiting factor. In these cases expedients such as ‘disk striping’ (that is, splitting the contents of a file across different disk volumes) might improve performance.

4.6.3 LAN

The performance of the LAN connecting the hosts to the network is also obviously important. The total volume of traffic to be carried on the LAN should be considered, not just the traffic generated by any single project. When planning new projects which will make extensive use of the LAN the advice of the network administrators responsible for it should be sought at the outset. It is often prudent to build a LAN from good quality components. Such kit might be somewhat more expensive, but it will often have fewer lost packets and hence give better performance.

4.7 Further Information

A useful recent article on TCP tuning is Brian Tierney’s *TCP Tuning and Network Troubleshooting*[24]. Also numerous sites are available which give hints on network tuning. Some useful ones are listed below.

GridMon Project:

http://gridmon.dl.ac.uk/tcp_tuning.html

Lawrence Berkeley National Laboratory:

<http://dsd.lbl.gov/TCP-tuning/TCP-tuning.html>

Pittsburgh Supercomputing Center:

http://www.psc.edu/networking/perf_tune.html

Stanford Linear Accelerator:

<http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/>

Chapter 5

File Transfer Protocols

Come, all young men of learning good,
A warning take by me.
I'll have you quit night walking
And shun bad company;
I'll have you quit night walking
Or else you'll rue the day,
And you will be transported
And go to Botany Bay.

Botany Bay,

Anonymous English folk-song, early nineteenth century,
collected early in the twentieth century by Cecil Sharp.

An important use of computer networks is to transfer files and similar types of data. Such usage is sometimes referred to as **bulk data transfer** to distinguish it from more interactive types of usage. The size of the files transferred can vary enormously, from a few bytes to Gigabytes or Terabytes. The routine transfer of large files is becoming increasingly common. Various file transfer protocols are available to move these data. Typically the protocols operate on top of the TCP/IP stack. In addition to giving different transfer speeds they will usually also offer different features and provide different levels of security and reliability. This chapter briefly introduces some of the more common transfer protocols and looks at techniques for improving their performance. Numerous different file transfer protocols are available. Some which are either common or are particularly suitable for transferring large amounts of data are listed in Table 5.1 and briefly discussed below. However, first the desiderata for a file transfer protocol are summarised.

5.1 Desiderata for a File Transfer Protocol

Some features which are usually desirable in a file transfer protocol are that it should be:

- reliable,
- efficient,
- secure,

Protocol	URL
FTP	-
SCP	http://www.openssh.com/ (OpenSSH)
GridFTP	http://www.globus.org/ (Globus home page)
BBFTP	http://doc.in2p3.fr/bbftp/
BitTorrent	http://bittorrent.com/

Table 5.1: File transfer protocols

- able to support third party transfer.

The first two criteria are largely self-evident. The protocol should be reliable and robust: it should deliver the data in their entirety and uncorrupted. If it fails to do so it should unflinchingly and unambiguously report an error message. The protocol should be efficient: it should deliver the data in a timely fashion. The requirement to support third party transfers is perhaps slightly less obvious. It is useful, for example, when a ‘master’ process is transferring a file from one remote machine to another.

The level of security required will bear some discussion. Control information directing the transfer, in particular usernames and passwords, should always be encrypted during transfer to deter hackers. Whether or not it is necessary to encrypt the data themselves whilst they are in transit depends on circumstances. In an academic research environment many data are public and clearly there is no need to encrypt these. Also, it is debatable whether it is worth encrypting some sorts of data which are notionally private, such as new experimental results to which access is restricted to the original investigators for a proprietary period. The difficulty of clandestinely finding and capturing such data whilst they are in flight and then subsequently interpreting them in a scientifically meaningful fashion makes the chances of piracy negligible. The situation is different for some sorts of financial data, where there can be sufficient incentive to eavesdrop on network traffic. Similarly, encryption may be required on some sorts of data for legal reasons, for example, medical records or population census details. Finally, this discussion has been entirely concerned with whether it is necessary to encrypt data during transit; unauthorised access to files on disk should be prevented.

5.2 FTP

FTP (File Transfer Protocol) is the standard method of transferring files across the Internet. It is defined in RFC 959[29]. FTP is ubiquitous and long-established, indeed its antecedents go back to the early days of ARPANET. However there are problems with it; specifically it is not secure. It uses separate TCP/IP streams to transfer commands and data and neither of them is encrypted, so passwords, for example, are sent as simple ASCII text. Nonetheless any replacement for FTP will need to provide many of its familiar and useful features. See Stevens[4], Chapter 27, pp419-439 for further details.

5.3 SCP

SCP (Secure Copy) is part of the SSH (Secure Shell) suite (see, for example, Barrett and Silverman[30] or the FAQ maintained by S. Acheson¹). SSH is not a shell in the conventional Unix sense. Rather, it is a set of commands to provide secure, reliable access to remote systems. It provides modern replacements for traditional remote login and file transfer commands such as TELNET and FTP (above) or the Unix ‘r-commands’ such as `rsh`, `rlogin` and `rcp`. SCP is simply the function in SSH for copying remote files.

The SSH suite provides authentication, encryption, compression and reliability. That is, it includes modern authentication techniques to verify that the user is allowed to access the remote resources. Both command information (including passwords) and the file contents themselves are encrypted before being transmitted over the network. Encrypting the command information is necessary for security. However, encrypting the file contents during transfer is often overkill (though there are certainly circumstances where it is justified) and, moreover, will impose a performance overhead. There are also options to compress the file during transfer. Finally, SSH guarantees that in the absence of an error report the file contents will be copied correctly.

There are actually two SSH protocols: SSH-1 and SSH-2, the second having been introduced to fix a number of problems and limitations in the first. The original SSH software was developed by Tatu Ylönen of the Helsinki University of Technology in 1995, and the SSH-1 protocol, released in the same year, essentially documented the behaviour of this program. The SSH-2 protocol followed in 1996. Various products, both commercial and free, which implement one or other of the protocols are now available for various platforms. One which is in widespread use is OpenSSH² which supports both protocols. SSH-1 is not as safe as SSH-2, though it is usually considered more configurable.

OpenSSH is entirely suitable for small and medium-sized data transfers. However, it has its own internal static transfer buffers which are unaffected by setting the TCP tuning parameters (as described in Section 4.3). Consequently these buffers can become a bottleneck in an end-host which has been tuned for a high BDP (bandwidth delay product) route. Consequently, OpenSSH/SCP is unsuitable for high-volume data transfers across such routes. If you need to use OpenSSH/SCP to transfer an extensive amount of data over a high BDP route there are two alternatives:

- use multiple streams (see Section 4.4) by running several SCP sessions simultaneously. This approach requires that the data to be transferred can be split up into a number of constituent files (at least one per stream),
- alternatively, the HPN-SSH patch for OpenSSH eliminates the problem by allowing the latter’s buffers to be defined dynamically at run time. HPN-SSH is described briefly in Section 5.3.2, below.

5.3.1 Hints on using the encryption and compression options

Several encryption techniques, offering varying degrees of security, are available, and SSH-1 and SSH-2 offer different alternatives. The ones available in OpenSSH on Linux are listed

¹See URL: <http://www.employees.org/~satch/ssh/faq/> This FAQ has links to draft IETF RFCs describing the SSH protocols.

²See URL: <http://www.openssh.com/>

in Table 5.2. For SSH-1 the degree of compression, if required, is expressed as a digit in the range 1 to 9. The interpretation of this scale is identical to that used by gzip. A value of 1 indicates the quickest to compute but least effective level of compression. A value of 9 specifies the most effective but slowest to compute level of compression. For SSH-2 only one level of compression is available and it is either on or off. It corresponds to SSH-1 level 6.

SSH-1	blowfish 3des
SSH-2	aes128-cbc 3des-cbc blowfish-cbc cast128-cbc arcfour aes192-cbc aes256-cbc

Table 5.2: Encryption techniques available in OpenSSH on Linux

The encryption technique is specified with the `-c` flag. Compression is requested using the `-C` switch and the level is specified by passing the `CompressionLevel` using the `-o` flag. So, for example, to use SSH-1 with blowfish encryption and a compression level of 5 you would specify:

```
scp -1 -c blowfish -C -oCompressionLevel=5 source-file destination-file
```

The `-1` flag specifies SSH-1. If the `-C` switch is omitted the file is transmitted without compression. The compression level may be specified for SSH-2, but will have no effect. Note that these options, and in particular the `-C` switch, are different in implementations other than OpenSSH. Also, in OpenSSH the defaults can vary depending on which version of Linux is being used and on settings made by the system administrator during installation. Compression reduces the volume of data which needs to be copied across the network but requires additional processing time on the source and destination hosts to perform the compression and subsequent decompression. The conventional wisdom is that compression is useful over modem lines and other slow connections, but with fast connections it will make no difference, or might even slow down the transfer.

In experiments with using OpenSSH over a WAN we found that:

- 3des-cbc (SSH-2) is significantly, and 3des (SSH-1) is substantially, slower than the other options,
- blowfish (SSH-1) is marginally slower,
- there is little difference between the speed of the other encryption methods,
- In SSH-1 the compression level made little difference to the speed. The only exception was with blowfish (SSH-1) encryption where high compression did indeed slow it down.

These results were found on a particular combination of hosts, LAN and WAN and are not necessarily generally applicable.

5.3.2 High Performance Network SSH

High Performance Network SSH (HPN-SSH)³ is a patch for OpenSSH which allows the latter's internal transfer buffer to be defined dynamically at run time rather than statically. Consequently, the buffers can be adjusted, for example to values suitable for high BDP links.

HPN-SSH was developed by Chris Rapier and Michael Stevens. It is fully inter-operable with other SSH servers. HPN-SSH clients should retrieve data from non-HPN-SSH servers at a higher rate than basic OpenSSH clients. Similarly, HPN-SSH servers should be able to upload data from non-HPN-SSH clients at a faster rate than basic OpenSSH servers. However, these results are only achieved if the hosts are running a properly tuned TCP stack (see Section 4.3).

HPN-SSH is available as a patch which is applied to the OpenSSH source files.

5.4 GridFTP

GridFTP[31] was developed by the Globus Project.⁴ It is specifically intended for use in scientific 'data grid' applications which typically need to move large amounts of data. It attempts to retain many of the advantages of traditional FTP whilst avoiding its shortcomings. Specifically, it provides the following features.

Automatic setting of TCP buffer/window sizes TCP implementations typically include a number of internal buffers and windows. Often their default settings are unnecessarily small for modern networks, leading to poor performance (see Section 4.3). The values can be changed manually, but this is tricky and error prone and often is not done. GridFTP attempts to set suitable values automatically.

Parallel transfer Typically a single TCP stream will use only a fraction of the available data transfer rate. Thus it is often possible to improve performance by using multiple streams (and FTP accelerators attempt to do this; see Section 5.6, below). GridFTP includes support for multiple streams.

Third-party control of transfer GridFTP includes third-party control of transfer. That is, the transfer can be initiated and controlled by processes running on neither the source nor destination hosts. This feature is useful, for example, for accessing large datasets which might be replicated at several different locations, or in circumstances where a 'master' process is copying data from one remote machine to another.

Partial file transfer GridFTP is able to transfer only a specified fraction of a file. This feature is useful in the common case where only a fragment of a large file is required.

Security GridFTP includes modern authentication features and provides secure file transfer.

Reliable Data Transport Clearly it is important that a file transfer protocol should be reliable and provide automatic, internal features for re-sending lost or corrupt packets. Standard FTP includes such features and they are also included in GridFTP.

GridFTP is a relatively new protocol and there are still some problems with it. It is not yet widely available and the software which implements it can be tricky to install. These

³See URL: <http://www.psc.edu/networking/projects/hpn-ssh/>

⁴The URL for the Globus Project home page is <http://www.globus.org/>

problems are likely to diminish with time. Its security features require that any user has acquired an ‘e-Science certificate’ from some recognised authority. Currently this process is something of an unusual chore. However, use of these certificates seems set to increase and acquiring them is likely to become increasingly routine.

5.4.1 GridFTP and firewalls

GridFTP uses an unusual range of port numbers which might be blocked by any intervening firewall. It uses port 2811 as a control channel and allocates for itself a range of ports for transferring data. Fortunately this range can be specified when GridFTP is invoked. In order to use GridFTP through a firewall you will need to:

1. ensure that port 2811 is open,
2. agree with the firewall administrator a range of ports to be used for data transfer, perhaps 40000-45000,
3. specify this range when you run GridFTP.

5.5 BbFTP

BbFTP is file transfer software developed by Gilles Farrache at the IN2P3 Computing Centre⁵ in Lyons. It was written for transferring data from the Babar experiment between SLAC in California and IN2P3. It is available from the CNRS⁶ in Paris. BbFTP implements its own transfer protocol which is optimised for large files (greater than 2 Gbyte). In particular it implements the specification for big transfer windows in RFC 1323[32] and uses multiple file transfer streams. It is particularly intended for transferring large volumes of data across a loaded WAN and so is suitable for use in scientific ‘data grid’ applications. The main features of bbFTP are:

- username and password are encoded at the connection,
- SSH and Certificate authentication modules,
- multi-stream transfer,
- big windows (as defined in RFC 1323),
- on-the-fly data compression,
- automatic retry,
- customisable time-outs,
- transfer simulation,
- integrated authentication with the AFS distributed file system,
- interface to the RFIO remote file access protocol.

⁵See URL: <http://cc.in2p3.fr/>

⁶See URL: <http://doc.in2p3.fr/bbftp/>

Dan Schragger of the Weizmann Institute has developed BBftpPRO,⁷ an enhancement to bbFTP which allows it to operate more easily over firewalls.

Unlike FTP and SSH, bbFTP is unlikely to be ready installed on a system. In order to use it it is necessary to first install a client and server on the two machines between which files are to be transferred. Normally the server will be installed on the machine where the files reside and the client on the one to which they are to be copied. However, files can be ‘pushed’ as well as ‘pulled,’ so the client and server can be installed as desired.

The usage of bbFTP is less interactive than that of SSH or FTP, perhaps unsurprisingly given that it is intended for copying large files. The operations to be performed by a bbFTP client are usually specified by configuration and control files. In particular the control file contains a series of entries comprising commands specifying the operations to be performed. Figure 5.1 shows a simple control file. The commands for copying files, changing local and remote directories *etc.* are similar to the corresponding FTP ones.

```
setnstream 3
setoption nocreatedir
df /home/acd
get /home/acd/timetest/testfiles/acbit.fit /home/acdscratch/
```

Figure 5.1: Example bbFTP control file

The server to be used can be specified on the command-line when the client is invoked. Alternatively, it can be included in with the file specifications given in the control file using a URL-like mechanism (a so-called ‘TURL’). Security in bbFTP can be provided by either SSH or certificate based mechanisms. Consequently, it is possible to use bbFTP without first obtaining a certificate. BbFTP obtains good performance both by using parallel streams (in a manner similar to an FTP accelerator, see Section 5.6, below) and by using a big transfer window, as prescribed in RFC 1323. The number of parallel streams can be specified in the configuration or control files.

5.5.1 Hints on installing and using bbFTP

1. If OpenSSL⁸ is not already available on your system it might be necessary to obtain and install it before installing the bbFTP server. It is not necessary to install OpenSSL before installing the bbFTP client.
2. By default the installation procedures for both the client and server attempt to put their files in system directories. If you do not have the privilege to write to these directories it is possible to specify an alternative location to the configure script run prior to installation:

```
./configure --prefix=chosen-directory
```

⁷See URL: http://www.weizmann.ac.il/physics/linux_farm/bbftp_PRO.html

⁸See URL: <http://www.openssl.org/>

3. The server demon writes its messages to the system log file:

```
/var/log/daemon.log
```

However, inspecting this file can be problematic as you might find that you do not have the necessary privileges.

4. In order to permit bbFTP to operate, any firewalls protecting either the client or server hosts should be configured to allow both incoming and outgoing traffic on a range of ports upwards of 5020. In practice ports 5020 to 5040 should be adequate. The client host should permit such traffic from the server and vice versa.
5. BbFTP is intended for transferring large files across a WAN, not for moving small files across a LAN. Indeed, caution should be exercised in using it to copy files across a LAN. In tests performed at the Royal Observatory Edinburgh the interface of the host running the bbFTP server regularly hung when more than one parallel stream was used to copy files across the local Ethernet.

5.6 FTP Accelerators

The FTP protocol works by copying data down a TCP stream from a source to a destination host. Typically such streams will (deliberately) use just a fraction of the available data transfer rate. Clearly it is possible to improve the transfer rate by using multiple streams (see Section 4.4). A number of utilities are available which provide this functionality; some are commercial products, others are free. These tools are usually called **FTP accelerators**. They piggy-back on top of standard FTP software and usually require no additional system software to be installed on either the source or destination hosts. They can significantly improve the transfer rate. However, because the number of streams available is finite some system administrators consider their use to be anti-socially hogging resources. Conversely, some resources, such as popular Web sites, are replicated (or **mirrored**) at several geographically dispersed sites. Some accelerators provide the additional feature of being able to simultaneously retrieve different segments of a resource from several such sites. Such simultaneous retrieval from multiple sites is often considered less anti-social than using multiple connections to a single site.

An additional consideration is that, because of the facilities provided by the host operating system, many accelerators will initially write multiple files, one per stream. Once the retrieval is complete they will concatenate these files into one final copy of the original. The consequent shuffling of bytes on disk might obviate the advantage of the multiple retrieval. However, on Unix there is a trick which allows files to be written beyond their notional end-of-file mark. This trick allows multiple streams to be written directly into their correct place in a single destination file, thus avoiding any subsequent shuffling of bytes. The Axel accelerator⁹ by Wilmer van der Gaast uses this technique.

5.7 Peer-to-Peer Protocols and BitTorrent

The protocols discussed previously in this chapter essentially operated using a client-server model: files reside on servers from whence clients can retrieve copies of them. Accelerators

⁹See URL: <http://wilmer.gaast.net/main.php/axel.html>

which retrieve segments of a file from multiple servers modify this model somewhat by having a single client access multiple servers. So-called peer-to-peer protocols have emerged as a novel alternative to this approach. A recent example, which has become quite widely used, is BitTorrent[33].¹⁰

In outline BitTorrent works as follows. A group of clients, or more accurately peers, all wish to obtain a copy of a file. This file is conceptually divided into a number of segments, which can be retrieved individually (and, when assembled, constitute the whole). Every peer must be prepared to supply, or upload, segments to other peers, as well as retrieving segments. A central server, called a tracker, keeps track of which peers have copies of which segments (but does not itself serve segments). Initially at least one peer, called the seed, must have a copy of the whole file. The other peers will typically initially have none of the file. The retrieval starts with each peer retrieving segments from the seed, but crucially each peer will retrieve different segments. Once each peer has a few segments the peers start retrieving from each other rather than from the seed, with the tracker keeping track of which-has-what. Because segments are being retrieved from multiple sites it is possible to achieve better performance than if all copies were retrieved from a single server.

BitTorrent, and similar protocols, are really intended for down-loading static files which are sufficiently popular that many users require copies of them. A common, but different, case is where a user, having created a bespoke file (in which only he is interested) on a remote host, wishes to retrieve a local copy. Neither peer-to-peer techniques such as BitTorrent nor more conventional accelerators retrieving from multiple sites are immediately suited to this case (though an accelerator being used to retrieve from a single origin using multiple streams may still yield a significant improvement).

¹⁰See URL: <http://bittorrent.com/> and also the FAQ at URL: <http://dessent.net/btfaq/>

Chapter 6

Discussion

We've all seen the error messages: *NFS server not responding; unable to locate server; please check the server name and try again; no route to host; host unreachable*. As fallible as our desk-top computers seem to be, somehow adding a network to the mix seems to make things break even faster than before.

Core Jini,
W. Keith Edwards, 1999.

This final chapter brings together some of the strands from earlier chapters of the document. It summarises some simple, practical steps which you can take now to improve the performance of applications which transfer data across the networks.

Obtaining good network performance is not simply a matter of tuning an existing setup or choosing suitable software. Rather, the network throughput required should be considered when specifying new computer systems and appropriate hardware bought. The performance of the LAN connecting the local computers to the wider MAN and WANs is also crucial, as is the performance and configuration of any firewall. Better performance is usually obtained by using modern hardware. Section 4.6 includes some specific suggestions about choosing computer hardware to give good network performance.

As a user you are unlikely to be able to solve all your network problems yourself. It will be necessary to liaise with your local and campus-wide network support staff, and it is advantageous to cultivate contacts with them. Also, they should be involved, from an early stage, in the planning for any new projects which will make substantial use of the networks and informed of the anticipated volume of traffic. Though these points might seem obvious they can often be overlooked. A list of 'ten questions' which you should try to answer before making heavy use of the network is given in Section 6.1, below.

The wide area networks are mostly empty and usually reasonably well-configured. Currently their capacity is adequate, and even transatlantic links do not appear to be a bottleneck. They are unlikely to be the cause of any problems which you encounter. Rather, most problems occur reasonably close to either the source or destination host; the 'last mile' problem. With perfectly ordinary, but modern, hardware and using the sort of simple tuning techniques mentioned in Section 4.3, but without resorting to more specialised techniques such as fast TCP stacks, it should be possible to obtain a transfer rate of 20 - 50 Mbyte/sec across JANET in the UK (see Chapter 4). There are a number of reasons why this transfer rate might not be achieved. Whilst it is difficult to generalise, some of the likely causes of poor performance are, in decreasing order of likelihood:

1. the end host hardware,
2. the application software (as distinct from the TCP/IP stack) running on the end hosts,
3. some local network limitation,
4. a firewall,
5. and only finally the WAN connecting the end hosts.

For example, in experiments between the Royal Observatory Edinburgh and the Jodrell Bank Observatory in Cheshire performed during 2004, the available rate was, at best, about 10 Mbyte/sec, well below the performance which should have been available. However, the bottleneck was in the final hop to the destination host in the Jodrell Bank LAN. This result is typical; the bottleneck is usually at or close to either the source or destination host.

The default values of the TCP tuning parameters are often inappropriate for modern computers and WANs. Improving the tuning should allow you to make better use of a partly-empty WAN, and as a first step to obtaining good performance it is always worth setting the tuning parameters to more suitable values.

It is sensible to run a series of representative tests before embarking on a major programme of transfers. Typically such tests might consist of running both example transfers and pchar probes over the route. Pchar should reveal any bottlenecks, though what remedial action will be possible depends on where the problem lies. In some cases it might be necessary to upgrade local equipment. Nowadays it is almost never possible to simply run pchar successfully, and it will probably be necessary to negotiate with local and remote network staff to allow pchar's traffic through firewalls.

The tests that you run should use a representative set of data. The behaviour of the protocols and applications will depend, to an extent, on the number of files transferred and their size distributions, as well as on the total number of bytes copied. File compression is usually more efficient on one large file than on many small ones. Firewalls are potentially one cause of anomalously slow performance, and sometimes simply changing the order of firewall rules can lead to improved performance. Try to ensure that the rules are as few and simple as possible. Rules are processed sequentially, so the rules applying to high-volume traffic should be placed towards the start of the list.

FTP is a familiar and robust mechanism for transferring files. However, it does not employ encryption, even for control information such as usernames and passwords. Encryption is often not necessary on academic or scientific data, though there are certainly cases where it is needed. However, the absence of encryption on usernames and passwords is potentially a security problem with FTP, and caution should be exercised in using it. FTP accelerators, such as Axel, can provide significant improvements in performance, but should be used benignly, so as not to affect the performance of other users. You should check with the administrators of the remote host before undertaking a major programme using an accelerator. If using an accelerator produces no improvement it is worth checking that multiple streams are permitted by the remote FTP site.

SCP is a modern, secure mechanism for copying files and is generally to be preferred over FTP. It encrypts both the control information (such as passwords) and the file content, though the latter is often not necessary in a scientific or academic context. SCP is part of the SSH suite and there are two SSH protocols: SSH-1 and SSH-2. The latter is usually considered the more secure. Though SCP is efficient and reliable, there is a specific problem using the

OpenSSH implementation over high bandwidth delay product (BDP) routes ('long fat pipes'). OpenSSH/SCP has its own internal transfer buffers which are static and unaffected by any TCP tuning that has been done to the end hosts. The consequence is that though SCP is suitable for transferring small and medium amounts of data over high BDP routes, it is not suitable for sending large volumes of data over such routes.

If you need to use OpenSSH/SCP to transfer a large amount of data over a high BDP route you should obtain and install the HPN-SSH patch, which solves the problem. Alternatively, you can use several concurrent SCP sessions to transfer different sections of the data simultaneously over separate streams.

SSH has several encryption and compression options. For OpenSSH the 3des-cbc (SSH-2), 3des (SSH-1) encryption options seem slower than the other options, and blowfish (SSH-1) is slightly slower, but there is little to choose between the remaining alternatives (see Table 5.2 for a list). SCP supports optional compression, and it is usually worth using, though whether this is so depends on the details of the files being copied and the route used. The level of compression used in SSH-1 does not make much difference. Lower values may be preferable, though again this will vary depending on the characteristics of the transfer.

BbFTP and similar modern protocols offer advantages similar to SCP in terms of security and compression. However, they also offer higher transfer rates through the use of multiple streams, big windows and automatic TCP tuning. Note, however, that bbFTP is intended for copying large files (say, greater than 2 Gbyte) over a WAN. It is not intended for copying small files or copying files over a LAN.

If none of the simple transfer protocols gives adequate performance, it is possible to experiment with one of the fast TCP stacks. However, you should be aware (and beware) that you are entering a specialised and complex area. Before using these stacks it is necessary to obtain permission to run them over the entire route.

6.1 Ten Easy Questions

This section is adapted, with permission, from Robin Tasker's '*Ten Easy Network Questions – So You Think You Know Your Network*'[34]. It is aimed at people running or implementing the network aspects of projects which are going to make heavy use of the network. It was written for the GridPPP project, but is widely applicable. Whilst intended to be brief and light-hearted it provides a challenge whose purpose is to ensure that you really understand your local network and, just as importantly, you know who to speak to when things go wrong or you need to find out additional information. If you want to drive your network hard you need to know this stuff!

Answer the following questions using your skill and judgement. Conferring with your local and Institutional network support staff is allowed. Actually it is encouraged! Share your answers with your colleagues.

1. Provide the names and contact details of your local (Departmental) and Institutional network support staff.
2. Provide details of the responsibilities, together with the demarcation of those responsibilities, of your local (Departmental) and Institutional network support staff.
3. What is a Regional Network Operator (RNO), and why does it matter to you?

4. What is SuperJANET4? And, more importantly, what is SuperJANET5?
5. Draw a simple diagram showing your local (Departmental) network and sufficient of your Institutional network such that you can trace a line from *your* end-system to the connection from your Institute's network into the RNO infrastructure.
6. On the diagram produced in answer to question 5, show the capacity of each link in the network and provide a note against each link of its contention ratio (briefly, the number of other users with whom you might have to share it).
(*Hint: just how many 100 Mbit/sec links are being fed into that 1 Gbit/sec uplink?*)
7. On the diagram produced in answer to question 5, colour and distinguish the switches and routers and for each device provide a note of its backplane capability.
(*Hint: just how many (frames per second|packets per second) can the backplane shift?*)
8. What is the average and peak traffic flow between your local (Departmental) network and the Institutional network?

What is the average and peak traffic flow between your Institutional network and the RNO?

What is the total capacity of your Institutional connection to the RNO?

What are the upgrade plans for your local (Departmental) network, your Institutional network and the network run by the RNO?

9. Do you implement Information Systems (IS) Security? Does your Institute implement IS Security?
Do you implement firewalls? Does your Institute implement firewalls?
On the diagram produced in answer to question 5 colour in the firewall(s) and any other security devices.
Provide information on how changes are made to the rule set of each firewall.
Provide a note on the capacity of each firewall. Include what happens when this capacity is exceeded.
10. What is the best performance you can achieve from your end-system to an equivalent system located in some geographically remote (but friendly!) Institute?

For your end-system:

- Do you understand the kernel, the bus structure, the NIC and the disk system?
- Do you understand TCP tuning and what it can do for you?
- Do you understand your application and what it can do to your performance?

Conclusions (and Answers)

We hope that with the completion of this exercise you will have a clearer understanding of the components of the network that affect your work (or at least the local portion of it; there will be similar complexity at the destination Institute). Furthermore, that you have pieced together the end-to-end path upon which you are reliant, and better understand the many issues that affect performance. Most of all we hope you have built up a set of contacts who can help you to make your network work for you.

And as for the answers?

Well, they are probably out of date already, so keep talking.

The ten questions were originally intended for use in the GridPP Project. The answers provided by its participating sites can be seen at:

http://wiki.gridpp.ac.uk/wiki/GridPP_Answers_to_10_Easy_Network_Questions

Appendix A

The Internet Standardisation Process

The Internet is famously the most successful, best functioning and largest anarchy that there has ever been, a circumstance which is not without a certain amount of irony given its origins in military command and control systems. However, some controls are needed, particularly for the development and approval of the standards which allow the Internet to function, such as TCP/IP.

A.1 Standardisation Bodies

There are ultimately four groups responsible for developing and maintaining the Internet standards.

ISOC (Internet Society)¹ is a professional society to facilitate, support and promote the evolution and growth of the Internet.

IAB (Internet Architecture Board)² is the technical oversight and coordination body for Internet standards. In particular it provides architectural oversight of the Internet standardisation process and is the body to which appeals against decisions taken during this process can be made. It also appoints the RFC editor (below). It is chartered as both as a committee of the IETF (below) and as an advisory body of the ISOC.

IETF (Internet Engineering Task Force)³ develops the specifications that become Internet standards. The actual technical work of the IETF is done in numerous working groups. The IETF is concerned with developing practical, usable standards for deployment in the short-term, rather than long-term research. Architectural oversight of the IETF is provided by the IAB (above).

IRTF (Internet Research Task Force)⁴ complements the IETF by pursuing long-term research projects. Like the IETF, architectural oversight is provided by the IAB.

¹See URL: <http://www.isoc.org/>

²See URL: <http://www.iab.org/>

³See URL: <http://www.ietf.org/>

⁴See URL: <http://www.irtf.org/>

A.2 RFCs

All the Internet standards are issued as so-called RFCs (Requests for Comments), a term which originated in the early days of the Internet and is something of a misnomer. Once an RFC is agreed and published it is an Internet standard and its contents are largely immutable. Each RFC has a unique sequence number which is assigned chronologically.

A.2.1 Obtaining RFCs

All RFCs are publicly available on-line. There are a number of ways to access them. Perhaps the simplest is via the Web. To obtain RFC *xxxx* access URL:

```
http://www.ietf.org/rfc/rfcxxxx.txt
```

For example RFC 793 is available at:

```
http://www.ietf.org/rfc/rfc793.txt
```

To obtain a complete listing of the various ways of accessing RFCs send an e-mail message as follows:

```
To:      rfc-info@isi.edu
Subject:  getting rfcs
Contents: help: ways_to_get_rfcs
```

It might take a while for a reply to arrive. There is an index of RFCs at:

```
http://www.ietf.org/iesg/1rfc_index.txt
```

It is important to check the index before starting to make serious use of an RFC because the RFCs are evolving. Some become obsolete, replacements are approved *etc.* The index always indicates RFCs which have newer replacements and updates. For further details see either:

```
http://www.ietf.org/rfc
http://www.rfc-editor.org/
```

Acknowledgements

We are grateful to various people who have advised or assisted during the preparation of this document, or made helpful comments on an earlier draft of it. We would like to thank (in alphabetical order): Colin Greenwood, Anthony Holloway, Nicola Pezzi, Robin Tasker, Brian Tierney and, particularly, Horst Meyerdierks. In addition, much of Chapter 4 is based on material by Brian Tierney and the ‘ten easy questions’ of Chapter 6 are by Robin Tasker.

The present document is derived from the earlier report *Computer Networks for the AVO* by A.C. Davenhall, A. Lawrence, E.T.W. Sutorius and R.G. Mann (11 February 2005). Numerous network managers and similar staff in several institutions assisted with the work for this earlier report. We would like to thank (in alphabetical order) Pete Bunclark, Tim Jenness, Benoit Pirenne, Mike Read, Henry Stilmack, Dieter Suchar, Jan vandenBerg and Andy Wright. We are particularly grateful to Horst Meyerdierks and the long-suffering Anthony Holloway.

The document was prepared as part of the work for the Exploitation of Switched Light Paths for e-Science Applications (ESLEA) Project. ESLEA is funded by the Engineering and Physical Science Research Council, the Medical Research Council and the Particle Physics and Astronomy Research Council. *Computer Networks for the AVO* was compiled as part of the work for the Astrophysical Virtual Observatory (AVO) Project. The AVO received financial support from the European Commission through contract HPRI-CT-2001-50030 under the Fifth Framework Programme for research, technological development and demonstration activities.

Bibliography

- [1] The notice appeared in *Mon. Not. R. Astron. Soc.*, 1873, **33**, pp369-370, and is reproduced by A.C. Davenhall in *The Observatory*, 2000, **120**, pp332-333. The first message transmitted under the agreement, reporting the discovery of the 129th minor planet, appeared in *Mon. Not. R. Astron. Soc.*, 1873, **33**, p368. For an account of the early development of transatlantic cables see T. Standage, 1999, *The Victorian Internet* (Phoenix: London), pp84-87 or L. Solymar, 1999, *Getting The Message: A History of Communications* (Oxford Univ. Press), pp68-86. 1
- [2] A.S. Tanenbaum, 2003, *Computer Networks*, Fourth Edition (Prentice Hall: Upper Saddle River, New Jersey). 2, 5, 9, 10, 13, 14, 22, 25, 42
- [3] P.J. Irving, 2003, *Computer Networks* (Crucial/Learning Matters: Exeter). 2, 5, 6
- [4] R.W. Stevens, 1994, *TCP/IP Illustrated*, Volume 1 (Addison-Wesley: Boston). 2, 16, 17, 18, 34, 52
- [5] C. Hunt, 2002, *TCP/IP Network Administration*, third edition (O'Reilly: Sebastapol, California). 2
- [6] K. Hafner and M. Lyon, 1996, *Where Wizards Stay Up Late* (Simon and Schuster: New York). 3, 27
- [7] J. Strand, A.L. Chiu and R. Tkach, February 2001, *IEEE Comm*, pp81-96. 6
- [8] See, for example, Oskar Andreasson's *Iptables Tutorial* which is available at URL: <http://www.faqs.org/docs/iptables/>. There is also a Linux `man` page and several books, such as, G.N. Purdy, 2004, *Linux Iptables Pocket Reference* (O'Reilly: Sebastapol, California). 11
- [9] D.M. Piscitello and A.L. Chapin, 1993, *Open Systems Networking: TCP/IP and OSI* (Addison-Wesley: Reading, Massachusetts). 14
- [10] A. Marine and G. Malkin, March 1994, RFC 1594: *FYI on Questions and Answers: Answers to Commonly asked "New Internet User" Questions*. 15
- [11] J.B. Postel (ed), September 1981, RFC 793: *Transmission Control Protocol*. Some errors are corrected in the Host Requirements RFC. 17
- [12] J.B. Postel, August 1980, RFC 768: *User Datagram Protocol*. 17
- [13] J.B. Postel (ed), September 1981, RFC 791: *Internet Protocol*. 17
- [14] J.B. Postel (ed), September 1981, RFC 792: *Internet Control Message Protocol*. 18
- [15] S.E. Deering, August 1989, RFC 1112: *Host Extensions for IP Multicasting*. 18

- [16] P. Loshin, 2004, *IPv6: Theory, Protocol and Practice*, Second Edition (Morgan Kaufmann: San Francisco). 22
- [17] The IPv6 RFCs are: 22
- S. Deering and R. Hinden, December 1998, RFC 2460: *Internet Protocol, Version 6 (IPv6) Specification*.
 - T. Narten, E. Nordmark and W. Simpson, December 1998, RFC 2461: *Neighbor Discovery for IP Version 6 (IPv6)*.
 - S. Thomson and T. Narten, December 1998, RFC 2462: *IPv6 Stateless Address Autoconfiguration*.
 - A. Conta and S. Deering, December 1998, RFC 2463: *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*.
 - M. Crawford, December 1998, RFC 2464: *Transmission of IPv6 Packets over Ethernet Networks*.
 - D. Haskin and S. Onishi, December 1998, RFC 2465: *Management Information Base for IP Version 6: Textual Conventions and General Group*.
 - D. Haskin and S. Onishi, December 1998, RFC 2466: *Management Information Base for IP Version 6: ICMPv6 Group*.
- [18] J. Morgan, 2004, *The Secrets of Rue St. Roche: Hope and Heroism Behind Enemy Lines in the First World War* (Allen Lane: London). Also a paperback edition was published by Penguin (London) in 2005. 27
- [19] N. Brownlee and C. Loosely, 2001, ‘Fundamentals of Internet Measurement: A Tutorial,’ *CMG J. Computer Resource Management*, no. 102. See URL: www.avoka.com/resources/keynote/Fundamentals_of_Internet_Measurement_A_Tutorial.pdf 28
- [20] The IPPM network performance metric RFCs are: 28, 29
- V. Paxson, G. Almes, J. Mahdavi and M. Mathis, May 1998, RFC 2330: *Framework for IP Performance Metrics*.
 - J. Mahdavi and V. Paxson, September 1999, RFC 2678: *IPPM Metrics for Measuring Connectivity*.
 - G. Almes, S. Kalidindi and M. Zekauskas, September 1999, RFC 2679: *A One-way Delay Metric for IPPM*.
 - G. Almes, S. Kalidindi and M. Zekauskas, September 1999, RFC 2680: *One-way Packet Loss Metric for IPPM*.
 - G. Almes, S. Kalidindi and M. Zekauskas, September 1999, RFC 2681: *A Round-trip Delay Metric for IPPM*.
 - M. Mathis and M. Allman, July 2001, RFC 3148: *A Framework for Defining Empirical Bulk Transfer Capacity Metrics*.
 - R. Koodli and R. Ravikanth, August 2002, RFC 3357: *One-way Loss Pattern Sample Metrics*.
 - C. Demichelis and P. Chimento, November 2002, RFC 3393: *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*.

- V. Raisanen, G. Grotefeld and A. Morton, November 2002, RFC 3432: *Network Performance Measurement with Periodic Streams*.
 - S. Shalunov and B. Teitelbaum, April 2004, RFC 3763: *One-way Active Measurement Protocol (OWAMP) Requirements*.
- [21] B. Lowekamp, B. Tierney, L. Cottrell, R. Hughes-Jones, T. Kielmann and M. Swany, May 2004, *A Hierarchy of Network Performance Characteristics for Grid Applications and Services*, See URL:
<http://www.gridforum.org/documents/GFD.23.pdf> 28
- [22] E.D. Zwicky, S. Cooper and D.B. Chapman, 2000, *Building Internet Firewalls*, second edition (O'Reilly: Sebastapol, California). 34, 36
- [23] See, for example, M.Dodge, December 2001, 'Mostly Cloudy, Clearing Later: Network Weather Maps', *Mappa Mundi* (http://mappa.mundi.net/maps/maps_025/). 38
The *Atlas of CyberSpace* is an extensive collection of graphical representations of computer networks. There is a mirror at:
http://www.geog.ucl.ac.uk/casa/martin/atlas/isp_maps.html
- [24] B. Tierney, 17 November 2005, *TCP Tuning and Network Troubleshooting*, see URL:
http://www.onlamp.com/pub/a/onlamp/2005/11/17/tcp_tuning.html 41, 49
- [25] P. Clarke, 2004, comment made at the *Networks for Non-Networkers* workshop held at University College London, 13-14 July 2004. 41
- [26] The Linux document describing the parameters can usually be found at: `/usr/src/linux/Documentation/networking/ip-sysctl.txt`. Oskar Andreasson has prepared an *Ipsysctl Tutorial* based on this documentation, which is available at URL:
<http://bec.at/support/ipsysctl-tutorial/index.html> 44
- [27] A. Falk, T. Faber, J. Bannister, A. Chien, R. Grossman and J. Leigh, November 2003, *Comm. ACM*, **46**, no. 11, pp43-49. 48
- [28] E. He, P. Vicat-Blanc Primet and M. Welzl, November 2005, *A Survey of Transport Protocols other than 'Standard' TCP* (GFD-E.055). See URL:
http://www.ggf.org/ggf_docs_final.htm 48
- [29] J. Postel and J. Reynolds, October 1985, RFC 959: *File Transfer Protocol (FTP)*. 52
- [30] D.J. Barrett and R.E. Silverman, 2001, *SSH The Secure Shell: The Definitive Guide* (O'Reilly: Sebastapol, California). 53
- [31] The Globus Project, 2000, *GridFTP Universal Data Transfer for the Grid*, Globus Project white paper, See URL:
<http://www-fp.globus.org/datagrid/deliverables/C2WPdraft3.pdf> 55
- [32] V. Jacobson, R. Braden and D. Borman, May 1992, RFC 1323: *TCP Extensions for High Performance*. 56
- [33] B. Cohen, May 2003, *Incentives Build Robustness in BitTorrent*. See URL:
<http://bittorrent.com/bittorrentecon.pdf> 59
- [34] R. Tasker, 30 September 2005, '*Ten Easy Network Questions*' – *So You Think You Know Your Network*, CCLRC Daresbury Laboratory. 62

See Appendix A for details of how to obtain copies of RFCs.