

Network Monitoring Tools

Kate Lance
Department of Computer Science
The University of Newcastle
Callaghan NSW 2308
clance@cs.newcastle.edu.au
7 July 1995

Abstract

This paper describes a number of TCP/IP tools for studying the activity of local and remote network traffic. Most of the packages are freely available at Internet archive sites. They cover a broad range of functionality for a variety of situations, such as network congestion, consistency checks, site reachability, traffic accounting, packet analysis, and long-term planning.

1 Introduction

When network problems occur it's often hard to know where to begin. There are many different components of a network: the local backbone of your own system, perhaps a larger segment such as a campus backbone, the Australian Internet (AARNet), and the intricate interconnections of the overseas Internet. Failure at any point can have a multitude of causes, ranging from physical components such as broken routers or Telecom line problems, to simply too much traffic at a bottleneck point.

Network monitors are a diverse collection of packages. I will discuss some of the ones I've run across over the last few years: new ones appear, some undoubtedly exist that I haven't used, but this is a fairly complete list of the most common ones. They fall roughly into one of the following categories:

- **Host monitors:** host activity as an indicator of network activity.
- **Information tracers:** tools to build up databases of information about the parameters and the state of network nodes.
- **Network access tools:** to discover the reachability or the state of each hop on the path to a distant site.
- **Local displays:** representations of local network activity, by protocol or as a percentage of maximum load.
- **Inter-network displays:** representations of traffic activity between local or remote networks.
- **Traffic counters:** tools to accumulate numbers of packets for accounting purposes, categorised by protocols, sources and destinations.
- **Packet analysers:** can catch all the information in network packets, examine their protocol-specific headers, sources and destinations, sizes, flags and other settings, and sometimes data content.

2 Brief discussion of TCP/IP protocols

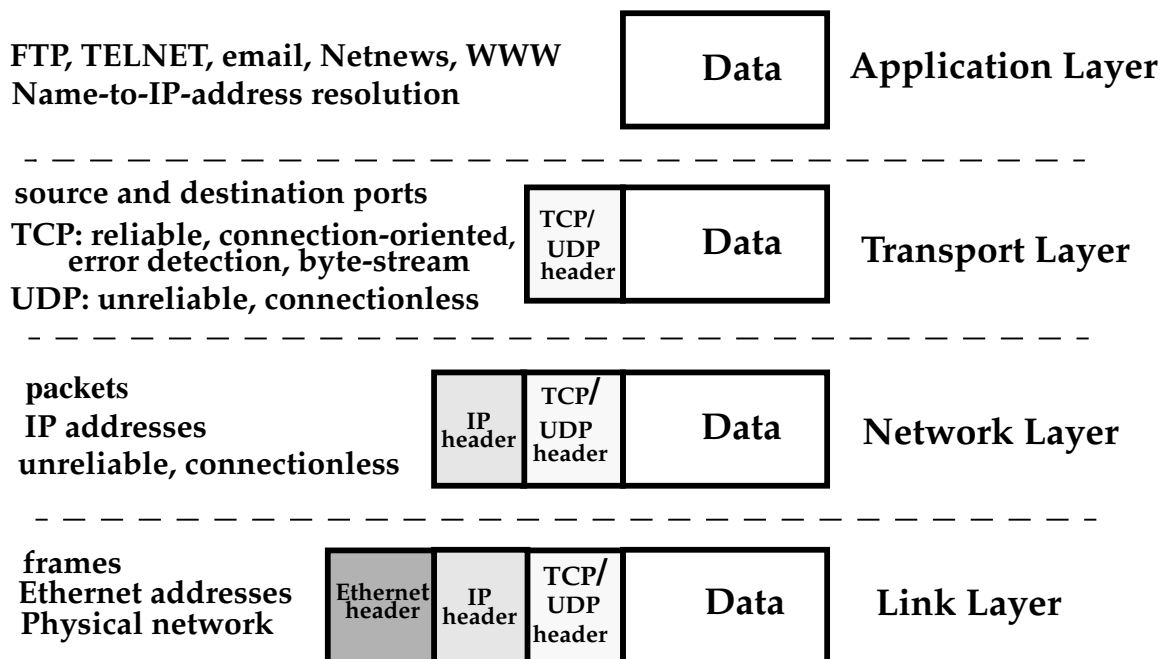
All of the monitors are useful in different contexts. To understand the information they provide, it's necessary to explain a little about the protocols that define how input generated at the user level of a computer is broken down into signals that can be passed over wire and fibre, from router to router, from one side of the world to the other, to be correctly re-assembled into meaningful information at the other end.

The set of standards that permit world-wide networks to communicate is called TCP/IP, the Transmission Control Protocol/Internet Protocol suite. It defines four layers of data communication:

- **Application:** user utilities such as email, telnet, ftp.
- **Transport:** controls data flow between hosts (TCP/UDP).
- **Network:** controls packet transport over networks (IP).
- **Link:** interfaces to the physical network hardware.

Any communication between users over a network is generated at the Application level, passed to the Transport level to be split into datagrams, then to the Network layer to be routed to another computer or network, and finally to the Link layer to be sent out as frames on the physical hardware. When the frames arrives at the host machine, they're received by the Link layer, forwarded to their destination by the Network layer, re-assembled into data by the Transport layer, then acted upon by the Application layer. When you set out to monitor or diagnose network activity, it's the three lower layers, Transport, Network and Link, that provide the most useful information.

TCP/IP Protocol Stack



Some network protocol definitions:

- **Connectionless** means that a protocol does not maintain any state information about successive pieces of data: they are all handled (and perhaps even routed) independently of each other. No end-to-end connection is involved.
- **Connection-oriented** means that a connection between two servers must first be set up before data can pass between them (the analogy is usually the telephone system).
- **Unreliable** means that a best effort delivery service is used. If something goes wrong a datagram is simply discarded and an attempt is made to send a warning back to the source.
- **Reliable** means that the protocol maintains data integrity with checksums, acknowledgement of incoming data, and retransmission of lost outgoing data. It also performs packet disassembly and reassembly, and buffer space flow control.

Protocols at the Transport layer:

- **TCP**: provides a reliable, connection-oriented service. It splits data arriving down from the Application level into byte-stream segments, and waits for acknowledgement of segment arrivals by the destination host—otherwise it times out and retransmits any missing segments. It performs integrity checks, and re-assembles and re-orders datagrams arriving up from the Network layer.

TCP protocols include **FTP** (File Transfer Protocol), **SMTP** (Simple Mail Transfer Protocol), **X Window System**, **telnet** and **rlogin**.

- **UDP**: provides a simple unreliable, connectionless service—it does not guarantee arrival, acknowledgement, or integrity. Data is split into datagrams or fragments of datagrams for transmission down to the Network layer. **UDP** protocols include **DNS** (Domain Name Service), **route** (routing protocol), **TFTP** (Trivial File Transfer Protocol), **BOOTP** (Bootstrap Protocol), **SNMP** (Simple Network Management Protocol), and **NFS/RPC** (Network File System/Remote Procedure Call).

Both TCP and UDP interpret the **port numbers** which define the application source or destination, so that data is sent to the appropriate user process, for instance, port 25 for email.

Protocols at the Network layer:

- **IP**: is the workhorse network layer protocol for all TCP, UDP, ICMP and IGMP data. It is an unreliable, connectionless service. It passes Transport level datagrams down to the Link layer as packets.
- **ICMP**: is used to exchange error messages and status information between the IP layers of different machines (**ping** and **traceroute**).
- **IGMP**: is used for multicasting: sending UDP datagrams to multiple hosts, for instance, the **MBONE** (Multicast Backbone).

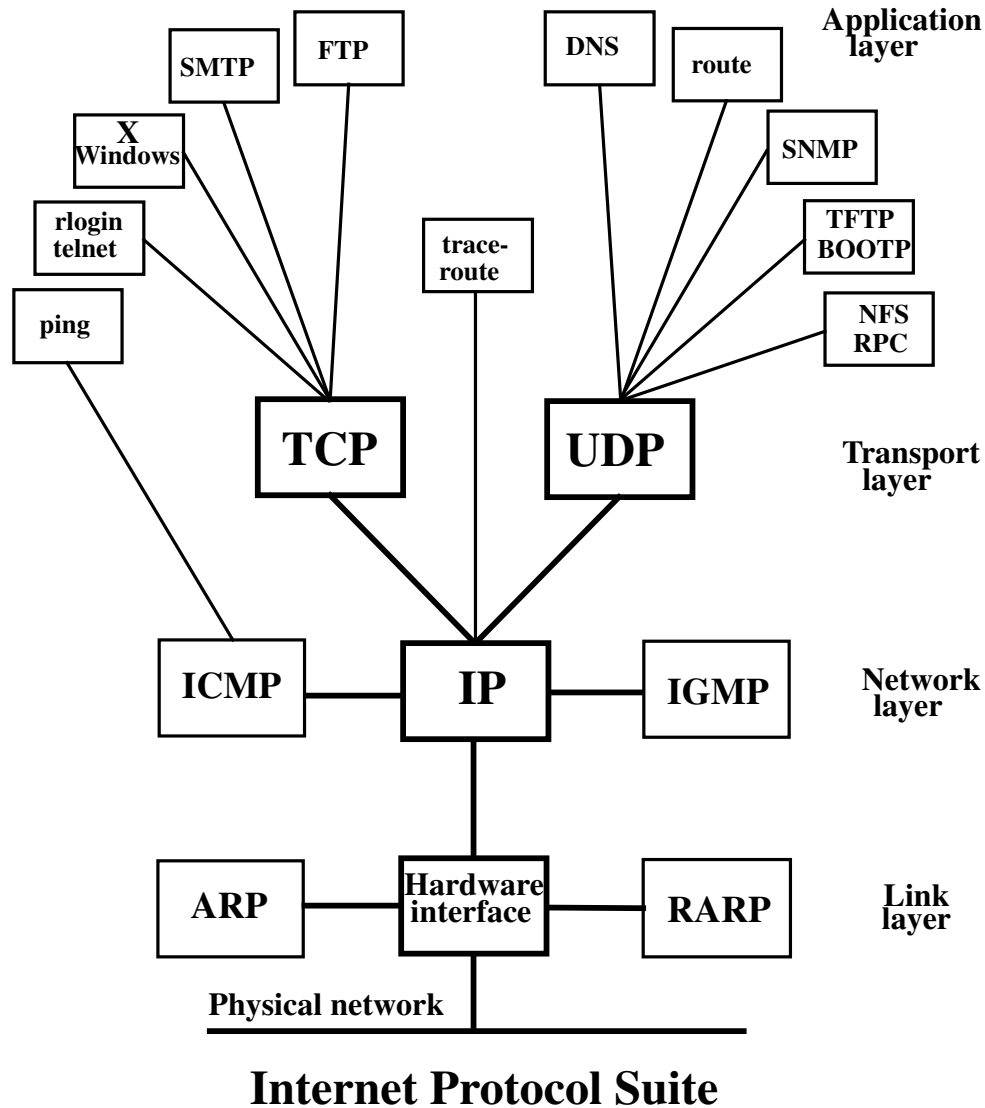
Protocols at the Link layer:

Packets arrive from the Network layer with 32-bit destination IP addresses (e.g. 134.148.96.116) which must be converted to unique hardware address formats, usually 48-bit addresses (e.g. 08:00:20:0b:36:e9) before being transmitted onto the network.

The Link layer varies according to the type of physical network, e.g. FDDI, Ethernet, PPP, token ring, but in general, it encapsulates IP packets in frames before transmitting them. When it re-

ceives a frame it performs the reverse operation before passing the packet up to the Network layer. The two Link layer protocols are:

- **ARP:** the Address Resolution Protocol, which is what converts the IP addresses to Ethernet ones.
- **RARP:** Reverse Address Resolution Protocol—a diskless machine like an X-terminal uses RARP to broadcast its Ethernet address, requesting other machines to reply and tell the diskless machine what its IP address is.



3 Network monitor prerequisites

- You must have root privilege to run almost all network monitors. This is because it is possible with some of them to read the contents of packets, including passwords and private data.

- The host interface to the Ethernet connection must be put into “promiscuous” mode, so it can read all the packets on the network to which it’s attached, not just its own incoming or outgoing packets. This should never be done on a multi-user, non-secure machine, because some cracking programs can be run without root privilege so long as the interface is in promiscuous mode.
- If a PC is attached to the network it can run monitor programs without restriction, which is why the use of PCs on Unix networks must be closely controlled, and why the network must be as physically secure as possible.
- Most monitoring tools were originally written for SunOS 4.x. Many will work on other systems like SGI, DEC and Hewlett-Packard. A few Solaris (SunOS 5.x) versions exist, but porting is very difficult, as the entire network interface was changed in the move between the two systems. We get around this problem by leaving our monitoring machine at SunOS 4.1.3, even though most of the other machines have moved to Solaris.

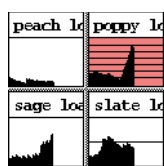
4 Host monitors

4.1 xmeter

xmeter was written by Bob Schwartzkopf. It displays a chart of system statistics for individual hosts, such as CPU usage, job load, collisions, swapping, errors, paging, system use, etc. It can be configured for four different warning levels, to change colour or highlight if the measured usage goes over a certain level. While it’s not strictly a network monitor, conditions on local machines often indicate local network conditions.

For instance, I always have open about eight tiny xmitters, showing the load on each of the multi-user machines. This is a very good warning indicator of heavy machine usage, which often impacts on the network. When an xmeter changes colour it catches the eye and warns that the usage level is rising; when it goes bright red, it’s time to intervene.

Advantages: indicates when machines are overloaded or collision levels are high, so network traffic is lost and needs re-transmitting, throughput becomes degraded and local network mounts are slow. Drawbacks: each invocation of xmeter needs 0.8 to 1.6 Mbytes of memory to run.



4.2 netstat

netstat is a standard Unix utility. With the **-i** or **-I** option it shows the number of packets in and out of the Ethernet interface, the errors and the collisions. It is not a monitoring package, but provides essential information that can not be otherwise obtained.

Name	Mtu	Net/Dest	Address	Ipkts	Ierrs	Opkts	Oerrs	Collis	Queue
le0	1500	134.148.96.0	sol	93296653	138	9341817	1	287745	0

Input errors (Ierrs) may result from electrical problems on the network corrupting packets, or lack of buffer space for receiving packets. Output errors (Oerrs) may indicate a faulty local network cable or a heavily loaded network (Oerrs excludes collisions). Ierrs and Oerrs should be tiny; an error rate of more than 0.025% indicates problems.

The figure for collisions (Collis) shows the number of packets that had to be re-transmitted because the signal occurred at the same time as another host on the network was also transmitting. The collision rate (Collis/Opkts) percentage should average less than 5%. Over 5% shows high network utilisation, and over 10% suggests the network should be partitioned (see **analyser** below). There are no tools available to automate error and collision checking, but a simple script can be used to calculate the rates:

```
netstat -i | awk '{if ($1 ~ /le/) { ie=($6/$5)*100 ; oe=($8/$7)*100 ; coll=($9/$7)*100 ; printf "%4s %6.3f %5s %6.3f %6.2f\n", "in%", ie, "out%", oe, "coll%", coll } }'
```

which gives, for example: in% 0.000 out% 0.000 coll% 3.08

To find the average network collision rate, add up the individual Collis and Oerrs values for all of the machines on the network before calculating the percentage, e.g. [(total Collis)/(total Oerrs)]*100.

Advantages: can be run by a cron script to keep regular statistics of network collision levels.
Drawbacks: not a package so you must collect and reduce data yourself.

5 Information tracers

5.1 etherhostprobe

etherhostprobe was written by Paul E. McKenney. It probes a range of specified IP addresses, by sending ARP packets to each one to find out the Ethernet numbers. Output is a list in the format Ethernet number, IP number, IP name.

Advantages: fast way to check on accuracy and completeness of /etc/ethers files. Drawbacks: the man page suggests it can affect host performance while running, but I did not notice this.

5.2 getethers

getethers was written by David Curry. It is similar to **etherhostprobe**, but has options to write output in a form useful for an Excelan Lanalyzer or a Network General Sniffer.

Advantages: it shows the vendor as well as the hostname, IP address and Ethernet address. Drawbacks: although the man page says it finds all the hosts on a network, it only found the localhost when I tried it.

5.3 netuse

netuse was written by Lee Liming and Michael L. Neil. It dynamically reports about hosts on a network, updating load averages, number of login sessions, display use, free space in /tmp, hostnames, vendor and model types. Daemon processes run on the machines being monitored, which report to a server process on a central host. (I have not used it, included here for completeness.)

5.4 Fremont

Fremont was written by David C. M. Wood, Sean S. Coleman and Michael F. Schwartz. It is a set of modules that seek out network information, timestamp and record it in a database cross-correlated to form a complex network picture. Analysis programs then highlight address conflicts, misconfigurations and any inconsistent information.

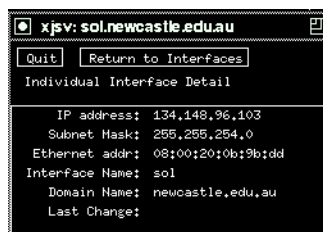
It uses ARP, ICMP, DNS and RIP (routing protocol) to explore the network, then records interface, gateway and subnet data in a Journal database. Data is then presented using X-window displays, of either lists of interfaces and their associated data, or as a picture of the network structure.

Advantages: checks consistency of large amounts of essential information that may have been entered into the system over long periods of time. (On our 75-node network it quickly found 2 incorrect subnet masks.) Drawbacks: Installation instructions are fairly obscure.



The screenshot shows a window titled "xjv: sol.newcastle.edu.au" with a "Quit" button and a "Return to Subnets" button. Below the buttons, it says "Interfaces of 134.148.96.0". The main content is a table with two columns of IP addresses and interface names.

134.148.96.0	134.148.96.81	tek1
134.148.96.2	134.148.96.82	tek2
134.148.96.6	134.148.96.83	tek3
134.148.96.9	134.148.96.84	tek4
134.148.96.10	134.148.96.85	tek5
134.148.96.11	134.148.96.101	miranda
134.148.96.12	134.148.96.102	bigfoot
134.148.96.13	134.148.96.103	sol
134.148.96.15	134.148.96.104	bbsjpc
134.148.96.16	134.148.96.106	mocha
134.148.96.17	134.148.96.107	grape
134.148.96.18	134.148.96.108	peach
134.148.96.19	134.148.96.109	slate
134.148.96.30	134.148.96.110	poppy
134.148.96.31	134.148.96.111	umber
134.148.96.33	134.148.96.113	caaxis
134.148.96.36	134.148.96.114	burgundy
134.148.96.37	134.148.96.115	bw
134.148.96.38	134.148.96.116	lily
134.148.96.39	134.148.96.117	violet
134.148.96.40	134.148.96.120	jungle
134.148.96.41	134.148.96.245	saturn
134.148.96.42	134.148.96.250	cisco-es-os



The screenshot shows a window titled "xjv: sol.newcastle.edu.au" with a "Quit" button and a "Return to Interfaces" button. Below the buttons, it says "Individual Interface Detail". The main content is a list of interface details for the 'sol' interface.

IP address:	134.148.96.103
Subnet Mask:	255.255.254.0
Ethernet addr:	08:00:20:0b:9b:dd
Interface Name:	sol
Domain Name:	newcastle.edu.au
Last Change:	

6 Network access tools

6.1 ping

If you can't access a remote machine, **ping**, a standard Unix command, will tell you if the machine is turned on and attached to a network. (It can't tell you whether or not it is accepting multi-user traffic.) **ping -s** gives more information: a high packet loss rate, or a long maximum round-trip time suggest that the problem may be that the network is very congested at some point.

```

PING ftp.uu.net: 56 data bytes
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=0. time=495. ms
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=2. time=1909. ms
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=3. time=910. ms
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=8. time=726. ms
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=10. time=514. ms
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=11. time=550. ms
64 bytes from ftp.UU.NET (192.48.96.9): icmp_seq=13. time=408. ms
----ftp.uu.net PING Statistics----
14 packets transmitted, 7 packets received, 42% packet loss
round-trip (ms)  min/avg/max = 405/739/1909

```

Advantages: straightforward reachability diagnosis. Drawbacks: there's no indication where on the path to a machine that problems are happening.

6.2 fping

fping was written by Roland J. Schemers, for a network of over 12,000 hosts. It quickly pings large numbers of addresses with ICMP packets to determine if a machine is up. It can be used at the command line or accept input from a file, and can be used in scripts (supplied with the code) to report lists of unavailable hosts by email, for instance.

Advantages: it doesn't wait for timeouts, and it doesn't normally flood the network with packets. Drawbacks: no obvious ones.

6.3 traceroute

traceroute was written by Van Jacobson. It tracks the route that a packet follows to reach its destination. It sends three IP packets to each routing node along the way, and times the return ICMP response. If there's no response before a timeout, the display shows an asterisk.

```

traceroute to ftp.uu.net (192.48.96.9) 30 hops max, 40 byte packets
 1 cisco-es-cs (134.148.96.250)  3 ms  2 ms  2 ms
 2 cisco-mc (134.148.2.1)      2 ms  2 ms  2 ms
 3 un.gw.au (134.148.24.251)   3 ms  2 ms  2 ms
 4 nsw.gw.au (139.130.80.1)    37 ms 33 ms 43 ms
 5 act.gw.au (139.130.192.1)   55 ms 62 ms 51 ms
 6 national.gw.au (139.130.188.1) 170 ms 174 ms 160 ms
 7 usa.gw.au (139.130.240.2)   177 ms * 176 ms
 8 border4-hssi1-0.SanFrancisco.mci.net (204.70.35.5) 395 ms * *
 9 * * *

```

Advantages: it gives a good indication of where the network point of failure is. Drawbacks: excessive use of **traceroute** can contribute to network congestion.

7 Local displays

7.1 xnetload

xnetload is currently maintained by Roger Smith (see the man page for previous authors). It shows the load on the Ethernet with a scrolling stripchart. It is one of the single most useful util-

ities available—I have one running at all times for our local Ethernet.

Advantages: sudden large network file transfers, periodic activity spikes, the load from network dumps, the general usage level, are all immediately obvious. It's usually the first indicator of congestion problems.

Drawbacks: the Ethernet statistics server, `rpc.etherd`, must be explicitly started up beforehand (or started in the `/etc/rc.local` file) to put the network into promiscuous mode (most other programs put the interface into promiscuous mode automatically).



7.2 nfwatch

On local area networks many systems use the NFS (Network File System) originated by Sun Microsystems. This permits disks that are physically located on different machines (servers) to appear as if they are locally mounted.

nfwatch was written by David A. Curry and Jeffrey C. Mogul. It is a utility that monitors all Ethernet traffic to an NFS server. The number and percentage of packets received by the host for each IP protocol is displayed on the screen in a continuously updated display. The program can watch traffic to an NFS client (a remote machine mounting a local disk), save data to a logfile, or check to see if specified files are being accessed. (Another NFS monitor program is **nfstrace**, by Matt Blaze, which provides traces of low-level NFS commands.)

Advantages: useful to see if excessive traffic is going to a single disk—which may be the result of one problem application rather than general traffic overload. Drawbacks: It cannot see any traffic that the monitoring host generates in response to NFS requests.

```

all hosts                Thu Jun 22 14:56:45 1995      Elapsed time: 00:01:36
Interval packets:      1460 (network)      1460 (to host)      0 (dropped)
Total packets:        39559 (network)     39559 (to host)     10 (dropped)
Monitoring packets from interface le0

```

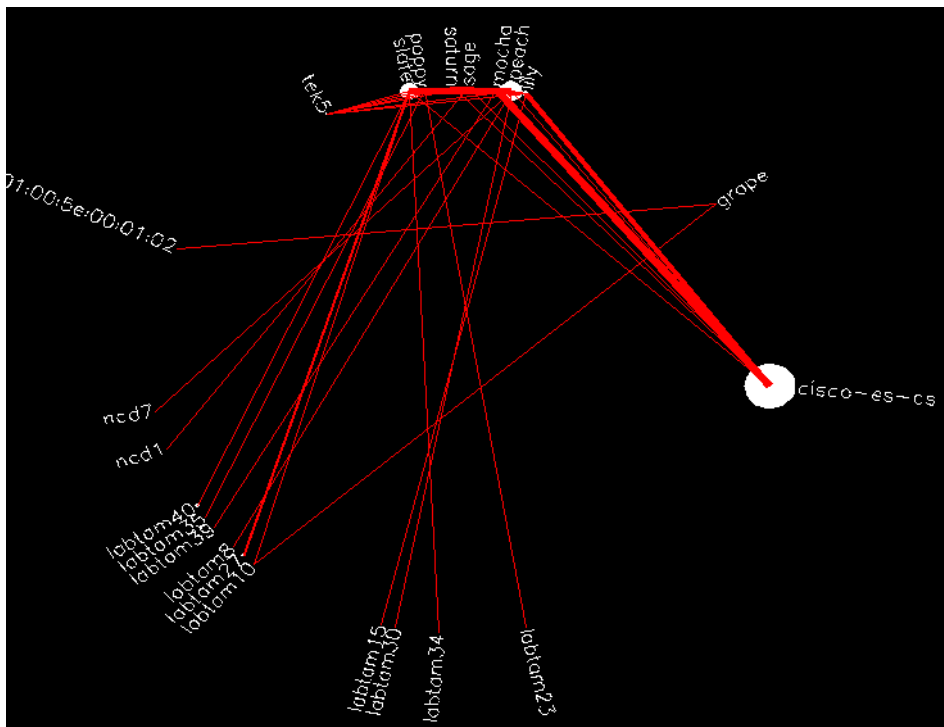
	int	pct	total		int	pct	total
ND Read	0	0%	0	TCP Packets	1258	86%	29356
ND Write	0	0%	0	UDP Packets	172	12%	9502
NFS Read	23	2%	616	ICMP Packets	3	0%	42
NFS Write	8	1%	222	Routing Control	1	0%	7
NFS Mount	1	0%	4	Address Resolutio	3	0%	31
Yellow Pages/NIS	5	0%	2583	Reverse Addr Reso	0	0%	6
RPC Authorization	53	4%	4296	Ethernet Broadcas	3	0%	102
Other RPC Packets	19	1%	517	Other Packets	24	2%	622
6 file systems							
File Sys	int	pct	total	File Sys	int	pct	total
lily(128,14)	8	26%	37				
peach(128,14)	7	23%	449				
poppy(128,14)	0	0%	1				

7.3 etherman

etherman was written by Mike Schulze, Craig Farrell and George Benko of Curtin University. It is a colourful X-based tool which dynamically displays a representation of real-time Ethernet traffic. It also provides protocol summaries, usage statistics, and PostScript snapshots of network activity. You need an up-to-date local `/etc/ethers` file so that Ethernet number-to-hostname translations can be done. It also requires installation of the **hershey** font libraries.

Advantages: heavy traffic between particular hosts, or between the network router and a host, is highlighted. The line linking two hosts gets larger with load, and the spot indicating the host which is the source of the traffic also gets larger in real time. You can save the data to files which summarise the traffic between all the hosts during the monitoring period. It is very good for immediate problem diagnosis.

Drawbacks: default mode picks up odd Ethernet packets, like AppleTalk broadcasts, that may not be of interest and there's no way to define your own preferred defaults—you need to reset filters for, say, just the IP traffic, each time you start it. It's an interactive real-time display, so you can't run it from the command line or cron to grab a sample of traffic at some later time.



7.4 analyser

analyser was written by Mazahir Songerwala, Craig Farrell and Mike Schulze. It is an X-based utility which suggests segmentation topologies for networks. It operates on the short summary output from **etherman** or files in the same format (source, destination, bytes, packets). For instance, if your network is frequently congested it may be useful to separate it into several subnets, so that recurring high traffic levels occurring between specific hosts (like a student server and a lab of X-terminals) are restricted to one subnet.

analyser uses three algorithms to recommend the best splits given the input traffic samples. Cluster (K-Means) is fast (a few seconds) but fairly simple-minded; Genetic Algorithms takes several minutes and provides useful suggestions; Simulated Annealing takes so long (hours) that I've never bothered to wait for the output.

Advantages: an unusual and potentially very useful tool for suggesting solutions to local network congestion.

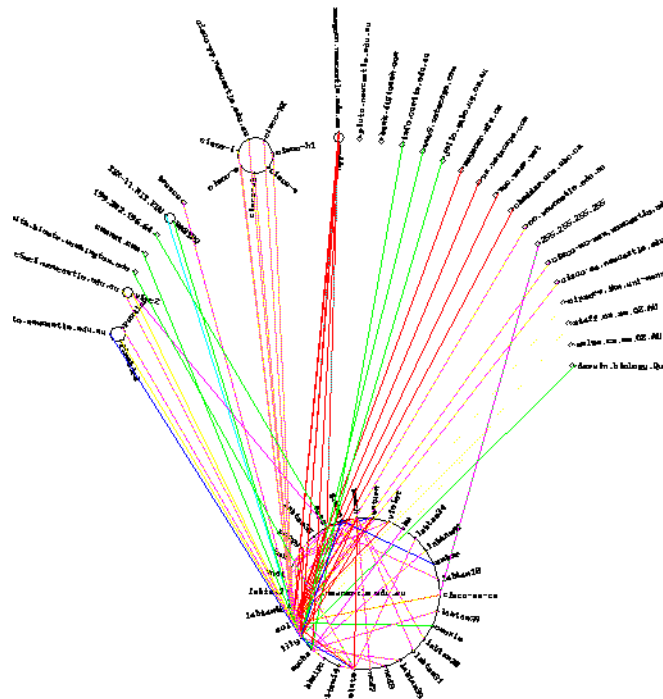
Drawbacks: samples need to be representative of normal traffic conditions, so will have to be large, which may substantially increase analysis time. Currently it can only make recommendations for two subnets and up to 256 hosts.

8 Inter-network displays

8.1 interman

interman is a companion package to **etherman**, written by the same people. It shows the real-time IP traffic only, displaying data transfers between local and remote networks as lines with different colours for different protocols. It provides protocol summaries, usage statistics and Post-Script snapshots. It also requires installation of the **hershey** font libraries.

Advantages: same as **etherman**. Drawbacks: same as **etherman** except there's no need to filter out odd packets as it's restricted to the IP traffic anyway.



8.2 ethertop

ethertop was written by Guy Cardwell. It has a periodically updating list of amount of traffic sent to and received from hosts on the network. It is like a text-based version of the old Sunview utility **traffic**, and can be used to quickly identify the source of excessive traffic. It accepts on-the-fly terminal input, like "l" to redraw the screen. As well as the host lists it shows per-second statistics of bytes, packets, broadcasts, tcp, udp, icmp, arp, nd and other traffic, and a breakdown of packet sizes.

Advantages: text-based, extremely useful summary of source-destination activity. Drawbacks: sometimes appears to crash previously started rpc.etherd daemons (e.g. if **xnetload** is already running).

Network load as seen from sol

bytes	pkts	bcst	tcp	udp	icmp	arp	nd	oth
54.53K	333.48	0.39	292.49	33.44	0.00	0.77	0.00	6.77

packet sizes:

64-154	155-245	246-336	337-427	428-518	519-609	610-699	700-790
292.49	11.60	3.48	1.93	2.51	0.00	0.00	0.00
791-881	882-972	973-1063	1064-1154	1155-1245	1246-1336	1337-1427	1428-1518
0.00	3.48	0.19	0.00	0.00	0.00	0.00	17.79

HOSTNAME	SENT	HOSTNAME	RECV
bronco.newcastle.edu.au	217.00	sol.newcastle.edu.au	233.40
peach.newcastle.edu.au	30.39	barracuda.newcastle.edu.au	20.95
lily.newcastle.edu.au	23.23	vlmc2.newcastle.edu.au	16.49
vlmc2.newcastle.edu.au	22.84	lily.newcastle.edu.au	16.30
miranda.newcastle.edu.au	10.65	miranda.newcastle.edu.au	10.67
saturn.newcastle.edu.au	5.42	saturn.newcastle.edu.au	7.37
barracuda.newcastle.edu.au	3.48	peach.newcastle.edu.au	7.18

8.3 ipwatch

ipwatch was written by Ivan R Judson. It gives a text-based dynamic display of network traffic. Advantages: doesn't need a graphical interface. Drawbacks: no manual page.

```

Load:                3% *
  arps                0      0
  IPmcast             1      0
TCP                  928    73% *****
  telnet             126     9% ****
  rlogin             340    26% *****
  Xwindow            348    27% *****
  mail                0      0%
  nntp                8      0%
  rsh                 0      0%
  ftpdata            21     1%
  ftp                 0      0%
  other              85     6% ***
UDP                  340    26% *****
  nfs                223    17% *****
  domain             0      0%
  ntp                 2      0%

```

who	0	0%
syslog	0	0%
snmp	0	0%
other	115	9% ****

9 Traffic counters

9.1 NeTraMet and NeMaC

NeTraMet and **NeMaC** were written by Nevil Brownlee. They are the first implementation of the Internet Accounting Architecture, defined in RFC 1272. **NeTraMet** is an accounting meter which builds up packet and byte counts for traffic flows between end-point addresses. It is managed by **NeMaC**, which downloads rule-sets and uses SNMP to collect the traffic data from the **NeTraMet** meters.

Advantages: very useful for providing detailed information about traffic flows and activity patterns—no packet content information is recorded, so privacy and security are not breached. It is still undergoing development by the IETF Accounting Group (join the mailing list by sending a request to accounting-wg-request@wugate.wustl.edu).

Drawbacks: requires a good understanding of TCP/IP protocols and SNMP terminology. The rulesets are complex, and to arrive at meaningful information requires post-processing of the collected data.

```
##NeTraMet v2.1: -c1 -r rules sol 3000 flows starting at 15:58:55 Fri 29 Jul 94
#Format: flowruleset flowindex firsttime sourcepeertype sourcepeeraddress dest
peeraddress sourcedetailtype sourcedetailaddress destdetailaddress topdus fro
mpdus tooctets fromoctets
#Time: 15:58:55 Fri 29 Jul 94 sol Flows from 1 to 1600
#Stats: aps=151 apb=0 mps=245 mpb=0 lsp=0 avi=0.0 mni=0.0 fiu=3 frc=0 gci=5 rpp=
1.0 tpp=0.0 cpt=0.0 tts=0 tsu=0
1 2 200 12 3D-3E-00-00-00-00 3F-40-00-00-00-00 0 0 0 49 0 4705 0
1 3 200 2 11.12.0.0 13.14.0.0 0 0 0 2034 0 313094 0
1 4 300 7 41.42.0 43.44.0 0 0 0 33 0 2028 0
4 5 1600 2 134.148.96.0 134.148.96.0 6 3762 6000 4 9 268 972
4 6 1600 2 134.148.96.0 134.148.32.0 6 58745 6000 1 4 134 292
4 7 1600 2 134.148.96.0 134.148.140.0 6 3242 6000 1 3 134 206
4 8 1600 2 134.148.96.0 134.148.96.0 17 663 44285 0 1 0 106
4 9 1600 7 0.0.0 0.0.0 1 0 0 1 1 63 63
4 10 1600 2 134.148.96.0 134.148.96.0 6 58750 6000 12 22 1524 1932
4 11 1600 2 134.148.96.0 134.148.96.0 6 4485 6000 0 7 0 550
```

9.2 nnstat

nnstat was written by Robert T. Braden and Annette L. DeSchon. It is a tool for the distributed collection of traffic statistics. Data is acquired at network entry-points, then collected and stored by a single host. It does not provide display or analysis programs. It was used to gather usage statistics for the NSFNET when that was the US backbone. (I have not used it, but include it for completeness.)

10 Packet analysers

10.1 xnetmon

xnetmon was written by L. Th. Weerpas and W. A. P. Haverhals. It is a set of dynamic X windows that display statistics and network parameters for the protocols TCP, UDP, IP, ICMP, memory buffers, and TCP and UDP sockets.

Advantages: fast, informative displays of active connections; actual data contents are hidden.
Drawbacks: no save-to-file option for later traffic analysis.

```
NetMon v1.0, (c) '92 by W.A.P.Haverhals & L.Th.Weerpas
Uptime: 483.35 [s] Interval: 1.02 [s]
----- TU Delft, the DNPAP group -----

TCP_GENERAL #COUNTER TCP_SEND #COUNT PER/SEC TCP_RECEIVE #COUNT PER/SEC
conn. attempts 19990 send pack total 7825994 3.92 rcvd pack total 12950344 3.92
conn. accepted 32616 data packet 4167029 2.94 pack in seq. 10947346 0.38
conn. established 48991 data bytes 1966101895 1847.04 bytes in seq. 2942650314 125.49
conn. dropped 795 data pack rx 7802 0.00 pack bad sum 58 0.00
embryonic dropped 3492 data byte rx 804837 0.00 pack bad ofs 0 0.00
conn. closed 63541 ack only pack 1330374 0.98 pack tooshort 0 0.00
timing segments 3010498 window probe 1266 0.00 pack dup.only 21453 0.00
timing updates 2982839 urg only 0 0.00 byte dup.only 5311052 0.00
delayed segments 1211148 win update 2246473 0.00 pack dup some 93 0.00
conn drop_timeout 11 other ctrls 72254 0.00 byte dup some 18065 0.00
r_xmit_timeout 11317 pack oob 88728 0.00
persist_timeout 1291 byte oob 57610509 0.00
keepalive_timeout 992 pack overflow 380 0.00
keepalive_probes 69 byte overflow 179232 0.00
keepalive_drops 325 pack after c1 126 0.00
pack win prob 297 0.00
pack dup ack 38918 0.00
pack ack2much 0 0.00
pack w/ack 3276989 2.94
byte acked 1966231740 1847.04
pack win updt 17431 0.00
```

```
-TCP SOCK XNetmon v1.0 by Weerpas & Haverhals
NetMon v1.0, (c) '92 by W.A.P.Haverhals & L.Th.Weerpas
Uptime: 309.61 [s] Interval: 1.94 [s]
----- TU Delft, the DNPAP group -----

STATE REMOTE_ADDRESS_/_PORT_ LOCAL_ADDRESS_/_PORT_ BYTE_SENT_ BYTE_RCVD_ SOCK_OPTIONS_
ESTABLISHED lily.newcastle., ident sol.newcastle.e., 1886 11 1
ESTABLISHED lily.newcastle., 43764 sol.newcastle.e., smtp 1 1
ESTABLISHED merlin.psmel.b., 22510 sol.newcastle.e., smtp 1 1
ESTABLISHED lily.newcastle., 6000 sol.newcastle.e., 1882 222969 3785
ESTABLISHED lily.newcastle., 6000 sol.newcastle.e., 1882 281657 4041
```

10.2 tcpdump

tcpdump was written by Van Jacobson, Craig Leres and Steven McCanne. It prints out the headers and part of the data of the packets on a network. All packets or just those of particular protocols can be selected. Link, Network, Transport or Application-layer data can be collected, with detailed timestamps, and saved to a file for later analysis. (Files containing one hour's data ranged from 10 to 50 Mbytes in size for our university backbone). It also requires installation of **libpcap**, a system-independent interface for packet capture.

Since it can select packets on the basis of ports, information is available about the usage of applications like FTP, email, WWW, gopher, etc., but deriving total statistics requires post-processing of the collected data. The total packet size, headers and data, can be shown with the `-e` flag (the size of the transferred data only is shown by default).

Advantages: it is a very good learning tool (see Reference [1], which uses it extensively to illustrate TCP/IP protocol activity). The manual page is good, but does require some previous knowledge of TCP/IP.

Drawbacks: packet data can be examined, breaching privacy and security. Data files are large. **tcpdump** works well under SunOS 4.x and SunOS 5.4 (Solaris 2.4). Don't compile it under Solaris 2.3 because it creates data files about 10 times larger than normal. Also, when **tcpdump** finishes, it reports the number of packets received and dropped. Under Solaris the dropped packets field is always zero, which is not correct.

Examples (timestamps and source and destination names have been abbreviated for clarity):

Link layer—ARP:

```
(time) 8:0:2b:32:b0:84 8:0:20:1f:35:54 arp 60: arp who-has seagoon tell bluebottle
(time) 8:0:20:1f:35:54 8:0:2b:32:b0:84 arp 60: arp reply seagoon is-at 8:0:20:1f:35:54
```

Fields: (time) is the datestamp, source Ethernet address, destination Ethernet address, protocol, size of complete packet, protocol-specific request and reply.

Network Layer—ICMP (ping):

```
(time) 0:0:c:3:df:1a 8:0:20:9:6a:a0 ip 98: viper > neddy: icmp: echo request
(time) 8:0:20:9:6a:a0 0:0:c:3:df:1a ip 98: neddy > viper: icmp: echo reply
```

Transport layer—UDP (DNS):

```
(time) 0:0:c:3:df:1a 8:0:20:9:6a:a0 ip 99: dewey.1661 > neddy.domain: 2+ (57)
(time) 8:0:20:9:6a:a0 0:0:c:3:df:1a ip 168: neddy.domain > dewey.1661:2NXDomain*0/1/0 (126)
```

Transport layer—TCP (telnet):

```
(time) 0:0:c:0:84:31 8:0:2b:32:b0:84 ip 60: un.gw.au.telnet > bluebottle.4506:61:64(3) ack 110 win 2035
(time) 8:0:2b:32:b0:84 0:0:c:0:84:31 ip 60: bluebottle.4506 > un.gw.au.telnet:.ack 64 win 33580 [tos 0x10]
```

Fields common to both TCP packets above: datestamp, source Ethernet address, destination Ethernet address, protocol, size of complete frame, source name or IP address, source port, destination name or IP address, destination port.

Subsequent fields, 1st line: Push flag, packet sequence numbers and number of data bytes sent, acknowledgement of previous packet number, receive window size.

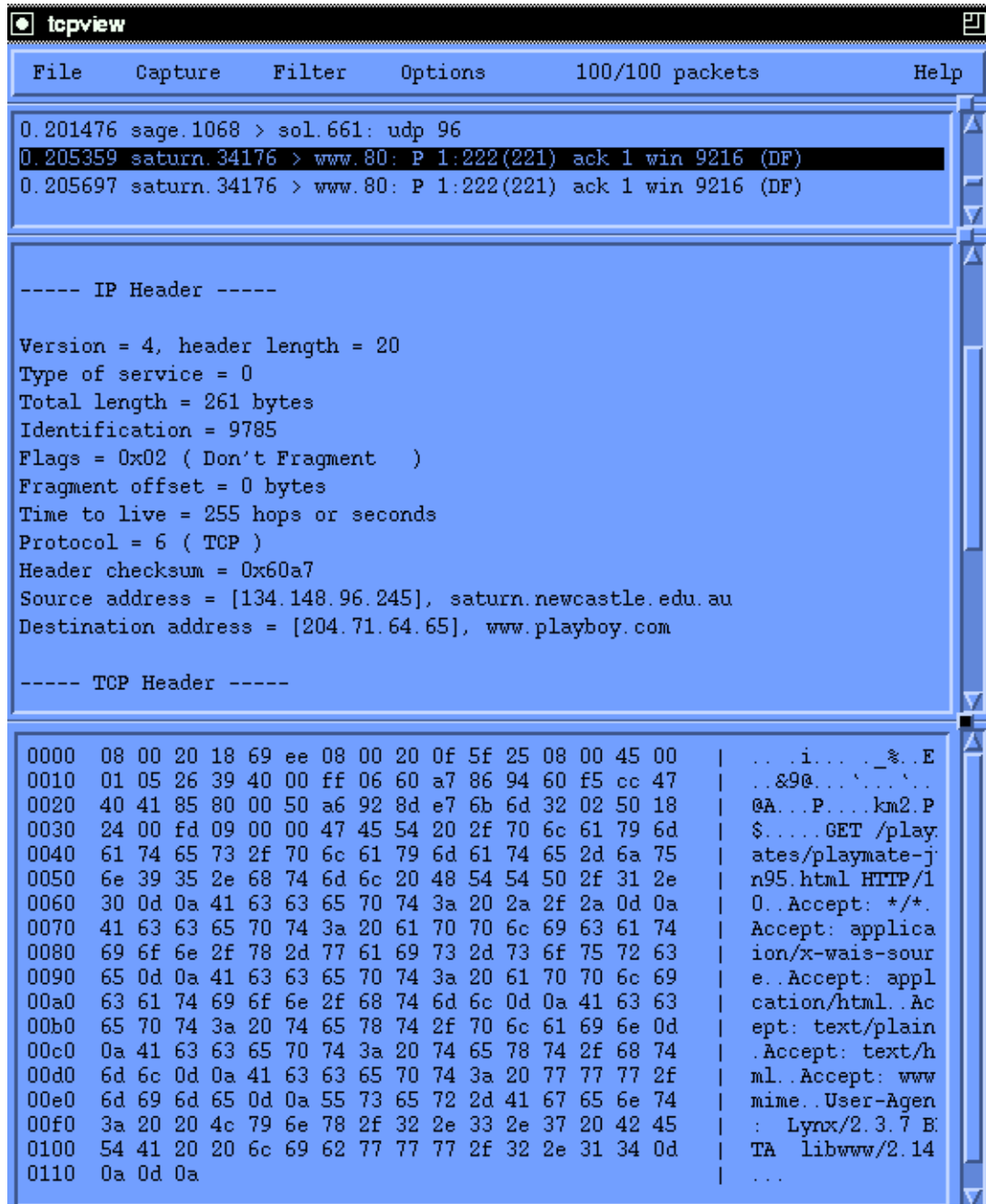
Subsequent fields, 2nd line: no set flag, acknowledgement of previous packet number, receive window size, type-of-service setting (not currently supported in TCP/IP).

10.3 tcpview

tcpview is a Motif interface to **tcpdump**. It can read previously collected **tcpdump** data, or capture current data. It has three windows—the top one shows the data, as in **tcpdump**. Selecting a line in the top window activates the other two: the middle window then shows a detailed decoding of the frame and the bottom a hexdump of the whole frame.

Advantages: interactive interface, follows out-of-order TCP frames, enhanced protocol decoding.

Drawbacks: requires Motif graphical user interface system. Packet data can be examined: the example below (selected entirely at random) shows information that the sender would probably have preferred to remain hidden!



10.4 packetman

packetman is another companion program to **etherman**, written by George Benko and Greg Barron. It is an X-based utility that captures packets and examines their headers and contents. It has a display similar to **tcpview**.

Advantages: same as **tcpview**, but does not require Motif. Drawbacks: as with other packet analysers, data can be examined.

10.5 netlog

netlog was written by staff at Texas A&M University. It records TCP and UDP traffic, to the screen, or to a file for later processing.

Advantages: straightforward format. Packet data is hidden. Drawbacks: it does not provide as much detail as other packet analysers (e.g. no byte size).

```
06/27/95 10:48:31.58 1CA0201 pluto.newcastle.edu.au 1023 ->peach.newcastle.edu
.au printer
06/27/95 10:48:34.77 98430400 slate.newcastle.edu.au 4396 ->tesla.newcastle.edu
```

10.6 etherfind and snoop

Proprietary programs exist with many of the features and similar syntax to **tcpdump**: SunOS 4.x has **etherfind**, and **snoop** is the Solaris equivalent. As with the other utilities you must be root to use either of them. On Silicon Graphics systems, **snoop** (7) provides a programming interface for writing packet capture applications.

10.7 tcptop

tcptop was written by Steven Grimm. It is a front-end to Sun's **etherfind**, and shows a dynamic display of active TCP connections on a network, in columns of data with source host and port, total bytes sent, connection status, delay, destination host and port, time started.

Advantages: easy to see unusual active connections. Packet data is not accessible.

Drawback: You must be using SunOS 4.x with NIS (Yellow Pages) running. Unless you set update time to at least 5 seconds it's hard to follow. It's very difficult to exit from it cleanly. No save-to-file facility.

11 Monitors useful in emergencies

- When a host is unreachable at any level of the network, from local to remote, **ping** and **traceroute** are the tools of first choice. They will indicate whether or not the problem is at the remote machine, or at any of the hops in between.
- For local traffic problems, such as slow connections, poor response, or high error rates, the first tools to use are **xmeter** and **xnetload**. **xmeter** will show if it is the machines themselves that are the problem, and **xnetload** will indicate if it's heavy network traffic.
- If the problem seems to be heavy traffic, identify the source with **ethertop** to show traffic levels from both local and remote hosts. **etherman** and **interman** would also be useful to identify sources of excessive traffic.

12 Monitors for performance and planning

- High local traffic levels over the long-term could be studied with **etherman/analyser** samples, to find out if physically separating the network into subnets would help.
- Network collision rates can be collected using **netstat** and again, if high, **analyser** might be useful.
- For measuring traffic levels between local and remote sites, either total amounts, per-protocol usage, or long-term changes, **NeTraMet/NeMaC** appears most useful: it has been used at many sites in New Zealand for several years where volume accounting and charging are standard.

13 Ethical questions

Apart from the packet analysers, most monitors permit the contents of packets to remain private and do not provide any direct means of identifying users. In the case of the packet analysers (apart from **tcptop**, **netlog** and **xnetmon**) however, there is a strong possibility of breaching the privacy and security of user communications.

To examine traffic for protocol usage, or to collect accounting statistics does not really require packet analysers (which generate enormous data files anyway)—**etherman**, **netlog**, **nnstat** or **NeTraMet/NeMaC** would be more useful for these purposes.

Packet analysers should be carefully restricted to educational uses or research into traffic format and structure, by people who are aware of the responsibility such privileged access entails. But for most system administrators, that responsibility is part of the job, so it could be argued that the ethical questions arising from monitoring network traffic do not really differ from those we must meet in the normal course of everyday work.

14 Sources

Most of the packages described in this paper are available by anonymous FTP from:

ftpcs.newcastle.edu.au, in **/pub/network_monitors**.

15 References

[1] W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, ISBN 0-201-63346-9.

Summary

	gui	text	inter	hrea	nact	nrea	time	spro	spac	sbyt	dpro	dpac	dbyt	sour	dest	size	head	data	file
xmeter	*																		
netstat	*	*		*					*										
e'hostprobe	*	*		*					*										*
getethers	*	*		*					*										*
netuse	*	*		*					*										*
Fremont	*	*		*					*										*
ping	*	*		*					*										*
fping	*	*		*					*										*
traceroute	*	*		*					*										*
xnetload	*	*		*					*										*
nfswatch	*	*		*					*										*
etherman	*	*		*					*										*
analyser	*	*		*					*										*
interman	*	*		*					*										*
ethertop	*	*		*					*										*
ipwatch	*	*		*					*										*
NeTraMet	*	*		*					*										*
nnstat	*	*		*					*										*
xnetmon	*	*		*					*										*
tcpdump	*	*		*					*										*
tcpview	*	*		*					*										*
packetman	*	*		*					*										*
netlog	*	*		*					*										*
eth/find/snoop	*	*		*					*										*
tcptop	*	*		*					*										*

Key:

gui:	GUI display	text:	text display	inter:	interactive	hrea:	host reachability
hpar:	host parameters	nact:	network activity	nrea:	network reachability	time:	timestamps
spro:	summarizes protocols	spac:	summarizes packets	sbyt:	summarizes bytes	dpro:	dynamic protocols
dpac:	dynamic packets	dbyt:	dynamic bytes	sour:	sources	dest:	destinations
size:	packet sizes	head:	header contents	data:	data contents	file:	writes to file

