expand over time. ***RRDtool*** refers to a suite of tools that allow you to create and modify RRD databases, as well as generate useful graphs to present the data. It is used to keep track of time-series data (such as network bandwidth, machine room temperature, or server load average) and can display that data as an average over time.

Note that RRDtool itself does not contact network devices to retrieve data. It is merely a database manipulation tool. You can use a simple wrapper script (typically in shell or Perl) to do that work for you. RRDtool is also used by many full featured front-ends that present you with a friendly web interface for configuration and display. RRD graphs give you more control over display options and the number of items available on a graph as compared to MRTG.

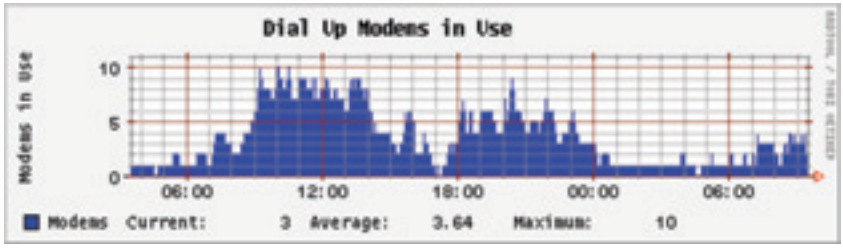


*Figure 6.15: RRDtool gives you a lot of flexibility in how your collected network data may be displayed.*

RRDtool is included in virtually all modern Linux distributions, and can be downloaded from *http://oss.oetiker.ch/rrdtool/.*

## ntop

*http://www.ntop.org/*. For historical traffic analysis and usage, you will certainly want to investigate **ntop**. This program builds a detailed real-time report on observed network traffic, displayed in your web browser. It integrates with rrdtool, and makes graphs and charts visually depicting how the network is being used. On very busy networks, ntop can use a lot of CPU and disk space, but it gives you extensive insight into how your network is being used. It runs on Linux, BSD, Mac OS X, and Windows.

Some of its more useful features include:

- Traffic display can be sorted by various criteria (source, destination, protocol, MAC address, etc.).
- Traffic statistics grouped by protocol and port number
- An IP traffic matrix which shows connections between machines
- Network flows for routers or switches that support the NetFlow protocol
- Host operating system identification

- P2P traffic identification
- Numerous graphical charts
- Perl, PHP, and Python API

Ntop is available from *http://www.ntop.org/* and is available for most operating systems. It is often included in many of the popular Linux distributions, including RedHat, Debian, and Ubuntu. While it can be left running to collect historical data, ntop can be fairly CPU intensive, depending on the amount of traffic observed. If you are going to run it for long periods you should monitor the CPU utilization of the monitoring machine.

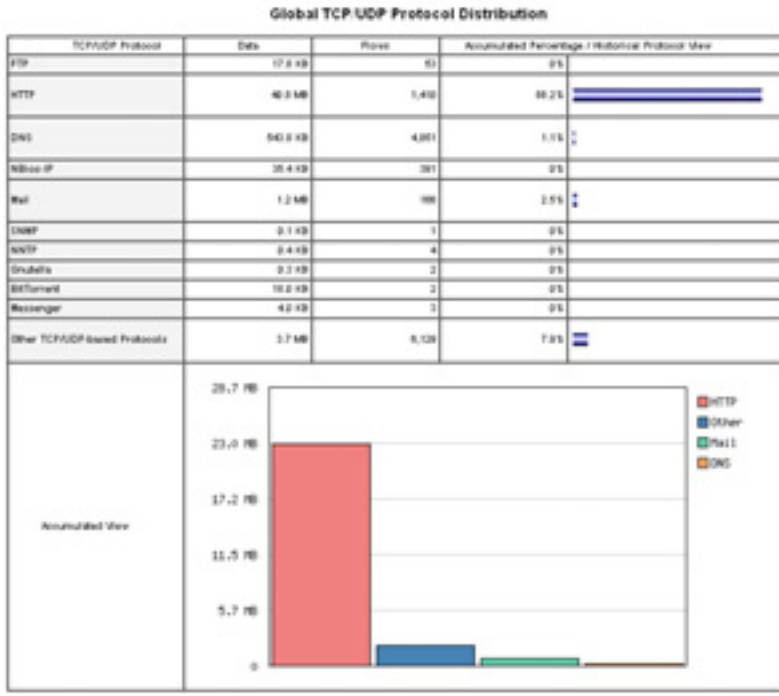


*Figure 6.16: ntop displays a wealth of information about how your network is utilized by various clients and servers.*

The main disadvantage of ntop is that it does not provide instantaneous information, only long-term totals and averages. This can make it difficult to use to diagnose a problem that starts suddenly.

## Cacti

*http://www.cacti.net/*. ***Cacti*** is a front-end for RRDtool. It stores all of the necessary information to create graphs in a MySQL database. The front-end is written in PHP. Cacti does the work of maintaining graphs, data sources,

and handles the actual data gathering. There is support for SNMP devices, and custom scripts can easily be written to poll virtually any conceivable network event.

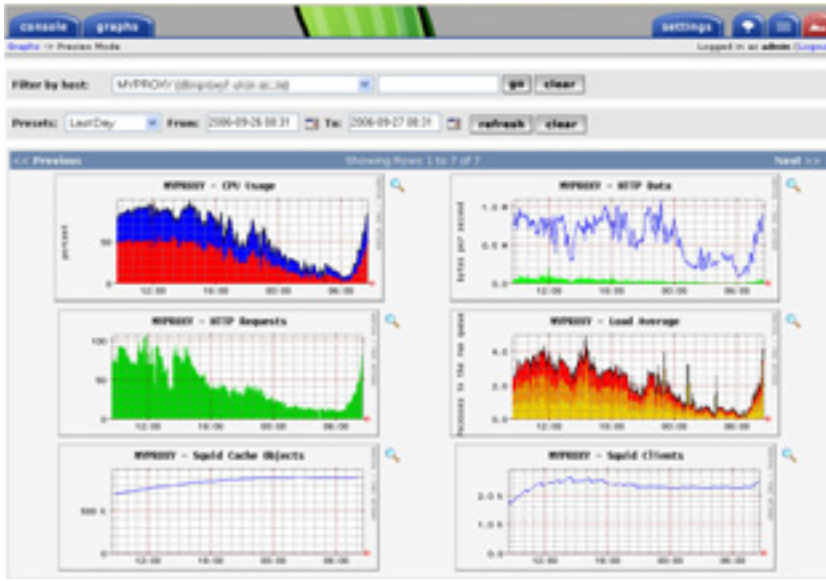

*Figure 6.17: Cacti can manage the polling of your network devices, and can build very complex and informative visualizations of network behavior.*

Cacti can be somewhat confusing to configure, but once you work through the documentation and examples, it can yield very impressive graphs. There are hundreds of templates for various systems available on the cacti website, and the code is under rapid development.

## NetFlow

NetFlow is a protocol for collecting IP traffic information invented by Cisco. From the Cisco website:

> *Cisco IOS NetFlow efficiently provides a key set of services for IP applications, including network traffic accounting, usage-based network billing, network planning, security, Denial of Service monitoring capabilities, and network monitoring. NetFlow provides valuable information about network users and applications, peak usage times, and traffic routing.*

Cisco routers can generate NetFlow information which is available from the router in the form of UDP packets. NetFlow is also less CPU-intensive on Cisco routers than using SNMP. It also provides more granular information than SNMP, letting you get a more detailed picture of port and protocol usage.

This information is collected by a NetFlow collector that stores and presents the data as an aggregate over time. By analyzing flow data, one can build a picture of traffic flow and traffic volume in a network or on a connection. There are several commercial and free NetFlow collectors available. Ntop is one free tool that can act as a NetFlow collector and probe. Another is Flowc (see below).

It can also be desirable to use Netflow as a spot check tool, by just looking at a quick snapshot of data during a network crisis. Think of NetFlow as an alternative to SNMP for Cisco devices. For more information about NetFlow, see *http://en.wikipedia.org/wiki/Netflow* .

## Flowc

*http://netacad.kiev.ua/flowc/*. **Flowc** is an open source NetFlow collector (see NetFlow above). It is lightweight and easy to configure. Flowc uses a MySQL database to store aggregated traffic information. Therefore, it is possible to generate your own reports from the data using SQL, or use the included report generators. The built-in report generators produce reports in HTML, plain text or a graphical format.

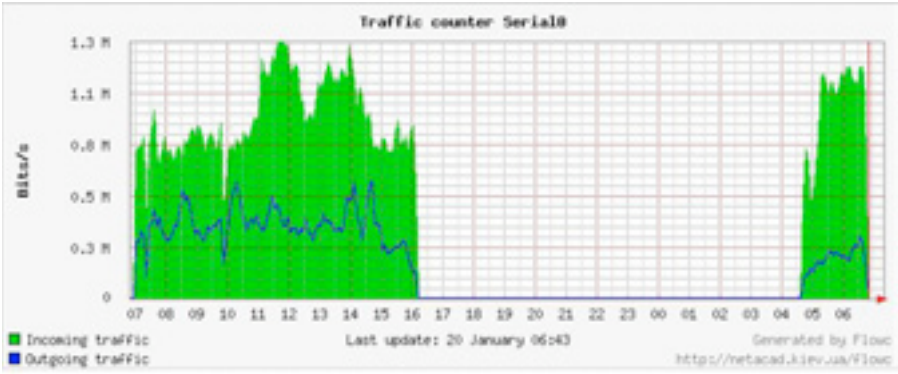


*Figure 6.18: A typical flow chart generated by Flowc.*

The large gap in data probably indicates a network outage. Trending tools typically will not notify you of outages, but merely log the occurrence. To be notified when network problems occur, use a realtime monitoring tool such as Nagios (see **Page 200**).

## SmokePing

*http://oss.oetiker.ch/smokeping/*. ***SmokePing*** is a deluxe latency measurement tool written in Perl. It can measure, store and display latency, latency distribution and packet loss all on a single graph. SmokePing uses RRDtool for data storage, and can draw very informative graphs that present up to the minute information on the state of your network connection.

It is very useful to run SmokePing on a host with good connectivity to your entire network. Over time, trends are revealed that can point to all sorts of network problems. Combined with MRTG (see **Page 190**) or Cacti (see **Page 192**), you can observe the effect that network congestion has on packet loss and latency. SmokePing can optionally send alerts when certain conditions are met, such as when excessive packet loss is seen on a link for an extended period of time. An example of SmokePing in action is shown in **Figure 6.19**.

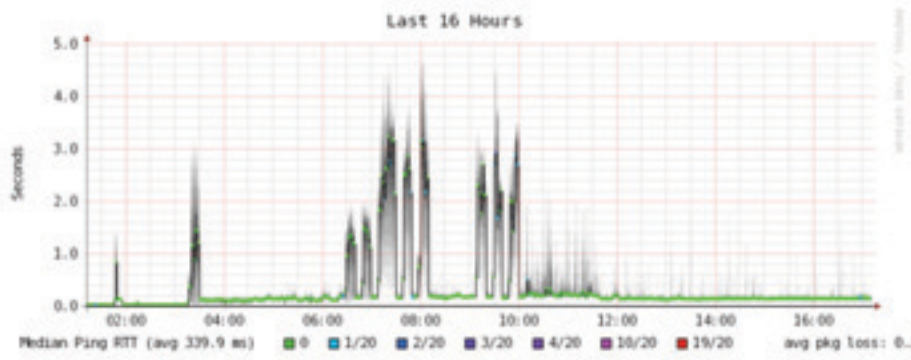


*Figure 6.19: SmokePing can simultaneously display packet loss and latency spreads in a single graph.*

## EtherApe

*http://etherape.sourceforge.net/*. **EtherApe** displays a graphical representation of network traffic. Hosts and links change size depending on the amount of traffic sent and received. The colors change to represent the protocol most used. As with wireshark and tcpdump, data can be captured "off the wire" from a live network connection or read from a tcpdump capture file.

EtherApe doesn't show quite as much detail as ntop, but its resource requirements are much lighter.

## iptraf

*http://iptraf.seul.org/*. **IPTraf** is a lightweight but powerful LAN monitor. It has an ncurses interface and runs in a command shell. IPTraf takes a moment to measure observed traffic, and then displays various network statistics including TCP and UDP connections, ICMP and OSPF information, traffic flows, IP checksum errors, and more. It is a simple to use program that uses minimal system resources.

While it does not keep historical data, it is very useful for displaying an instantaneous usage report.

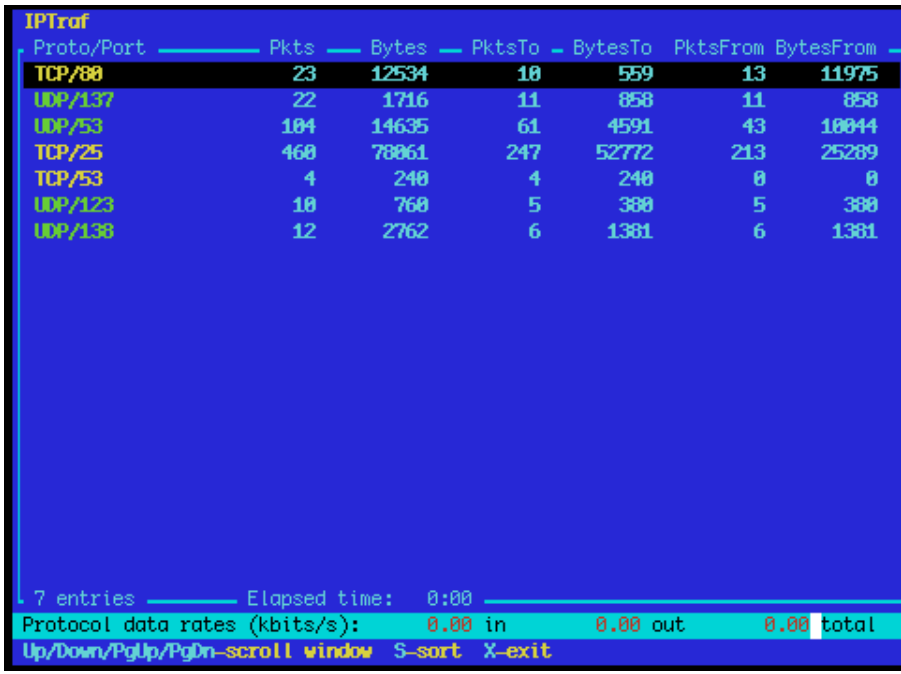


*Figure 6.20: iptraf's statistical breakdown of traffic by port.*

## Argus

*http://qosient.com/argus/*. ***Argus*** stands for ***Audit Record Generation and Utilization System***. Argus is also the name of the mythological Greek god who had hundreds of eyes.

From the Argus website:

> *Argus generates flow statistics such as connectivity, capacity, demand, loss, delay, and jitter on a per transaction basis. Argus can be used to analyze and report on the contents of packet capture files or it can run as a continuous monitor, examining data from a live interface; generating an audit log of all the network activity seen in the packet stream. Argus can be deployed to monitor individual end-systems, or an entire enterprises network activity. As a continuous monitor, Argus provides both push and pull data handling models, to allow flexible strategies for collecting network audit data. Argus data clients support a range of operations, such as sorting, aggregation, archival and reporting.*

Argus consists of two parts: a master collector that reads packets from a network device, and a client that connects to the master and displays the usage statistics. Argus runs on BSD, Linux, and most other UNIX systems.

### NeTraMet

*http://freshmeat.net/projects/netramet/.* ***NeTraMet*** is another popular flow analysis tool. Like Argus, NeTraMet consists of two parts: a collector that gathers statistics via SNMP, and a manager that specifies which flows should be watched. Flows are specified using a simple programming language that define the addresses used on either end, and can include Ethernet, IP, protocol information, or other identifiers. NeTraMet runs on DOS and most UNIX systems, including Linux and BSD.

## Throughput testing

How fast can the network go? What is the actual usable capacity of a particular network link? You can get a very good estimate of your throughput capacity by flooding the link with traffic and measuring how long it takes to transfer the data.


*Figure 6.21: Tools such as this one from SpeedTest.net are pretty, but don't always give you an accurate picture of network performance.*

While there are web pages available that will perform a "speed test" in your browser (such as *http://www.dslreports.com/stest* or *http://speedtest.net/*), these tests are increasingly inaccurate as you get further from the testing source. Even worse, they do not allow you to test the speed of a given link, but only the speed of your link to a particular site on the Internet. Here are a few tools that will allow you to perform throughput testing on your own networks.

## ttcp

*http://ftp.arl.mil/ftp/pub/ttcp/*. Now a standard part of most Unix-like systems, **ttcp** is a simple network performance testing tool. One instance is run on either side of the link you want to test. The first node runs in receive mode, and the other transmits:

```
node_a$ ttcp -r -s

node_b$ ttcp -t -s node_a
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp -> node_a
ttcp-t: socket
ttcp-t: connect
ttcp-t: 16777216 bytes in 249.14 real seconds = 65.76 KB/sec +++
ttcp-t: 2048 I/O calls, msec/call = 124.57, calls/sec = 8.22
ttcp-t: 0.0user 0.2sys 4:09real 0% 0i+0d 0maxrss 0+0pf 7533+0csw
```

After collecting data in one direction, you should reverse the transmit and receive partners to test the link in the other direction. It can test UDP as well as TCP streams, and can alter various TCP parameters and buffer lengths to give the network a good workout. It can even use a user-supplied data stream instead of sending random data. Remember that the speed readout is in kilobytes, not kilobits. Multiply the result by 8 to find the speed in kilobits per second.

The only real disadvantage to ttcp is that it hasn't been developed in years. Fortunately, the code has been released in the public domain and is freely available. Like ping and traceroute, ttcp is found as a standard tool on many systems.

## iperf

*http://dast.nlanr.net/Projects/Iperf/*. Much like ttcp, **iperf** is a commandline tool for estimating the throughput of a network connection. It supports many of the same features as ttcp, but uses a "client" and "server" model instead of a "receive" and "transmit" pair. To run iperf, launch a server on one side and a client on the other:

```
node_a$ iperf -s

node_b$ iperf -c node_a
------------------------------------------------------------
Client connecting to node_a, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  5] local 10.15.6.1 port 1212 connected with 10.15.6.23 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-11.3 sec   768 KBytes   558 Kbits/sec
```

The server side will continue to listen and accept client connections on port 5001 until you hit control-C to kill it. This can make it handy when running multiple test runs from a variety of locations.

The biggest difference between ttcp and iperf is that iperf is under active development, and has many new features (including IPv6 support). This makes it a good choice as a performance tool when building new networks.

### bing

*http://fgouget.free.fr/bing/index-en.shtml*. Rather than flood a connection with data and see how long the transfer takes to complete, **Bing** attempts to estimate the available throughput of a point-to-point connection by analyzing round trip times for various sized ICMP packets. While it is not always as accurate as a flood test, it can provide a good estimate without transmitting a large number of bytes.

Since bing works using standard ICMP echo requests, so it can estimate available bandwidth without the need to run a special client on the other end, and can even attempt to estimate the throughput of links outside your network. Since it uses relatively little bandwidth, bing can give you a rough idea of network performance without running up the charges that a flood test would certainly incur.

## Realtime tools

It is desirable to find out when people are trying to break into your network, or when some part of the network has failed. Because no system administrator can be monitoring a network all the time, there are programs that constantly monitor the status of the network and can send alerts when notable events occur. The following are some open source tools that can help perform this task.

### Snort

***Snort*** (*http://www.snort.org/*) is a packet sniffer and logger which can be used as a lightweight network intrusion detection system. It features rule-based logging and can perform protocol analysis, content searching, and packet matching. It can be used to detect a variety of attacks and probes, such as stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and many other kinds of anomalous traffic patterns. Snort has a real-time alert capability that can notify administrators about problems as they occur with a variety of methods.

Installing and running Snort is not trivial, and depending on the amount of network traffic, will likely require a dedicated monitoring machine with considerable resources. Fortunately, Snort is very well documented and has a strong user community. By implementing a comprehensive Snort rule set, you can identify unexpected behavior that would otherwise mysteriously eat up your Internet bandwidth.

See *http://snort.org/docs/* for an extensive list of installation and configuration resources.

## Apache: mod_security

ModSecurity (*http://www.modsecurity.org/*) is an open source intrusion detection and prevention engine for web applications. This kind of security tool is also known as a ***web application firewall***. ModSecurity increases web application security by protecting web applications from known and unknown attacks. It can be used on its own, or as a module in the Apache web server (*http://www.apache.org/*).

There are several sources for updated mod_security rules that help protect against the latest security exploits. One excellent resource is GotRoot, which maintains a huge and frequently updated repository of rules:

*http://gotroot.com/tiki-index.php?page=mod_security+rules*

Web application security is important in defending against attacks on your web server, which could result in the theft of valuable or personal data, or in the server being used to launch attacks or send spam to other Internet users. As well as being damaging to the Internet as a whole, such intrusions can seriously reduce your available bandwidth.

## Nagios

***Nagios*** (*http://nagios.org/*) is a program that monitors hosts and services on your network, notifying you immediately when problems arise. It can send notifications via email, SMS, or by running a script, and will send notifications to the relevant person or group depending on the nature of the problem. Nagios runs on Linux or BSD, and provides a web interface to show up-to-the-minute system status.

Nagios is extensible, and can monitor the status of virtually any network event. It performs checks by running small scripts at regular intervals, and checks the results against an expected response. This can yield much more sophisticated checks than a simple network probe. For example, ping (**page 185**) may tell you that a machine is up, and nmap may report that a TCP port responds to requests, but Nagios can actually retrieve a web page or make a database request, and verify that the response is not an error.

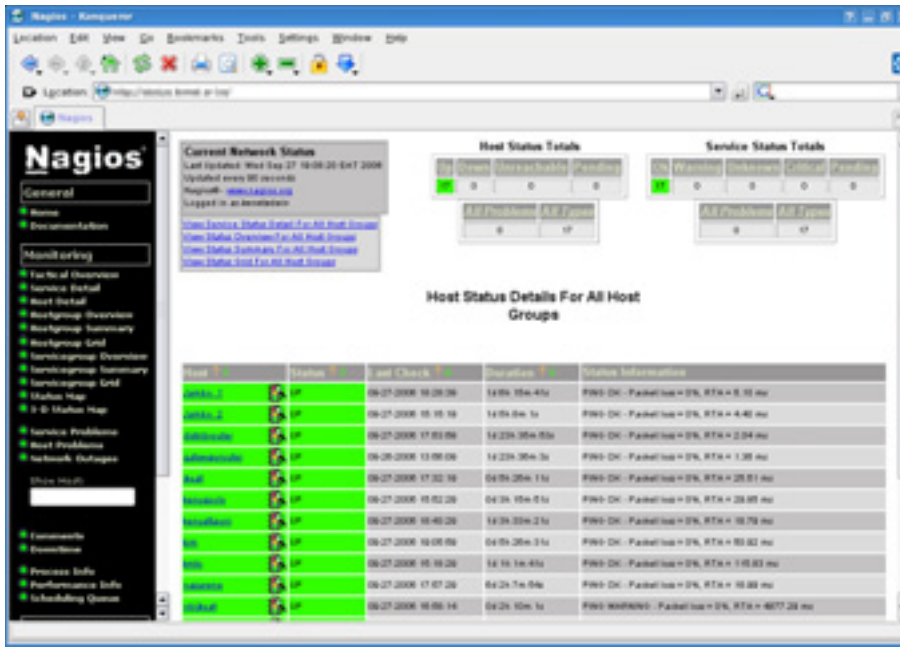


*Figure 6.22: Nagios keeps you informed the moment a network fault or service outage occurs.*

Nagios can even notify you when bandwidth usage, packet loss, machine room temperature, or other network health indicator crosses a particular threshold. This can give you advance warning of network problems, often allowing you to respond to the problem before users have a chance to complain.

### Zabbix

***Zabbix*** (*http://www.zabbix.org/*) is an open source realtime monitoring tool that is something of a hybrid between Cacti and Nagios. It uses a SQL database for data storage, has its own graph rendering package, and performs all of the functions you would expect from a modern realtime monitor (such as SNMP polling and instant notification of error conditions). Zabbix is released under the GNU General Public License.

## Other useful tools

There are thousands of free network monitoring tools that fill very specialized needs. Here are a few of our favorites that don't quite fit into the above categories.

### Driftnet and Etherpeg.

These tools decode graphical data (such as GIF and JPEG files) and display them as a collage. As mentioned earlier, tools such as these are of limited use in

troubleshooting problems, but are very valuable for demonstrating the insecurity of unencrypted protocols. **Etherpeg** is available from *http://www.etherpeg.org/*, and **Driftnet** can be downloaded at *http://www.ex-parrot.com/~chris/driftnet/*.


*Figure 6.23: A web collage generated by Etherpeg.*

## ngrep

***Ngrep*** provides most of GNU grep's pattern matching features, but applies them to network traffic. It currently recognizes IPv4 and IPv6, TCP, UDP, ICMP, IGMP, PPP, SLIP, FDDI, Token Ring, and much more. As it makes extensive use of regular expression matches, it is a tool suited to advanced users or those that have a good knowledge of regular expressions.

But you don't necessarily need to be a regex expert to be able to make basic use of ngrep. For example, to view all packets that contain the string GET (presumably HTTP requests), try this:

```
# ngrep -q GET
```

Pattern matches can be constrained further to match particular protocols, ports, or other criteria using BPF filters. This is the filter language used by common packet sniffing tools, such as tcpdump and snoop. To view GET or POST strings sent to destination port 80, use this command line:

```
# ngrep -q 'GET|POST' port 80
```

By using ngrep creatively, you can detect anything from virus activity to spam email. You can download ngrep at *http://ngrep.sourceforge.net/.*

# What is normal?

If you are looking for a definitive answer as to what your traffic patterns **should** look like, you are going to be disappointed. There is no absolute right answer to this question, but given some work you can determine what is normal for your network. While every environment is different, some of the factors that can influence the appearance of your traffic patterns are:

- The capacity of your Internet connection
- The number of users that have access to the network
- The social policy (byte charging, quotas, honor system, etc.).
- The number, types, and level of services offered
- The health of the network (presence of viruses, excessive broadcasts, routing loops, open email relays, denial of service attacks, etc.).
- The competence of your computer users
- The location and configuration of control structures (firewalls, proxy servers, caches, and so on)

This is not a definitive list, but should give you an idea of how a wide range of factors can affect your bandwidth patterns. With this in mind, let's look at the topic of baselines.

## Establishing a baseline

Since every environment is different, you need to determine for yourself what your traffic patterns look like under normal situations. This is useful because it allows you to identify changes over time, either sudden or gradual. These changes may in turn indicate a problem, or a potential future problem, with your network.

For example, suppose that your network grinds to a halt, and you are not sure of the cause. Fortunately, you have decided to keep a graph of broadcasts as a percentage of the overall network traffic. If this graph shows a sudden increase in the amount of broadcast traffic, it may mean that your network has been infected with a virus. Without an idea of what is "normal" for your network (a baseline), you would not be able to see that the number of broadcasts had increased, only that it was relatively high, which may not indicate a problem.

Baseline graphs and figures are also useful when analyzing the effects of changes made to the network. It is often very useful to experiment with such changes by trying different possible values. Knowing what the baseline looks like will show you whether your changes have improved matters, or made them worse.

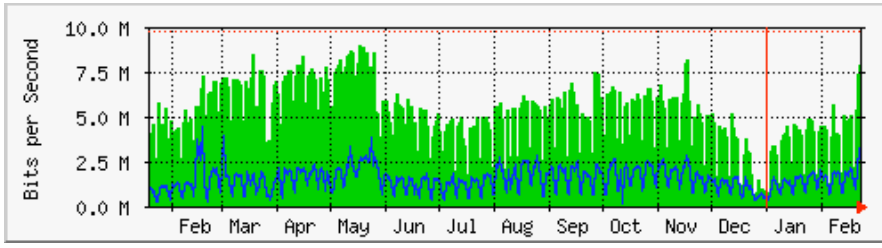


*Figure 6.24: By collecting data over a long period of time, you can predict the growth of your network and make changes before problems develop.*

In **Figure 6.24**, we can see the effect the implementation of delay pools has made on Internet utilization around the period of May. If we did not keep a graph of the line utilization, we would never know what the effect of the change over the long term was. When watching a total traffic graph after making changes, don't assume that just because the graph does not change radically that your efforts were wasted. You might have removed frivolous usage from your line only to have it replaced by genuine legitimate traffic. You could then combine this baseline with others, say the top 100 sites accessed or the average utilization by your top twenty users, to determine if habits have simply changed. As we will see later, MRTG, RRDtool, and Cacti are excellent tools you can use to keep a baseline.

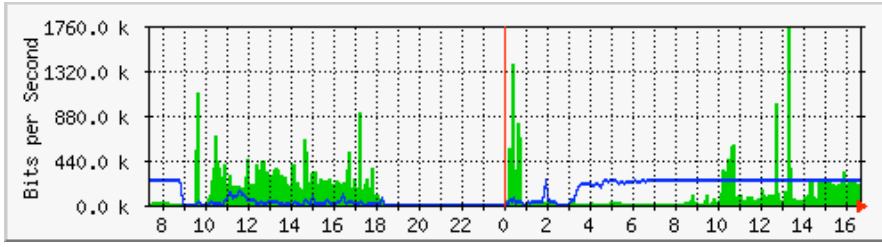


*Figure 6.25: The traffic trend at Aidworld logged over a single day.*

**Figure 6.25** shows traffic on an Aidworld firewall over a period of 24 hours. There is nothing apparently wrong with this graph, but users were complaining about slow Internet access.

**Figure 6.26** shows that the upload bandwidth use (dark area) was higher during working hours on the last day than on previous days. A period of heavy upload usage started every morning at 03:00, and was normally fin-

ished by 09:00, but on the last day it was still running at 16:30. Further investigation revealed a problem with the backup software, which ran at 03:00 every day.

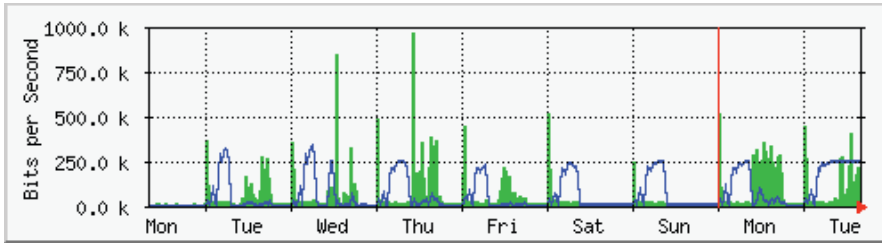


*Figure 6.26: The same network logged over an entire week reveals a problem with backups, which caused unexpected congestion for network users.*

**Figure 6.27** shows measurements of latency on the same connection as measured by a program called SmokePing. The position of the dots shows the average latency, while the gray smoke indicates the distribution of latency (jitter). The color of the dots indicates the number of lost packets. This graph over a period of four hours does not help to identify whether there are any problems on the network.

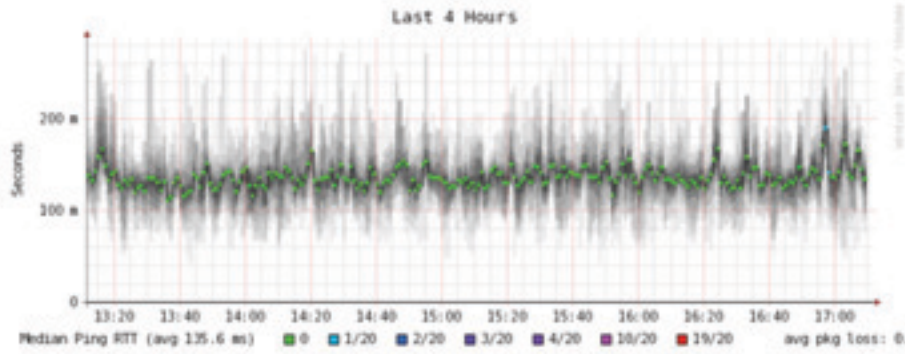


*Figure 6.27: Four hours of jitter and packet loss.*

The next graph (**Figure 6.28**) shows the same data over a period of 16 hours. This indicates that the values in the graph above are close to the normal level (baseline), but that there were significant increases in latency at several times during the early morning, up to 30 times the baseline value. This indicates that additional monitoring should be performed during these early morning periods to establish the cause of the high latency, which is probably heavy traffic of some kind.

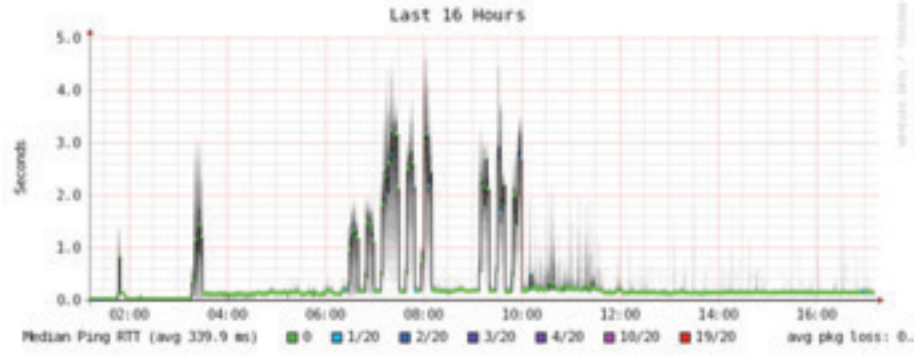


*Figure 6.28: A higher spread of jitter is revealed in the 16 hour log.*

**Figure 6.29** shows that Tuesday was significantly worse than Sunday or Monday for latency, especially during the early morning period. This might indicate that something has changed on the network.


*Figure 6.29: Zooming out to the week long view reveals a definite repetition of increased latency and packet loss in the early morning hours.*

## How do I interpret the traffic graph?

In a basic network flow graph (such as that generated by the network monitor MRTG), the green area indicates ***inbound traffic***, while the blue line indicates ***outbound traffic***. Inbound traffic is traffic that originates from another network (typically the Internet) and is addressed to a computer inside your network. Outbound traffic is traffic that originates from your network, and is addressed to a computer somewhere on the Internet. Depending on what sort of network environment you have, the graph will help you understand how your network is actually being used. For example, monitoring of servers usually reveals larger amounts of outbound traffic as the servers respond to requests (such as sending mail or serving web pages), while monitoring cli-

ent machines might reveal higher amounts of inbound traffic to the machines as they receive data from the servers.

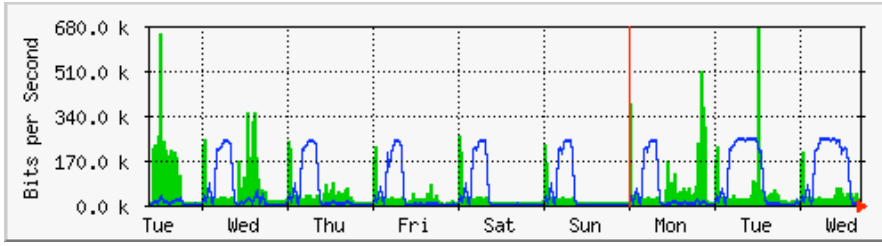


*Figure 6.30: The classic network flow graph. The dark area represents inbound traffic, while the line represents outbound traffic. The repeating arcs of outbound traffic show when the nightly backups have run.*

Traffic patterns will vary with what you are monitoring. A router will normally show more incoming traffic than outgoing traffic as users download data from the Internet. An excess of outbound bandwidth that is not transmitted by your network servers may indicate a peer-to-peer client, unauthorized server, or even a virus on one or more of your clients. There are no set metrics that indicate what outgoing traffic to incoming traffic should look like. It is up to you to establish a baseline to understand what normal network traffic patterns look like on your network.

## Detecting network overload

**Figure 6.31** shows traffic on an overloaded Internet connection.

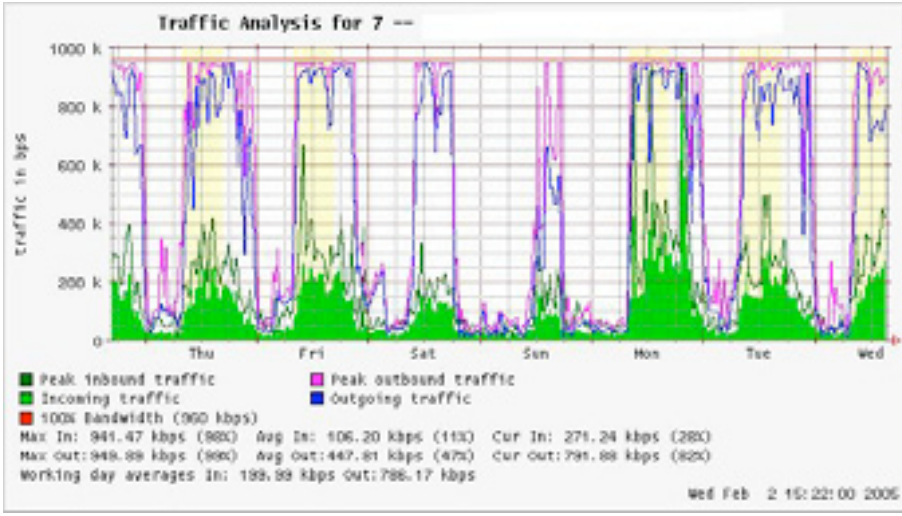


*Figure 6.31: Flat-topped graphs indicate that a line is using the maximum available bandwidth, and is overutilized during these times.*

The most apparent sign of overloading is the flat tops on outbound traffic during the middle of every day. Flat tops may indicate overloading, even if they are well below the maximum theoretical capacity of the link. In this case it may indicate that you are not getting as much bandwidth from your service provider as you expect.

## Measuring 95th percentile

The 95th percentile is a widely used mathematical calculation to evaluate regular and sustained utilization of a network pipe. Its value shows the highest consumption of traffic for a given period. Calculating the 95th percentile means that 95% of the time the usage is below a certain amount, and 5% of the time usage is above that amount. The 95th percentile is a good value to use to show the bandwidth that is actually used at least 95% of the time.

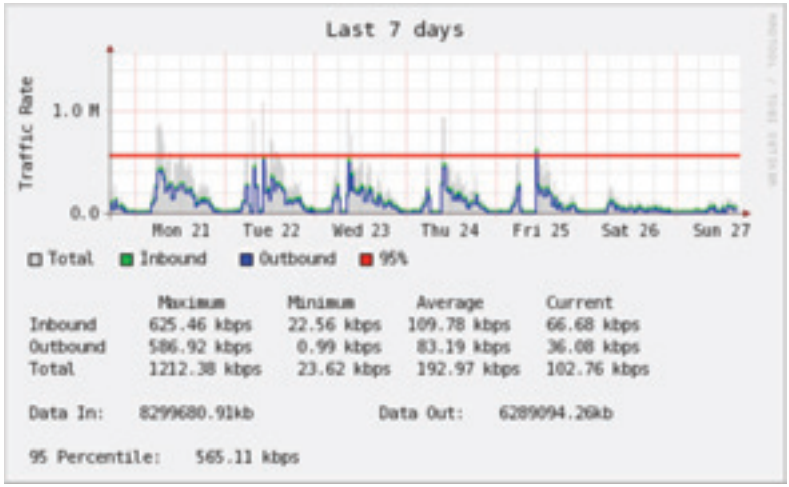


*Figure 6.32: The horizontal line shows the 95th percentile amount.*

MRTG and Cacti will calculate the 95th Percentile for you. This is a sample graph of a 960 kbps connection. The 95th percentile came to 945 kbps after discarding the highest 5% of traffic.

## Monitoring RAM and CPU usage

By definition, servers provide critical services that should always be available. Servers receive and respond to client machine requests, providing access to services that are the whole point of having a network in the first place. Therefore, servers must have sufficient hardware capabilities to accommodate the work load. This means they must have adequate RAM, storage, and processing power to accommodate the number of client requests. Otherwise, the server will take longer to respond, or in the worst case, may be incapable of responding at all. Since hardware resources are finite, it is important to keep

track of how system resources are being used. If a core server (such as a proxy server or email server) is overwhelmed by requests, access times become slow. This is often perceived by users as a network problem.

There are several programs that can be used to monitor resources on a server. The simplest method on a Windows machine is to access the Task Manager using the **Ctrl Alt + Del** keys, and then click on the Performance tab. On a Linux or BSD box, you can type **`top`** in a terminal window. To keep historical logs of such performance, MRTG or RRDtool (on **Page 190**) can also be used.

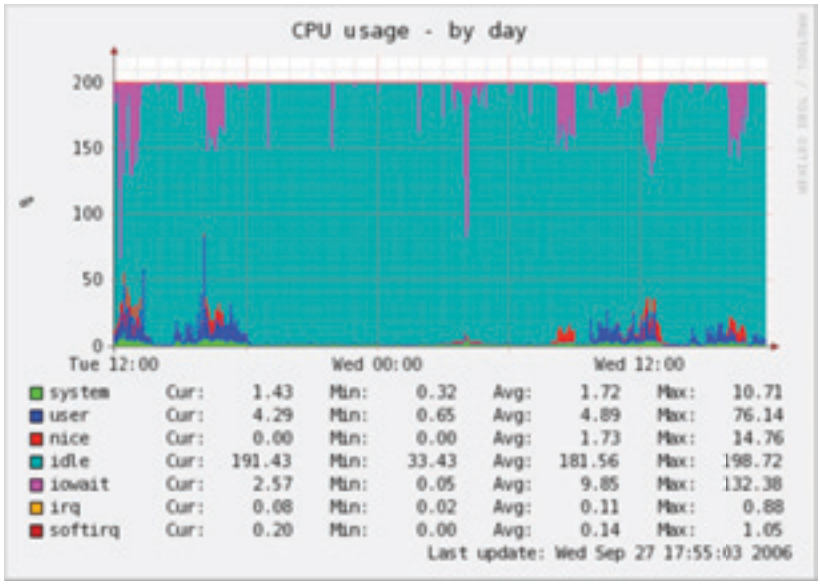


*Figure 6.33: RRDtool can show arbitrary data, such as memory and CPU usage, expressed as an average over time.*

Mail servers require adequate space, as some people may prefer to leave their email messages on the server for long periods of time. The messages can accumulate and fill the hard disk, especially if quotas are not in use. If the disk or partition used for mail storage fills up, the mail server cannot receive mail. If that disk is also used by the system, all kinds of system problems may occur as the operating system runs out of swap space and temporary storage.

File servers need to be monitored, even if they have large disks. Users will find a way to fill any size disk more quickly than you might think. Disk usage can be enforced through the use of quotas, or by simply monitoring usage and telling people when they are using too much. Nagios (see **Page 200**) can notify you when disk usage, CPU utilization, or other system resources cross a critical threshold.

If a machine becomes unresponsive or slow, and measurements show that a system resource is being heavily used, this may be an indication that an upgrade is required. If processor usage constantly exceeds 60% of the total, it may be time to upgrade the processor. Slow speeds could also be as a result of insufficient RAM. Be sure to check the overall usage of CPU, RAM, and disk space before deciding to upgrade a particular component.

A simple way to check whether a machine has insufficient RAM is to look at the hard disk light. When the light is on constantly, it usually means that the machine is constantly swapping large amounts of data to and from the disk. This is known as ***thrashing***, and is extremely bad for performance. It can usually be fixed by investigating which process is using the most RAM, and killing or reconfiguring that process. Failing that, the system needs more RAM.

You should always determine whether it is more cost effective to upgrade an individual component or purchase a whole new machine. Some computers are difficult or impossible to upgrade, and it often costs more to replace individual components than to replace the entire system. Since the availability of parts and systems varies widely around the world, be sure to weigh the cost of parts vs. whole systems, including shipping and taxes, when determining the cost of upgrading.