More through accident than intention, the information returned by *gethostbyname* corresponds quite closely with the status information defined above. Wouldn't it be great if all networking objects where listed there too.

8 Interpreting Queries

monet is concerned with monitoring the status of network objects, thus a definition of status is required. The status information must be general enough to cover all types of network objects by a set of common attributes. In addition to the status at one point we need to keep track of the status history so that back references can be made to particular events and so that statistics over the whole or part of the history can be made.

The raw statistics will need to be processed by monetd to produce results in generic format. In addition these may need to be further processed by clients, for example to produce statistics over a period or to produce *current status* values.

A number of meta-statistics may be generated by monetd such as *behaviour*. Although clients should be able to produce this for themselves from the more basic values, it is very convenient to have this done by monetd.

In addition to instantaneous status values, it would be useful to have *mean* values for a period. This would be basically to save space, as much data for a period would be compressed into single values. It is likely that instantaneous values would be required for the most recent period, with means for periods further back, the size of the mean periods increasing as we go further back. For example keep the instantaneous values for the past day, the mean values for every day in the past month and the means for every day in the past year. Of course an instantaneous values is just a mean value for a very small period.

Keeping history lists could use up memory. In addition to its retention by monetd it may also be cached by the clients. Periodic saving to disc would also be required in a format readable by humans and by monet on startup.

In future we might like monetd to handle all consistency checking for network objects, this would be used by clients such as the xmonet mapper to keep the maps in touch with reality.

8.1 Pragmatics

The nature of network objects varies greatly as do the protocols that can be used to monitor them. So for some object classes it is not possible to get some or any of the specified data. Some of the reasons for this are listed below.

No reliable source You can get a lot of information from the nameserver for IP objects but where do you find out the official name of an X.25 host?

Shakey Definition The capacity of an ethernet is pretty clearly 10Mbps. But what is the ordinal capacity of a sun workstation or an IP address.

Difficult to determine Determining a list of nearest neighbours for an IP host is trivial. Finding the domain name of the machine at the other end of the leased line connected to your local cisco is possible at least in some circumstances but is not simple.

7.3 X.25

The reachability of an X.25 host is done simply by opening a virtual circuit to that host. This means we could only ever send X.25 queries out one at a time because many networks limit the number open at any one time, in addition this caused at least one log file to over flow.

The user data field should be set up for a particular host usually this is very simple but some hosts will refuse a pad connection and have special user data field for monitoring purposes.

Address translation is done using a very simple lookup table that comes with the distribution but each site will need to update this themselves. The user data field for a host can be set here too. This mechanism is not terribly powerful and so may need to be improved later. When this file is altered it will be automatically re-read.

7.4 SNMP

Currently monitoring is done by sending separate requests for the host and each one of its interfaces. A few variables are read such as traffic in and up time from which load and reachability are derived.

The name of an interface is given as <host.name.xy>+<#>interface, so obviously each interface could have more than one name.

It would be nice to query all the interfaces at the one time rather than sending out request for each separately.

Code has been included for listening to Snmp Traps but this has not been tested yet.

7.5 RIPE Database

This is an IP Network database maintained by the European IP coordinating body RIPE, using the *whois protocol* for access to information. It might be nice to have monetd work as a whois server and so merge these sets of information. This would provide such useful information as technical contacts and their addresses.

7.6 X.500

This would be a good source of contact information one day. It could be accessed via scripts or by maybe using the ISODE stuff in here directly. It should be a source of contacts and addresses but it is also possible to draw maps of the with the links representing primary and secondary servers between directory service agents and the parts of the tree they cover.

7.7 Domain Name Server

This is a very good source of information. It provides an accurate source of naming and topology for networks and hosts but is limited because only necessary information is ept up to date.

Geography Sometimes nodes should stay in a particular region.

Zoom Some systems are more important than others and should be drawn larger, with a zoom for smaller stuff.

Shape Graphs where nodes are drawn as single points are the simplest, rectangles are a bit more difficult.

Batch/Incremental With batch layout, all topology is known at once and an optimal layout can be performed. When nodes are added incrementally/ interactively, then it is undesirable if the whole layout changes in front of the users eyes.

Magnetic Heuristic Treat linked nodes as attractors and unlinked nodes as repulsors. From this work out the force each node exerts on all the others and move the nodes accordingly. This is at least n-squared but is quite a good untangle heuristic.

Regional Reduction Algorithm Find the regions in a network. Treat these as nodes in a new network, links represent the places where regions touched. If multiple links between two nodes are reduced to one the number of regions should reduce at each step. Finally we have a heirarchy of subnets that can be drawn quite easily with the minimum number of overlaps.

Finishing As a final measure for an untangled network would be an algorithm to shift nodes so that they are aligned and the links between them are draw straight.

In addition if networks can be reduced to a group of trees or single cycles. A good tree/cycle layout algorithm can be used.

Strength of links Some links can be given priority over each other.

7 Information Sources

7.1 IP

A lot of information can be obtained from the nameserver. Status information is found by using ICMP pings. Currently, an IP address and a host are treated as separate objects but they could be queried together.

7.2 TCP

The system can ping tcp servers using the port number and hostname to define them. The request is done simply by trying to bind to the port. For example, monitoring a nameserver could be done using,

ping = "tcp:&+53"

xmonet would replace *&* by the name of the object to be pinged.

6.2 Behaviour

It is nice for to know if some system is behaving in a strange way, then it can be fixed directly without having to seach for the problem. Monet uses a general behaviour indicator for this purpose, a system can be behaving normally, oddly or chaotically. The difficulty is in deciding what constitutes these behaviours for different types of system. In addition they must be made given only partial information.

If an network object goes up and down three or four times in an hour we might think it odd, more than 10 we could even use the word chaotic. Currently this is how behaviour if defined. The precise reason is writen in the log file as a comment, but the idea is really for a network manager to look at a network map and see something blinking (or whatever) and see if they can find out what is happening.

It would be nice to use other events to change the behaviour parameter, simple (easily implementable) examples would be,

- If the traffic going into a system on an interface is very high and similar to that going out. Here there may be packets bing bounced back and forth.
- Very veriable times in responding to queries.
- One interface overloaded and the others almost empty.
- A system using only a tiny proportion of its supposed capacity.

6.3 Auto-discovery

Getting the network topology is trivial with IP networks, but difficult or impossible for other types of network. It would be possible for SNMP if the names of neighbours were given in the MIB.

There is also the difficulty of determining when data gained from the network overrides that entered by hand.

6.4 Automatic Layout

With computer networks as big and complex as they are becoming it is not really practical to keep them up to date by hand. Assuming the topology is available it would be very helpful to have them laid out in a pleasing manner.

Layout / Untangle Just getting the nodes roughly in position is a big help, which can be finished of by hand.

Links It is pleasing for links to always be either straight or at 45 degrees. It is not desirable for links to be made up of many segments, since it is difficult to keep one line straight a compromise is to use up to two segments. [4]

[4] This is especially useful when the link itself carries information.

The status attributes listed below are a subset of those available for each entity [3]. Some of these can be quite accurately determined, such as the speed of an SNMP interface, but others, such as reachability are more shakey.

Reachability Is the system up or down, or has it been down for a long period of time and thus become *critical.*

Future Is this just a temporary system or about to be taken down. Is is being planned by has not yet been installed or is its future pretty certain.

Behaviour No details, just some unusual events.

Capacity For a network, its average speed. Not really definable for a host.

Official Name monet uses domain names to uniquely identify all objects. Fine for some objects but problematic for others.

Load In Incoming traffic for some recent period expressed as a percentage of its total capacity.

Load Out Same as for Load In but for outgoing traffic.

Number of external Connections

List of Connected Systems This has not really been implemented yet. It is used for the Explore Command described in the chapter of xmonet.

Contact Address E-mail address of someone to contact if there is problem with the network entity.

6 Issues

This section discusses some of the issues, problems and solutions involved in monet. It is far from complete or extensive but should serve as an illustration.

6.1 Reachability

This could be looked at as if the system is up or down. It depends to a large extent on where the monitoring is done from. A system might be up but not reachable. A broken network might be only partially reachable. An overloaded system may not respond to queries.

There is no realiable way to acertain if a system is up or down. Even for protocols with an explicitmeans of querying the status one can never be sure. For protocols that don't some arbitary means must be found.

[3] A full list, with explanation, is available with the command *monetsh> attributes*

Interactive, Automatic and Visual Maps provide an overview of the network, they are also useful for planning and troubleshooting. Interesting situations should be marked clearly and very interesting events should be notified to managers via bleeps or automatic mail.

All the data needed is already there in the network. The amount of data the user has to enter themselves must be minimised. This is especially important for large databases and for consistency.

Uses existing databases, such as the nameserver, snmp agents, whois database for information. This obtained dynamically or via scripts.

Many interesting situations can be analysed easily by software.

Distributed Monitoring and Logging Monitoring is done via local agents which passes on requests to other agents for remote monitoring status. This means that the monitoring information need be gathered only once without doing queries over long distances, and that the same information is available, interactively to anyone who can connect to an agent.

Maps can be stored locally or at aremote site which is accessable by anyone for depositing or retrieving maps. This allows the work of many people to be shared, and to help build a total map of the world.

The very large number of entities monitorable requires that maps be drawn at least semi-automatically [2].

Some logging is done, but is rather primitive. A much more powerful system is planned which would automatically compress logs and allow remote interactive access and automatic production of graphs and reports.

Monet is not designed for monitoring low level traffic, the contents of packets, etc. It is more oriented towards higher level management, although if someone out there has such a package they could be merged.

5 Model

The system consists of a set of communicating agents. These have two tasks, which are to act as database servers making monitoring information available to clients; and secondly to send out queries and cook the results into a set of status attributes.

The database follows a simple object oriented model that allows dynamic definition of new classes and allows individual entities to inherit the attributes of their parent class.

Users can view parts of the network database by defining a *map* which is basically a list of entities with some geometry information.

[2] The very simple drawings on maps are an effort to use the minimum space possible

and is also the surname of the french painter *Claude Monet* which should give a hint at something visual.

3 Rational

Networks are increasing in size and complexity not only due to purchases but also because of the increase in interconnectivity between them. There are quite often many bodies and people involved which makes it very hard to find out what is actually going on. This is further complicated by different protocols, and the many management tools developed without interoperability built in.

SNMP at least provides a common interface but requires proxy agents to work with non IP networks. It has no history mechanism and you cannot start at one agent and explore outwards from it [1]. Lastly it can only deal with physical hosts, you could maybe have a proxy to monitor an ethernet but it cannot describe higher level entities like servers such as mail transfer agents or nameservers.

These two problems, of inconsistent tools and no really common interface for network management are what monet tries to solve.

4 Features and Goals

Really the main design goal was to produce a common framework for all network monitoring. So that managers need write only the low level monitoring code, already having reporting, logging and visualisation tools avalilable.

Generic and Extensible Can be used to draw all sorts of networks, IP, SNMP, X.25, DNS, MX/SMTP, X.500. New classes of network map can be created quite easily by the user without additional code although some programming effort may be put in if they are to be monitored. It is very easy to add monitoring modules for new protocols, or to avoid writing new code, the output from existing monitor programs can be piped directly to the monitoring agent.

Access to the networking database is available from the command line, X or from scripts. In addition an C language API is planned.

The system is designed to be configurable, with as little as possible built in to the code. For example the X interface is almost totally configurable, through its applications defaults file, more so than most other X programs; and tools like the visual traceroute are implemented without building direct support into the system.

Monet is portable, free, public domain software. Written for *unix* in K&R C with *X11 Release 4* and tested on various machines including apollo, sun, gould and a previous version under silicon graphics and ultrix.

[1] A connectivity map could be built into snmp.

Contents

1 Introduction **1**

2 What's in a name? **1**

3 Rational **1**

4 Features and Goals **1**

5 Model **3**

6 Issues **4**
6.1 Reachability 4
6.2 Behaviour 4
6.3 Auto-discovery 4
6.4 Automatic Layout 5

7 Information Sources **6**
7.1 IP 6
7.2 TCP 6
7.3 X.25 6
7.4 SNMP 6
7.5 RIPE Database 7
7.6 X.500 7
7.7 Domain Name Server 7

8 Interpreting Queries **7**
8.1 Pragmatics 8

1 Introduction

Monet is a generic tool for distributed mapping, monitoring and logging of computer networks. Many network management tools are are often specific to one set of protocols or are not designed to work with other software. The basic design goal of monet was to build a tool that could be used with all kinds of network and also allowed interaction with existing tools.

The basic philosophy behind monet is to build a model of the networking universe that keeps itself up to date and produces maps, reports, graphs and warnings without involving people more than necessary.

2 What's in a name?

The name monet required some alternative lunch time conversations as all the obvious names have already been used up (net watch/view/visualise etc). It comes from *Mo(nitor) net(work)*

Guide to Monet

by James Barr

NIKHEF-H
Kruislaan 409
1009DB Amsterdam
The Netherlands

Version 0.2-Beta

20 July 1991