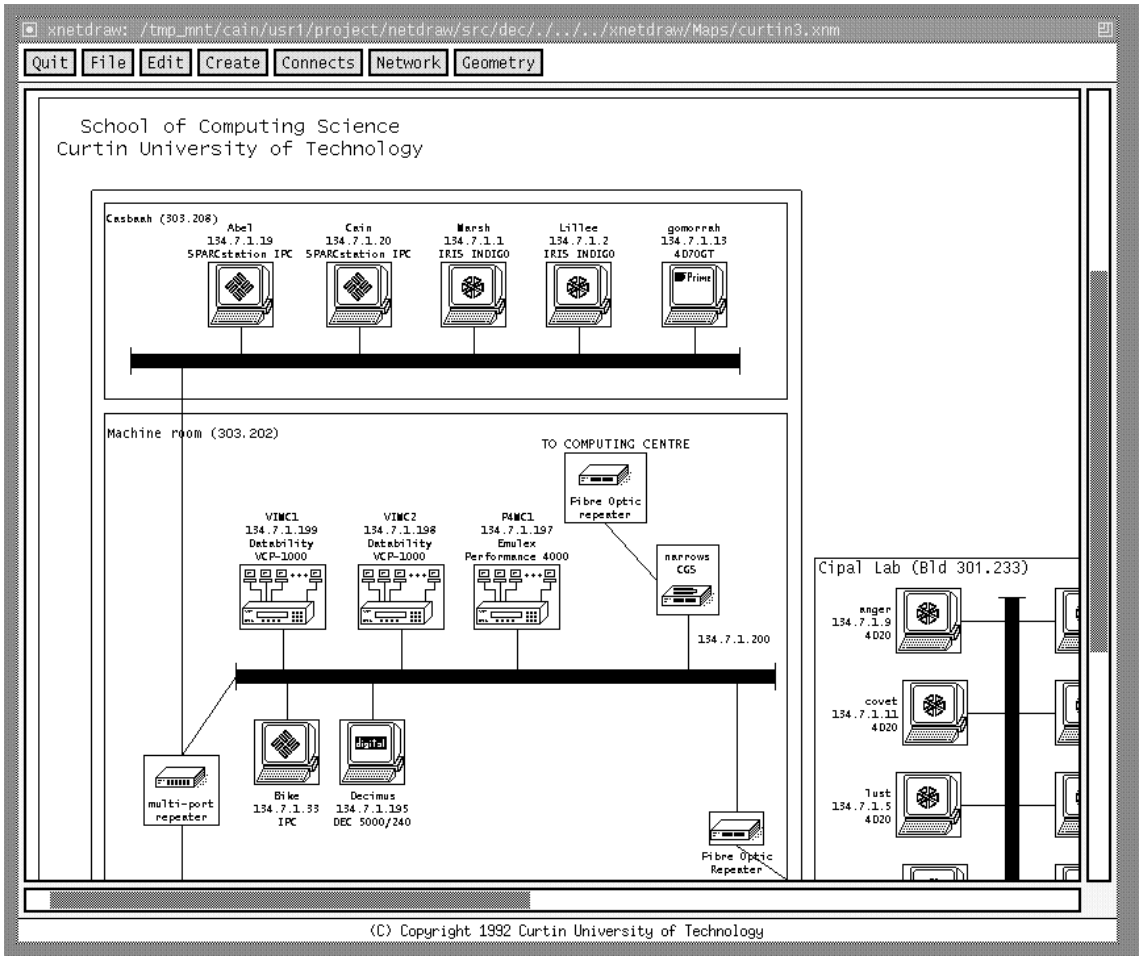
Netdraw v1.2
Computing Project 394



Name: Jarrod Douglas
e-mail: netdraw@cs.curtin.edu.au
Student No. : 900769B

19 February 1994

Supervisor: Craig Farrell.
e-mail: craig@cs.curtin.edu.au

Table of Contents

List of Figures....ii
Abstract....iii
Preface....iv
1.0 Introduction....1
2.0 Background....1
2.1 Network Diagrams....1
2.2 Network Drawing Software....2
3.0 The Existing System....2
3.1 History....2
3.2 Original Aims....3
3.3 Evaluation of Original Project....3
3.4 Development Platform....5
4.0 Aims....5
4.1 General Aim....5
4.2 Specific Objectives....5
4.3 Development Plan....6
5.0 New Enhancements....6
5.1 The Learn Mode Extension....6
5.1.1 Networked Machine Information....6
5.1.2 Network Packet Reading....9
5.1.2.1 Packet Reading Development Process....9
5.1.2.2 The Network Interfaces....9
5.1.3 Learned List Building....10
5.1.4 Diagram Building....10
5.1.4.1 Adding Nodes....10
5.1.4.1.1 Gathering Machine Information....10
5.1.4.1.2 Positioning Nodes....11
5.1.4.1.3 Creating The Nodes....12
5.1.4.2 Adding Backbones....12
5.1.4.2.1 Extending The Backbone Structure....12
5.1.4.2.2 Positioning Backbones....12
5.1.5 Integrating Learn Mode Into Netdraw....13
5.2 Application Resources and Parameters....13
6.0 Future Enhancements....14
7.0 Conclusions....14
8.0 Bibliography....15

List of Figures

Figure 1 : Netdraw's Main Screen 4
Figure 2 : Ethernet Frame Structure 7
Figure 3 : IP Packet Structure 7
Figure 4 : Bridge Unit Example 8

Abstract

This report presents an outline of the extensions made to an interactive network drawing package, Netdraw. Netdraw was developed originally in 1992 at Curtin University as part of the *netman* project.

Netdraw currently provides a purpose specific system for drawing network diagrams on both Sun SparcStation and a DECstation 5000 series platforms. This report will discuss the changes made to the existing Netdraw system to automate more of the drawing process by gathering information from the network itself.

Preface

I would like to thank all the people who helped me through this project from moral support, technical support to administrative support.

Firstly, I would like to thank my supervisor, Craig Farrell for his constant supply of verbal motivation and for giving me the opportunity to work on a project involving such a wide range of computing knowledge and skills.

Also I would like to thank Mike Schulze, for all his excellent technical help and pieces of really useful code.

I would also especially like to thank Associate Professor Steve Kessell for his administrative help in giving me a bit more time to really get stuck into the work. Without his help this project would have cost me so much more personal trauma.

Thanks should also go to Brad Allen and Peter Musca for all the work they put into the original project, giving me a good base to work from in extending the system.

Still more thanks and other comments go out to my family, Darren Brown, Steve Lewis and all the other people who provided the moral "haven't-you-finished-that-project-yet" support throughout the project's duration.

1.0 Introduction

Local area networks are becoming common throughout industry and commerce at a surprising rate. Shared bus-based communication networks like ethernet are becoming more and more popular. Network administrators are responsible for maintaining these ever expanding computer systems, at both the technical and managerial levels. They are responsible for not only the performance, integrity and technical details of the network but often responsible for the management of network financing and budgeting. To perform well at these many and varied tasks, network administrators need good tools to assist in analysing, maintaining and reporting network performance. Many differing types of tools such as protocol analysers, network traffic density monitors, packet tracing monitors, and more, are required to administer all the different parts of a network. The Netman project aims at developing a set of useful network administrative tools using the X windowing interface, for monitoring and administering ethernet and TCP/IP protocol based systems.

Netdraw is one of the tools being developed in the Netman project. It plans to provide network administrators with a quick and efficient means of producing quality network diagrams. The network diagrams produced reflect the logical and/or the physical connections between networked computers.

Development of this software package was started in late 1992 as a third year project of two Computing Science students at Curtin University of Technology. Section 3.0 of this report looks at the system produced by this original project. A year later, Netdraw was extended, as the aim of this project, to include network 'awareness' and thus automate even more of the diagram production process. The aims of this project are defined in Section 4.0 and section 5.0 discusses the features added to that system during the project. Some further potential enhancements to Netdraw are outlined in section 6.0. Sample output from the new system running learn mode routines is in Appendix A. A diagram produced using the original system, before modification, can be found in Appendix B.

2.0 Background

2.1 Network Diagrams

It is well known that a diagram can represent large amounts of data or complex information in a way that is more easily understood by the human mind. A diagram of a network gives a more abstract view of the physical network connections. This abstract view removes the need for technical knowledge or knowledge of network terminology in order to understand the network topology.

A network administrator not only has to control the technical aspects of the network, but also has to make financial request and reports to justify the effectiveness of the system to higher levels of management. Managers at these higher levels generally have

little or no technical background and so a diagram of a network's structure is not only a valuable reference for the network administrator but also a useful guide to higher level management.

Networks are very dynamic creatures. New machines are frequently added to a network, old ones removed and existing machines relocated to another part of the network. Due to this flexible nature of network structure, a network diagram that is correct today, may need many changes made to it tomorrow. It is important then, that the technique used to draw the network diagrams needs to be similarly flexible, so that the changes in the network can be reflected in the diagram quickly and efficiently. The best way to achieve this is with an efficient, tailor made software package.

2.2 Network Drawing Software

Until recently, general purpose drawing packages have been the only way to draw good quality network diagrams, in an automated way. Unfortunately, using a general graphics software package to draw such dynamic diagrams would be very time consuming. Small changes in the network layout could mean extensive changes to a single diagram. For example shifting one machine on the diagram could mean having to reposition other machines, their related text labels and redrawing the connections between machines and network backbones to keep the diagram tidy; all of which may have to be done line by line.

Netdraw provides a mechanism for drawing networks as a series of machine icons connected by lines. A suite of icons are provided to represent many different types of network devices. The package itself also maintains the links between devices, in a tidy manner, no matter where they are positioned in the diagram. This allows network diagrams to be produced and changes made to existing diagrams much quicker and easier.

3.0 The Existing System

3.1 History

Netdraw is a subsection of the Network Management (Netman) project that is being undertaken by Craig Farrell and Mike Schulze for the School of Computing Science at Curtin University. Craig Farrell is the Network Manager for the School of Computing Science. He found that when asked to produce a diagram of the current network at Curtin, he had no specialised application to do it. He also found there was no acceptable application dedicated to producing such a diagram available. This gave birth to the idea of Netdraw (Musca & Allen 1992).

Netdraw was originally undertaken, as a major third year project, by Brad Allen and Peter Musca, two third year Computing Science students at Curtin University of Technology. Outlined briefly below are the original goals and achievements of their project.

3.2 Original Aims

The objective of the original project was to develop a digital software application with the intent of producing graphical representations of ethernet and fibre optic computer networks. This application must provide an easy to use, interactive environment for the modification of those diagrams, as well as being able to produce the images in postscript format.

3.3 Evaluation of Original Project

As a result of the original project, Netdraw already provides the user with a basic graphics package for drawing network diagrams. It has a set of predefined icons for various types of hardware and computer manufacturers. Also a fully interactive user interface was developed using the X windows system. Figure 1, below is a snapshot of the current Netdraw systems's main screen on the DECStation version, also containing a working diagram.

The system allows for the creation of a number of network items and their many attributes ; nodes - representing connected machines, backbones - representing the shared communications line, lines and rectangles for adding clarity to the diagram (such as for showing physical separations or groupings of machines) and also text to add labelling and extra information to the diagram. The interface also provides a mechanism for later changing the attributes of any drawing item.

The original system was quite well developed on the Sun SparcStation platform, giving a polished almost commercial appearance. However, the DECStation implementation had a number of small bugs and performance deficiencies, some of which were later corrected.

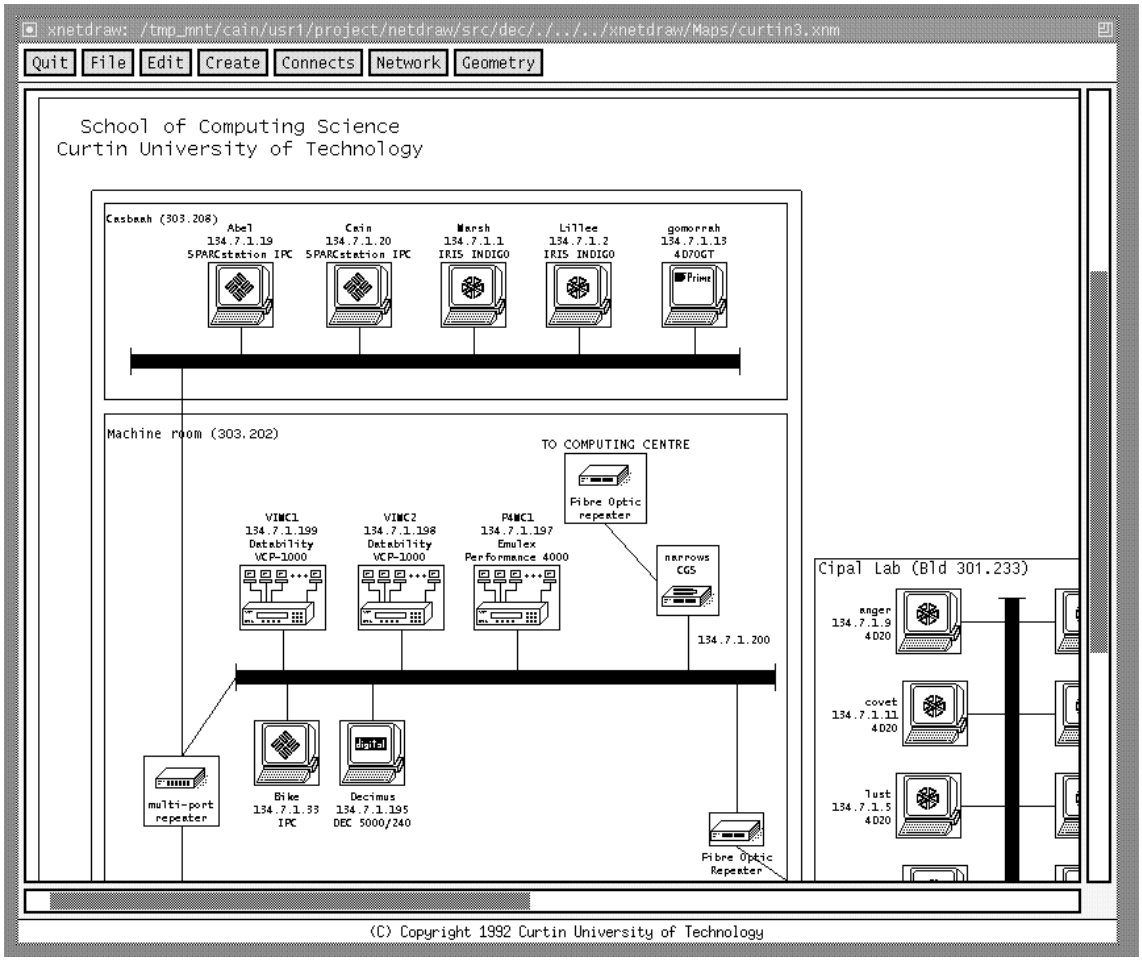


Figure 1 : Netdraw's Main Screen

Some of the minor problems that existed with the original system are :

The DECStation version was not fully working as there were a number of small syntactic errors in the code to prevent compilation.

The whole system takes a long time to load and has a very poor response time to most of the functions it performs.

The menu option to create a new network node does not work, it accepts the new node's details but does not place a new node in the diagram.

All newly created items are placed in the centre of the active diagram. This can cause complications of the user selecting the wrong item when there is another item already located in that position. So a number of nodes created in succession without re-locating them in between makes it very difficult to pick the desired node from the bunch.

The tool used for selecting files (for saving and retrieving diagrams) is very slow and does not show the user the full path of the file system being searched.

Most of these minor performance problems don't dramatically affect the function of the program, but they do reduce its overall appearance. Most of these problems also only appear in the DECStation implementation, whereas the Sun version seems to be quite more developed and bug-free. Despite these minor problems, the original Netdraw project achieved most of what was specified the formal objectives and provided a good set of functions for easy program extensibility.

3.4 Development Platform

The development of Netdraw occurred for both DECStation 5000 series machines running Ultrix 4.2a and Sun SparcStation architecture machines using SunOS 4.1.x. Netdraw requires at least X11 Release 4 libraries and protocols.

In order to perform network learning functions, Netdraw requires direct interface with an IP protocol based ethernet network.

4.0 Aims

4.1 General Aim

The aim of this project is primarily to extend the existing Netdraw system, preferably on both Sun and DEC platforms, so that it may make use of ethernet traffic to 'learn' many of the network connections and layout automatically. The 'learn' mode is intended to draw much of the network diagram without any user input. It was decided for this project, to implement this learn mode in a passive monitoring way. This means that only machines that are sending or receiving packets on the network will be discovered. An alternative to this is active discovery, which would involve sending packets to each possible network address to see if a machine exists there. It was decided that active discovery would be detrimental to the performance of the network and so passive discovery should be employed.

Netdraw should perform this passive learn mode by analysing the address contents of the ethernet and IP protocol packets being broadcast on the network. For each unique network address in a packet is a unique machine somewhere on the network. If a machine is attached to the same physical ethernet wire as the machine that is running Netdraw, then the manufacturer of the ethernet card (and often also the machine) can be determined and a company logo displayed as part of the drawing.

4.2 Specific Objectives

The specific goals intended for this project are :

- Produce a system to analyse network packet contents and determine the locality of the originator and recipient.

- Interface the system produced above to the current Netdraw system, to create

the 'learn' mode, allowing Netdraw to create much of the diagram with minimal user input. (This should be set up as the default startup mode of Netdraw).

• Support the ability to take an earlier produced diagram and update it using the 'learn' system.

• Extend the graphics system so that the user can place text anywhere within the diagram (this was an unsatisfied objective of the previous development work).

• Bring the whole Netdraw system as close as possible to a commercially viable product standard.

• Write sufficient user and reference documentation for this size system. Possibly including a Unix man page (or Xman page for X-windows) for basic synopsis and usage.

4.3 Development Plan

The main objective is to get the learn mode operating successfully. This requires that the network reading routines be developed first and then integrated into the existing package. A system to keep a list of networked machines already 'found' needs to be developed next, so that each machine only appears once on the diagram. The list of found machines then needs to be translated into nodes and connections as used by the existing system. The existing Netdraw will need menu modifications to allow user control over the network learning procedures. Once the learn mode is working then time permitting, the other objectives can be done in any order, as these are of lesser importance.

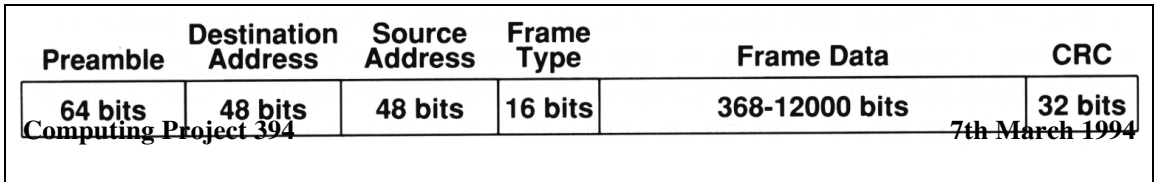
5.0 New Enhancements

5.1 The Learn Mode Extension

5.1.1 Networked Machine Information

In order to identify machines uniquely, IP addresses need to be collected. From an IP packet a source and a destination address can be extracted, so the two machines involved in communications can be potentially added to a list of 'discovered' machines on the network. Using the IP address combined with the ethernet address can yield useful information such as manufacturer name, possibly device type and machine hostname. This information can be added to the network diagram to enhance both it's meaning aesthetic value.

Below is a diagram representing the structure of an ethernet frame (Comer 1991).



From this frame can be determined the ethernet address (and thus the manufacturer name) of the source and destination machine (on that ethernet wire) and the type of higher protocol used in the frame (eg. IP). It is possible to obtain the machine's symbolic name from the ethernet address, using standard system files (ie. /etc/ethers) and tables.

A much more reliable and efficient way of obtaining this information is from the use of IP addresses. If the frame contains an IP packet we can extract the source and destination IP addresses for that packet also. This gives us a unique identification tag for each machine on the diagram and in the world. Below is a diagram representing the structure of an IP packet (Comer 1991).

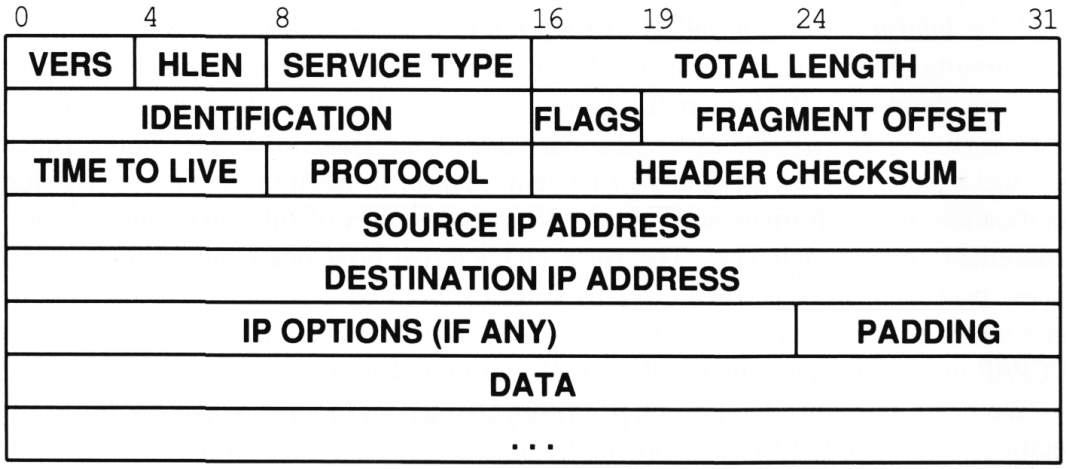


Figure 3 : IP Packet Structure

With just the ethernet and IP addresses of source and destination machines (and frame type number) we can determine enough information to draw a quite detailed diagram - providing we know the network domain within which the program is working.

Ethernet addresses are only addresses to network interfaces not machines, this means a device like a bridge only has one interface to the ethernet wire we are listening to and hence only one ethernet address (that we can read). From this address we can determine the manufacturer of the bridge device's ethernet card but because the function of a bridge is to pass on packets from other networks, the bridge itself will appear to have many IP addresses, or conversely many different machines from other networks will appear to have the same ethernet address. For example, the following diagram shows how two ethernet networks may be 'bridged' together.

In this example machine 'X' is running the packet reading program. A packet going

from machine 'J' to machine 'K' passes through the bridging device 'B'. Thus when X reads the packet, the destination ethernet and IP addresses are (for 'J') 08:09:0A:0B:0C:0D and 101.6.2.1 respectively, the source ethernet and IP addresses are (from 'K') 01:02:03:04:05:06 and 101.6.1.1.

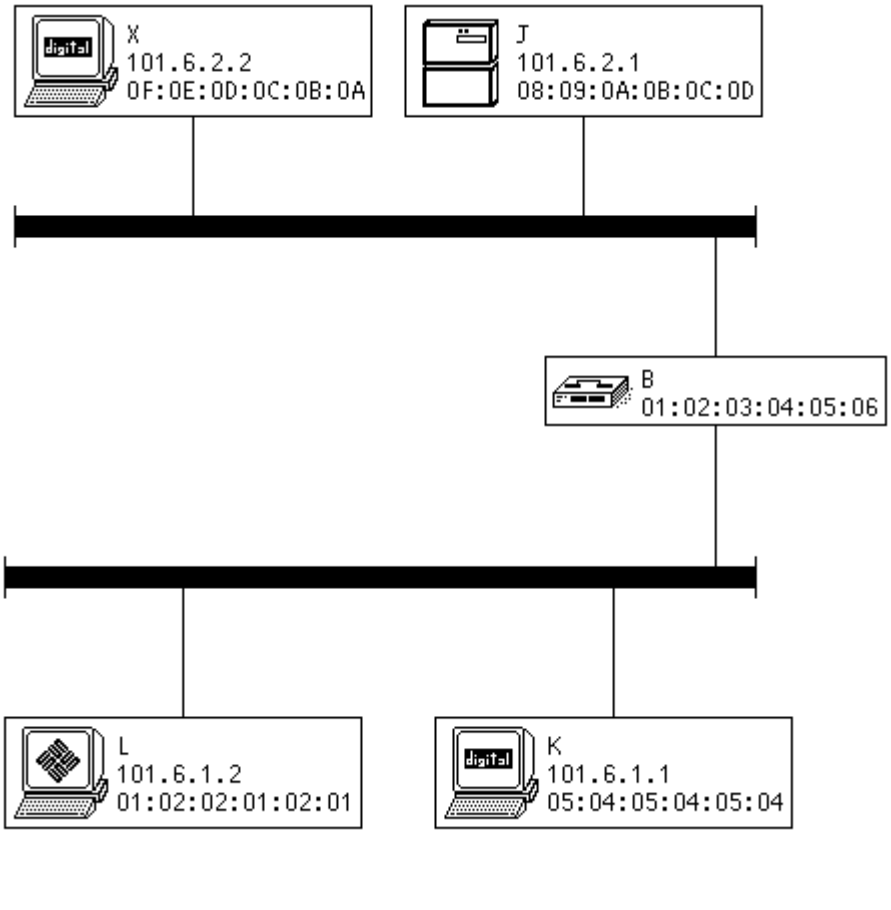


Figure 4 : Bridge Unit Example

Similarly, a packet from machine 'L' to machine 'J' has the source ethernet address of 01:02:03:04:05:06 because it is coming from bridging unit 'B' also. It is obvious from this example that the source ethernet address for all IP packets coming from anywhere beyond 'X's network will be that of 'B's address. For this reason it is necessary to identify individual machines by their IP addresses rather than ethernet addresses. This also shows that it is not possible to obtain an ethernet address for a machine that is not on the local network and therefore, we cannot determine the manufacturer names of those machines. Because all their ethernet addresses will come through as the same as the bridging or routing device's then the program will give them all the same manufacturer name and machine icon on the diagram.

5.1.2 Network Packet Reading

5.1.2.1 Packet Reading Development Process

At the start of this project I worked with code supplied by Mike Schulze to read packets from the network. After studying this code for a couple of weeks, playing with bits and recompiling several times, I managed to get a burst of header details from each packet travelling on the network, printed out onto the screen. Much of the code given was concerned with time stamp information, that I did not require, so further work was required to remove the unnecessary information. When I had finally produced a working packet reading program, I was informed by my supervisor that although this code worked, it was optimised specifically for DECStations machines and was therefore virtually useless for my purposes. However, I had at least gained more knowledge with how packet reading programs are set up and I had developed a better feel for what sort of information I could get from ethernet packets.

After a minor setback, I acquired more portable packet reading routines from the Etherman (Schulze 1993) source code that differed only in minor ways to the code I had previously been working with. This code also included routines for reading from Sun SparcStation machines. Once again I got to work on the packet reading program. When I had this working on the DECStation, I made a few adjustments to also get the code working on the Sun platform. Once the program worked on both machine types, I had to think about how to integrate this isolated program into the existing Netdraw.

5.1.2.2 The Network Interfaces

The network interface used by Netdraw in order to use learn mode requires promiscuous access to all network traffic. This means that any ethernet frame on the network can be read even if it is not addressed to the machine doing the learning. Normally a packet can only be read by the machine it's addressed to or if the packet is a broadcast packet, all machines can read it off the network. Promiscuous network access is usually only provided to system super users, because of the obvious security implications.

Netdraw uses two network interfaces one for each development platform environment. The devices used are ; for the DECStation 5000 / Ultrix 4.2a the Ultrix Packet filter and for the SUN SparcStation / SunOS 4.1.x the SunOS Network Interface Tap.

The Ultrix Packet filter allows for batch reads, packet filtering and promiscuous operation. The super user can configure the mode of operation of the packet filter device, allowing non super users access to promiscuous packet reading.

The SunOS Network Interface Tap (NIT) only supports promiscuous packet reading for the super user.

5.1.3 Learned List Building

As packets are being read off the network, a list needs to be built up so that a quick check can be made to determine if that machine has already be 'seen'. Once a machine has been found, the list is scanned sequentially for its IP number or its ethernet. If it is

not an IP packet, then the list can be scanned for its ethernet address, if not found then it can be added to the diagram and to the list - I have given non-IP packets an IP number of 0.0.0.0, just to differentiate them for now. If however, the packet is IP and its IP number is found on the list, then no further processing is required. If it isn't on the list then its IP number and ethernet address are added to the list and the machine is added to the diagram.

As the learned list is built, memory is dynamically allocated to store the new structure. Although there are more efficient methods of performing this (dynamic memory allocation is time & CPU expensive) the data structure of a machine record is very small (11 bytes long) and this is only done when an undiscovered machine is added. Once the diagram starts to develop, most machines that are active on the network have already been found and so no new memory needs to be allocated.

When the learn mode is turned off, if the information is not required again, then the memory allocated for the list should be disposed of. During the integration of these routines to the main program I chose to dispose of this list information for reasons discussed later.

Netdraw already contained a number of lists that it maintained in order to keep track of what machines are on the diagram and which ones are being selected for moving etc. I chose to implement a separate list for the learned list information, primarily for ease of programming. Although an alternative approach I had was to extend the structure of the existing 'node' record to include ethernet addresses and use the existing lists to scan during the learn mode process. This method may be a better way for integration purposes but it would be less efficient to have to search three lists instead of just one each time.

5.1.4 Diagram Building

5.1.4.1 Adding Nodes

As the learned list is built so the diagram on screen is updated accordingly. After much searching I discovered routines already built into the existing Netdraw system to add nodes to the current diagram, along with routines for changing the attributes associated with each node. Similar functions also existed for backbone creation and manipulating.

5.1.4.1.1 Gathering Machine Information

Once the ethernet and IP addresses have been collected for a node, certain functions are available to add more details to it. The information placed in the nodes by the new learn mode system is; the IP address, the host name and the manufacturer name. From the manufacturer name and machine type, the existing system will also add an appropriate company logo and device image icon if possible.

The IP and ethernet addresses are collected directly from the packet reading routines. The ethernet address is not stored in the node itself but merely used as a temporary tool

to obtain manufacturer information.

Netdraw uses a call to the library routine *gethostbyaddr* to find the symbolic name address of a host from its IP number. This uses the Internets Domain Name System (DNS) to resolve the map between IP address and symbolic name (this however requires the TCP and UDP protocols also). This name can also be obtained from a standard system file /etc/ethers, which contains a list of ethernet address / symbolic machine name pairs.

Every ethernet card has a unique address, built into its hardware. This address, has two parts ; the manufacturer's identification code and a card identification code (both unique codes). In order to determine the manufacturer name from the ethernet address, a section of source code from the Etherman application was used. The functions used build and access a reference table in memory, from a file of manufacturer names and codes.

Some manufacturers only make one type of network device and some manufacturers have more than one code that they use for different devices. This means the device type (eg. router, workstation etc) can sometimes be determined from the manufacturer's code.

Already built into the existing Netdraw system are some routines that automatically check the manufacturer name and device type for a node on the diagram and determine an appropriate icon. So once the manufacturer name is discovered and added, the existing system adds an icon for displaying on the diagram. If the system has no icon specific to that manufacturer it uses a generic default icon for that device type.

5.1.4.1.2 Positioning Nodes

In the existing system, when a node is created it is placed in the centre of the visible diagram. It is then left to the user to relocate that node to its position in the diagram. With the new learn mode, some attempt has to be made to organise the nodes in an automated way, so that the end result is at least tidy and useful to the user. If all discovered nodes were placed in the centre, then the job of re-arranging them all to actually make a useable diagram, would outweigh the time and effort saved by having the learn mode create them in the first place.

Originally I had planned to calculate the position of the next node after placing the currently discovered node. I had decided to place the nodes from left to right, top to bottom in the order discovered across the canvas. After extending the work further to include the backbones and connections I re-worked this idea, so that nodes were placed relative to the network backbones they were connected to. The nodes are still placed initially according to my original scheme but are then moved according to the new scheme, when they are connected up. The new positioning scheme is discussed below in section 5.1.4.2.2.

5.1.4.1.3 Creating The Nodes

Nodes are created using calls provided by the existing Netdraw system. In order to be useful, nodes must be upon initialisation, immediately added to one of Netdraw's internal lists. The discovered IP address and determined manufacturer name is stored in the node. Then the current diagram is searched for the backbone that the new node is attached to (if it exists). Once a backbone is found, existing functions are used to connect the node to the backbone. Then the node becomes part of the current diagram.

5.1.4.2 Adding Backbones

When a new node is discovered, we need to know what network it came from and which backbone item to connect it to on the diagram. If the required backbone doesn't exist, then it must be created and neatly positioned within the current diagram.

5.1.4.2.1 Extending The Backbone Structure

To enable newly discovered nodes to be connected correctly, an extra field was added to the backbone structure definition. This field is a subnet mask, which identifies what IP addresses are to be connected to that backbone segment. The mask can be compared to a new IP address's subnet part, if there is a match then it can be connected to that backbone, otherwise another backbone must be found or created.

5.1.4.2.2 Positioning Backbones

When a new backbone is created, its position is assigned from globally maintained data. Each backbone starts "growing" from a certain distance from left-hand side of the diagram, in a horizontal direction. Each new backbone is positioned a set distance on the diagram below the last placed backbone, this last position is stored in the programs global data. The distances set here are in no way intelligent, so if the nodes connected to the above backbone are extraordinarily large then they may conflict with the new backbone's nodes. The objective is mainly to get an accurate diagram on the screen as quickly as possible and let the user sort out the aesthetic problems.

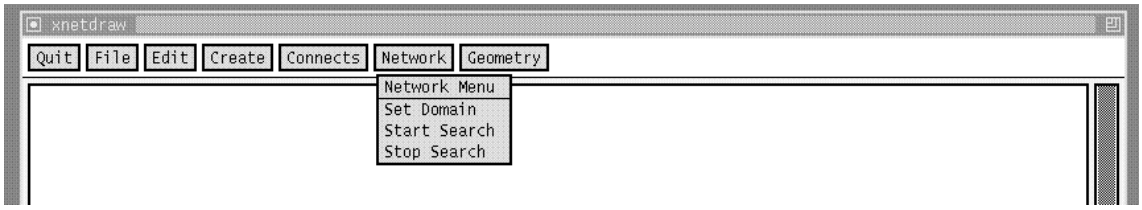
Once a new backbone is placed on the screen, the node attached to that backbone is repositioned to the top left position a set distance from the backbone. The backbone structure itself stores information on where the next node is to be placed along the backbone. As each node is added to the backbone, this information is updated (the width of the new node is added along with a standard between node gap space). The backbone also keeps track of how many nodes are attached to it, so that the node's positions can alternate between being above then below the backbone. This gives a more balanced, aesthetically pleasing diagram of the network.

Unfortunately, there are diagram size restrictions on the existing system. Instead of expanding to fit the diagram, Netdraw has a fixed size canvas. In combination with the new learn mode procedure, this means that a backbone with a large number of active machines may grow off the end of the accessible diagram.

5.1.5 Integrating Learn Mode Into Netdraw

The learn mode routines were quite easily interfaced with the existing Netdraw system by use of an X work-procedure. X windows programs generally execute a two step loop ; gather events, process events. X windows programming also allows regular non-event initiated routines called work procedures. Work procedures are executed after the processing of each registered event in the windowing system. The whole of the learn mode system is placed in one of these work procedures. When the learn mode is activated, the work procedure performs the grabbing a single packet and adding the source and destination nodes to the diagram. The learn mode is activated and de-activated through use of a simple boolean switch and selectable menu items to control it.

I have extended the menu interface on the existing Netdraw system. The 'Network' menu now offers items to start and stop the net work learning mode. Below is a diagram showing the new Network menu items (popped out).



In order to extend the menu interface, I was required to learn about how X programs worked in general and how actions and events were handled by the existing Netdraw system interface.

Changes also had to be made to the mainline perform the initialisation of the network interface device and setup the manufacturer code table in memory.

5.2 Application Resources and Parameters

I have added some new application resources and command line parameters to the existing system. These are for the use of the new learn mode routines. The *manfCodes* application resource, allows the user to specify the path and file name for the file that contains the list of manufacturer names and codes. The contents of this file are loaded for use in determining the manufacturer name of all discovered nodes. The application resource *deviceType* is used as the setting for the name of the network interface adaptor. If this is not set, then the program has some environment specific defaults it uses. The *deviceType* can also be set at the command line by *Xnetdraw -I deviceType*.

6.0 Future Enhancements

There are many changes and modifications that could be made to improve the interface, performance and general usefulness of Netdraw. I will attempt to highlight what I

consider areas of Netdraw that will need more work done on them.

The work by this project on the Sun SparcStation platform, only went as far as setting up the packet reading and the menu buttons, the rest of the project needs to be worked out for this environment.

A major drawback to the new system is the fixed canvas size. Changing the size of the canvas would require a lot of modifications to the existing code and particularly to some of the file saving functions, but the static size of the canvas limits the usefulness of Netdraw.

One of the unsatisfied aims of the original project was to load in an old diagram and continue to update it using the learn mode. This would require either saving learned list information with a diagram or finding some way of learning from the diagram itself to build the learned list. This would mean old diagrams created without the use of learn mode, may be able to be updated also.

The 'Set Domain' item in the Network menu currently does nothing. This should be activated so that the user can input the specification for the local domain, whereas at the moment this is hardwired into the program.

From the old system, the 'Undo' menu item needs to be implemented, currently it does nothing, and an 'undo' function is a handy thing to have for drawing editing. The old systems file selection tool is also very slow in response. I would suggest replacing or enhancing the existing widget with a known more efficient one.

One of the more important enhancement that could be made is the production of some useful support material, such as user's manuals, Unix man pages and programmer manuals.

7.0 Conclusions

Unfortunately, due to time constraints, some of the original minor objectives were not completed. The ability to update an existing network diagram was not implemented because of the large amounts of re-designing that would be needed.

Allowing the user to place text anywhere within the diagram, although one of the goals originally set as an objective, was discovered later to be already implemented and working. Contrary to the information given by the report of the previous project work.

The system is not yet at a state that is commercially viable as work needs to be done to remove some of the bugs and tidy up the interface. Complete user and reference documentation still needs to be written for the system, although until Netdraw is more developed and polished, this would probably be a waste of time.

The main objective of this project however, was to get the learn mode routine working correctly. For the DECStation development platform, this has been achieved with

reasonable success. The development on the Sun platform was halted at an early stage, due to the security implications of allowing root access to the application. The network learning mode was successfully integrated into the existing Netdraw package, to produce the desired 'network aware' drawing package.

As an extra result, from this project I also gained valuable experience working with large applications and in X windows programming environments. I have also learned the value of good time management when undertaking projects of this size.

8.0 Bibliography

Black, U.D. (1992) TCP/IP and Related Protocols. McGraw Hill: New York.

Comer, D.E. (1991) Internetworking with TCP/IP ; Principles , Protocols and Architecture. Prentice-Hall: London.

Jacobson, V. (1988) *Traceroute* source code. University of California. E-mail van@helios.ee.lbl.gov

Musca, P. and B. Allen (1992) *Xnetdraw v1.0* source code and report. Department of Computing Science, Curtin University of Technology: Perth, Western Australia.

Nye, A. and T. O'Reilly (1992) X Toolkit Intrinsics Programming Manual for X Version 11. O'Reilly & Associates Inc.: U.S.A

Schulze, M. (1993) *Etherman v1.1* source code. Department of Computing Science, Curtin University of Technology: Perth, Western Australia.

Schulze, M., G. Benko and C. Farrell (1993) Hombrew Network Monitoring: A Prelude to Network Management. Technical Report No. 9-1993, Department of Computing Science, Curtin University of Technology: Perth, Western Australia.

Appendix A

Network Diagrams From Netdraw V1.2*

* Produced using the Netlearn mode

Appendix B

Network Diagrams From Netdraw v1.0*

* Produced manually from the original system.