THE INTERNET EXTENDER

BY

JASON STEVEN SARICH

DESIGN PROJECT REPORT

Submitted in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in
Computer Engineering
In the School of Engineering
Santa Clara University, 1998

Contents

Appendices

ABSTRACT

"The Internet Extender"

by

Jason Steven Sarich

At this time, single users and small businesses connect to the Internet by either having multiple inexpensive low-bandwidth separate accounts, or single expensive high-bandwidth accounts with a service provider. The multiple low-bandwidth accounts require separate phone lines to connect through a modem for each account. The single high-bandwidth account requires very expensive equipment on top of the already costly service. Both methods of connection require unique IP addresses, which are being depleted quickly. This report describes the design of a project to be a median solution for connecting to the Internet. The project, which uses network address translation, consists of the program that manipulates outbound and inbound packets to allow for multiple computers on a local area network to access the Internet through a single PPP connection. This program is also a temporary solution for the problem of depleting IP addresses because it uses only one unique IP address. The program executes on a DOS base personal computer with a modem and a network interface card. The Internet Extender program described in this report is an inexpensive project that needs only one phone line to operate, and can handle enough local clients for any small business purposes.

1. Introduction

The Internet Extender is a software program designed to allow multiple machines to use a single IP address to access the Internet. When someone wants to use the Internet, his or her machine must have a valid and globally unique IP address. What ISP's (Internet Service Provider) do is, lease a unique IP address to your machine and you pay a hourly, monthly, or yearly service fee for use of that address. Since the address is dynamically assigned, you are not guaranteed that same address every time you log on. To receive the IP address, you dial-up the ISP using a modem or ISDN, then the ISP assigns an IP address for you to use. And only the machine that received the IP address is allowed to use that connection at any one time.

Now what if multiple people want to use the Internet at the same time or intermittently? They would have to have a separate machine, ISP account, and unique IP address to access the Internet simultaneously. The other option using single ISP accounts is to take turns using the connected machine. Single shared accounts can get very inefficient, and multiple accounts can get expensive because each machine must have a separate phone line and account. Now not everyone will use their account connection at the same time, so why not allow multiple computers connected to each other through an Ethernet Local Area Network to share a single connection to access the Internet? The Internet Extender is a solution to this problem.

When I began the design process of the project, I wanted the program to run minimized in Windows 95. But, the problem with that is that Windows likes to control all operations of the computer including hardware and networking software, so I would have had to learn how to program Windows Sockets, which in itself would have taken longer than I had time to spend on the project. So my design changed early to run on a DOS platform so I could easily access the hardware components of the system. I got the idea for the project from a previous experience with networking and accessing the Internet. Whenever I was on the Internet using my modem, and I wanted to use the phone, somehow, everyone else was on the Internet or phone. So I decided to create a program that would allow people to share a single connection to the Internet and free up the phone lines. The machines will communicate with each other over a Local Area Network.

When machines are communicating over a LAN running TCP/IP, they are usually set up with non-globally unique IP addresses as defined by the Internet Standards Organization. These addresses are not valid on the Internet because of duplication, or predefined special purposes for the addresses. For these machines to communicate over the Internet, there IP addresses are going to have to be replaced with a valid globally unique address. The Internet Extender will perform this task and more. The project is built from the idea of Network Address Translation, which I will be explaining a bit later. The basic theory is that the NAT box will change the non-unique addressing information to valid information when sending information onto the Internet, and when traffic comes back into the NAT, the addressing information is translated back, and sent to the correct local machine. The local machines treat the NAT box as their gateway to the Internet, and the Internet views all traffic as coming from a single address. This will also be a temporary solution to the shortages of IP addresses. Instead of having multiple machines use globally unique addresses, they can all share a single address, which will free up addresses for others to use.

2. Users Manual

2.1 *System Requirements:*

IBM PC or compatible with a 386 processor or higher
4MB of RAM
Microsoft DOS 5.0 or later
VGA, 16 colors recommended
14.4/28.8 modem, 56K recommended
Network Interface Card with packet driver
PPP packet driver, PPPD for DOS (freeware) recommended
ISP Dial-up account

2.2 *Installation*

The following instructions are assuming you are using a 3COM network interface card and the freeware PPP packet driver, PPPD for DOS. Create a directory on your main hard drive (usually C:) called INET_EXT. Copy all Internet Extender files into that directory. You can find the freeware PPP packet driver, PPPD for DOS on the Internet. Once you get that archive, extract all files into the INET_EXT directory of your hard drive. To make the installation easy, copy your Ethernet Packet driver (3C5X9PD.EXE for 3COM 10BaseT NIC) to the INET_EXT directory as well.

Now that all of the files are installed, you must create and edit two configuration files, WATTCP.CFG and PPPD.CFG. The WATTCP.CFG file must contain the local IP address and netmask of the Internet Extender machine, as well as the IP address and port of any server machines on the local subnet (see fig 1).

```
my_ip=10.64.1.32          # Class A IP Address 1.0.0.0–127.255.255.255
netmask=255.0.0.0         # Netmask information can be found in RFC1219
tcp_server=10.64.1.1:80   # TCP Web server at IP address:Port
```

fig 1: WATTCP.CFG

The second configuration file is the PPPD.CFG file used by PPPD for DOS to connect to your ISP, login in, and receive a valid IP address (see fig 2). In this file, you tell the program what COM port your modem is on, the maximum speed to connect at, your username/password, and phone number of the ISP.

```
com2                          # COM port your modem can be found on
57600                         # Maximum speed you wish to connect
modem                         # You are using a modem
connect "chat -e " ATZ OK ATDT2471089 ogin: userid word: password beginning"
                              # Phone Number, userid, and password
# see README file included with PPPD for more configuration options
```

fig 2: PPPD.CFG

After you have set up the configuration files, it would be easiest to add to your startup batch file (AUTOEXEC.BAT) for the computer to read and execute on start up so that the program starts automatically. You first load the Ethernet packet driver, then the PPP packet driver, and finally the Internet Extender program (see fig 3).

```
set wattcp.cfg=c:\inet_ext\wattcp.cfg # Tells program where to find config file.
cd inet_ext                           # Change directory to find program files.
3c5x9pd.exe 0x61                      # Ethernet packet driver and interrupt vector
pppd.exe                              # PPP packet driver
nat.exe                               # Internet Extender program
# The interrupt vector can be any value between 0x60 to 0x80.
```

fig 3: AUTOEXEC.BAT

2.3 *Operation*

Now that the configuration files are set up properly, reboot the machine and allow it to automatically initialize the packet drivers and start the program. The program will start and display the packet driver information for 30 seconds then execute as normal. Once started, the program will display the current active connections from each local machine to each remote machine. The display is scrollable, and if any errors occur, the error message will inform you what direction the packet was headed (inbound/outbound), what the source and destination addresss and ports were, and the protocol of the packet.

3. The Internet Extender Program Design

The Internet Extender is built from the idea of Network Address Translation or NAT. The basic idea behind NAT is to translate valid addressing information from one network to valid addressing information of another. The Internet Extender utilizes the NAT technology by translating multiple non-globally unique IP addresses into a globally unique IP address that is accepted on the Internet. Network Address Translation is described in greater detail in RFC1631. An example of its application is with my personal network consisting of eight computers running Windows 95 and statically assigned (class A) IP addresses. If these machines were to try to communicate on the Internet directly, all of the packets would be ignored. So the solution is to have the NAT box be a machine with a valid IP address assigned by the ISP. The NAT box will actually have two separate IP addresses. The PPP side of the box will have a globally unique address, and the Ethernet side will have a local class A address. The local machines set the local IP address of the Internet Extender to be their gateway address. Then when they send information directed onto the Internet, the Internet Extender will receive the information manipulate the addressing information and send it onto the Internet. The reverse version of the procedure is used to receive information from the Internet and send it back to the correct client on the local subnet.

The Internet Extender is designed to treat a local subnet of machines as a single node on the Internet. Outside sources will only see traffic come from a single IP and in turn create a firewall so undesirables cannot access vital information on the local subnet, unless authenticated by a local server. The program will be like a one-sided mirror for the subnet of machines to look through, and keep outside sources from seeing in. That characteristic creates a temporary solution to the depletion of IP addresses that exists today. If instead of having ten machines get their own unique IP addresses, and only use the connection 10% to 20% of the time, you can assign one address to the Internet Extender and have it allow those ten people to share the connection. This temporary solution would have freed up nine IP addresses for other people to use.

The Internet Extender program is designed to deal with information at the packet level. When information is received on the Ethernet side of the machine, the Ethernet packet is put into a buffer. When the program receives the packet from the buffer it references the Ethernet packet as an IP packet by moving the starting pointer from the head of the Ethernet packet to the head of the IP packet encapsulated within it. Once the program has the IP packet, it can access the header information, which contains information about source and destination IP addresses and ports, and what type of packet it is. The program will extract the addressing information and packet type, put the information in an inbound and outbound network address table. The program will then replace the non-unique source address and port with the globally unique address assigned by the ISP and a port number that will be created to associate the local client with the globally unique address.

Once the header information for the packet has been changed, the checksum of the packet needs to be recalculated to reflect any changes in packet size. Then the packet is sent out onto the Internet. The same basic procedure is used for the opposite transmission. Except, when a packet comes into the NAT box from outside, there must be a table entry or server entry that reflects its header information, or the packet is ignored. To insure that a connection doesn’t stay open between the source and destination longer than necessary, an expiration thread will be running to remove table entries after connections have been closed, or stalled for a long period of time. There will also be a display thread running that displays the program activity. A graphical program process flow can be found in appendix B.

3.1 *Programming Tools*

The program will be written in C using the Multi-C © from Mix Software functions and libraries. Waterloo © TCP/IP C library functions and routines will be used to receive, manipulate, and send IP packets. The reason for using Multi-C is for its application of threads and non-preemptive scheduling algorithms. The functions included in Multi-C will allow for multiple processes to independently execute together. The threads will also make the program run smoother and faster overall because they are not dependent upon each other to execute.

3.2 *Network Address Hash Table Structures*

The Internet Extender program keeps track of the sessions that are open between the local client machines and the remote Internet machines by keeping two arrays of linked lists that are accessed as Hash tables. An Inbound table is used to map entries for packets coming in from the Internet, and the Outbound table is used to map entries for packets coming from the local subnet. I originally only had one table to map entries going both ways. That configuration was much simpler, but it was flawed. Each entry was used for mapping inbound and outbound packets, so if two machines on the local subnet tried to access a mail server, and somehow mapped to the same entry, they would both be sent to the mail server with the same IP and port. Not only would the mail server get confused, but every time a packet came back from the server, there would be no way to distinguish between the two entries. The dual table arrangement makes it so there will always be separate entries, because as I will explain next, there will be no chance for the packets to get mapped to the same entry.

Packets that are bound to be mapped in the Outbound table get the index of the entry that the local information will be placed in by:

```
hashsize = the number of entries that can be mapped, found by available memory.
index = (local_IP  XOR  local_port)  mod  hashsize;
```

The actual mapped port is found by a much different method, not having anything to do with the data or index of the entry. I like to call the method “sliding window” not to be confused with the packet acknowledgment protocol. The program is set to use a maximum of 32,000 ports, but only about 7000 entries can be in use at any one time without running out of memory. So what happens is the program allocates one mapped port at a time to each new session in sequence from 5000 to 37000. The reason for the starting point of 5000, is that almost every known registered port used by the most common applications are less than 5000. When it is allocating these ports the window of roughly 7000 will move when the used ports get up to port 12000. If for some reason the memory runs out and a new entry needs to be added, the oldest entry will be deleted.

The oldest entry is found by searching the table and finding the entry with the oldest timeout. Every time an entry is created or accessed, the timeout for that entry is updated to the maximum of 6 hours and gets counted down until it times out. When an entry is being looked for to delete, the one with the shortest amount of time to live will be deleted. Every time an entry is created in the table, a timeout is set, a packet count is set to zero, and the addressing information of the local machine is added to the entry. Then the next time the entry is accessed, the timeout is reset and the packet count is incremented. When a connection is closed, that entry timeout has 30 seconds to receive any remaining packets and is then removed from the table to free up memory.

When an entry comes into the Internet Extender on the Inbound side, the timeout is reset and the packet counts are incremented as well. The way that the Inbound table entry is found is a bit simpler since the destination port number of the inbound packet is equal to the mapped port found in the table entry.

```
index =  mapped port  mod  hashsize;
```

That simple expression is used to direct the inbound thread to the correct index of the Inbound table entry. The dual address tables allow for the mapped port to be different every time since it is simply a field in an entry of the tables. Now there is no chance of duplicate mapped ports to be assigned to a packet.

The program is broken up into four main threads: Inbound thread, Outbound thread, Display thread, and an Expiration thread. The main program first initializes the packet drivers to make sure that both the Ethernet and PPP drivers exist. Then it initializes the address tables and display. Finally it calls each of the threads that will execute and yield to each other infinitely or until a kill command (ESC) is given to the program. Both packet manipulation threads will have a higher priority to run then the other two threads. The higher priority means that it will run more times than that of a lower priority thread. The program will be structured in a way that if the active thread goes into a wait state, or it has completed processing data, it will yield to another thread until the other threads yield back to it. Each thread is detailed further below.

3.3 *Outbound Thread*

When a client machine makes a request to send information to and from the Internet, it needs to communicate over the local area network. Since the only connection to the Internet is through the NAT box, there is a need to translate each packet and send it onto the Internet without the client's knowledge. The Ethernet packet driver accepts packets and asks the program for a buffer to put it in. The program responds by sending a pointer to an address of an empty buffer or a null pointer if the buffers are full. The Outbound thread is in a forever loop, which checks the packet buffer for new packets. If no packet exists, the thread yields the processor to another thread. If a new packet is received then the thread will extract the addressing information from the packet header like the local and remote IP addresses and ports, and the protocol. That information is then taken and mapped into an entry in the Outbound address table with the method described earlier.

Once the entry is mapped, the newly mapped entry information is returned to the Outbound thread. The Outbound thread will then substitute the local port and local IP address with the mapped port and the globally unique IP address. The mapped port is going to be used to find the entry that contains the local host addressing information so that the packets from the Internet can be sent to the correct local machine. Once this entry is established, the information mapping is much simpler, because the information already exists and can just be extracted from the table. After the packet manipulation has occurred and the size of the packet has changed if at all, the checksum needs to be recalculated and reinserted back into the packet. The timeout for the entry is then set to a predefined time (TCP=6 hours, UDP=2 minutes, ICMP=1 minute). If the packet being sent onto the Internet is the final packet of the connection, the timeout is reset to 60 seconds for any extra closing packets going back and forth. The packet sent count is initialized to 1 or incremented depending on if this is the first packet of the connection or not. The packet is now valid and ready to be forwarded onto the Internet, so it is forwarded to the PPP packet driver to be sent onto the Internet. Once the packet is sent on its way, the Outbound thread yields the processor to another thread.

Not all packets that go through the Outbound thread run so smoothly. There are packets that don’t follow the same addressing scheme as the previously explained. Applications like FTP have the IP address embedded within the data portion of the packet in plain ASCII text. Some ICMP packets have only partial information about the addresses and ports, so they go through special functions to properly handle them. But for the most common applications, the Internet Extender handles them correctly.

3.4 *Inbound Thread*

When a client machine sends an Ethernet packet onto the LAN meant for the Internet, it most likely is going to expect a response back from the remote destination in an Ethernet packet. The Inbound thread is designed to receive the IP packet from the Internet and change the header information back to what it was when the client sent it. The Internet Extender program relies on the PPP packet driver to send and receive packets from the Internet. The PPP packet driver works in the same way as the Ethernet packet driver, in that it requests a buffer for the packet, and when it receives the address of the buffer it puts the packet in the buffer and sets a flag to full. The Inbound thread executes in much the same manner as the Outbound thread in that it is in a forever loop checking the packet buffer for received packets. If there is a packet in the buffer then it sets the flag for that buffer to empty and manipulates the header information. If the buffer is empty, then the program yields the processor so other threads can execute.

The Inbound thread uses the destination port of the packet to search the Inbound table because it should be equivalent to the mapped port that was put into the packet when it was sent onto the Internet. If however it maps to an entry that doesn’t contain information about the connection, then the packet is ignored. If servers were established when the program was started, and the port is equivalent to the well-known port of the server, then a new entry will be mapped to reflect a new connection.

Once the entry is found and all the information matches, the procedure will manipulate the header information. The Inbound thread will replace the previously mapped destination IP and port with the local IP and port numbers that were inserted into the table when the entry was first created. Once the switch is complete, the checksum is calculated to make sure that the packet size is correct. Then the timeout of the entry will be reset to the predefined timeout, and the count for packets received will be incremented. Finally the packet is sent to the Ethernet packet driver that will encapsulate the packet within an Ethernet packet, recalculate the checksum, and forward it to the correct machine on the local subnet. Once the packet is sent, the Inbound thread will yield the processor to another thread. So both Inbound and Outbound threads will process only one packet at a time before yielding to each other, or another thread.

3.5 *Expiration Thread*

The Expiration thread is designed to remove table entries of connections that have been closed, or that have been idle for a long period of time. This thread has a lower priority than the first two, so it will be yielded to less of the time. The Expire thread will go through the address tables one entry at a time and decrement the timeout the amount of time that has gone by since the last visit to the entry, or by one second if the entry's timeout was just reset, then yield to another thread. When it is yielded to once again, it traverses the table to the next valid entry and repeats the process. The expiration thread will only manipulate one entry at a time. This will allow other more important threads to run more often with less wait states. Every time a packet is received and the table is updated, the timeout gets reset to the max value. Once an entry decrements to zero, it is removed from the table to free up memory and make that port available for use by another connection.

3.6 *Display Thread*

The Display thread is used to indicate and update what activity is happening within the program. The routine displays what connections are currently using the Internet at that specific time on the monitor. It is also run with a lower priority than the first two threads, so it will not execute as often. The display will accurately show what connections are open in real time, so when an entry is removed from the table, the display will reflect that action. The Display thread will show the local IP address and port, the remote IP and port, the protocol, packets sent and received, and the timeout status. Along with updating you on what connections are open, it will display what percentage of resources you have free. Since there are only 25 rows that can be displayed in a DOS window, there will be no way to display all of the connections when traffic is high, so you can scroll the display with the PGDN and PGUP keys. An example of the display is given in appendix C.

4. Testing

Once the implementation of the program was complete, I needed to test and debug the program for functionality. I built a basic Pentium 166MHz DOS based system with a modem and network interface card (NIC) to fulfill the requirements of the program. When the machine boots up, the batch file starts the Ethernet Packet Driver TSR which allows the Internet Extender to communicate with the NIC. It then loads a Point-to-Point Protocol (PPP) Driver TSR which allows the Internet Extender to communicate with the modem. The PPP Driver is a program that will dial the Internet Service Provider, send login information, and receive a globally unique IP address for the Internet Extender to use. Once those two TSR's are loaded, the program is run. When the program starts, it verifies that both packet drivers are present, sets up the globally unique IP address to use, and WATTCP sets up the local non-unique IP address for the local machines to use as their gateway address. In that same local configuration file, local server addresses and ports can be specified so that outside clients can access local servers.

The machine is put onto a Local Area Network of client and server machines, and each machine must set the IP address of the Internet Extender machine as their gateway to the Internet. When the program has been started, the local machines will be able to run Internet applications as per normal. All traffic directed to the Internet will go through the Internet Extender machine, while the local traffic will be ignored. When I tested the program, I ran applications such as Netscape, Telnet, MIRC, and even FTP in PASV mode. I was also able to setup and access a local FTP server from outside the LAN. The program did not, however, handle every application type that I tested.

4.1 *Problems Faced*

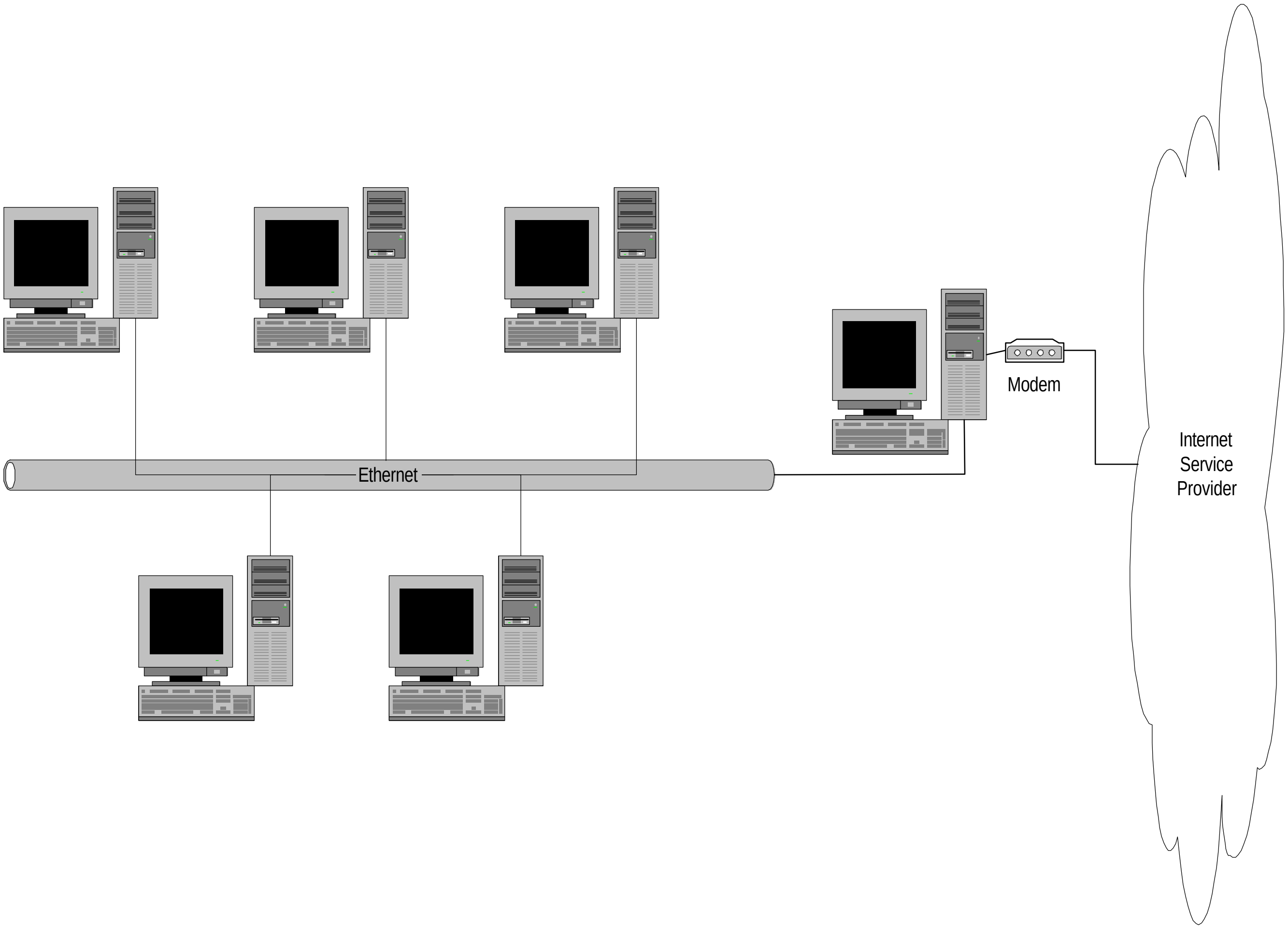
The first problems faced were applications that sent ICMP packets like PING. PING is an application that sends a message that requests an ECHO of the message to come back to the application to see if the target address specified exists. These ICMP packets contain embedded IP headers that must be translated, and once translated, the embedded checksum must be recalculated as well. ICMP packets don't follow the same convention as TCP and UDP packets, so every time an ICMP packet comes in, it must be handled by special functions. The other problems I faced were packets that have embedded IP addresses within the data portion of the packet, like FTP and IRC authentication. As of right now, if you run FTP in PASV mode, the addressing format works the same way as normal TCP packets, and the NAT box handles them appropriately. And luckily, the IRC authentication process is not mandatory to create a connection and use the software. So after the host authentication process has ended, the program will function as normal. I have not tried the more obscure applications that are not commonly used, but when I come across other applications that don't function properly, they will be handled accordingly.

4.2 *Results*

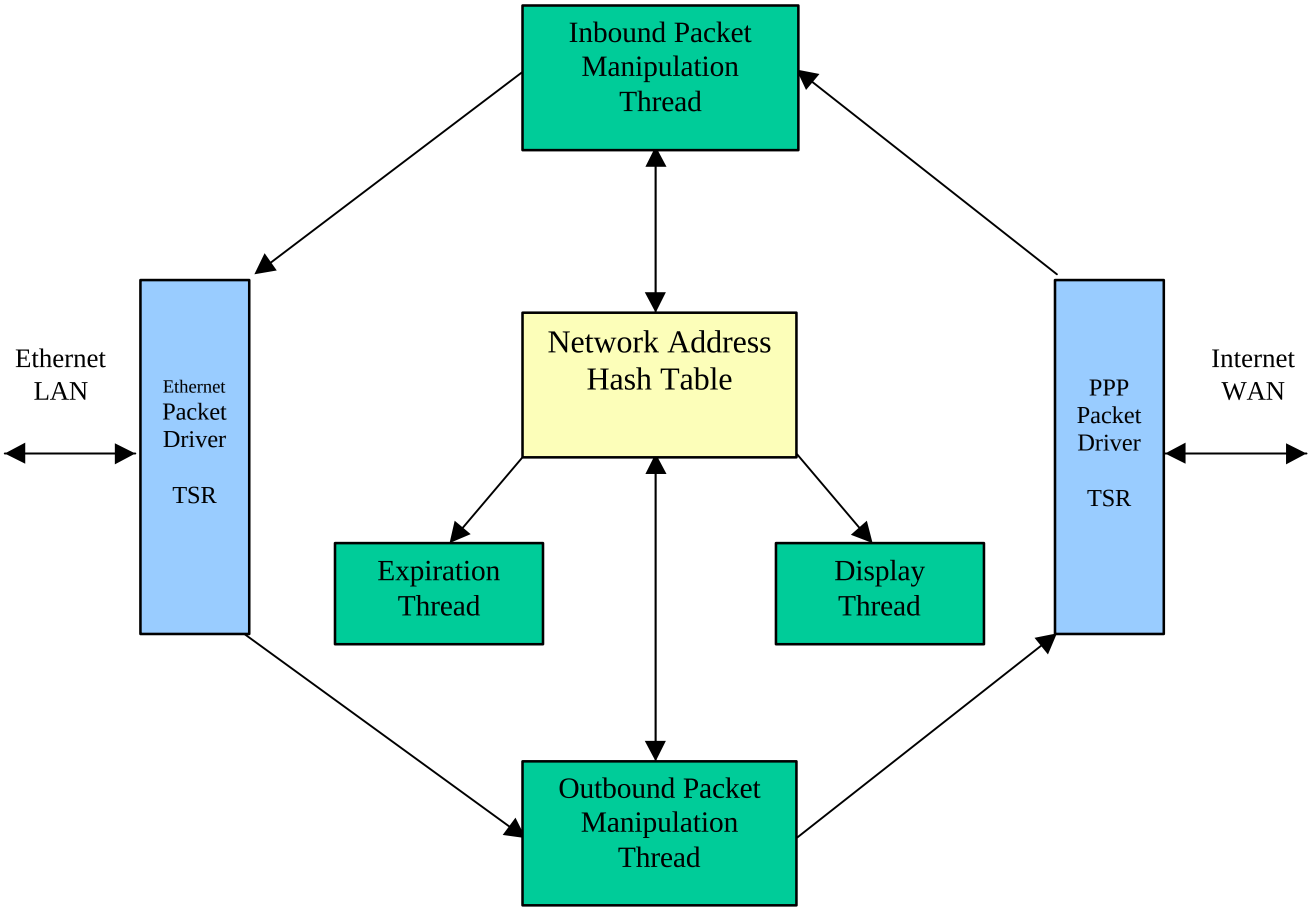
The Internet Extender executes correctly. At the end of the Spring quarter, the program has more functionality than what was expected during the design phase of the project. Features have been implemented that were thought to be a future enhancement of the program, like local server support, and the original intent of the project functions properly. The performance of the program has shown to be better than I had originally expected. The Internet Extender has been tested on numerous occasions by as many as 8 machines accessing information simultaneously. Testing will continue into the near future to make sure the program can handle the most common Internet applications as well as those less common. The source code for the Internet Extender is included in Appendix D.

5. Conclusion

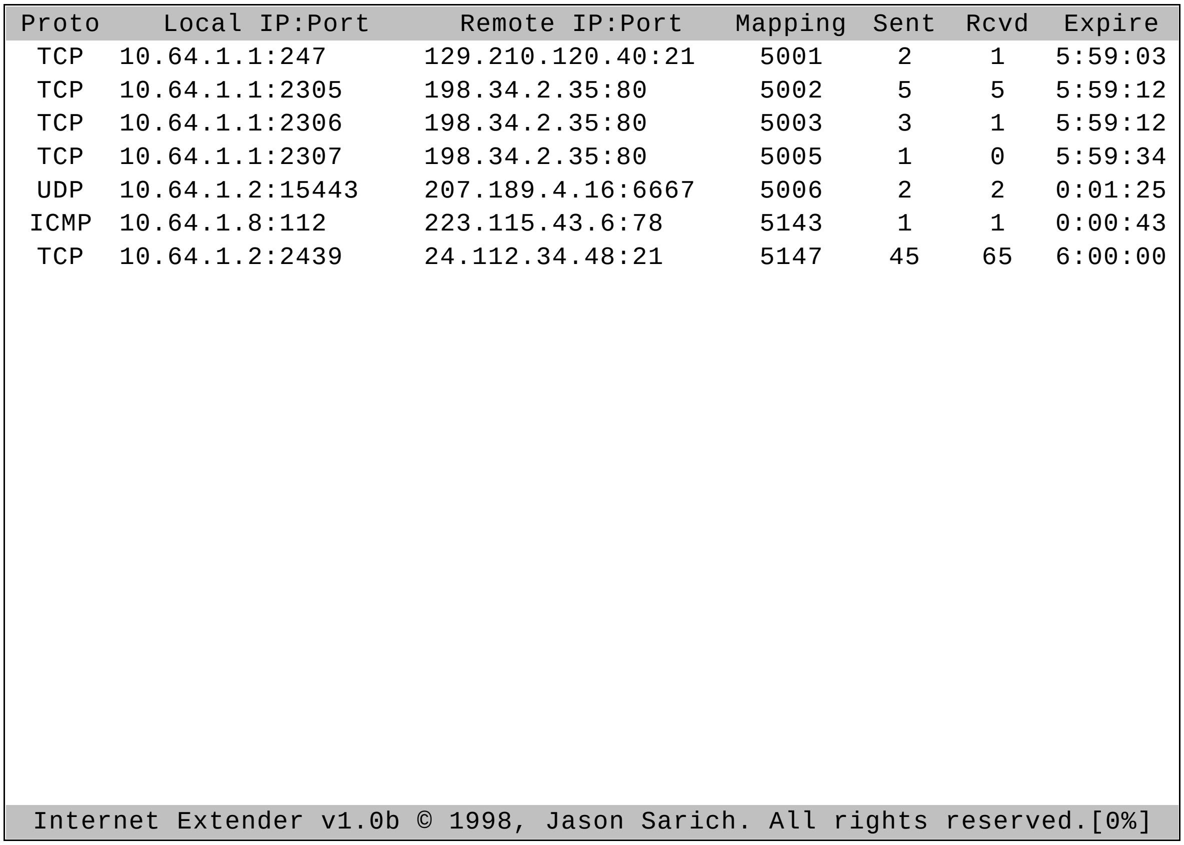
I have succeeded in creating a correctly functioning version of the Internet Extender. I have learned far and beyond what I had planned to learn about computer networking at its lowest form. I now have a better understanding of how Internetworks and LANs work, and why problems occur within them. I have also learned more about the circular process of the design, implementation, and testing phases of a project. This design project has furthered and strengthened my desire to pursue a career in the computer networking field.



A. Physical Network Connection Layout


Inbound Packet Manipulation Thread
Network Address Hash Table
Ethernet LAN
Ethernet Packet Driver TSR
Internet WAN
PPP Packet Driver TSR
Expiration Thread
Display Thread
Outbound Packet Manipulation Thread

B. General Operation of the Internet Extender Program



```
Proto    Local IP:Port         Remote IP:Port     Mapping  Sent  Rcvd   Expire
 TCP   10.64.1.1:247        129.210.120.40:21      5001      2     1   5:59:03
 TCP   10.64.1.1:2305       198.34.2.35:80         5002      5     5   5:59:12
 TCP   10.64.1.1:2306       198.34.2.35:80         5003      3     1   5:59:12
 TCP   10.64.1.1:2307       198.34.2.35:80         5005      1     0   5:59:34
 UDP   10.64.1.2:15443      207.189.4.16:6667      5006      2     2   0:01:25
ICMP   10.64.1.8:112        223.115.43.6:78        5143      1     1   0:00:43
 TCP   10.64.1.2:2439       24.112.34.48:21        5147     45    65   6:00:00

 Internet Extender v1.0b © 1998, Jason Sarich. All rights reserved.[0%]
```

C. Example of the Internet Extender Display