DOS Comms
Tech Notes

Winsock

Frequently Asked Questions About Windows Sockets Version 1.1

20 Sep 1993

This FAQ has been put together by Mark Towfiq, with much-appreciated assistance from Jay Allard, Bruce Backman, Paul Brooks, Martin Hall, Simon Hewison, Mike Morse, Bob Quinn, Ed Schwalenberg, Bill Tang, Dave Treadwell, and Fred Whiteside. If you have any modifications to this FAQ, send them to towfiq@Microdyne.COM, and I will fold them into the next revision.

First of all, the questions:

---

1. What is Windows Sockets?

Answer: The Windows Sockets specification defines a network programming interface for Microsoft Windows which is based on the "socket" paradigm popularized in the Berkeley Software Distribution (BSD) from the University of California at Berkeley. It encompasses both familiar Berkeley socket style routines and a set of Windows-specific extensions designed to allow the programmer to take advantage of the message-driven nature of Windows.

The Windows Sockets Specification is intended to provide a single API to which application developers can program and multiple network software vendors can conform. Furthermore, in the context of a particular version of Microsoft Windows, it defines a binary interface (ABI) such that an application written to the Windows Sockets API can work with a conformant protocol implementation from any network software vendor. This specification thus defines the library calls and associated semantics to which an application developer can program and which a network software vendor can implement.

Network software which conforms to this Windows Sockets specification will be considered "Windows Sockets Compliant". Suppliers of interfaces which are "Windows Sockets Compliant" shall be referred to as "Windows Sockets Suppliers". To be Windows Sockets Compliant, a vendor must implement 100% of this Windows Sockets specification.

Applications which are capable of operating with any "Windows Sockets Compliant" protocol implementation will be considered as having a "Windows Sockets Interface" and will be referred to as "Windows Sockets Applications".

2. What is the latest version?

Answer: The latest version of the specification is 1.1.

3. When is the next rev of the specification? Why not sooner?

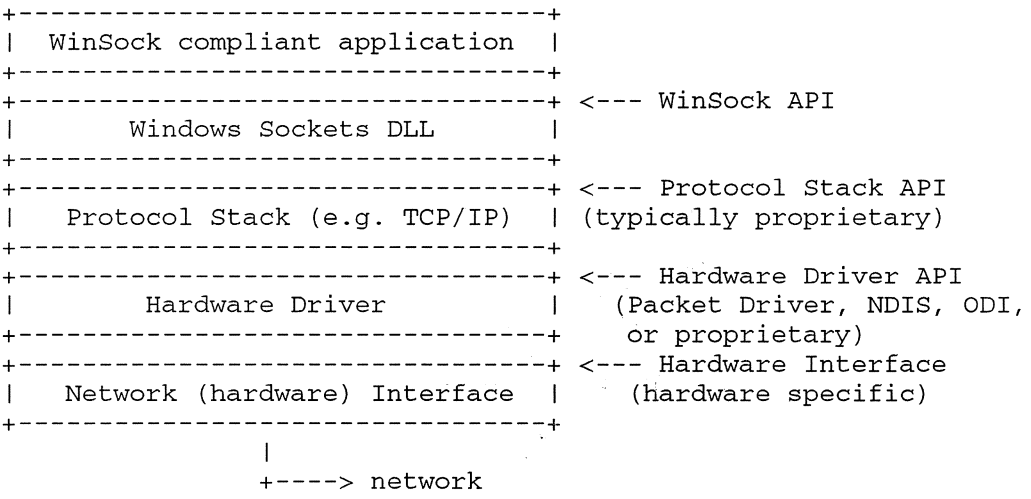
Answer: The next rev. (2.0) will not be until towards the end of 1993. We need 1.1 of the API to become firmly settled and implemented first.

4. Where can I get a/the WINSOCK.DLL?

Answer: You can most probably get one from the same place you got your TCP/IP software from.

5. Why isn't there just one WINSOCK.DLL? Do I need a TCP/IP already to use it?

Answer: The Windows Sockets specification defines the top level of the DLL, the part which is called by user programs. The method a given WINSOCK.DLL will use to access TCP/IP (or NetWare, or AppleTalk, or DECNet ...) depends on the networking package you have installed, and therefore must vary. A WINSOCK.DLL is therefore just an interface to whatever existing protocol you already have installed. An illustration would help:



6. Where can I contact Windows Sockets application and implementation vendors? (include list of address, phone contacts)

Answer:

Date: Tue, 6 Apr 93 12:53:37 PDT
From: tang@documentum.com (Bill Tang)
To: winsock@Microdyne.COM
Subject: vendor list (long)

Thanks to all that responded to my previous inquiry of vendor implementations. I am posing a summary and will keep the list updated if I receive more information.

SUMMARY

Company	(old list at 11/23/92) BETA	FINAL	Update (4/6/93)
3Com Corp	Q1'93	Q2'93 ?	
Beame & Whiteside	Q2'93	Shipping	Shipping v1.1 DLL with stack
Distinct Corp	Q3'92	Shipping	?
FTP Software	Q4'92	Shipping	shipping v1.1 TCP/IP stack with DLL; DLL available on BBS or anonymous-ftpable on vax.ftp.com; Development Kit being updated to include v1.1 Windows Sockets support.
Frontier Tech	Q3'92	Shipping	?
IBM	Q4'92	Q1'92	?
JSB Corporation	Q4'92	Q4'92	?
Lan Design	Q4'92	Q1'93	?
Lanera Corp			Shipping v1.1 TCP/IP stack
Microdyne	Q4'92	Q1'93	?
Microsoft(W NT 32b)	Q3'92	Q2'93	Win32 SDK March release
Microsoft(W NT 16b)	Q4'92	Q2'93	Win32 SDK March release
Microsoft(W 3.x 16b)	Q4'92	Q2'93	?
NetManage	Q4'92	Q4'92	shipping v1.1 DLL
Network Research	?	?	?
Novell	?	?	?
Spry			(stack) supports Winsock v1.1
Sun Microsystems	Q4'92	1stHalf'93	?
Ungermann Bass	?	?	v1.1 TCP/IP stack and DLL: Beta Q1'93, Final Q2'93
Walker Richer Quinn	Q4'92	Q1'93 ?	
Wollongong	Q4'92	Q1'93 ?	

7. Where can I get sample applications and tests?

Answer: Files and information related to the Windows Sockets API are available via FTP (user: "anonymous", password: your e-mail address) on the host SunSite.UNC.EDU, in /pub/micro/pc-stuff/ms-windows/winsock, which is a mirror of /pub/winsock on Microdyne.COM (SunSite has a much faster connection to the Internet, so you are advised to use that). Note: if you do not have FTP access to the Internet, send a message with the word "help" in the body to either ftpmail@SunSite.UNC.Edu, or ftpmail@DECWRL.DEC.Com (in the UK mail to ftpmail@doc.ic.ac.uk), to obtain information about the FTP to Mail service there.

8. Will Windows Sockets be in _____?

8.1. Will Windows Sockets be in Windows NT?

Answer: Yes. In 16 and 32-bit versions.

8.2. Will Windows Sockets be in Windows for Workgroups?

Answer: Yes.

8.3. Will Windows Sockets be in DOS?

Answer: Paul Brooks of TurboSoft (paul@abccomp.oz.au) tells me:

> As for a Windows Sockets for DOS - we have a library that is pretty much complete. The goal was to produce a DOS library and Binary API which would allow developers to code applications using the Windows Sockets 1.1 spec. and have it run exactly the same under DOS as Windows - blocking and non-blocking modes, Asynchronous calls, the works. Apart from some AsyncGetXXXbyYYY calls the rest works, although it is not possible to capture all the Windows-specific semantics of some areas.

I am sure you can contact him for more information. JSB also has a standardized Berkeley Sockets API for DOS which provides access to all TCP/IP implementations.

8.4. Will Windows Sockets be in UNIX?

Answer: Well, since it came from BSD UNIX, there's not much need.

8.5. Will Windows Sockets be in Win32-S?

Answer: Yes, Win32s 1.1 contains a thunking layer that converts 32-bit Windows Sockets calls to 16-bit Windows Sockets. It should work on top of any 16-bit WINSOCK.DLL.

9. What about standard APIs for ____?

9.1. What about standard APIs for FTP?

Answer: Not yet.

9.2 What about standard APIs for Telnet?

Answer: Not yet.

9.3 What about standard APIs for SNMP?

Answer: In the works. Send e-mail to winsnmp-request@Microdyne.COM to join the list.

9.4 What about standard APIs for RPC?

Answer: Microsoft will be providing an implementation of DCE RPC. Also in the works is the definition of a standard implementation of ONC RPC (known as RPC for Windows) which will be made available by multiple vendors. To contribute to the discussion, send email to rpc4win@wco.ftp.com.

9.5 What about standard APIs for TLI/XTI?

Answer: No, not yet. Vendors chose to do Windows Sockets because of the sockets application and knowledge base, however anyone out there is free to try a Windows TLI/XTI Specification.

10.1. Does Windows Sockets work over protocols other than TCP/IP?

Answer: Yes, it does. But we know of no vendors supplying one currently.

10.2. Will it?

Answer: Yes. Windows NT will include mechanisms for multiple protocol support in Windows Sockets, both 32-bit and 16-bit.

11. Why no SOCK_RAW?

Answer: SOCK_RAW is optionally implemented by many major Windows Sockets vendors. It is not mandated now because: 1) not every stack vendor can

supply a complete SOCK_RAW interface, and 2) we did not have time to define the minimal subset every vendor could provide (e.g. maybe just ICMP).

12. Why isn't it possible to share sockets between tasks?

Answer: The real answer is that it wasn't considered sufficiently important for 1.1, but is high on the list for inclusion in 2.0.

13.1. How do I get my IP address?

Answer: Do a gethostbyname() on the output from gethostname().

13.2. Why no SIOCGIFADDR?

Answer: It was not included because there were concerns it could not be supported across platforms. It may be included in version 2.0 of the spec.

14. When should I use blocking vs. non-blocking sockets?

Answer: Try to use non-blocking sockets where possible; certainly if you are coding a Windows application from scratch. Blocking sockets should only be used when trying to maintain portability between UNIX/MS-DOS and Windows.

15. What about other socket options that BSD supports? Ioctls?

Answer: Some vendors may support them. Do not rely on ones not explicitly mentioned in the specification, however.

16. How can I get the local username?

Answer: Based on a suggestion from Simon Hewison (p0063886@cs3.oxford-brookes.ac.uk), one idea is to call WNetGetUser() in the NETWORK.DRV. If a vendor has supplied a NETWORK.DRV then this will work, if not you can just use some other method. Thus you could write a bit of code thus:

```
WNetGetUser(szNetUserId,sizeof(szNetUserId);
if (strlen(szNetUserId)==0) {
        /* call some other method of getting
           userid, eg. the lan manager call NetWkstaGetInfo */
}
```

Information on this API call is in the Windows Device Driver Adaption Guide.

17. Do I need to initialize the WSAData structure before calling WSAStartup?

Answer: No, WSAStartup does not retrieve the contents of the WSAData structure pointed to, it fills it in.

18. If I write a Windows Sockets program, will I be able to communicate with a Sockets program on UNIX or any other non Windows platform?

Answer: Absolutely! This common question is the result of confusing protocols with the APIs; communicating programs need not have been created with the same APIs, as long as they are using the same (transport & network) protocols.

19. Is it possible to create sockets that map to a dll rather than an application? I have tried a WSAStartup() as part of my LibMain, but the sockets that are created are owned by the application, not by the DLL. It would be desirable for me to have some of the sockets that are opened have "A Global (DLL wide) Scope".

Answer: The way this situation has been dealt with by developers is to have your DLL create a "helper process" which will perform all Windows Sockets

operations on behalf of your applications (thereby having all sockets owned by the helper process' task).

20. A *Class* of questions that could be answered are related to porting extant BSD applications to Windows Sockets: "How to I implement the xxxx function call in my Windows Sockets application?" (e.g. fcntl(), readv(), etc).

Answer: In general, you will have to code such functions yourself, although it would not hurt to suggest them for the next revision of the specification as well.

21. Is there a Pascal/Visual Basic/Visual Cobol header file for Windows Sockets?

Answer: The Pascal and Visual Basic forms are already in the FTP archive. Look there for any other header files that people may have developed so far. It should be pointed out that not all the Winsock functions can be called from Visual Basic and the subset that can be called are not really enough for a true app (for example, you can't call gethostbyname()). To use Winsock from VB requires a translation DLL that will probably have to be written in C. To date, there is no public domain implementation, although several vendors have announced toolkits for Winsock for VB.

Trumpet Winsock

install.txt from the trumpet distribution

Trumpet Winsock

Version 1.0

By Peter R. Tattam

Managed by Trumpet Software International

Introduction

Thank you for using the Trumpet Winsock. It is through the kind support of many users out there that quality networking software has been available at affordable prices to the Internet community. The Trumpet Winsock is a Windows Sockets 1.1 compatible TCP/IP stack which provides a standard networking layer for many Windows(tm) networking applications to use, and has itself been a major vehicle in achieving widespread use of Windows Sockets 1.1. The product is a shareware item and as such, you are permitted to evaluate it for a period of 30 days. If you are satisfied with its usefulness, a registration form is provided which you can fill out and send to Trumpet Software International. A registration fee is requested to maintain the development and support of this software. Suitable arrangements have been made for site licenses, and details can be found in a later section.

Disclaimer & Copyright

They are provided as shareware with the following limitations:

> These programs are shareware and are not to be resold or distributed for sale with other programs which are for sale. There is no warranty or claim of fitness or reliability. The programs are distributed AS IS, and as such neither the author, nor Trumpet Software International nor the University of Tasmania shall be held liable for any loss of data, down time, loss of revenue or any other direct or indirect damage or claims caused by these programs.

Instructions for Installing the Trumpet Winsock.

The Trumpet Winsock will only run on your PC under the following conditions. You must have either a packet driver available for use by network programs, or if you wish to use SLIP, a free comms port. Additionally, packet drivers can only be used reliably under enhanced mode using WINPKT. Standard mode can be used, but care must be taken to avoid system crashes. NDIS and ODI can be used via packet driver shims, but their use is not supported. PKTMUX may also be used instead of WINPKT, and must be version 1.2c or later, but again its use is not supported.

If you already have some kind of TCP/IP networking package installed, it is most likely that the Trumpet Winsock will not run and you will have to massage your system configuration to install the Trumpet Winsock,

possibly even to the extent of uninstalling that networking package. Alternatively, there may be a Winsock available for your package in which case the Trumpet Winsock will not be required.

Installing Trumpet Winsock over Packet Driver.

Firstly, if you don't know what a packet driver is, it is normally a small piece of software which sits in between your network card and your TCP program. This provides a standard interface which many programs can use in a similar manner to BIOS calls using software interrupts.

Why is it called a packet driver? This is because modern networks send information using packets of information rather than sending information one byte or character at a time. For example, Ethernet sends information in frames of up to 1514 bytes long. The reason for sending things in packets is that information can be transmitted much more efficiently in packets.

Central to the concept of the packet driver is the vector which is used to communicate with it. The 80x86 family of processors allows programs to communicate with the operating system through what is called a "software interrupt", which always has a number in the range 0 to 255. This is termed a "vector" and is the one of the key mechanisms to pass control to the MS-DOS operating system. Usually the vectors are in hexadecimal, making the range expressed as 0x00 to 0xFF. The 0x in front of the number means that we are using hexadecimal numbers instead of decimal numbers. They may also be expressed in the notation 00H to FFH, or $00 to $FF. If you are dealing with packet drivers, hexadecimal notation is much more common, but occasionally they are expressed in decimal. Examples of software interrupts in use on PC's are 0x10 for the video BIOS, or 0x21 for calls to DOS.

Packet drivers are only allowed to have a software interrupt vector in the range 0x60 to 0x7F. Normally, you will pick 0x60 as the default place to install your packet driver, but certain machine configurations may make that vector unavailable. Just choose one that is free - the packet driver should tell you if you can use it or not.

The Trumpet Winsock also uses a special virtual packet driver "wrapper" which enables your packet driver to function correctly in Windows. While the packet driver is an efficient way to communicate with your network card, it will not work correctly from Windows without a little assistance. The program "WINPKT" was written by some clever people on the Internet to allow a packet driver to work correctly within Windows by making sure that packets get directed to the correct "virtual machine" under Windows enhanced mode. A "virtual machine" can be either the entire Windows session, or any

dos session active within Windows. Refer to the Windows system documentation for more details.

In addition to this, you will need to have some understanding of IRQ vectors and I/O addresses that may be relevant to installing your network card.

Where do I obtain packet drivers from?

These days, packet drivers are usually provided with your network card, but a comprehensive collection of public domain packet driver can be obtained from a packet driver collection called the "Crynwr Packet Driver Collection." Information on where to get this packet driver collection from is provided as an appendix to this document.

Actually Installing the Winsock.

Before you do anything, copy the files winsock.dll, tcpman.exe, winpkt.com, hosts, services and protocol to a suitable directory.

eg. c:\trumpet

the essential files:

winsock.dll	the guts of the TCP/IP driver
tcpman.exe	interface program for managing the winsock
winpkt.com	virtual packet driver interface for windows
hosts	list of host names & aliases
services	list of Internet services
protocol	list of Internet protocols

Modify the path line in your autoexec.bat to contain a reference to that directory.

eg. path c:\dos;c:\windows;c:\trumpet

Make sure it is active by rebooting or executing autoexec.bat again.

The most basic setup of packet driver and WINPKT would look something like this example :

```
ne2000 0x60 2 0x300
WINPKT 0x60
```

The first line installs an NE2000 packet driver on vector 0x60 using IRQ 2 and I/O address 0x300

The second line installs the WINPKT virtual packet driver using the same vector that the ne2000 packet driver was installed on.

This is of course an example so your mileage will of course vary. Some example configurations are described later in this document. Choose the one which suits you the best and modify it to your requirements.

Now you are ready to start windows. Start it up!!

From windows, start up tcpman. From the file manager, go File/Run, and then "tcpman". If this fails, the path is probably not set up correctly, so fix it. Later on, you can install it as an icon to use it directly.

Assuming you are a first time user, a setup screen will appear giving you a number of options to fill in. You will need to fill in a few details to enable the TCP package to function. Fill in the following details. If you are unclear on any of them, try to seek some help from qualified Internet support staff - it will save you a lot of time.

IP address	your Internet IP address, "bootp", or "rarp". lower case please. If you use BOOTP, be sure to have a BOOTP service on the network or the winsock will not load.
Netmask	your Internet network mask. (eg. 255.255.0.0)
Default Gateway	your default Internet gateway. (IP address)
Name server	your name server IP address for DNS searches. You may provide more than one address by separating the addresses with spaces. (IP addresses only)
Time server	at present unused - future winsock API's may support this. (IP addresses only)
Domain suffix	a space separated list of domain suffixes to be used when resolving names in the DNS system.
Packet Vector	either leave this as 00 to search for the packet driver, or the vector that you installed the packet driver under. The number is required in hexadecimal without the leading "0x". In our example, you would provide "60". (numeric)
MTU	Maximum Transmission Unit. (numeric) For Ethernet, 1500 is the maximum, and is recommended.
TCP RWIN	TCP Receive Window (numeric) eg. (defaults to 4096 but can be larger)
TCP MSS	TCP Maximum Segment Size (numeric) (usually MTU - 40)

The rest of the details should be greyed out and you need not try to fill them in. The Internal SLIP check box should not be checked.

The first four parameters and the packet vector are required for successful functioning of the winsock, while the rest can be tailored to suit your needs.

When you are done, click on <OK> and if all goes well, the Trumpet Winsock will be initialised. You are now ready to start using the winsock.

What to do if something goes wrong

Firstly...

The Trumpet Winsock requires that your have the correct combination of tcpman.exe, winsock.dll and winpkt.com. When upgrading to a new release, replace each of these files to be sure that everything is up to date.

If you get the messages about not finding a packet driver or unable to load TCP, then check that the packet driver loaded properly, that WINPKT managed to find it, and that the correct vector was chosen from tcpman.

At the moment, only Ethernet and SLIP packet driver types are supported.
Token ring is only available via the ibmtoken packet driver, and should work, but is untested by the author.

ODI can be used via the ODIPKT shim, and NDIS via the DIS_PKT shim. examples are provided later on. Also, examples are provided of installation using NetWare.

Possible causes for tcpman load errors specific to packet drivers.

unable to bind protocol 0806	another TCP stack is using the packet driver... remove it.
WINPKT or pktdrv not found	couldn't find the correct packet driver. Also check the vector number in TCPMAN.
unable to allocate network buffers	critical error... try to free up some special driver memory by removing windows device drivers.
network buffers low	not critical but unadvisable... see above

If WINPKT can't load (No packet driver found), check your packet driver vector number. Some drivers may choose a default vector which is not at 0x60. eg. ODIPKT default is 0x69

If you are using ODIPKT and you can't get any response, you probably accessing the wrong protocol. If you have the ARP trace on, you will possibly get "ARP timed out" messages as well. The first parameter of ODIPKT selects the correct protocol. Try adjusting this.

Anything else... contact me. I'll try to figure out what's wrong, but first browse the samples provided.

Installing Trumpet Winsock over Internal SLIP

SLIP is a simple protocol which allows an Async serial connection to send Internet Protocol (IP). You usually need to have access to a server which can understand SLIP. Usually, SLIP is accessed via a phone line, and with the advent of high speed modems, TCP/IP is a reality over a dial-up connection.

The Trumpet Winsock has facilities for managing a SLIP connection as well as the ability to use dialling scripts for logging in and out of your SLIP server.

Actually Installing the Winsock.

Before you do anything, copy the files winsock.dll, tcpman.exe, hosts, services and protocol to a suitable directory.

eg. c:\trumpet

the essential files:

winsock.dll	the guts of TCP/IP driver
tcpman.exe	interface program for setting up the winsock
hosts	list of host names
services	list of Internet services
protocol	list of Internet protocols

Modify the path line in your autoexec.bat to contain a reference to that
directory.

eg. path c:\dos;c:\windows;c:\trumpet

Make sure it is active by rebooting or executing autoexec.bat again.

Now you are ready to start windows. Start it up!!

From windows, start up tcpman. From the file manager, go File/Run, and then "tcpman". If this fails, the path is probably not set up correctly, so fix it. Later on, you can install it as an icon to use it directly.

Assuming you are a first time user, a setup screen will appear giving you a number of options to fill in. You will need to fill in a few details to enable the TCP package to function. Fill in the following details. If you are unclear on any of them, try to seek some help from qualified Internet support staff - it will save you a lot of time.

Firstly, click on Internal SLIP. Some of the parameters will be greyed and others ungreyed.

IP address	your Internet IP address or "bootp". lower case only. Only use BOOTP if you are not intending to use a dial in script. If using a dialler script with the address extracted by the script, or BOOTP later, just leave it with default value of 0.0.0.0 Only use BOOTP if your server supports it, otherwise the winsock will delay for about 15 seconds and the message "Unable to load TCP" will come up.
Name server	your name server IP address for DNS searches. You may provide more than one address by separating the addresses with spaces. (IP addresses only).
Time server	at present unused - future winsock API's may support this. (IP addresses only).
Domain suffix	a space separated list of domain suffixes to be used when resolving names in the DNS system.
MTU	Maximum Transmission Unit. Related to TCP MSS... usually TCP MSS + 40. (Numeric)
TCP RWIN	TCP Receive Window. It is recommended that this value be roughly 3 to 4 times the value of TCP MSS. (Numeric)
TCP MSS	TCP Maximum Segment Size, It is recommended that this be a smallish value when using SLIP - say 512 bytes for SLIP and lower for CSLIP. CSLIP is able to compress data more efficiently when it is less than 255. (numeric)
SLIP port	your comms port number ..1=com1, 2=com2 etc. (numeric)
baud rate	the speed you wish to run at. (numeric)
hardware handshake	recommended if your link supports it.
Van Jacobson CSLIP Compression	if your server will support it. You may also have to adjust MTU, MSS & RWIN. to be suitable.
Online Status Detection	if your modem will support it, select DCD or DSR on-line status detection.

The rest of the details should be greyed out and you need not try to fill them in.

When you are done, click on <OK> and if all goes well, the Trumpet Winsock will be initialised. You are now ready to start using the winsock.

Logging in to the server.

You can use either the manual login or the automatic scripting to access your server. For the time being, choose manual and log into your server with the appropriate commands. Don't forget to use the <esc> key to get out when you have finished dialling in. After logging in, you may need to go and set your IP address if it is allocated dynamically.

If you wish to use another terminal program to dial in to the server, don't forget to issue AT&D0, or disable DTR dropping when exiting the program, or the connection will be severed when the application closes the comms port.

Try out pingw to a well known host IP address to see if all is well.

Problems

Check your baud rates...
If using hardware hand shaking with an external modem, make sure the cable is correctly wired.
At the moment, all dialling must be done with 8bits, no parity. This may not work for you... you will then need to use an external dialler. The next revision will have an extension to the dialler to allow this.

If all else fails... contact me !!

Once you have determined your login sequence, you can set up a login script. A sample script is provided along with a listing of a typical session.

Automatic dialling.

Minimal scripting is supported, and the script commands are

input <timeout> <string>	wait for string received.
output <string>	send string.
display <string>	display string on display.
wait <timeout> { DSR | CTS | RLSD | DCD}	wait for DSR or CTS or RLSD(DCD)
trace (on | off)	useful for debugging scripts
echo (on | off)	defaults to on
password <prompt>	message box for password
username <prompt>	message box for username
address <timeout>	parse IP address
set (DTR | RTS) (on | off)	set/reset the modem lines.
sleep <seconds>	pause for so many seconds.
exec <string>	program will be started up concurrently using winexec().
online	enter SLIP mode. commands depending on received characters will not work

	correctly after this command is issued since the winsock will interpret data as SLIP frames. Useful before an exec command which uses the winsock.
BOOTP	inform the winsock that a BOOTP will be required after the script has finished.

means start comment except inside string

string arguments
\l line feed
\r return
\n cr/lf pair
\f form feed
\t tab
\b backspace
\nnn ASCII value in decimal
\i IP address
\p password
\u username
\c comm port number (as you have configured it)

a sample script is given for logging in to our Xylogics terminal server.

```
output atz\13
input 10 OK\n
#output atd242284\13
output atd241644\13
input 30 CONNECT
input 30 \n
wait 30 dsr
output \13
input 30 username:
output tattam\13
input 30 password:
password Enter your password
output \p\13
input 30 >
output who\13
input 30 >
output slip\13
input 30 Your address is
address 30
input 30 \n
display \n
display Connected. Your IP address is \i.\n
exec pingw 131.217.10.1
```

Here's a log of a typical session. Names have been blanked out for security.

```
Trumpet Winsock Version 1.00 Alpha #18
```

```
Copyright (c) 1993 by Peter R. Tattam
All Rights Reserved.
SLIP ENABLED
Internal SLIP driver COM3 Baud rate = 38400 Hardware
handshaking
My ip = 131.217.8.4 netmask = 255.255.0.0 gateway =
131.217.250.1
Executing script c:\dev\tcpip\winsock\login.cmd
SLIP DISABLED
atz
OK
atd241644
CONNECT 38400


Annex Command Line Interpreter   *   Copyright 1991
Xylogics, Inc.

Checking authorization, Please wait...
Annex username: xxxxxx
Annex password:

Permission granted
University of Tasmania
AARNet Terminal Server

SLIP users:
Use a maximum segment size (MSS) of 209
and a maximum transmission unit (MTU) of 255.

Async AppleTalk users:
Configure MacTCP to use the Computing Centre zone.
        *** Note change in procedures for starting
async AppleTalk.
        *** After typing atalk you will be prompted
for your password
        *** again.
AARNET TS5 >who
Port What User             Location          When
Idle  Address
 2   CLI  xxxxxxxx         ---               8:01pm
[local]
      +1    'telnet tasman.cc'
 3   CLI  xxxxxxxx         ---               8:15pm
[local]
      +1    'rlogin franklin.cc'
 4   SLIP modem4           ---               8:19pm
ants
 5   CLI  xxxxxxxx         ---               8:34pm
[local]
 6   CLI  xxxxxxxx         ---               7:19pm
[local]
      +1    'rlogin baudin.cc'
 17  SLIP modem18          ---               6:39pm
ants
AARNET TS5 >slip

Switching to SLIP.
```

```
Annex address is 131.217.250.10.  Your address is
131.217.8.5.

Connected.  Your IP address is 131.217.8.5.

Script completed
SLIP ENABLED
```

Dialler problems.

Q. tcpman just pauses when starting up, then gives the message "unable to load TCP".

A. You've probably got BOOTP set. Replace it by 0.0.0.0 before dialling and try again. RARP is impossible to send via SLIP so don't bother with that.

Q. The connection appears to be too slow compared to Xmodem.

A. Possibly the MTU/MSS & RWIN settings are not right. Try to make RWIN about 3 to 4 times MSS and an exact multiple if possible. Turn on the IP trace to see if fragmentation is occurring on TCP connections. If so, then reduce MSS until it stops. UDP packets will still be fragmented, but nothing can be done about that. On the trace, TCP is type 6 while UDP is type 17.

Q. Some input commands in the script don't work.

A. Check for upper case/lower case conflicts. Also check for blanks at the end of the lines.

For other problems, contact me at

trumpet-bugs@petros.psychol.utas.edu.au, or subscribe to the Trumpet discussion group and ask your question. Details are at the end of this document. As time goes on, various FAQ's will be constructed to cope with the more common problems.

Sample Configurations for Packet Driver.

1. Plain ne2000 packet driver using WINPKT.

```
ne2000 0x60 2 0x300
WINPKT 0x60
```

2. Ne2000 packet driver with Novell NetWare access using WINPKT.
Important is the specification of the -n switch of the packet driver. Some packet drivers don't support this switch. In that case, you may be forced to use ODI instead. An example could be the Xircom Pocket Adapter.

```
ne2000 -n 0x60 2 0x300
WINPKT 0x60
pdipx
netx
path c:\dos;c:\network\win31
f:
login
```

3. Ne2000 packet driver with Novell NetWare access using PKTMUX. Notice that WINPKT is not required since PKTMUX does a similar job.

```
ne2000 -n 0x60 2 0x300
pktmux 4
pktdrv
pktdrv
pktdrv
pktdrv
pdipx
netx
path c:\dos;c:\network\win31
f:
login
```

4. ODI setup with NetWare access.

You will need ODIPKT. The latest known release is 2.4 It is important that ODIPKT reference the correct protocol for IP access. This can be specified as the first parameter to ODIPKT (0=1st, 1=2nd and so forth)

Here's a sample of my network attach batch file.

```
@echo off
cd \
lh lsl
lh \odi\ne2000
cd \net
lh ipxodi
lh odipkt
lh WINPKT 0x69
lh netx
path c:\dos;c:\net\win31
```

```
f:
echo on
login
```

Also, your net.cfg must be suitably configured. Here are the relevant excerpts from my net.cfg

```
Link Support
        Buffers 8 1586
        MemPool 16384

Link Driver NE2000
        Port #1 300 20
        Int #1 2
        Frame Ethernet_II
        Frame Ethernet_802.3
        Protocol IPX 0 Ethernet_802.3
```

The ordering of the frame protocols is important for the default setup of ODIPKT. Also, users should be aware that there are two versions of ODIPKT, one released I believe by FTP Software, and the public domain one. I refer to the public domain version. Also note that there are two programs with the same name of "ne2000.com". One is a packet driver and is referred to in an earlier section. The one referred to in this section is actually an ODI driver and won't function as a packet driver at all.

5. NDIS & Windows for Workgroups setup. (courtesy of Peter Whisker, WhiskerP@LGWCT.LOGICA.COM)

Installation of Trumpet Winsock makes use of the DIS_PKT9.DOS or DIS_PKT.DOS shims which provide a Packet Driver interface the NDIS. The version I have tested is found on a number of sites as DISPKT11.ZIP, and contains DIS_PKT.DOS dated 28/4/93. The example is based upon an actual configuration using DEC Pathworks version 4.1 (DECNET version) with a DEPCA Ethernet card. This installation assumes that you have NDIS up and running and have a valid PROTOCOL.INI file.

You need to perform three basic steps:

Load the DIS_PKT driver in CONFIG.SYS following the load of PROTMAN.SYS:

```
DEVICEHIGH=\DECNET\PROTMAN.SYS /I:C:\DECNET
DEVICEHIGH=\DECNET\DEPCA.DOS
DEVICEHIGH=\DECNET\DIS_PKT.DOS
```

Add a few lines to PROTOCOL.INI (here labelled [PKTDRV]) in order to bind it to the Ethernet driver. In this example, DIS_PKT.DOS is configured with packet vector 60.

```
[DEPCA.DOS]
DRIVERNAME = DEPCA$
```

```
.
.
.
[PKTDRV]
  DRIVERNAME = PKTDRV
  BINDINGS = DEPCA.DOS
  INTVEC = 0x60
```

In AUTOEXEC.BAT, load WINPKT or PKTMUX as described in examples 2 and 3 above. This should be done after the NETBIND command in your NDIS startup has been executed. In the case of Pathworks, this command is normally contained in STARTNET.BAT. For example:

```
call \decnet\startnet.bat
WINPKT 0x60
```

6. Some more packet driver installations courtesy of Ashok Aiyar (ashok@biochemistry.bioc.crwu.edu)

Configuration for Cabletron Network Cards. The packet driver provided by Cabletron is a little confusing as it doesn't use the same parameters as packet-drivers that use the Crynwr skeleton.

Typically the Cabletron driver is loaded as:

"csipd_e /s:62 /h:7 /p:300"

In this example the software interrupt is 0x62. Load winpkt.com as

"WINPKT 0x62"

Release 11 of the Crynwr packet drivers includes a driver for Cabletron cards written by Kai Getrost using the Crynwr skeleton that uses the same parameters as the other Crynwr drivers. This driver (CTRONDNI.COM) seems to work well with E1020/1040 and E2020 Cabletron cards. Indeed I see a performance gain over the Cabletron driver. Your mileage may vary.

C/SLIPPER with PKTMUX. Although the Trumpet Winsock has built in support for C/SLIP, there are situations when in addition to Winsock applications there is a need to run packet driver applications simultaneously over a SLIP link. For such situations, PKTMUX is of utility.

Example:

```
CSLIPPER vec=65 com1 irq=04H baud=57600 ether
PKTMUX 4 65 /4  .... (support for  a maximum of 4 virtual
packet drivers)
PKTDRV 60 65
```

Configure the Trumpet Winsock to use the virtual packet driver at 0x60. All other virtual packet drivers

(PKTDRV) can be loaded in the DOS Windows in which they are used. They need not be loaded before entering Windows.

(Ed. Note... You may also require the use of a special comms buffer to enhance the buffering capabilities of Windows when using slipper/cslipper. An FAQ on doing this is available from biochemistry.bioc.cwru.edu via gopher or FTP. It is not needed when using the internal SLIP functions of the Winsock)

Extra Info

You may use environment variables or command line options to override some of the network parameters. They have the same names as the saved parameters in trmpwsk.ini. This file normally resides in the winsock directory rather than the windows directory since this facilitates setting up the winsock in a networked environment. IP addresses can be overridden by using the environment variables, or the command line.

example of command line.

```
tcpman -ip=123.231.213.123 -netmask=255.255.255.0
```

example of environment variable

```
set ip=123.231.213.123
set netmask=255.255.255.0
```

Here's a list of parameters.

ip/myip	your IP address or 'bootp' or 'rarp' (lower case only)
netmask	your netmask. eg. 255.255.0.0
gateway/mygateway	your gateway (IP address)
dns	list of DNS IP addresses
time	list of time server IP addresses
domain	list of domain name suffixes
vector	packet driver vector in hex
MTU	Maximum Transmission Unit
RWIN	TCP Receive Window.
MSS	TCP Maximum Segment Size
slip-enabled	0 = off, 1 = on
slip-port	port number (1-9)
slip-baudrate	baud rate in decimal
slip-handshake	0 = off, 1 = on
slip-compressed	0 = off, 1 = on

The Crynwr packet driver collection

Availability

The Crynwr packet driver collection is available by mail, by FTP, by e-mail, by UUCP and by modem. The drivers are distributed in three files: drivers.zip, which contains executables and documentation, drivers1.zip, which

contains the first half of the .ASM files, and drivers2.zip, which contains the second half of the .ASM files.

Mail:

Columbia University distributes packet drivers by mail. The formats are 9-track 1600 bpi tapes in ANSI, tar, or OS SL format, or PC diskettes (360K 5.25" and 720K 3.5"). The exact terms and conditions have yet to be worked out, please call (212) 854-3703 for ordering information, or write to:

Kermit Distribution, Dept PD
Columbia University Center for Computing Activities
612 West 115th Street
New York, NY 10025

or send e-mail to kermit@watsun.cc.columbia.edu (Internet) or KERMIT@CUVMA (BITNET/EARN).

FTP/e-mail:

The packet driver collection has its own directory devoted to it, pd1:<msdos.pktdrvr>. The drivers are there, along with many free programs that use the packet drivers.

SIMTEL20 files are also available from mirror sites OAK.Oakland.Edu (141.210.10.117), wuarchive.wustl.edu (128.252.135.4), ftp.uu.net (192.48.96.9), nic.funet.fi (128.214.6.100), src.doc.ic.ac.uk (146.169.3.7) or rana.cc.deakin.oz.au (128.184.1.4), or by e-mail through the BITNET/EARN file servers.

Modem:

If you cannot access them via FTP or e-mail, most SIMTEL20 MSDOS files, including the PC-Blue collection, are also available for downloading from Detroit Download Central (313) 885-3956. DDC has multiple lines which support 300/1200/2400/9600/14400 bps (103/212/V22bis/HST/V32bis/V42bis/MNP). This is a subscription system with an average hourly cost of 17 cents. It is also accessable on Telenet via PC Pursuit and on Tymnet via StarLink outdial. New files uploaded to SIMTEL20 are usually available on DDC within 24 hours.

CD-ROM:

Public, private or corporate institutions and libraries interested in the SIMTEL20 MSDOS collection in CD-ROM format bundled with library card-catalog type access and duplication software can contact Coyote Data, Ltd. by

mail at 1142 N. Main, Rochester, MI 48307 or by FAX at (313) 651-4071.

UUCP:

The packet driver files are available from UUNET's 1-900-GOT-SRCS, in uunet!~/systems/msdos/simtel20/pktdrvr. See UUNET.DOC for details.

ODIPKT location

I am told that the originating site for ODIPKT is the following...

Host hsdndev.harvard.edu

Location: /pub/odipkt
FILE -rwxr-xr-x 2915 Aug 21 20:01 odipkt.com

A copy of the NDIS shim is there also.

Trumpet General Discussion Group.

The machine petros.psychol.utas.edu.au is now running a local news service with the news groups

trumpet.announce
trumpet.bugs
trumpet.feedback
& trumpet.questions

If you do not have access directly to this service, these news groups are gatewayed to the following mailing list.

You may join the new Trumpet mailing list by sending a message to

listproc@petros.psychol.utas.edu.au

with just one line in the body.

subscribe trumpet-user Your Full Name

Where "Your Full Name" should be replaced by your actual full name.

The list is called

trumpet-user@petros.psychol.utas.edu.au

and is running on a 486/50 FreeBSD system. Hopefully it will cope ;-)

You can ask questions, or discuss any aspect of any Trumpet program on this group. Feedback is always welcome. There is also an anonymous FTP area with all the latest Trumpet programs and pre-releases. If you do use a pre-release, be prepared for unexpected problems since such programs are in alpha/beta test.

Bugs or Comments

Send to

trumpet-bugs@petros.psychol.utas.edu.au

For bug reports, please send a copy of config.sys, autoexec.bat, trumpwsk.ini, and any other relevant network configurations. In the case of ODI, also send net.cfg. We will do my best to sort out your problem. Due to the high demand for the Trumpet Winsock, my mail box can be overloaded at times. Be patient... someone will answer you.

TCPMAN - The Trumpet Winsock TCP Manager

Menu options.

File/Setup calls up the setup dialog for configuration

IP address	your IP address, "bootp" or "rarp" (lower case). BOOTP will only work if there is a BOOTP service on-line. RARP will only work if using Ethernet, and there is an RARP service on-line.
Netmask	your network mask.
Default gateway	your default Internet gateway or router.
Name server	your Domain Name Server address.
Time server	(unused leave empty)
Domain Suffix	A space separated list of suffixes to be tried when looking up names via the name server.
Packet Vector	for accessing the packet driver in hex.
MTU	Maximum Transmission Unit.
TCP RWIN	TCP Receive Window
TCP MSS	TCP Maximum Segment Size
Demand Load Timeout	Number of seconds tcpman stays loaded after the application has finished with it.
Internal SLIP	Click on this for internal SLIP support & dialler support.
SLIP port	which comms port to use.
Baud Rate	speed of the connection.
Hardware Handshaking	turn on for RTS/CTS handshaking. May require the AT&K3 modem command to function properly.
Van Jacobson CSLIP compression	turn on for CSLIP TCP header compression.
Online Status Detection	needed for dialler autologin / autologout enabling.
None	no online status detection
DCD (RLSD) check	may require AT&C1 modem command to function.
DSR check	may require AT&S1 modem command to function.

File/Register calls up the registration dialog.
File/Exit quits the TCP manager, forcing the winsock to be unloaded.

Edit/Copy copy selected text on tcpman display to the clipboard
Edit/Clear clear the tcpman display

Tracing options. Use with care since some applications may crash when the traces are active. Should a program crash with stack overflow, the winsock may remain loaded in memory even though tcpman has exited. It is advisable to restart windows if this happens and possibly even to reboot your machine. Also, timing measurements of the winsock throughput will be severely affected by the trace options.

Trace/TCP turn TCP trace on/off
Trace/UDP turn UDP trace on/off
Trace/IP turn IP tracing on/off
Trace/ARP turn ARP tracing on/off
Trace/RARP turn RARP tracing on/off
Trace/Ethernet add Ethernet headers to IP/ARP/RARP traces.
Trace/Extra detail add some extra detail to TCP, UDP & IP traces.
Trace/Socket calls trace each winsock call.
most parameters are displayed as well.
Trace/DNS trace Domain Name Server operations.
Use with care, stack overflows can be frequent.
Trace/Messages trace Async Socket messages.

Dialler/Login invoke the login.cmd dialler script.
Dialler/Bye invoke the bye.cmd dialler script.
Dialler/Other invoke other scripts.
a file selection dialog of *.cmd will be
displayed.
Dialler/Options call up the dialler options dialog.

No automatic login
Automatic login on startup only.
Automatic login and logout on demand.

SLIP inactivity timeout (minutes) Number of minutes to wait before exiting winsock. (when no application is using the winsock.) Automatic login & logout must be enabled for this to close the SLIP connection. A value of 0 disables the timeout.

Dialler/Manual Login invoke the dialler manually.
Use <esc> to exit from the manual dialler.
Dialler/Edit Scripts invokes notepad to edit any script.

Help/About Display the version number and copyright.

Registration

Registration of the Trumpet Winsock is encouraged since it funds further development of the winsock. It involves sending in a registration form filled in with your registration name and other details. On receipt of your registration, you will receive a password which will remove the UNREGISTERED VERSION notice and replace it with your registration name. As part of this registration, you will receive enough support to get you going within the existing capabilities of the winsock at the present time, and preference will be given to registered users when it comes to bug fixes or future enhancements to the winsock. Packet drivers using Ethernet and SLIP are presently the only supported network access. The winsock will function through the use of packet driver shims for ODI, NDIS and token ring, but the use of these is not supported, neither is the winsock supported should you be using PKTMUX.

The Trumpet Winsock is currently distributed as shareware. You may use the Trumpet Winsock for 30 days to evaluate its usefulness. If at the end of that time you are satisfied with the Trumpet Winsock as a product, you can register it. The basic registration fee for a single user version of the Trumpet Winsock is US$20. See a later section for details on multi-user site licenses.

Australian users should contact me regarding Australian pricing information and availability.

Cheques or Postal Orders should be made out to

Trumpet Software International

and sent to

Trumpet Software International
GPO Box 1649,
HOBART, TAS AUSTRALIA 7001

You should fill out the following order form and send it along with your cheque or Postal Order to the above address.

FAX/Phone International 61-02-487049, Australia 002-487049

Please note that International mail can be rather slow, and it may take up to 2 months to receive your registration.

```
+-----------------------------------------------+
|              O R D E R   F O R M              |
| for Trumpet Winsock version 1.0 Software      |
+-----------------------------------------------+
```

Ship to: Bill to:

[ ] [
]
[ ] [
]
[ ] [
]
[ ] [
]
[ ] [
]

Please supply the following items:

Licence to use Trumpet Winsock 1.0 for [ ] users
. . . .
. US$[ ]

Tick at least one of the following options.
[ ] 5.25" disk with the latest version of Trumpet Winsock + password
[ ] 3.5" disk with the latest version of Trumpet Winsock + password
[ ] registration password via post
[ ] registration password via e-mail

Your registration name (required) (will appear on program)

[_____]

Your e-mail address (optional - print clearly)
[_____]

Date sent [_____] Expected delivery Date [_____]

Site Licenses

A site license is defined as being a sale to an organisation or company, and may not be resold or redistributed for profit. It may only be used within that organisation.

prices valid until 30-Jun-1994

Single User license

1 user $20 US

Multi-user site license

Trumpet Winsock will be charged by the number of simultaneous users.
The pricing structure for commercial users is thus

1-99 users $20 US per user
100-499 users $2000 US + $10 US per additional user over 100
500-999 users $6000 US + $5 US per additional user over 500
1000+ users $8500 US + $2 US per additional user over 1000

site restriction 10km radius (negotiable)

Unlimited Commercial Site License

$10,000 US for first year.
subsequent years, 25% of unlimited site license fee for that year.
site restriction 100km radius (negotiable)

The pricing structure for educational users is thus

1-100 users $20 US per user
100+ $2000 US

site restriction unlimited.

Your site license will give you support for up to 12 months from the date of purchase. Such support will include upgrades and bug fixes within that 12 months within the constraints of the program's existing capabilities. Future upgrades will be 25% of the original license fee per annum. Arrangements will also be made for conversion of smaller licenses to larger ones.

Should you wish to obtain the Trumpet Winsock to distribute with other programs, you should make a suitable offer to Trumpet Software International, and it will be considered. Source code will not be made available under any circumstances, and Trumpet Software International reserves the right not to accept any offer which is not considered acceptable.

Trumpet Software International
GPO Box 1649,
HOBART, TAS AUSTRALIA 7001

FAX/Phone International 61-02-487049, Australia 002-487049

Microsoft Windows Sockets Support A

This appendix discusses PC-NFS support for the Microsoft Windows Sockets specification. The following information is supplied for developers who wish to use the PC-NFS implementation of Windows Sockets.

Windows Sockets

With this PC-NFS release, SunSelect™ supports the Windows Sockets 1.1 interface, commonly known as WINSOCK. This interface consists of a library of application program interface (API) routines that reside in the file `WINSOCK.DLL`. These routines can be used by programmers who want to create network applications that use sockets. Consult the Windows Sockets specification, which is available from a variety of sources, including the `winsock-request@microdyne.com` alias, for further information about the WINSOCK library.

The sections that follow describe the PC-NFS implementation of Windows Sockets.

General Notes

- SunSelect supports version 1.1 of Windows Sockets.
- The PC-NFS implementation of Windows Sockets uses the default windows sockets `FD_SETSIZE` as it appears in the `winsock.h` file.

- Version 5.0 of `RTM.EXE` and `TKLIB.DLL`, which are part of this release, are necessary to run Windows Sockets applications.
- If you open the maximum number of sockets (16), you cannot subsequently make calls to the `getXbyY` routines. See "rtm – Load Resident Transport Module" on page 120 for information about RTM switches for setting the number of TCP and UDP sockets.

Optional `setsockopt()/getsockopt()` Options

There are `setsockopt()/getsockopt()` options that are optional (see section 2.4 of the Windows Sockets specification). SunSelect supports `SO_RCVBUF`, `SO_SNDBUF` and `SO_DEBUG`.

For `SO_RCVBUF` and `SO_SNDBUF`, the values will default to an increment of 512 bytes.

`SO_DONTROUTE` is not supported, but causes a silent return, as allowed for in the Windows Sockets specification.

Files Needed

The files needed for development include:

```
include/winsock.h
include/netdb.h
include/arpa/inet.h
include/netinet/in.h
include/sys/socket.h
include/sys/time.h
winsock.lib
winsock.nlp
```

These files are installed during the installation of PC-NFS in `\nfs\winsock`.

The files needed to run Windows Sockets applications include:

```
winsock.dll
wshelper.exe
```

The file `wshelper.exe` is necessary for all Windows Sockets applications. It is automatically loaded from within Windows with the first call to `WSAStartup()`, and remains in memory until an exit from Windows.

Run `wshelper.exe` from within Windows with the `/U` flag to remove the helper from memory. The task will unload only if no applications are currently using Windows Sockets.

Users can display debugging information concerning sockets by running `wshelper` from within Windows with the `/D` flag. This will load `wshelper.exe` with a visible window and will display informational debug messages for all sockets.

Implementation Anomalies

`SO_LINGER` and `SO_DONTLINGER` are not handled properly. The current default is to an `SO_LINGER` value of 20 seconds instead of the Windows Sockets default of 0. Setting `SO_LINGER` to 0 will work, but data may be lost. Setting `SO_DONTLINGER` does not work. If a socket is non-blocking, and `SO_LINGER` is non-zero, a `close()` will never return `WSAEWOULDBLOCK`; instead it will block.

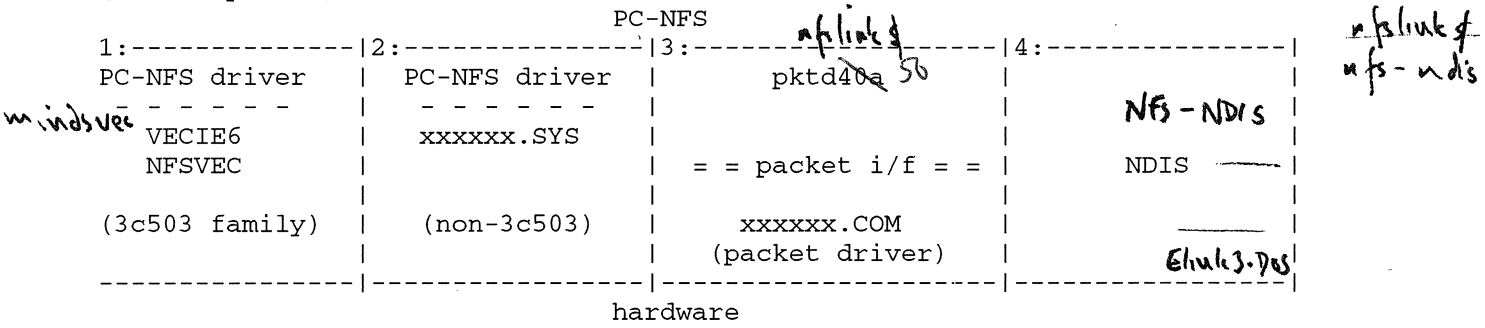
PC-NFS

JLB.

LHB Tech Note 1: Concerning Packet Drivers

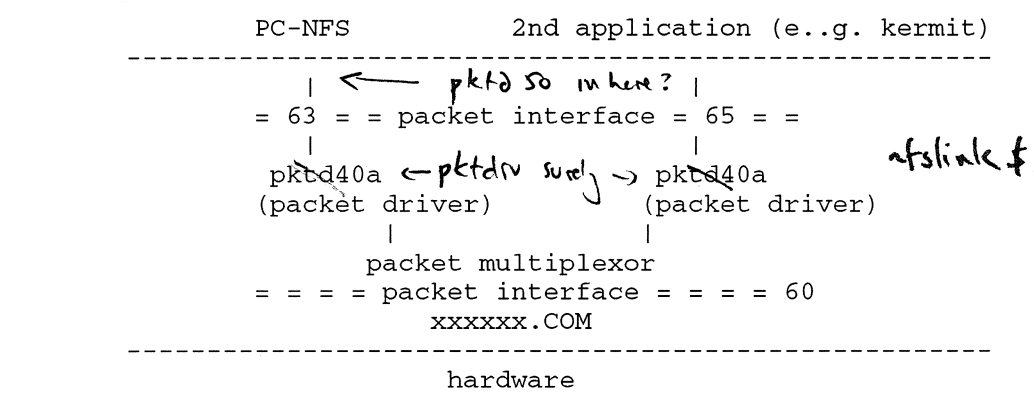
There are a confusing variety of ways of driving Ether cards from applications such as PC-NFS. There are four alternatives of immediate interest:

(Unmultiplexed):



"xxxxxx" will be some name associated with a particular ether card, e.g. wd8003e.sys or wd8003e.com. The 3c503 family is a slightly special case and uses the two components VECIE6 and NFSVEC unless used with a packet driver in which it uses 3c503.COM.

This model will only allow PC-NFS access to the Ether card. If more than one ether-using application must co-exist then a packet multiplexor must be used. This is an extension of case 3: above but instead of one packet interface there are two, one above the other with a multiplexor between them, thus:



pktd40a is a packet driver specific to PC-NFS v4.0a. xxxxxx.COM is the ether driver TSR as before. The drivers and multiplexor have to be loaded and be able to detect each other. Loading is done via lines in AUTOEXEC.BAT and CONFIG.SYS; detection is handled by binding the modules to interrupts as follows (example is 3c503)

In CONFIG.SYS:

```
device=c:\nfs\pktd40a.sys              ; loads the PC-NFS packet driver
```

In AUTOEXEC.BAT:

```
3c503.com 0x60 5 0x300 1               ; loads the 3c503 Ether packet driver -
                                       ;    packet interrupt no = 0x60
                                       ;    h/w int 5, vector 300
                                       ;    cable type 1 (thin)

c:\nfs\pktmux 3                        ; loads the multiplexor with 3 channels

c:\nfs\pktdrv 63 60                    ; load a driver offering int 63 to the
                                       ; application and expecting ether card
                                       ; on packet interrupt 0x60

c:\nfs\pktdrv 65 60                    ; load a 2nd driver on int 65, ditto.
```

NDIS

protman tools for protocol.ini (1i)

PNFS

PKTDRVR
q in whatever

PACKET SHIM = DIS_PKT 9.

PACKET I/F

NDIS I/F

NDIS

PCNFS IP

PCNFS STUB <? NFS-NDS

NDIS

PC NFS 5.0 ref. manual

Example

The following example is a typical AUTOEXEC.BAT file for a PC running PC-NFS software. It assumes that Windows has been installed on drive F in the \WINDOWS directory. It also assumes that the PC-NFS utilities reside in the \PCNFS\UTILS directory on drive G.

```
PATH=C:\NFS;C:\DOS;C:\BIN
PROMPT $p$g
SET TZ=EST5EDT
SET NFSDRIVE=C
SET NFSPATH=C:\NFS
PRT *
NET INIT
PATH=F:\WINDOWS;C:\NFS;C:\DOS;C:\BIN;G:\PCNFS\UTILS
RTM
SNMP
LISTENER
WIN
```

CONFIG.SYS

The CONFIG.SYS file, typically residing in the root directory, contains configuration information for DOS. It includes the names of device drivers to load, the sizes of DOS data structures, and other information. (For a detailed description of CONFIG.SYS, refer to your DOS documentation.)

File Format

This is a text file that includes one command per line.

What Changes Does PC-NFS Software Make to the File?

When you install PC-NFS, or when you change the network driver used by PC-NFS software, the `install` program reads, edits, and modifies CONFIG.SYS. It adds entries for the PC-NFS device driver (PCNFS.SYS), the socket device driver (SOCKDRV.SYS), and whatever drivers are necessary for the network board or serial port that you selected.

These lines have the following format:

```
DEVICE=C:\NFS\PCNFS.SYS <options>
DEVICE=C:\NFS\SOCKDRV.SYS
DEVICE=C:\NFS\linkdriver.SYS <options>
```

where:

<options> are those options available for use with `PCNFS.SYS` and your network interface driver. The options are described in the next section.

linkdriver is the name of your network interface driver.

The `nfsconf` configuration program also makes changes to `CONFIG.SYS`. In response to user input about the number of drives desired for mounting remote file systems, it adds the following line, which sets the size of the DOS drive table:

`LASTDRIVE=`*n*

where *n* is a letter that specifies a drive name that is three more than the total number of drives desired—PC-NFS printers use those last three drive letters. For example, if you wanted to mount file systems on drives D through M, enter a `LASTDRIVE=` setting of `P:`.

What Changes Can You Make to the File?

You edit `CONFIG.SYS` to change certain PC-NFS configuration options. There are two types of options. Options used with `PCNFS.SYS` control the way the PC-NFS application operates. Options used with *linkdriver*`.SYS` control the way your network interface driver operates. It is only necessary to edit this line if you are configuring drivers from third parties; you should make all other changes using the menus in `nfsconf`.

The following list describes all the options you can use with `PCNFS.SYS`. If you are not sure of the effect of an option, you should save your `CONFIG.SYS` file and change only one option at a time. Each option consists of a slash (`/`) followed by a letter and an optional argument. The case of the letter is unimportant. All numeric arguments are decimal. Add the options to the end of the line containing `PCNFS.SYS`, and use spaces to separate each option.

/b*n*

This option sets the default Internet broadcast address to contain only 1s (if *n*=1) or only 0s (if *n*=0). The default (and the standard) is 1.

/c*x*

This option sets the file name mapping character to *x*. The default is the tilde (~) character. The file name mapping character is used in forming mapped names when PC-NFS software encounters an NFS file name that cannot be represented under DOS. See the *PC-NFS User's Guide* for more information on file names.

/f*n*

This option sets the maximum number of open NFS files to *n*. The default is 8 and the maximum is 127. Note that the FILES= directive in your CONFIG.SYS file applies only to the number of files open on local DOS drives; you must use the /f*n* option to configure the number of NFS files.

/i0

This option disables the reassembly of incoming Internet Protocol (IP) datagram fragments. This option will save about 4 Kbytes of memory, but use it with great caution. Never use this option if your network is gatewayed to remote networks, or if you are using Serial Line Internet Protocol (SLIP). (Note that "0" is a number, not a letter.)

/l*n*

(Note that this is lowercase letter "L," not the digit 1.) This option sets the number of lock-table entries. Each active DOS record lock requires one entry. The default is 16 and the maximum is 64. Specifying *n*=0 disables the use of file locking. (This is not recommended.)

/m

This option indicates that you are using a monochrome PC. When you install PC-NFS software, it detects that you have a monochrome system, and the /m option is added to the PCNFS.SYS line. nfsconf and other PC-NFS utilities may check this and adapt their user interface accordingly.

/p*n*

This option sets the User Datagram Protocol (UDP) source port for all NFS operations to *n*. This option should be used by an experienced user for testing purposes only.

/q*n*

This option sets the UDP destination port for all NFS operations to *n*. This is used only in testing with an NFS server that uses a nonstandard port. (Since the value applies to all NFS operations, it is only useful when testing with a single server.)

/r*n*

This option specifies the number of times the PC-NFS application should retry an NFS request, such as reading a file or looking up a name in a directory, before it times out and displays the message: `Abort, Retry, Fail?` The default value is 4.

If you use the `retries=` option with `nfsconf` or `net use` when mounting a resource, whatever value you specify for `retries=` will override the /r*n* option. If you specify 0 for the `/rn` option, the PC-NFS application will retry indefinitely—regardless of what you entered for the `retries=` option in `nfsconf` or `net use`.

/s

This option enables space significance in file names. Include this option for Compaq® DOS and most versions of MS-DOS; omit it for IBM PC-DOS. If you want the PC-NFS application to acknowledge embedded spaces in a file name, use this switch. Otherwise, PC-NFS software will ignore embedded spaces.

/u*n*

This option configures the use of the UDP checksum. If *n* is 0, PC-NFS software does not compute or verify UDP checksums. If *n* is 1, PC-NFS software computes UDP checksums for outgoing packets and verifies UDP checksums for incoming packets, unless an incoming packet does not have

one. If *n* is 2, PC-NFS software behaves the same as if *n*=1, except that it only computes outgoing UDP checksums if the destination is off the network or if the interface is SLIP. The default is 1.

UDP checksums ensure the integrity of your data, but they reduce performance slightly. Change this option only with caution.

`/v`

Normally, the PC-NFS application truncates file and directory names to comply with the DOS 8.3 character format. If you use this option, the PC-NFS application will not truncate file and directory names; instead, it will allow names that exceed the 8.3 character format required by DOS.

`/x`

This option configures PC-NFS software for operation over a serial, point-to-point connection. This option is normally set by the `install` or `nfsconf` program.

`/y`

(Do not use this option if you are running Windows or if you are running multiuser DOS applications that use the INT2F and INT21 5F0X interrupt requests.) This option inhibits the interception of INT2F and INT21 5F0X requests. This option is used to avoid conflicts with other resident software, or with applications that assume that the presence of these services implies the availability of network services not supported by the PC-NFS application.

Note – The `/t` (user time-out) and `/d` (drives) options supported in earlier versions of the PC-NFS product are no longer used.

Different PC-NFS device drivers support different combinations of options. For drivers supplied with PC-NFS software, all configuration options can be set using `nfsconf` or `install`; you should not need to edit the options directly. For drivers supplied by board vendors and others, you should consult the documentation provided with the driver.

The following list describes the options you can use for the device drivers supplied with the PC-NFS application. The values you choose should match jumper or NVRAM (non-volatile RAM) settings on the board for correct operation.

- 3Com 3C503

```
DEVICE=C:\NFS\VECIE6.SYS /i[vector] /p[iobase] /d[dma_chan] /m[n]
    /t[xcvr]
DEVICE=C:\NFS\NFSVEC.SYS
```

where:

`/i[`*vector*`]` is the interrupt vector to be used. The default is interrupt `3`.

`/p[`*iobase*`]` is the base of the I/O registers; the default is `300` (hex).

`/d[`*dma_chan*`]` is the DMA channel to be used; the default is channel `1`.

`/m[`*n*`]` enables shared memory where *n*=`1` is equivalent to the memory address DC00:0, *n*=`2` is equivalent to D800:0, *n*=`3` is equivalent to CC00:0, and *n*=`4` is equivalent to C800:0.

`/t[`*xcvr*`]` selects the transceiver to be used: `1` (the default) selects the on-board transceiver and a thin Ethernet connection, while `2` selects an external transceiver. (Note that for the 3C503 board, the PC-NFS application installs two device drivers: a board-specific vector driver, and `NFSVEC.SYS`, which handles the interface between the PC-NFS application and vector drivers.)

- 3Com 3C523

```
DEVICE=C:\NFS\VEC523.SYS
DEVICE=C:\NFS\NFSVEC.SYS
```

As for most MCA devices, there are no `CONFIG.SYS` options for the 3C523. As in the case of the 3C503, the PC-NFS application uses a board-specific vector driver and `NFSVEC.SYS`.

- 3Com 3C505

```
DEVICE=C:\NFS\PSH.SYS /i[vector] /p[iobase] /d[dma_chan]
DEVICE=C:\NFS\3C505.SYS
```

where:

`/i[`*vector*`]` is the interrupt vector to be used.

`/p[`*iobase*`]` is the base of the I/O registers.

`/d[`*dma_chan*`]` is the DMA channel to be used. The driver `PSH.SYS` expects

to find all three values or none: in the latter case, the defaults are 3, 300, and 1, respectively.

Note that for the 3C505, PC-NFS software installs two device drivers: one which manages the 3C505 board, and 3C505.SYS, which handles the interface between the PC-NFS application and the software that is downloaded onto the board. (At startup time, AUTOEXEC.BAT invokes the program, \LDR.EXE, that reads the file, \LDR.CFG. Then \LDR.CFG directs AUTOEXEC.BAT to load the drivers \NFS\PROA.SYS, \NFS\VECIE3.SYS, and \NFS\3C505DX.SYS.)

- Interlan NI5010

  DEVICE=C:\NFS\NI5010.SYS /i[*vector*] /p[*iobase*]

  where:

  /i[*vector*] is the interrupt vector to be used. The default is interrupt 3.

  /p[*iobase*] is the base of the I/O registers; the default is 300 (hex).

- Ungermann-Bass PC NIC

  DEVICE=C:\NFS\NIC.SYS /i[*vector*] /m[*sh_mem_base*]

  where:

  /i[*vector*] is the interrupt vector to be used. The default is interrupt 2.

  /p[*sh_mem_base*] is the base of the shared memory, expressed as a segment base. The default is D000 hex, corresponding to an address of D000:0000.

- Western Digital WD8003E (and similar boards)

  DEVICE=C:\NFS\WD8003E.SYS /i[*vector*] /p[*iobase*] /m[*sh_mem_base*]

  where:

  /i[*vector*] is the interrupt vector to be used. The default is interrupt 2. (Note that some of these boards are supplied with interrupt 3 as the default, so check your board before using interrupt 2.)

  /p[*iobase*] is the base of the I/O registers; the default is 280 (hex).

/m[*sh_mem_base*] is the base of the shared memory, expressed as a segment base. The default is D000 (hex), corresponding to an address of D000:0000.

- NDIS

```
DEVICE=C:\LANMAN\PROTMAN.SYS [/I:path]
DEVICE=C:\LANMAN\yourlink.SYS
DEVICE=C:\LANMAN\NFS-NDIS.SYS
```

where:

/I:*path* specifies the search path to the PROTOCOL.INI file. The default path is C:\LANMAN.

yourlink is the name of your network interface driver. For example, if you are using a Token Ring driver, *yourlink*.SYS would be IBMTOK.SYS.

For NDIS operation, the PC-NFS application installs the NDIS protocol manager, PROTMAN.SYS, and the PC-NFS/NDIS interface module NFS-NDIS.SYS. You are responsible for editing CONFIG.SYS to load the link level driver of your choice. You also have to create the file \LANMAN\PROTOCOL.INI based on the board vendor's documentation and the sample \LANMAN\PROTOCOL.NFS installed by PC-NFS software. Alternatively, you can run the separate quikndis program.

For more information about using NDIS and quikndis, refer to Chapter 5 of the *PC-NFS Installation Guide.*

- Serial Line IP (SLIP)

```
DEVICE=C:\NFS\SLIP.SYS /pport /bspeed [/m]
```

where:

port is the COM port to use. The PC-NFS application supports only COM1 (the default) or COM2, coded as 1 or 2.

speed is the baud rate of the link, which should be 1200, 2400, 4800, or 9600. The default is 9600. The /m option should be included for a dial-up configuration and omitted for a hard-wired link. (This option is used only by install and nfsconf; it is not processed by SLIP.SYS.)

~src24/nfs50 on castle

PC-NFS 5.0

Requirements:
This requires a gread deal of space, but can be customized for partial installation. You require:

1 Mb disk space for an installation without DOS utilities.
(Dos utilities can be used over the network.)

3.5 Mb disk space - with DOS utilities.

2.5 Mb disk space - full PC-NFS set up for windows.

You need 640 KB usable RAM, if not using Windows, and a minimum of 4 MB Ram if using windows. Dos 3.3 and Windows 3.0 are the earlist versions acceptable.

Installation (Upgrade).
The files you need to upgrade are in:
'/usr/local/dos/public/sh01/nfs50.cmp' on 'pcserver1.ed.ac.uk'.
[These files are exported to PCs I know about. Please let me know if you wish any PCs to be added to the list.]

First mount the files - use drive P:, say.

Then type 'P:' and 'install', and follow the prompts.

For a windows installation, you will also have to do 'P:' and 'winstall' after the initial installation.

Alternatively, install from floppy disks, if you have them.

Patches.
Latest dates and sizes:
348128 May 13 1993 telnetw.exe
- Corrections to windows telnet
40592 Jun 8 1993 wshelper.exe
- For applications which use the 'winsock' interface
25448 May 26 1993 int14.zip
- For applications which require the 'int14' interface.

Please apply patches from
'/usr/local/dos/public/sh01/nfs50.pch' on 'pcserver1.ed.ac.uk'.
[These files are exported to PCs I know about. Please let me know if you wish any PCs to be added to the list.]

Copy 'telnetw.exe' to 'nfs\telnetw\telnetw.exe' and
'wshelper.exe' to 'nfs\wshelper.exe'.

int14 - int14.doc and int14.exe are available if you need them. This is a new facility, not released as part of 5.0

'telnetw.exe. and wshelper.exe' are also available by anonymous ftp from ftp.york.ac.uk, directory pub/pc-nfs/Patches.
'int14.zip' is available by anonymous ftp from ftp.york.ac.uk, directory pub/pc-nfs/int14
'int14.zip' and wshelper.exe' are also available by anonymous ftp from src.doc.ic.ac.uk, directory pub/sun/pc-nfs

Server Installation.
To install a copy of the compressed files on your server, create a directory for the files, mount the installation files (as above) and do 'P:' and 'install -n directory'.
To install a copy of the uncompressed files on your server, to save space on your PC, create a directory for the files, mount the installation files (as above) and do 'P:' and 'install -nu directory'. This actually asks for the NDIS disk - but you can use <ctrl> C to get out at this stage.

Changes.
There is a new Telnet, which has many more commands, and features.
In particular, Telnet has to be killed after logging out.

Windows FTP rewritten. Quite nice user interface.

Problems:
1. Windows - Telnet overwriting colour menu, and funny colours
appearing on screen.
Fixed by installing patch.

2. Windows FTP - occasionally gets timeout errors from which it cannot
recover.

3. Windows 'network information query' facility to look up NIS maps
- map 'netgroup' not in default list.

4. List of servers not very helpful. Would be nice if we could customize
by adding frequently used servers.

Novell version - 16 Apr 93.

1. Loading novell's 'telapi' causes hang if using ODI driver.

2. vtcpip.386 for windows causes trouble (cannot find tcp)

3. Seems to work with either encapsulation. - /b2 not req.

4. Printers should not be mounted during 'net init' - mount
separately after loading.

5. MS-DOS window not working.
Now using 'netx.exe' from netware\drivers\dosup7.zip. Seems
much better.

Otherwise - working using ODI. Using autoexec.015.

Using '3c503' odi driver. Not tried packet driver yet.

TELNET (Windows)

In PC-NFS 5.0 Windows Telnet:

To display hash as pounds, use SETUP menu, DISPLAY.
This also has 'auto wrap', and 132 column mode.

If you want to run more than one session in a single window, you can use SETUP menu, SESSIONS to select multi-session mode. Then you can use Connections menu to connect to other hosts (or more connections to the same host) and to swap between sessions. You can also run other windows, of course.

You can create a special version of telnet to call castle directly, and send your usual user id.
Using the original telnet, use file menu, setup. Select filename 'castle.set' and create it using 'save'. Then use 'setup' menu 'command' to change the default command file to 'castle'.
In directory c:\nfs\telentw, create file castle.ecf containing:
and add at the end:

```
! Changes for castle.ecf
!standard initialisation
@tnwinit.ecf
set host castle
write host "ercc24"
```

Copy the icon to a new one, change the properties to give title 'telnet

castle' and command line with parameter 'castle'.

Other changes you make will be saved in 'castle.set'.

My current 'festival.ecf' contains:

```
!standard initialisation
@tnwinit.ecf
set host festival
wait "login:"
DEFAULTUSER="ercc24"
inquire/status USER "Festival username ("+DEFAULTUSER+") :"
if USER .eqs. "" THEN USER=DEFAULTUSER
write host USER
wait "password:"
inquire/noecho/status PASSWORD "Festival Password for '"+USER+"' :"
write host PASSWORD
```

Windows FTP.
Seems unable to transfer files except from home directory C:.

! Stephen Hayes, Edinburgh University Computing Service.
! S.T.Hayes @ edinburgh.ac.uk, Tel: 031-650 4990.

1,,
Summary-line: 17-Mar ercc24@festival.ed.ac.uk #Re: Packet Drivers, NDIS
Received: from festival.ed.ac.uk by dcs.ed.ac.uk id aa27179; 17 Mar 94 14:28 GMT
From: "S.T.Hayes" <ercc24@festival.ed.ac.uk>
Subject: Re: Packet Drivers, NDIS
To: John Butler <jhb@dcs.ed.ac.uk>
In-Reply-To: John Butler's message of Thu, 17 Mar 1994 10:50:30 +0000
Reply-To: Stephen_Hayes <S.T.Hayes@ed.ac.uk>
Date: Thu, 17 Mar 94 14:21:06 GMT
Message-ID: <9403171421.aa18905@uk.ac.ed.festival>

*** EOOH ***
From: "S.T.Hayes" <ercc24@festival.ed.ac.uk>
Subject: Re: Packet Drivers, NDIS
To: John Butler <jhb@dcs.ed.ac.uk>
In-Reply-To: John Butler's message of Thu, 17 Mar 1994 10:50:30 +0000
Reply-To: Stephen_Hayes <S.T.Hayes@ed.ac.uk>
Date: Thu, 17 Mar 94 14:21:06 GMT

> I've set up configurations with Packet Drivers
> I've set up configurations with PC-NFS + NDIS
> I've never done the two together in the one configuration!
> Do you have an example? I'll get there eventually but one would help!

There are two ways you might wish to do this, namely:

1..Hardware/Packet Driver/NDIS/Pc-NFS

2. Hardware/Ndis Driver/Packet/PC-NFS

My file ~ercc24/cutcp/drivers.doc on 'castle' gives you general information about all this area.

Essentially, you can use 'DIS_PKT9.DOS' to run a packet interface over an NDIS driver (Case 2). As far as I know, it is not possible to run an NDIS interface over a packet driver (Case 1).

Config.sys should contain:

```
DEVICE=C:\NFS\PCNFS.SYS /f20 /s /c^
DEVICE=C:\NFS\SOCKDRV.SYS
LASTDRIVE=V
device=c:\lanman\protman.sys /i:c:\base.004\prot.005
device=c:\lanman.dos\drivers\ethernet\elnkii\elnkii.dos
device=c:\packet\dis_pkt9.dos
DEVICE=C:\packet\pktd50.sys
```

protocol.ini (directory in /i: on protman line) contains:

```
[PROTOCOL MANAGER]
DRIVERNAME = PROTMAN$

[PKTDRV]
DRIVERNAME = PKTDRV$
INTVEC = 0X60        - Packet driver software vector
NOVELL = NO

[ETHERLINK]
; protocol.ini section for the 3Com Etherlink II Adapter Card
; transciever=ONBOARD is the default   - external for thick wire
IOADDRESS = 0X300
INTERRUPT = 5
MAXTRANSMITS = 40
DRIVERNAME = ELNKII$
TRANSCEIVER = ONBOARD
```

Autoexec.bat contains:

```
# version for PC-NFS with multiplexed packet driver, for 'degas', Mar 92.
myname=ercc24
myip=129.215.200.110
# myip=rarp
netmask=255.255.255.0
hardware=wd8003e
ioaddr=280
address=D000
# should have 'ftp=no'
# or 'passfile=' to enable password checking.
ftp=yes
passfile=c:\f265\passfile.001
# ftp=no
# passfile=nul.???
# video emulation mode
video=auto
# enable tektronix emulation
tek=yes
#
name=default
# erase=rubout/backspace...'127' echos as a small triangle!
erase=rubout
nfcolor=YELLOW
nbcolor=black
rfcolor=black
rbcolor=green
ufcolor=CYAN
ubcolor=black
# enable wrap at 80 columns
vtwrap=y
# name=castle
# erase=backspace
# host=castle.edinburgh.ac.uk
# hostip=129.215.200.80
# name=capricorn
# host=capricorn.ucs.ed.ac.uk
# hostip=129.215.200.8
name=cancer
host=cancer.ucs.ed.ac.uk
hostip=129.215.200.7
nameserver=1
# name=ercvax
# host=ercvax.edinburgh.ac.uk
# hostip=129.215.128.30
name=gw-EUCS-Offices
host=gw-EUCS-Offices.ucs.ed.ac.uk
hostip=129.215.200.254
gateway=1
domaintime=1
domainretry=4
domainslist="ucs.ed.ac.uk,ed.ac.uk,ac.uk"
\032
```

```
# version for PC-NFS with multiplexed packet driver, for 'degas', Mar 92.
myname=ercc24
myip=192.41.103.62
# myip=rarp
netmask=255.255.255.0
hardware=wd8003e
ioaddr=320
address=D000
# should have 'ftp=no'
# or 'passfile=' to enable password checking.
ftp=yes
passfile=c:\f265\passfile.001
# ftp=no
# passfile=nul.???
# video emulation mode
video=auto
# enable tektronix emulation
tek=yes
#
name=default
# erase=rubout/backspace...'127' echos as a small triangle!
erase=rubout
nfcolor=YELLOW
nbcolor=black
rfcolor=black
rbcolor=green
ufcolor=CYAN
ubcolor=black
# enable wrap at 80 columns
vtwrap=y
# name=castle
# erase=backspace
# host=castle.edinburgh.ac.uk
# hostip=129.215.200.80
# name=capricorn
# host=capricorn.ucs.ed.ac.uk
# hostip=129.215.200.8
name=cancer
host=cancer.ucs.ed.ac.uk
hostip=129.215.200.7
nameserver=1
# name=ercvax
# host=ercvax.edinburgh.ac.uk
# hostip=129.215.128.30
name=gw-EUCS-Offices
host=gw-EUCS-Offices.ucs.ed.ac.uk
hostip=129.215.200.254
gateway=1
domaintime=1
domainretry=4
domainslist="ucs.ed.ac.uk,ed.ac.uk,ac.uk"
\032
```

```
# version for PC-NFS with multiplexed packet driver, for 'degas', Mar 92.
myname=ercc24
myip=192.41.103.129
# myip=rarp
netmask=255.255.255.0
hardware=packet
ioaddr=0
address=0
# should have 'ftp=no'
# or 'passfile=' to enable password checking.
ftp=yes
passfile=c:\f265\passfile.001
# ftp=no
# passfile=nul.???
# video emulation mode
video=auto
# enable tektronix emulation
tek=yes
#
name=default
# erase=rubout/backspace...'127' echos as a small triangle!
erase=rubout
nfcolor=YELLOW
nbcolor=black
rfcolor=black
rbcolor=green
ufcolor=CYAN
ubcolor=black
# enable wrap at 80 columns
vtwrap=y
# name=castle
# erase=backspace
# host=castle.edinburgh.ac.uk
# hostip=129.215.200.80
# name=capricorn
# host=capricorn.ucs.ed.ac.uk
# hostip=129.215.200.8
name=cancer
host=cancer.ucs.ed.ac.uk
hostip=129.215.200.7
nameserver=1
# name=ercvax
# host=ercvax.edinburgh.ac.uk
# hostip=129.215.128.30
name=gw-EUCS-Offices
host=gw-EUCS-Offices.ucs.ed.ac.uk
hostip=129.215.200.254
gateway=1
domaintime=1
domainretry=4
domainslist="ucs.ed.ac.uk,ed.ac.uk,ac.uk"
\032
```

```
@ECHO OFF
rem file c:\base.004\autoexec.002
rem autoexec.bat for NFS with Windows 3.1, NFS driver for 3c503.
SET TZ=GMT0BST
C:\WINDOWS\SMARTDRV.EXE
PROMPT $p$g
PATH C:\WINDOWS;C:\;c:\dos;c:\nfs;c:\emacs;c:\f265;g:\pdsoft\h56;g:\pdsoft\kermit;c:\bin
SET TEMP=C:\TEMP
MODE CON CODEPAGE PREPARE=((437) C:\DOS\EGA.CPI)
MODE CON CODEPAGE SELECT=437
KEYB UK,,C:\DOS\KEYBOARD.SYS
rem Require 'driver=c:\packet\pktd.sys' in config.sys
rem First load packet driver
rem c:\f265\packet\wd8003e.com 0x60 2 0x280 0xd000
rem Packet driver requires shared memory enabled on 3c503
c:\f265\packet\3c503.com 0x60 5 0x300 1
rem Now mux to allow Clarkson too.../lf listener for ftp
c:\f265\pktmux\pktmux 3
c:\f265\pktmux\pktdrv
c:\f265\pktmux\pktdrv /lf
c:\f265\pktmux\pktdrv
set configtel=c:\f265\config.004
SET NFSDRIVE=C
SET NFSPATH=C:\NFS
C:\NFS\PRT *
C:\NFS\NET INIT
SET TMP=C:\TMP
c:\MOUSE\MOUSE.COM
rem C:\DOS\DOSSHELL
rem Load the rtm for PC-NFS under Windows
c:\nfs\RTM.exe
rem Load listener for Windows - Display until dismissed by 'alt'
c:\nfs\listener.exe -p 0
doskey
```

```
SET TZ=GMT0BST
c:\lanman\netbind
rem Now mux to allow Clarkson too.../lf listener for ftp
c:\f265\pktmux\pktmux 3
c:\f265\pktmux\pktdrv
SET NFSDRIVE=C
SET NFSPATH=C:\NFS
C:\NFS\PRT *
C:\NFS\NET INIT

   pktmux and pktdrv in above are optional.

! Stephen Hayes, Edinburgh University Computing Service.
! S.T.Hayes @ edinburgh.ac.uk, Tel: 031-650 4990.
\037
```

Packet Drivers

from ib.rl.ac.uk (see doc 2)

/* C:\DOC\PKTMUX.DOC **** 16 FEB 94 ** 10:08 ******; G W Robinson

R A L I B M P C U S E R G U I D E

PKTMUX Packet Driver Multiplexor

Contents

1. Introduction and Disclaimer

The author and his employers accept no responsibilty for any damage done by this software. It is run strictly at the user's risk and all necessary precautions, such as backing up of discs, should be taken before hand.

Similarily the vendors/authors of applications and Packet Drivers accept no responsibility for problems and malfunctions, and will give no support, when their software is used with PKTMUX.

This document describes PKTMUX which is a program that provides a multiplexing interface to a Packet Driver. It thus allows several IP protocol stacks to be run in parallel either under DOS or a control program such as Windows 3 or DESQview.

Multiplexing IP protocol stacks is a non trivial problem and this program is only likely to meet about 90% of user requirements. There will always be 10% who need a more sophisticated product. Because PKTMUX makes certain probability assumptions it is also highly likely that it will make a mistake every now and then when they are not valid. At best an application will recover from this situation and at worse something, possibly even the whole PC, will fail. It is therefore likely that PKTMUX will never be 100% reliable and in some situations it may be so flakey as to be unusable.

This documents version 1.2h and differences from previous versions are detailed in Section 11. The programs from this version must not be mixed with those from previous versions as they are incompatible. Many thanks to those around the world who have reported problems and helped with their solution.

2. Installation

The system comes as a single program named PKTMUXxx.EXE where xx is the version number, currently 12. When this program is run it expands into several files. These are:

PKTMUX.EXE	The packet multiplexor
PKTDRV.EXE	The pseudo Packet Driver interface to PKTMUX
PKTSTATS.EXE	A program state and statistics display program
PKTMUX.DOC	This documentation

In addition a free program WINPKT.COM and a source listing WINPKT.ASM is supplied since this provides some of PKTMUX's functionality and is rather smaller.

The programs should be copied into a directory in the Path such as C:\BIN on RAL machines. PKTMUX.DOC should be put in a documentation directory such as C:\DOC on RAL machines. The BAT file INSTALL.BAT does this.

3. Overview

The Packet Driver interface was developed by FTP Software Inc as a standardised way of accessing different makes of communications cards. It is widely used especially over ethernets and a large amount of communications software is available for it. The ethernet implementation makes use of the fact that two bytes in the header define the packet type and this is used to provide a multiplexing mechanism between several packet types. However it looks no further into the protocol stack and thus this feature is of limited use when protocols of the same type, such as those from the TCP/IP family, are used.

PKTMUX attempts to remove this limitation by providing a multiplexing facility for Internet Protocols (IP). This is done by not only switching on the packet type (which denotes IP) but also the IP protocol type (which can denote IP protocols TCP, UDP, ICMP and others). In the case of TCP and UDP it can also switch on the Port number being used. It is therefore possible to run more than one IP protocol stack at the same time. It is not possible to multiplex other protocol stacks or more than one of each IP protocol type other than TCP, UDP or ICMP. One protocol stack can be IEEE 802.3 (ISO 8802/3) protocol.

PKTMUX was originally written to meet the needs of the RAL MOS2 system which is a TSR (Terminate and Stay Resident) program that provides IBM 3270 emulation over asynchronous and ethernet communications. In this context it provides two functions for the TCP/IP version of MOS2. One is that it allows additional communications applications to run alongside MOS2. This is useful because, since MOS2 is a TSR, you can hot key back to DOS and run other commands. Thus an LPR or FTP can be run alongside the MOS2 3270 session.

A second requirement was to allow MOS2 to run under a control program such as Windows 3 or DESQview. This is an instance of a more general problem in running an application which uses a Packet Driver under a control program. It arises because the Packet Driver, which is loaded before Windows, calls the application when it has received a packet. This is quite satisfactory under DOS but under Windows there is no certainty that the application is currently running and there is therefore the risk of jumping into the middle of another program with dire consequences. A Packet Driver option -w gets over this by checking that part of the application program code is present and throwing away the packet if not. This leads to a rather slow data rate as the protocol timeout and retry mechanisms have to be brought into play to recover from the situation. A better solution is provided by the free software WINPKT which only works under Windows 3 Enhanced mode and uses Windows 3 facilities to make sure the application is running. A copy of WINPKT is provided in this package. WINPKT has the drawback that it does not work with other control programs such as DESQview and also with certain ethernet cards such as the BICC 16 bit varient. PKTMUX meets this requirement and gives the additional feature of being able to run communications applications in more than one window.

Given the widespread use of Packet Driver it is surprising that nobody has written a PKTMUX before. It is probably because a general purpose multiplexor is impossible to implement. PKTMUX only attempts to meet the needs of IP protocols and is therefore likely to be of little use in other situations. It must however be emphasised that PKTMUX will only work when the probability of various identifying values being duplicated is low and that when duplication does occur then the various retry mechanisms can recover from the mess. If this is not the case the PKTMUX is of little use.

PKTMUX has so far been tested satisfactorily with PC/TCP, PC-NFS, NCSA, CUTCP, B&W TCP, Waterloo TCP and Novell Netware. It has also been tested with various interfacing shims such ODIPKT, DIS_PKT, PDETHER and the RAL LLCPKT and LLCPKT2 programs that emulate the BICC MPS.

Further technical details are given in the Program and Technical Description sections.

4. Usage Guidelines

The following give basic usage guidelines. For more detail consult the Program Descriptions and Examples sections. All programs except WINPKT give help when run with the /h option.

The normal procedure for running a communications application is to load a Packet Driver and then run the application. To introduce multiplexing into this PKTMUX and PKTDRV must be loaded after the Packet Driver and before the application.

Thus under DOS type the following after you have loaded the Packet Driver:

```
PKTMUX n                ; n is the maximum number of packet
                        ; driver channels to support - default 2
PKTDRV                  ; repeated n times - displays Interrupt
                        ; used
```

Then run the applications as required. Note the first ones must usually be TSRs otherwise you cannot get back to DOS to load subsequent applications. Applications which search for a Packet Driver will find the first free PKTDRV. Once an application starts using PKTDRV it becomes busy and is no longer recognised as a Packet Driver. Note that this only happens once the application has started communicating over the network so each application must be got to this state before further applications are loaded. If not then two applications will link into the same PKTDRV and fail. Applications for which you have to specify the Packet Driver Interrupt should be set for the highest PKTDRV Interrupt to avoid conflicts.

For usage under Windows 3 load PKTMUX and PKTDRV as above making sure there are enough copies of PKTDRV to cater for all applications that will be running concurrently. Spare PKTDRVs cause a minimal overhead. Then load Windows and run the communications applications as required. Note that this is a change to the previous recommendation and results from an improved interface between PKTDRV and the application under Windows 3.

All Windows sessions that use PKTMUX must have PIF settings so that Execution in the Background is enabled and the Background Priority should be at least 50. Windows must be running in Enhanced Mode. The PKTDRV /ne option can be used to ensure that the PKTDRV parameters are correct. An alternative to PKTMUX/PKTDRV is WINPKT if only one Windows communications application is to be run.

Alternatively for applications that run in a DOS session under either Windows or other control programs such as DESQview then only PKMUX should be loaded before the control program. Then run one copy of PKTDRV in any DOS session from which you wish to use a communications application that runs over a Packet Driver. This technique can be mixed with the previous one for Windows 3.

The program PKTSTATS gives details of program states and various statistics.

5. Program Descriptions

5.1 PKTMUX.EXE

This is a TSR that provides, in conjunction with PKTDRV, multiple IP protocol channels to a Packet Driver. Its format is:

```
PKTMUX  chan_cnt pkt_drv_int chan_time_out /options
```

which installs the multiplexor on the first packet driver it finds or hex interrupt "pkt_drv_int" is this is specified (note 1). "Chan_cnt" channels are supported - default is 2 (note 2). "chan_time_out" is the time in seconds a timed out channel waits before being reset (note 6). Any failure during loading will cause PKTMUX to abort and set the DOS ERRORLEVEL to 1. PKTMUX memory usage ranges from 15K for one channel to 34K for eight channels.

The following options modify the action taken:

- a display additional information on loading.
- b look for and use buffer in Packet Driver (note 12).
- d drop packets if application has no buffers (note 5).
- h display this help information.
- i support one IEEE 802.3 channel (note 9).
- o override use of specified interrupt (note 1).
- p pause if error found (note 13).
- q query state of a PKTMUX (note 11).
- r reset timed out channels; rr resets internal flags as well and should be used with care (note 6).
- s silence output except warnings (unless /ss) or errors (unless /sss) (note 10).
- t terminate PKTMUX and Packet Driver (note 3).
- u unload PKTMUX if not being used by a PKTDRV (note 3).
- v set DOS Environment variable PKT_INT to interrupt used or found (notes 1 & 11).
- x multiplex all received packets - testing only (note 7).
- 1 to 9 allocate buffers for this channel count (note 4).

Examples are:

```
pktmux              ; normal use - 2 channels
pktmux 4            ; install to multiplex four applications
pktmux 2 62         ; install on Packet Driver at Int 62
pktmux 4 /b         ; try using Packet Driver buffer
pktmux 4 /bi        ; support IEEE 802.3 and also try and use
                    ; Packet Driver buffer
pktmux /u           ; unload PKTMUX (dont forget to unload
                    ; PKTDRVs first)
pktmux 8 /4         ; install to multiplex eight applications
                    ; but only allocate enough buffers for four
pktmux /r           ; reset timed out channels
```

Notes:

1. By default PKTMUX searches for a Packet Driver and, on finding one, takes over its interrupt effectively hiding it so it cannot be found by any other application. There is then no confusion between the real Packet Driver and the pseudo ones provided by PKTDRV. If the /v option is given then the interrupt used is recorded in the Environment variable PKT_INT. Note that PKTMUX will refuse to load if during its search for a Packet Driver either another PKTMUX or a PKTDRV is found or if one of these is the specified pkt_drv_int. This latter restriction can be overcome by the /o (override) option.

2. By default PKTMUX supports two Packet Driver channels. This means it will communicate with up to two copies of PKTDRV and provide two pseudo Packet Driver interfaces. Each PKTDRV acts, within limits, as if it were a Packet Driver with its own ethernet card albiet with the same MAC address as any others. PKTMUX supports up to eight busy copies of PKTDRV. Note that supporting only one channel achieves the same effect as a normal Packet Driver and is only useful where PKTMUX provides additional functionality such as allowing an application to run under Windows 3.

3. The /u option causes PKTMUX to unload unless a PKTDRV is still busy. The /t option does the same thing but also sends a terminate request to the Packet Driver, which will also unload if it supports this facility.

4. By default PKTMUX allocates a basic set of buffers and then adds additional ones for each channel configured. Tests suggest this is adequate for most cases. However this can be overridden by giving /n where n is a decimal number in the range 0 to 9. PKTMUX will then allocate buffers as if this number of channels were to be used. This can be used to either save memory by reducing the buffer allocation or increasing the number of buffers when experience suggests it is needed. The memory overhead per channel is a little over 2K.

The buffer allocation is shown by PKTMUX when it is loaded and in detail by running PKTSTATS /a after PKTMUX has been loaded.

5. PKTMUX does its best to avoid losing data by holding packets in its buffers when an application is unable to accept them. In some cases this can cause a shortage of buffers. The /d (drop) option causes PKTMUX to behave exactly as a Packet Driver and to drop any packet that is refused by an application because it has insufficient buffer space. This option acts on all channels and overrides a similar option on PKTDRV which works on a per channel basis.

6. When running under Windows 3 or DESQview with a PKTDRV and the application in a DOS session then, if the session is terminated without closing down the application, PKTMUX is left with the channel marked as Busy. It will either be freed after "Chan_time_out" seconds (default infinity) or the option /r (reset) can be used on a call to PKTMUX to reset all such channels. If /rr is given then the busy flags in PKTMUX are also reset which could cause it to fail. See also the PKTDRV /r option and the section on Channel Management in the Technical Description below for further details.

7. By default PKTMUX sends data direct to the application when only one channel is in use. If the /x option is given then packets are copied to a buffer and multiplexed by the normal mechanisms as if several channels were operational. This is therefore a test facility to check if PKTMUX is able to support an application irrespective of any other application that is running. Note that if /x is used with Novell and Windows 3 it can crash Windows.

8. PKTMUX only supports Accesses for Packet Driver Class 1 (DEC/Intel/Xerox "Bluebook" Ethernet). It will not refuse other Classes but the result is undefined and a failure is likely if the same Class is used over more than one channel.

9. If the /i option is given then PKTMUX opens the Packet Driver for all types and does its own filtering on the Type field in order to decide to which handle a packet is destined. This is in order to allow one channel to Access as Packet Driver Class 1 and to accept IEEE 802.3 type packets where the Type field is the data length and is a value of 1500 bytes or less. This dramatically reduces the efficiency of PKTMUX so should only be used when IEEE 802.3 packets are to be used. Unfortunately the overhead remains irrespective of whether such packets are being used but this can be mitigated by the /b option.

Note this is a new facility to evaluate feasibility and may not be very satisfactory. In particular if used with Novell and Windows 3 it can crash Windows.

10. If the /s option is given then PKTMUX does not give out any messages unless an error or warning occurs. The option /ss inhibits any warning messages and /sss stops error messages as well.

11. The /q option causes PKTMUX to return the state of "pkt_drvr_int" in both text form and also via the ERRORLEVEL as follows:

0	No PKTMUX found
1	PKTMUX found

If the "pkt_drvr_int" is not specified then a search is made from Interrupts 60 to 7F looking for PKTMUX. If the /v option is given then the interrupt used is recorded in the Environment variable PKT_INT.

12. The /b option causes PKTMUX to try and locate the data buffer within the Packet Driver so that it can look at the data before taking a copy of the packet. This option is specific to the Packet Drivers of certain ethernet cards and has been tested successfully on the following:

Card	Buffer Size Found in Bytes
BICC 16 bit ISA and MCA versions	64
NE2000	23
WD8003	64
DIS_PKT.xxx	64

Note that if PKTMUX is unable to find the data buffer the option is disabled so it can be used safely in any situation. However if the card you are using is not in the above list then it would be prudent to check that all is well just in case the algorithm fails to locate the correct buffer. If the /x option is given then PKTMUX goes through the motions of trying to locate the buffer but does not use it. Note that PKTSTATS displays the current state of the algorithm and with the /a option gives more details.

The use of the /b option is recommended since it has the potential to significantly improve the performance for PKTMUX.

13. One of the problems in running PKTMUX from a batch file is that if an error occurs then the message often disappears off the screen before it can be noted. The /p option causes output to pause if an error is detected when PKTMUX is first loaded and wait for input from the user before proceeding.

5.2 PKTDRV.EXE

This is a TSR which provides the pseudo Packet Driver interface in conjunction with PKTMUX. Its format is:

```
PKTDRV pkt_drvr_int mux_int all_type /options
```

which uses the multiplexor on hex interrupt "mux_int" or by default searches for it. It installs itself as a Packet Driver on hex interrupt "pkt_drvr_int" or, by default, the first free interrupt after the multiplexor (note 1). The "all_type" parameter (repeatable) defines the packet Type values used when an application asks for all types (eg PC-NFS v3.0). The default setting is 0800, 0806 and 8035 (note 2). Any failure during loading will cause PKTMUX to abort and set the DOS ERRORLEVEL to 1. PKTDRV memory usage is a little over 1K.

The following options modify the action taken:

```
 c   copy send buffers (note 3).
 d   drop packets if application has no buffers (note 5).
 e   extend search area under a control program (note 11).
 f   force PKTDRV to be Free; !f forces to Busy (note 1)
 h   display this help information.
 i   asking for all types gives IEEE 802.3 channel (note 7).
 l   act as listener for any protocol type (note 4).
!l   dont act as listener for any protocol type.
 lf  act as listener for TCP FTP; !lf negates.
 li  act as listener for IP; !li negates.
 lt  act as listener for TCP; !lt negates.
 lt# act as listener for TCP port # (decimal); !lt# negates.
 lu  act as listener for UDP only; !lu negates.
 lu# act as listener for UDP port # (decimal); !lu# negates.
 n   only load PKTDRV if needed ie. there is no free PKTDRV
     already (note 8).
 o   override use of specified interrupt (note 1).
 p  pause if error found (note 12).
 q   query state of a PKTDRV (note 10).
 r   reset a PKTDRV that is busy (note 1).
 s   silence output except warnings (unless /ss) or errors
     (unless /sss) (note 9).
 t   terminate PKTDRV, PKTMUX and Packet Driver (note 6).
 u   unload last loaded PKTDRV if not Busy (note 6).
 uu  unload last loaded PKTDRV even if Busy.
 ur  repeatedly unload last loaded PKTDRV if not Busy.
 v   set DOS Environment variable PKT_INT to interrupt used or
     found (notes 1 & 11).
 w#  wake up using ethernet IRQ # (decimal) (note 13).
```

Examples are:

```
pktdrv                ; normal use
pktdrv /c             ; copy send buffers
pktdrv 66 62          ; multiplexor on 62, put Packet Driver on 66
pktdrv /t             ; terminate PKTDRV, PKTMUX and Packet Driver
pktdrv /u             ; unload last PKTDRV to be loaded
pktdrv /uu            ; unload last PKTDRV to be loaded even if Busy
pktdrv 63 /u          ; unload PKTDRV on Int 63
pktdrv 0 0 800 806 1000 ; Application wanting all packet types will
                          ; just get 0800, 0806 and 1000
pktdrv /!l /lf        ; not a listener for any service except FTP
pktdrv /lt21          ; listener for TCP port 21
pktdrv /w3            ; wake up with ethernet card interrupt IRQ 3
```

Notes:

1. PKTDRV by default uses the next free Interrupt after that used by PKTMUX. It avoids Interrupts 61, 62 and 64, as these are used by PC/TCP, Vista eXceed and Novell respectively, and also 67 as this is the EMS entry point. Interrupts 70 - 76 are normally in use on all but the XT type PCs. If the /v option is given then the interrupt used is recorded in the Environment variable PKT_INT. If the Interrupt is specified it is checked to see if it already in use by PKTDRV, PKTMUX or a Packet Driver and if so PKTDRV will refuse to load unless the /o option is given.

Note that under a control program, such as Windows 3, PKTDRV will, by default, use the same Interrupt in each DOS session it is loaded in. This is because each DOS session has its own version of the Interrupts. For the same reason PKTSTATS will only see the PKTDRV, if any, in its DOS session yet will display the correct total that are busy.

Part of the Packet Driver definition is the use of the string PKT DRVR just after the interrupt entry point in order to identify it. A communications application which does not require the explicit definition of the Packet Driver interrupt searches from Interrupt 60 up to 7f until it finds one. PKTDRV uses the same mechanism and so appears to be a genuine Packet Driver. However once a communications application has accessed PKTDRV the identification is changed and only reverts back when the application has given a release command. Thus several copies of PKTDRV can provide the impression that multiple Packet Drivers are present. The state of each PKTDRV is shown by calling PKTSTATS - Free means it can be used by an application and Busy means its identification has been changed and it is in use.

One problem with this mechanism is when two applications can use the same Packet Driver - for example PC/TCP running alongside a Novell Packet Driver using TYPE 8137 protocol - then doing this over one PKTDRV will fail. One solution is to use separate PKTDRV for each. Alternatively once the first application has been started and the PKTDRV is now Busy then a command of the form:

```
pktdrv pkt_drvr_int /f
```

forces the state back to Free. A second application can then be loaded which finds PKTDRV and can use it.

Note that this is not possible when one application is running under DOS and the other is running under Windows and the PKTDRV remains busy should this be the case.

The opposite problem is when an application is loaded, locates a Free PKTDRV but does not issue any commands and thus change the PKTDRV state to Busy. The call:

```
pktdrv pkt_drvr_int /!f
```

forces the PKTDRV to be Busy thus subsequent applications will not find it.

If a PKTDRV is marked as Busy and the application has terminated (or crashed!) then calling PKTDRV with /r option will reset the specified "pkt_drvr_int" PKTDRV. Note that /r resets the channel whereas /f simply resets the program identification and leaves all the call details intact. If the PKTDRV was also terminated at the same time as the application, such as when a Windows DOS session is terminated, then the channel is Busy but there is no PKTDRV to send a Reset to. In this case PKTMUX will have registered that the PKTDRV has gone and giving the command:

```
pktmux /r
```

will reset any Busy channel for which there is no PKTDRV.

Note that in the PKTDRV calls above if the "pkt_drvr_int" is not specified then a search is made from the last possible PKTDRV Interrupt number (7f) back to the first (60). Thus a generalised use of the feature is possible provided the PKTDRV being used is the last one that was loaded and/or the highest Interrupt number. See also note 11.

The /f, /!f and /r requests are refused if they are directed at a PKTDRV which was loaded under DOS and the PKTDRV issuing them is running in a DOS sessionunder Windows or DESQview. This can be overriden by repeating the option for example /ff, /!f!f or /rr.

A copy of PKTDRV can be loaded at any time. If it is required to unload the system at some later time it is perhaps wise to load all the PKTDRV copies that are required under DOS just after PKTMUX. This minimises the chance of problems with interrupt chains but means that any use of the /f or /r options need the "pkt_drvr_int" specifying.

2. Instead of requesting certain packet types some applications, notably PC-NFS v3.0, ask for all packet types. PKTMUX does not allow this since it makes its job extremely difficult so PKTDRV intercepts such a request and replaces it by specific packet types. By default these are 0800 (IP Protocols), 0806 (ARP - Address Resolution Protocols) and 8035 (RAPR - Reverse ARP) and these are all PC-NFS really needs for its own use. Where further packet types are required these can be overridden and alternatives supplied.

One limitation of this implementation is that an application with a genuine requirement for all packet types cannot be supported. An exception is IEEE 802.3 (ISO 8802/3) over a Packet Driver Class 1 as used by the RAL LLCPKT product and this can be supported via the /i option.

Note that a single PKTDRV will only support a single application when implementing all packet types (such as PC-NFS) or IEEE802.3. If another application is required, such as Novell, then it must use another PKTDRV.

3. Under Windows 3 (and possibly DESQview) some communications cards which use DMA (direct memory access) for sending data don't work properly. The same thing can happen when the application is located in upper memory (ie above 640K). The solution is to copy the data from the application's buffer into one in PKTMUX and send it from there. This is done by the /c option and it only applies to data sent via that PKTDRV. The only card so far found to require this is the BICC 16bit ethernet card though this may be dependent on the PC hardware and the EMM in use.

4. The /l and /!l options indicate whether or not the application using this PKTDRV should act as a listener for well known services. See Technical Description below for further details.

5. The /d (drop) option makes this PKTDRV channel behave as a normal Packet Driver and drop any packet for which the application has no buffer rather than holding it in a buffer which is the default. The PKTMUX option /d implements the same feature for all channels and overrides the PKTDRV setting.

6. The /u option causes PKTDRV to unload unless it is still busy with an application. Adding an /r requests all PKTDRVs be unloaded. Note this may not be possible if other TSRs where loaded between PKTDRVs. If it is known that an application is not Busy, such as when it has crashed, then the /uu option will force the unload. The /t option does the same thing but also sends a terminate request to PKTMUX which, if it is acceptable, will also send a terminate request to the Packet Driver. These requests are refused if the PKTDRV they are directed at was loaded under DOS and the PKTDRV issuing them is running in a DOS session under Windows or DESQview. The /t or /u option can also be used to request the termination of a Packet Driver if PKTMUX is not loaded.

If the "pkt_drvr_int" is not specified then a search is made from the last possible PKTDRV Interrupt number (7f) back to the first (60). Thus a generalised use of this feature is only possible provided the PKTDRV being unloaded is the last one that was loaded and also the highest Interrupt number. See also note 11.

7. If the /i option is given then a request for all types will result in IEEE 802.3 (ISO 8802/3) type packets (ie those 1500 bytes long or less) being routed up this channel. The PKTMUX /i option must also be given for this to work.

Thus implementations of ISO CONS (Pinkbook in the UK academic community) can run over PKTMUX where Access is done using Packet Driver Class 1. UK Academic users of the RAINBOW software who wish to use a Packet Driver alongside to run say PC-NFS or any other TCP/IP stack thus have two options. They can either run the RAL LLCPKT2 product which provides a packet driver interface. This interface can then be either used directly by an application or via PKTMUX for several applications.

Alternatively they can run LLCPKT direct to a PKTDRV with the /i option. Note that the overheads of both methods are significant. However if the /b option is successfully used on PKTMUX then the latter is dramatically more efficient. The LLCPKT program does not make the PKTDRV busy so it is necessary to run PINKBOOK in order to avoid other applications reusing this free PKTDRV.

Note this is a new facility to evaluate feasibility and may not be very satisfactory.

8. The /n option instructs PKTDRV to check either the specified "pkt_drvr_int" or, if this is not given, then Interrupts between 60 to 7f, to see if a PKTDRV is already running and whether it is Free. If this is true it does not load and returns an ERRORLEVEL similar to the /q option as follows:

0	No Free PKTDRV found - PKTDRV loaded ok
2	Free PKTDRV found - PKTDRV not loaded
3	As 2 but Free PKTDRV loaded before Control Program
4	No Free PKTDRV found - PKTDRV load failed for another reason such as no PKTMUX present.

The parameter settings of the free PKTDRV are also set to those of the PKTDRV call except for the /w (ethernet board interrupt) option which cannot be changed from its original setting.

This option allows a BAT file to only load a PKTDRV if it is required and also to reset the parameters of a free PKTDRV to those it requires. See also note 11. Note that the ERRORLEVEL denoting a failure is now 4 instead of 1 as it is when the /n option is not present.

9. If the /s option is given then PKTDRV does not give out any messages unless an error or warning occurs. The option /ss prevents any warning messages and /sss prevents error messages as well.

10. The /q option causes PKTDRV to return the state of "pkt_drvr_int" in both text form and also via the ERRORLEVEL as follows:

0	No PKTDRV found
1	Busy PKTDRV found
2	Free PKTDRV found
3	As 2 but Free PKTDRV loaded before Control Program

If the "pkt_drvr_int" is not specified then a search is made from Interrupts 60 to 7F looking for any PKTDRVs. Reporting a Free PKTDRV takes precedence over a Busy one. See also note 11.

11. The rules for searching for a PKTDRV are modified when running in the DOS session of a control program such as Windows or DESQview. In this case, even if the "pkt_drvr_int" is specified, only the PKTDRV programs loaded in this DOS session are checked. To expand the search to all PKTDRVs, that is to include those loaded before the control program was run, the option /e must be given. Thus it is possible to target the effect of any option to either the current DOS session or all the PKTDRVs in the machine excluding those in other DOS sessions. If the /v option is given then the PKTDRV interrupt is recorded in the Environment variable PKT_INT. These search rules affect options /f, /!f, /n, /q, /r, /t and /u.

12. One of the problems in running PKTDRV from a batch file is that if an error occurs then the message often disappears off the screen before it can be noted. The /p option causes output to pause if an error is detected when PKTDRV is first loaded and wait for input from the user before proceeding.

13. Some applications use the ethernet hardware interrupt to drive their communications code. This gives problems with PKTMUX since when more than one channel is busy any received data is sent to the application after this interrupt has been processed. The outcome is that the application runs satisfactorily on its own but once a second channel becomes busy the application then slows down. The solution is to make the ethernet interrupt wake up PKTDRV by using the /w option. The IRQ value of the ethernet hardware interrupt must be specified in decimal and should be in the range 2 to 15.

This option should thus be used with any application that requires the ethernet hardware IRQ or interrupt to be specified. It may also give improved performance if used for all applications. The only exception is when PKTDRV is running in a Windows or DESQview session when the option is ignored as it would crash the PC.

Note that IRQs map onto the PCs interrupt values as follows:

IRQ (Decimal)	:	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ (Hex)	:	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Interrupt (Hex)	:	71	B	C	D	E	F	70	71	72	73	74	75	76	77
Interrupt (Dec)	:	113	11	12	13	14	15	112	113	114	115	116	117	118	119

Note that IRQ2 is redirected by the PC hardware to IRQ9 and it is the latter that must be linked to for this feature to work in this case. PKTDRV takes care of this by default.

Applications that have been found to need this option are B&W NFS, Wollongong Pathway NFS and PC/TCP v2.11 IDRIVE when used under Windows. Other applications are likely to run faster with this option and it is recommended that /w is always included unless problems occur.

5.3 PKTSTATS.EXE

This program displays program details and statistics from PKTMUX.
Its format is:

```
PKTSTATS /options
```

The following options modify the action taken:

```
a  display further information - can be repeated (note 1).
e  extend PKTDRV search area under a control program (note 5).
h  display this help information.
q  query state of a Packet Driver, PKTMUX and PKTDRV (note 4).
s  silence output from /q option.
v  set DOS Environment variable PKT_INT to interrupt found by
   /q option (note 5).
```

Examples are:

```
pktstats            ; normal use
pktstats /a         ; show further information
pktstats /q         ; query state
pktstats /qe        ; query state under Windows including
                    ; PKTDRV copies loaded under DOS
```

Notes:

1. The option /a can be repeated up to 3 times to give increasing levels of information. This is mainly intended for debugging purposes. Repeating /a 3 times can get the program in a loop or give misleading information since it scans queues which may be changing. This is intended only for cases when PKTMUX has stopped with a system error.

2. The counts given by PKTSTATS are 16 bit integers so will overflow over a period of time. The counts are not read at the same time but are obtained as they are required by PKTSTATS. As PKTMUX is processing data at the same time there may be some inconsistencies in fast moving values such as Broadcast and Ignored counts.

3. In the output from PKTSTATS where the name before a count value is in CAPITAL letters then this indicates that data is being lost or discarded for some reason. Further details are given in the section on Problem Solving.

Each PKTDRV channel may have one or more of the following qualifiers againsts it:

```
/Drop_buff      PKTDRV /d option given
/DV             PKTDRV running under DESQview
/DOS_to_Window  PKTDRV running under DOS but application
                under Windows
/TIMED_OUT      PKTDRV appears to be no longer active.
/TX_Copy        PKTDRV /c option given
/Win            PKTDRV running under Windows
/Zero_type      All Packet Types requested so mapped onto
                specified Types - usually 0806 and 0800.
/802.3          PKTDRV /i option given
```

4. The /q option causes PKTSTATS to search Interrupts 60 to 7f for the state of a Packet Driver, PKTMUX and PKTDRV and report back in both text form (unless /s option given) and also via the ERRORLEVEL as follows:

```
0    No Packet Driver found
1    Packet Driver found
2    PKTMUX and Packet Driver found
3    Busy PKTDRV, PKTMUX and Packet Driver found
4    Free PKTDRV, PKTMUX and Packet Driver found
5    As 4 but Free PKTDRV loaded before Control Program
```

Reporting a Free PKTDRV takes precedence over a Busy one. See also note 5.

5. The rules for searching for a PKTDRV with the /q option are modified when running in the DOS session of a control program such as Windows or DESQview. In this case only the PKTDRV programs loaded in this DOS session are checked. To expand the search to all PKTDRVs, that is to include those loaded before the control program was run, the option /e must be given. Thus it is possible to target the effect of the /q option to either the current DOS session or all the PKTDRVs in the machine excluding those in other DOS sessions. If the /v option is given then the interrupt of whatever is found is recorded in the Environment variable PKT_INT.

5.4 WINPKT.COM

This program is not part of the PKTMUX system but since it provides a subset of the facilities for a smaller memory requirement it is included. In is not supported by RAL. In accordance with the distribution licence the source is also supplied in WINPKT.ASM. The program COPYING mentioned at start up is not supplied as I dont have a copy. WINPKT is also part of the Crynwr (ex Clarkson) Packet Driver collection v10 and may be a later version.

WINPKT acts as an interface between an application running under Windows 3 in Enhanced mode and a Packet Driver. It uses Windows 3 calls so is specific to this case. Its format is:

```
WINPKT new_pkt_drvr_int old_pkt_drvr_int
```

where "old_pkt_drvr_int" is the interrupt of the Packet Driver in either decimal or hex preceeded by 0x. "new_pkt_drvr_int" is the new interrupt to use and cannot be the same as "old_pkt_drvr_int". There are no documented options.

Examples are:

```
winpkt 0x63 0x64        ; Packet driver on Interrupt 64, WINPKT
                        ; accessed via Interrupt 63
winpkt                  ; provide help information
```

Notes:

1. WINPKT should be loaded after the Packet Driver and before Windows 3 is loaded. It is recommended that "new_pkt_drvr_int" is before "old_pkt_drvr_int" since applications that search for a packet driver will find the driver and not WINPKT.

2. WINPKT has no unloading mechanism so if unloading is required the RAL LOADSYS system or similar must be used.

3. WINPKT may not work with certain ethernet cards. The BICC 16 bit card is the only know example so far found. PKTMUX should be used in these cases along with the /c option on PKTDRV.

6. Examples

The following examples illustrate the use of PKTMUX and attempt to show the various possible uses of the system. It assumes a degree of familiarity with the setting up and use of the various systems exampled. Examples include the RAL MOS2 IBM 3270 emulator since this system was one of the reasons for writing PKTMUX. The RAL LOADSYS program to load and unload TSRs and Device Drivers can also be useful in running multiple protocol stacks. Details of both are given in the Reference Section.

It is usually the case that a sequence of commands is put into a BAT file and one section below gives examples of techniques that facilitate this.

The examples in the main give the essential features and omit any frills. The addition of /s or /ss will remove a lot of unnecessary messages and /p will pause if an error occurs. The options /b for PKTMUX and /w for PKTDRV are not included but are strongly recommended since they can give a significant performance improvement. Their addition would give examples of the form:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver for NE2000 card
pktmux /b                ; Support 2 channels and use data buffer
pktdrv /w5               ; PKTDRV driven by the IRQ5 used by the
                         ; Packet Driver
```

In general there are two classes of communications applications. The simplest are those that just need a Packet Driver in order to work, for example the CUTCP programs PING, LPR and FTPBIN. A more complicated type are those applications which require their own TSR to be loaded first. The applications then communicate via this instead of directly to the Packet Driver. Examples are applications that run over PC/TCP and PC-NFS.

Another twist are those applications that either are, or can become, TSRs and thus allow you to return to DOS. Thus further applications can be run. Examples are MOS2 which is a TSR and FTP which via the command ! becomes a TSR and starts up a DOS session.

6.1 Packet Driver, PC/TCP and PC-NFS Applications under DOS

The following illustrates how to run a Packet Driver application alongside those requiring their own TSR to be loaded. The first is for PC/TCP and assumes IPCUST.SYS and IFCUST.SYS are loaded in CONFIG.SYS if they are being used.

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver for NE2000 card
pktmux                   ; Support 2 channels
pktdrv                   ; PKTDRV for PC/TCP to use
pktdrv                   ; PKTDRV for Packet Driver application
                         ; to use
ethdrv                   ; PC/TCP Packet Driver interface
```

You can now run applications from the PC/TCP program suite, such as FTP, PING or LPR. Programs that just require a Packet Driver can also be run such as FTPBIN from the CUTCP program suite.

The following illustrates how to run a Packet Driver application alongside the PC-NFS TSR. It is assumed that SOCKDRV.SYS, PKTD.SYS and ANSI.SYS are loaded in CONFIG.SYS.

```
pcnfs.sys /b1            ; loaded in CONFIG.SYS; /b1 must be set
                         ; at RAL and probably most other sites

mbdndpd 0x63 /I10 /D3    ; Load Packet Driver for BICC 16 bit
                         ; card
pktmux                   ; Support 2 channels
pktdrv                   ; PKTDRV for PC-NFS to use
pktdrv                   ; PKTDRV for Packet Driver application
cd \nfs
prt *                    ; Normal PC-NFS loading
net init
```

PC-NFS applications such as NFSPING can now be run as well as those just requiring a Packet Driver such as FTPBIN from CUTCP.

6.2 MOS2 under DOS

The RAL MOS2 IBM 3270 Emulator v2.3 is a TSR. It supports the Waterloo TCP/IP protocol stack and is normally run by loading the Packet Driver and then running the file MOS2T.BAT. To achieve this using PKTMUX type:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux                   ; Support 2 channels
pktdrv                   ; PKTDRV for MOS2 to use
pktdrv                   ; PKTDRV for applications to use
mos2t                    ; Run MOS2
```

Once MOS2 is running you can then hot key (Alt-Esc) back to DOS and then any other communications application which runs over a Packet Driver can be used. For example PING from the Waterloo TCP/IP suite or FTPBIN or LPR from the CUTCP suite. For example:

```
ftpbin ib.rl.ac.uk       ; establish FTP communications with RAL IBM
```

To run MOS2 alongside applications from the PC/TCP applications suite the following is suggested:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux                   ; Support 2 channels
pktdrv                   ; PKTDRV for PC/TCP to use
pktdrv                   ; PKTDRV for MOS2 to use
ethdrv                   ; PC/TCP Packet Driver interface
mos2t                    ; Run MOS2
```

Once MOS2 is running the PC/TCP applications can be used.

A combination of the two cases, that is the ability to run applications from the PC/TCP program suite or any other application that runs directly over a Packet Driver can be achieved by the following:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux 3                 ; Support 3 channels
pktdrv                   ; PKTDRV for PC/TCP to use
pktdrv                   ; PKTDRV for MOS2 to use
pktdrv                   ; PKTDRV for applications to use
ethdrv                   ; PC/TCP Packet Driver interface
mos2t                    ; Run MOS2
```

Once MOS2 is running either the PC/TCP applications or those requiring a Packet Driver can be used.

Note that it is possible to run MOS2 before loading PC/TCP but this is not recommended. Loading PC/TCP first removes any problems with the provision of well known services such as an FTP listener. If MOS2 must be loaded first then its PKTDRV should have the /!1 option so that any incoming calls for well known services are routed to the next PKTDRV which should be the one for PC/TCP.

6.3 Packet Driver and IEEE 802.3 (ISO 8802/3) applications

The following illustrates how to run an IEEE 802.3 application using a Class 1 access alongside one or more Packet Driver applications. One obviously must be a TSR in order to allow a return to DOS in order to run the other. If the Packet Driver application was a TSR (eg PC-NFS) then the sequence could be:

```
ne2000 0x63 0x5 0x320     ; Load Packet Driver
pktmux /ib                ; Support 2 channels
pktdrv                    ; PKTDRV for application to use
pktdrv /i                 ; PKTDRV for IEEE 802.3 use
```

You then load the Packet Driver application TSR then on returning to DOS run the IEEE 802.3 application. Note that PKTMUX and the PKTDRV used by the IEEE 802.3 application must have a /i option. In fact all PKTDRVs could have the /i option so that the IEEE 802.3 application could use any one. However the exception to this is an application, such as PC-NFS v3.0, which asks for all packet types. Its PKTDRV must not have a /i option.

The /b option on PKTMUX is very strongly recommended for IEEE 802.3 work since, if it works, it should dramatically reduce the overheads.

6.4 Packet Driver and Rainbow Applications

UK academic users of the Rainbow product can run Packet Driver applications alongside by use of a similar sequence to the previous section. The Rainbow software (eg PINKBOOK) must be run over the IEEE 802.3 PKTDRV via the LLCPKT interface provided by RAL. A feature of PINKBOOK means you must not use Interrupt 60 for the Packet Driver. Thus to run it alongside PC-NFS could be done as follows:

```
pcnfs.sys /b1             ; loaded in CONFIG.SYS; /b1 must be set
                          ; at RAL and probably most other sites

mbdndpd 0x63 /I10 /D3     ; Load Packet Driver for BICC 16 bit
                          ; card
pktmux /ib                ; Support 2 channels
pktdrv                    ; PKTDRV for PC-NFS to use
pktdrv /i                 ; PKTDRV for PINKBOOK to use
cd \nfs
prt *                     ; Normal PC-NFS loading
net init

llcpkt                    ; LLCPKT Packet Driver to BICC MPS
                          ; interface
pinkbook                  ; PINKBOOK TSR
rainbow                   ; Rainbow application
```

6.5 Packet Driver Applications under Windows 3

The following illustrates how to run Packet Driver applications under Windows 3 in Enhanced mode.

```
ne2000 0x63 0x5 0x320     ; Load Packet Driver
pktmux 8 /4               ; Support 8 channels but only 4 will
                          ; will be active at once
pktdrv                    ;
pktdrv                    ; PKTDRV's for up to
pktdrv                    ; 4 channels to use
pktdrv                    ;
win                       ; Run Windows 3
```

For a Packet Driver application just open a DOS session and run it. For example the MOS2 emulator BAT file:

```
mos2t
```

To run further applications just open more DOS sessions and run the application up to the limit of PKTMUX channels configured.

There is a slight complication if there is a possibility that there will not be a free PKTDRV. One solution is to load more PKTDRVs than are required. A second is to run the following command before the application is run:

```
pktdrv /ne                ; Load PKTDRV if one not free
```

which loads PKTDRV only if there is not a Free one available in the whole of the PCs memory and not just the DOS session. Note that if the Free PKTDRV was originally loaded with the /w option then a warning message will occur. This can be removed by adding the same /w option to this PKTDRV call or by adding the /ss option to supress the warning message, for example:

```
pktdrv /ness
```

This method is also used if there is a need to set options on the PKTDRV. For example the /d option can improve the operation of TRUMPET so the sequence would be:

```
pktdrv /ned               ; Load PKTDRV if one not free and set /d
news                      ; Run TRUMPET
pktdrv /ne                ; Reset PKTDRV to default options
```

The /d option is set on either the Free PKTDRV or the new PKTDRV if one is loaded. The second PKTDRV call ensures that its options are reset to the correct value for a use by a subsequent application. This is explained more fully in section 6.10 on BAT files.

For IEEE 802.3 applications the PKTMUX call must have the option /i added (and /b is also recommended) and the PKTDRV used should also have a /i. Thus to run the Rainbow application in their Windows DOS session would type:

```
pktdrv /nei             ; Load PKTDRV if one not free and set /i
llcpkt                  ; LLCPKT Packet Driver to BICC MPS
                        ; interface
pinkbook                ; PINKBOOK TSR
rainbow                 ; Rainbow application
```

Where a PIF file is used then the associated BAT file could contains commands similar to above. A little care is needed with error conditions since if there is no free channel then the PKTDRV load will fail. For example:

```
pktdrv /nep
IF ERRORLEVEL 1 GOTO EXIT          ; Jump if failed
.
run application
.
:EXIT
```

If the loading of PKTDRV fails for any reason the /p option holds up processing allowing the user to see the error message generated. The complete BAT file for the MOS2 emulator would therefore be:

```
pktdrv /nep
IF ERRORLEVEL 1 GOTO EXIT          ; Jump if failed
call mos2t
command
:EXIT
```

Be warned that Windows, especially 3.0, can become unstable if you have insufficient memory and opening too many DOS sessions may result in a Unrecoverable Application Error. This usually does no damage and other sessions are unaffected. Note also that, unless the PC is quite powerful, applications may fail as they will not get enough CPU to process their communications in time and protocols may time out. This is especially so if IEEE 802.3 protocols are used.

6.6 Packet Driver, PC/TCP and PC-NFS Applications under Windows 3

This is essentially the same as under DOS but with the application run under a DOS session. For example to run both PC/TCP and Packet Driver applications the following would suffice.

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux 8 /4              ; Support 8 channels but only 4 will
                         ; will be active at once
pktdrv                   ; PKTDRV for PC/TCP to use
pktdrv                   ; PKTDRV for application to use
pktdrv                   ; PKTDRV for application to use
pktdrv                   ; PKTDRV for application to use
ethdrv                   ; PC/TCP Packet Driver interface
win                      ; Run Windows 3
```

For a Packet Driver or a PC/TCP application just open a DOS session and run it. If the PC/TCP NFS implementation (IDRIVE) is being used then the PC/TCP PKTDRV must have a /w option (/w5 in the example).

6.7 Windows 3 Applications

Windows 3 applications are no different from DOS applications unless they need to modify the PKTDRV options or they run over an interface program. For these cases the setting up must be done before Windows is loaded and in such a manner that the correct PKTDRV is used by the application.

For example WINQVT requires the interface program PKTINT to be run and the following would achieve this and support other Packet Driver applications.

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux 8 /4              ; Support 8 channels but only 4 will
                         ; will be active at once
pktdrv                   ; PKTDRV for WINQVT to use
pktdrv                   ; PKTDRV for application to use
pktdrv                   ; PKTDRV for application to use
pktdrv                   ; PKTDRV for application to use
pktint                   ; WINQVT Packet Driver interface
win                      ; Run Windows 3
```

Then run WINQVT as normal with QVT_TCP.RC edited to contain the line:

```
packet_vector=65
```

This is done because WINQVT has to have its Packet Driver interrupt specified and the first PKTDRV will use 65 by default in the above example. If there is any doubt then the PKTDRV could have its interrupt number specified as the first parameter for example:

```
pktdrv 65
```

If WINQVT is not going to be loaded immediately then there is a problem that another application may use the PKTDRV intended for WINQVT since it would the the first one found when searching down the Interrupts. To combat this it is recommended that WINQVT is given a high interrupt number (eg 7F) to guarantee it will be free. For example:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux 8 /4              ; Support 8 channels but only 4 will
                         ; will be active at once
pktdrv                   ; PKTDRV for application to use
pktdrv                   ; PKTDRV for application to use
pktdrv                   ; PKTDRV for application to use
pktdrv 7f                ; PKTDRV for WINQVT to use
pktint                   ; WINQVT Packet Driver interface
win                      ; Run Windows 3
```

Then run WINQVT as normal with QVT_TCP.RC edited to contain the line:

```
packet_vector=7f
```

For Windows Enhanced mode only the PKTDRV and other calls used only by Windows applications can be loaded via the WINSTART.BAT file thus removing them from DOS sessions. Thus in the above example the following lines could be moved to WINSTART.BAT:

```
pktdrv 7f                ; PKTDRV for WINQVT to use
pktint                   ; WINQVT Packet Driver interface
```

To run further Packet Driver applications open a DOS session and run the application as before. To run PC/TCP or PC-NFS applications then modify the above to include the TSR before loading Windows. Alternatively for PC/TCP the TSR can be run under Windows 3 inside a DOS session since it is effectively a Packet Driver application. IPCUST.SYS and IFCUST.SYS, if they are being used, must have previously been loaded in CONFIG.SYS or could be loaded by the RAL LOADSYS system.

6.8 Novell Netware

Novell using a Type 8137 packet can be run in a variety of ways. Note that because PKTMUX hides the real Packet Driver by taking over its interrupt then the Novell Packet Driver interface IPX must use a PKTDRV. For example:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux 2                 ; Support 2 channels
pktdrv                   ; PKTDRV for Novell to use
pktdrv                   ; PKTDRV for application to use
ipx                      ; Load Novell 8137 Packet Driver
                         ; interface
etc
```

An improvement on this is to note that PKTDRV, like a normal Packet Driver, can support several protocol types so it is possible for one PKTDRV to support both Novell and an application. However the difference is that PKTDRV sets itself busy when it is Accessed so it must be made Free again before it can be Accessed again. For example:

```
ne2000 0x63 0x5 0x320    ; Load Packet Driver
pktmux 2                 ; Support 2 channels
pktdrv                   ; PKTDRV for Novell and one
                         ; application to use
ipx                      ; Load Novell 8137 Packet Driver
                         ; interface
pktdrv /f                ; make PKTDRV free again
pktdrv                   ; PKTDRV for second application to use
etc
```

Note that in this case the second application must not be one that uses all packet types (eg PC-NFS v3.0) or IEEE802.3 or be running under Windows. Further note that if the application does any form of filestore redirection such as PC-NFS or IDRIVE from PC/TCP then this must be loaded before IPX is run. This is because although at Packet Driver level there are no problems it appears that only the Netware filestore redirector can cope with another redirector following it in the chain. It must therefore be loaded last.

If the Novell service is using the Standard Netware protocol, that is using a standard IEEE802.3 packet, then the Packet Driver will map this to Type 8137 if the -n option is given. Thus the first line in examples above would become:

```
ne2000 -n 0x63 0x5 0x320 ; Load Packet Driver
```

6.9 Packet Driver Applications under DESQview

The following illustrates how to run Packet Driver applications under DESQview. It is similar to the Windows 3 case but the PKTDRVs are loaded in each DOS session instead of before DESQview is loaded.

```
ne2000 0x63 0x5 0x320     ; Load Packet Driver
pktmux 8 /4               ; Support 8 channels but only 4 will
                          ; will be active at once
dv                        ; Run DESQview
```

To run an application open a DOS session and type:

```
pktdrv                    ; PKTDRV for application to use
```

followed by the application. The application could be the MOS2 emulator BAT file for example:

```
mos2t
```

To run further applications just open more DOS sessions and run PKTDRV then the application.

6.10 Packet Driver/PKTMUX/PKTDRV BAT files

It is frequently the case that a general purpose BAT file has to be written to load communications software. PKTMUX provides various tools to assist in this as are illustrated in the following examples. Note that the /s or /ss option can be added to all the calls to avoid confusing the user with the messages that are generated. It is also recommended that the /p option is added to all calls to load PKTMUX and PKTDRV so that the BAT file pauses if an error is found thus allowing the user to read the message. The examples that follow are for DOS usage and modifications for their use under Windows 3 and DESQview are detailed later in this section.

The following example checks if a Packet Driver or PKTMUX has already been loaded before loading both along with a PKTDRV ready to run a communications application.

```
pktstats /qp                      ; Query state
IF ERRORLEVEL 2 GOTO GOT_PKTMUX   ; Jump if got a PKTMUX
IF ERRORLEVEL 1 GOTO GOT_PACKET   ; Jump if got a Packet Driver
ne2000 0x63 0x5 0x320             ; Load Packet Driver

:GOT_PACKET
pktmux                            ; Load PKTMUX
IF ERRORLEVEL 1 GOTO FAIL         ; Jump if failed

:GOT_PKTMUX
pktdrv /n                         ; Load PKTDRV if none Free
IF ERRORLEVEL 4 GOTO FAIL         ; Jump if failed

run application
```

If the application was one that terminated and then released the Packet Driver, for example LPR or TRUMPET, then the line:

```
pktdrv /u
```

could be added after the application had been run to unload the PKTDRV, if this is possible. This would reduce the memory occupancy and increase the possibility of being able to unload any other TSRs.

If the application required options to be set on PKTDRV, for example /d is recommended for TRUMPET, then the latter part of the BAT file could be:

```
:GOT_PKTMUX
pktdrv /nd                        ; Load PKTDRV if none Free
IF ERRORLEVEL 4 GOTO FAIL         ; Jump if failed
news                              ; run TRUMPET
pktdrv /n                         ; Reset PKTDRV options
pktdrv /u                         ; Unload if possible
```

The second PKTDRV /n call is included in order to reset the options for the case where PKTDRV was already loaded and the PKTDRV /u is unable to unload it. If this is omitted then the options set on the PKTDRV would be used by the next application unless its BAT file included a PKTDRV /n call before the application was run.

Note that in all the above cases the /v option could have been added thus causing the interrupt used to be recorded in the DOS Environment variable PKT_INT. For example the call:

```
pktdrv /nv
```

would set PKT_INT to either the Free interrupt it found or the Interrupt used by the new PKTDRV. Hence an application requiring to know the Interrupt value in its call could be passed this detail.

However if such an application held the interrupt number in a file so that it could not be easily changed in a BAT file then checks for a PKTDRV on this fixed interrupt number, 7f in the example, could be done as follows:

```
pktdrv 7f /n                    ; Load PKTDRV on 7f if not one
                                ; already loaded
IF ERRORLEVEL 4 GOTO FAIL       ; Jump if failed

run application

pktdrv 7f /u                    ; Unload PKTDRV if possible
```

If there was already a Busy PKTDRV on Interrupt 7F then it would not be possible to run the application.

When running under Windows 3 the Packet Driver, PKTMUX and sufficient PKTDRVs should have been loaded before Windows. So only the PKTDRV part of the examples above are relevent. They will all work from a Windows 3 DOS session provided the /q options are made /qe and the PKTDRV /n option is made /ne. This is because without the /e they would only search the DOS session for PKTDRVs and would ignore those loaded before Windows. Thus a sequence to run an application could be of the form:

```
pktdrv /ne
IF ERRORLEVEL 4 GOTO FAIL       ; Jump if failed

run application

pktdrv /usss                    ; Unload PKTDRV if one loaded
```

Note the last line is only needed if it is required to remove a PKTDRV that was loaded in the DOS session (ie there were no Free ones) and will cause an error otherwise hence the /sss to supress the error message.

Again any parameters, such as /d or /lf, which were on the PKTDRV /ne call would be implemented by which ever PKTDRV was used. However it is essential that the PKTDRV options are reset as indicated earlier in this section.

All the DOS examples would also work under DESQview provided certain conditions are met concerning any PKTDRV programs loaded under DOS before DESQview was loaded. They must either be busy, for example being used by PC/TCP or PC-NFS or if they are still free then they must either be made busy or they must use a high interrupt such as 7F. The reason behind all this is to avoid an application in a DOS session searching from Interrupt 60 and finding a free PKTDRV running under DOS rather than in the DOS session itself. Whilst an application will work in this case the response will be a lot slower and the likelyhood of failure much higher. If it can be guaranteed that any Free PKTDRV loaded under DOS has an Interrupt at the end of the range, eg 7F, then the examples above will work.

If not then any Free PKTDRV under DOS must be made Busy to avoid any problems. The following suggests how and uses the /e option to widen the search outside the DOS session and the /v option to note the PKTDRV interrupt number:

```
pktdrv /nv                      ; Load PKTDRV unless one Free
                                ; in this DOS session
IF ERRORLEVEL 4 GOTO FAIL       ; Jump if failed

:TEST_AGAIN                     ; Have a Free PKTDRV in the
                                ; DOS session
pktdrv /qve                     ; Check entire machine
IF ERRORLEVEL 3 GOTO GOT_DOS    ; Jump if Free PKTDRV under DOS
run application                 ; All ok so run application
GOTO EXIT

:FAIL                           ; Failure to load PKTDRV
Echo Cannot load PKTDRV
GOTO EXIT

:GOT_DOS                        ; Free PKTDRV under DOS
pktdrv /e!f!f %PKT_INT%         ; Busy it
GOTO TEST_AGAIN                 ; Test again in case another

:EXIT
```

This BAT file forces to Busy any Free PKTDRV running under DOS with a lower Interrupt number than any in the DOS session so that an application does not use it. Note that this BAT file will run equally happily under DOS provided a PKTDRV at a high interrupt number is not Free.

7. Technical Description

This section describes how PKTMUX goes about its task and is intended for those who wish to understand how the system works and why it has the limitations it has. An understanding of the Packet Driver interface is assumed. The various PKTMUX counts and states detailed below are shown the command:

```
pktstats /a
```

Additional states are given by the option /aa.

7.1 Basic Methodology

In essence the system is very simple. PKTMUX talks to the Packet Driver and receives data packets from it. Each PKTDRV passes all commands from an application onto PKTMUX with the addition of the channel number. PKTDRV also sits on a timer interrupt and asks PKTMUX once per system tick if there are any data packets for this channel and if so then gets them passed over. Optionally the ethernet card interrupt can be used to drive PKTDRV. A PKTDRV is only Busy from when it has been asked by an application to Access a packet Type to when that is Released. In between it remains Free.

Whilst PKTMUX makes every attempt to be efficient it does create a significant overhead when multiplexing between several applications. This is because the Packet Driver interface only tells you it has a packet and does not give a pointer so that you can see if you are interested in its contents. Where only one application is using this packet type then the packet is sent direct to the application. Otherwise it is necessary for PKTMUX to read the packet into its own storage, analyse its contents and then send it when asked by a PKTDRV to the appropriate application(s). Thus every packet received in this manner has to be copied once more than necessary unless devious cunning is employed via the /b option.

7.2 Buffer Strategy

When an application is unable to accept a packet, usually because it has no free buffer space, then PKTMUX keeps the packet in its buffers and every timer tick keeps asking the application to accept it. It does this even when it normally sends data direct to the application, thus those applications which operate on a small buffer pool may lose less data when under PKTMUX especially when they receive less CPU cycles when running under Windows 3. The decision whether to keep a packet when an application is unable to accept it is a difficult one and depends on the available buffer pool and activity on other channels. The ultimate criterion is the age of the buffer and after between 2 and 3 seconds it is dropped. Whilst this mechanism is satisfactory for most applications there are some that give problems. One that has been noted is TRUMPET which, when an interaction has been completed and it is waiting for user input, refuses to accept any more packets. Whilst PKTMUX's buffer strategy will cope in normal circumstances, under heavy loading this could give problems. The /d option is therefore available on PKTDRV which causes a packet to be always dropped when an application is unable to accept it. Thus it behaves exactly as a Packet Driver. The /d option is also available on PKTMUX to provide this feature on all channels and could be used in cases of extreme loading.

7.3 Channel Management

Normally a channel is freed when the application Releases all the packet TYPEs is has Accessed. If this does not occur, usually because the application has crashed, then there are two possible cases.

The first, and most normal, is where the PKTDRV being used by the application is still running and a call the PKTSTATS shows it to be Busy. The command:

```
pktdrv /r
```

will reset the PKTDRV and free the channel.

The second case is when the PKTDRV is not running. PKTMUX detects this by the absense of any timer interrupts and frees the channel after about two seconds. However this technique fails under a control program such as Windows 3 or DESQview since the DOS session can be locked thus preventing the PKTDRV from sending its timer interrupts. An example is when an area is Selected under Windows 3.0 for such actions as cut and paste.

To overcome this PKTMUX does not immediately free such channels when running under a control program but this gives the new problem that it now has no means of knowing the channel can be freed. Such channels are marked as having Timed Out and this is displayed by PKTSTATS. To reset such a channel use the following command:

```
pktmux /r
```

in a DOS session and this will make all such channels free again.

An automatic means of recovery from this situation is provided by the third PKTMUX parameter, Chan_time_out. This is the time in seconds the call stays in a timed out state before being freed automatically. However this should be set with care since if you spend too long on your cut and paste the channel may be freed and your application will fail.

One side effect of an application crashing is that it may leave PKTMUX in one of its internal busy states. This is shown by the PKTSTATS /aa output in the line "Busy Flags". If this occurs then PKTMUX will effectively go to sleep. When this is the case the call

```
pktmux /rr
```

will also reset these flags and PKTMUX may resume working. It may also crash!

The maximum number of Busy channels supported by PKTMUX is 8 but realistically a rather lower number is recommended. This value is permitted in order to mitigate the problems detailed above by preventing timed out channels from blocking other applications. If say only 4 channels are actually going to be required but 8 are specified then to save memory the number of buffers should be reduced by a call such as:

```
pktmux 8 /4
```

7.4 Control Programs

One of the problems with the Packet Driver interface is that when it receives a packet it then calls a routine in the application. This will not fail under DOS but if the application is running under a control program such as Windows 3 or DESQview, where the application can be swopped in and out of the current virtual memory, then there is a need to establish that the current application is the correct one. The PKTDRV program does this by noting the code it is to jump to when a packet is received and checking if that is present. This works satisfactorily unless two copies of the same application are running and in this case the application has to be tagged in a unique way.

Note that there are two ways of using PKTDRV under a control program. Under Windows 3 the preferred way is to load the PKTDRV before Windows. Because it has special code to interface to Windows the PKTDRV program will request that the correct Windows session is scheduled so that the received packet can be transferred.

Under other control programs such as DESQview the preferred way is to load PKTDRV in a DOS session after the control program has been started and then run the application. In this way when PKTDRV asks PKTMUX if it has received a buffer it can be sure the application is in memory and so minimises any delay.

Alternatively when an application runs directly under the control program it is therefore not possible to load PKTDRV in the same virtual memory so it has to be loaded under DOS before the control program. It therefore has no certainty that the application is running when it asks PKTMUX if any data has been received so sometimes has to wait until its timer interrupt coincides with the time slice of the application. Except for the case of Windows 3 this can slow things down considerably and may require PKTMUX to have a larger buffer pool in order to cope.

7.5 Listeners and /l Options

One of the problems PKTMUX has to solve is to which application to route packets for which it has no previous knowledge. Examples are ARP requests and calls to previously unused TCP or UDP ports. Experience has shown it is best to send ARP requests to all applications. For TCP and UDP ports the requests can be either for well know services (such as TELNET or FTP) in which case the default is to send it to the first application to sign up for that packet type in the hope that it has a listener for this service. This avoids more than one response to such a request.

Alternatively it may be to a port previously notified by the application (an example is FTP) and in this case it is sent to all applications who use that packet type. This latter technique works provided the application is tolerant of such unsolicited messages. Tests so far indicate that PC-NFS and CUTCP dont mind. However PC/TCP, Waterloo TCP and WINQVT are more strict and send back an error message. For TCP this is a RST (reset) and for UDP it is an ICMP Port Undefined message. Since this upsets the service being used such cases are trapped and the error message filtered out. Note that the WINQVT log may report "Packet received for invalid port - reset sent". In the PKTSTATS output such packets are counted under "Ignored: err Resp" and under "Ign" in the "PKTDRV channels:" tables. There is currently no way of preventing such unsolicited messages being sent to an application and the /l options only apply to packets for well known ports.

The definition of a well known port is a little vague these days. Originally it was 0-255 but the Unix fraternity officially extended this to 1024 for Unix Standard Services and the current RFC 1106 list goes way beyond this value. As some applications assume the original rule in their port usage so PKTMUX designates ports 0-255 as well known and routes them to only one channel. Any other services outside this range are likely to be provided by a specialist server so sending to all should locate it. Experience seems to support this.

The various /l options on PKTDRV override the default setting for well know services and indicate where such a request should be sent or not. Options of the form /l indicate this application has servers of this type and /!l indicates it does not. The absence of an /l or /!l option means the application provides all servers but it is used only when no other application has an /l option.

The /l option types are implemented as an hierarchy with the specific protocol ports for TCP and UDP taking precedence over the protocols themselves which in turn take precedence over the protocol family (IP). Last in precedence is a general listener. The applications are also searched in reverse order of loading so that a later application can take precedence over an earlier one.

The simplest method of making sure the main services such as FTP are found is to load that application first. Where this is not possible then the PKTDRVs for applications that do not support servers over a protocol should be marked as such by an /!l (ie not a listener) type option so that it is avoided when looking for the default. And any application that wants to take over from the default can be marked by an /l (ie I am a listener) type option. Note that requests will be discarded if no listener is found.

PKTSTATS will display the listener settings for each channel.

7.6 Port Duplication

When an application makes a call to a service it specifies the port to which replies should be sent. How this port number is generated is dependent on the application. There is therefore a possibility that two applications could generate the same reply port number. To combat this PKTMUX inspects all reply port numbers in outgoing packets for TCP and UDP and replaces any new and duplicated number by the next one higher, if that is not in use. It also resets the packet header sumcheck if this is being used. Port numbers on incoming packets are similarily mapped along with ICMP packets containing TCP and UDP. Thus it is possible to run two copies of the same application without any problems of port duplication.

However there are two areas that cannot be fixed. One is where the application specifies the reply port via the protocol. For example FTP usually specifies the port to be used for the file transfer via the PORT command in its controlling data stream. Since PKTMUX does not analyse the data going through it this is not noted. There is therefore the possibility that such a port number may be duplicated if two copies of the same application are run. Fortunately tests with the FTP implementations from PC/TCP and CUTCP have not shown this to be a problem.

The second is where the reply port is a well known port and an example of this is BOOTP. In this case it is assumed that any duplicate use of this port is by an application taking over the function this well known port supports. Thus a BOOTP exchange on one channel will be assumed complete when a second channel uses BOOTP. If this is not the case, as it would be if two applications started up at the same time, then hopefully the BOOTP retry mechanisms will recover the situation.

7.7 IP Fragmentation

IP Fragmentation is a means whereby a large packet is carried through a network whose packet size limit is too small. It is done by simply putting the extra data into the data part of one or more IP packets ie. where you would normally expect the TCP or UDP header to be. The constituent packets of the fragment are linked by having the same IP Identification.

PKTMUX notes the channel that the first fragment is sent to and then routes all further fragments with the same Identifier to that channel. This works satisfactorily for most cases but has some potential problems.

The first is when the fragments arrive out of order. As PKTMUX needs the first packet in order to get the TCP or UDP header out then any fragment that arrives before this packet will be discarded. Such cases are recorded in the NO FRAGMENT count of PKTSTATS output. The protocol retry mechanisms should retransmit the packet and hopefully the first packet of the fragment will arrive first and everything will be ok.

The second is a more difficult problem in that if the same IP Identification is used by two fragmentation sources then PKTMUX has no way of distinquishing between the two. Hopefully this will be very rare and the receiving application(s) should spot that they have the wrong fragments and their retry mechanisms should recover the situation.

Note that when fragmentation is occurring then the number of received and copied fragments (excluding the first) is displayed by PKTSTATS.

7.8 Other IP Protocols

PKTMUX is only able to multiplex on IP protocols it knows about. These currently are TCP, UDP and ICMP. Any other IP protocol type will be handled correctly provided there is only one channel using it. Multiple usage of another IP protocol will therefore fail. Provided any such protocol has a port mechanism of some form it would be technically possible to add support to PKTMUX.

7.9 IEEE802.3 (ISO 8802/3) Protocol Support

Since this protocol changes the meaning of the Type field to be the length, PKTMUX has to ask the Packet Driver to send it all packets and not just those of a certain TYPE. This dramatically increases the number of packets copied into PKTMUX and hence the overhead on the PC. It also increases the response time, especially under control programs such as Windows 3, so can give protocol problems as well. Use of the /b option (see next section) can improve this.

This option is for those IEEE802.3 implementations that Access the Packet Driver as a Class 1. The correct way to support IEEE802.3 is via Class 11 but this is only available on recent Packet Drivers and gives problems as detailed in section 7.12 below.

Users of the RAL LLCPKT2 product which allows the Rainbow software to run alongside a Packet Driver interface should note that PKTMUX now supports the simpler LLCPKT product which, if used with the /b option, can give a more efficient system. Note that the Rainbow PINKBOOK program uses Interrupt 60 so this should not be used for the Packet Driver.

7.10 Use of Packet Driver Internal Buffer

One of the limitations of the Packet Driver interface is that when it receives a packet it then calls a routine in the application asking for a buffer of a certain size. Thus PKTMUX only knows how big the data is but has no idea of the contents other than that provided via the handle used in the call. The generic Packet Driver code is actually provided with a pointer to the Type field which is also used to access the next byte but unfortunately the registers pointing to this are overwritten before PKTMUX is called.

It is therefore reasonable to assume that somewhere within the Packet Driver program are held the first few bytes of the incoming packet and that they are valid at the point when PKTMUX is asked to provide a buffer in order to read this packet. The /b option instructs PKTMUX to search the Packet Driver for this buffer, to copy it and then to test it against the data it actually obtains from the Packet Driver. Should a consistent match be found then the Packet Driver buffer is used to filter out those packets that are of no interest and to send those packets for which there is in only one channel direct to the application. This should therefore reduce the copying of data and in the case of IEEE802.3 support can dramatically reduce the overhead. The gain in other cases is

How you locate the buffer is an interesting problem. So far three types of Packet Driver implementations have been found. One, exampled by NE2000, uses a fixed buffer every time so this is fairly easy to locate. A second type uses a buffer pool and this is rather more difficult to track. The BICC 16 bit card uses this technique and experimentation has shown that a segment register always points to the correct area of memory. The third is ODI-PKT which holds the buffer address at a known position in its stack. An algorithm that copes with all these cases has been implemented. Testing with other cards will no doubt require it to be modified. PKTSTATS gives an indication of where the algorithm is and the eventual outcome of the testing.

Although this has proved a very reliable mechanism please note that once again PKTMUX is breaking all the rules and so problems may occur. How effective this option is depends on the size of buffer available. When the available buffer is 64 bytes you get the full benefit and for all cases there should be a significant improvement in performance. Where the buffer is smaller then below 48 bytes the benefits tail off considerably. However if you are using IEEE802.3 then the /b option avoids the copying of all packets not for this address so gives a significant reduction in overheads.

7.11 Novell Protocol Support

PKTMUX has been tested satisfactorily with Novell Netware using their Type 8137 protocol. It has also been tested with the 802.3 varient and this protocol will work provided the -n option is given to the Packet Driver as shown in the examples section. Further details are given in the next section.

7.12 Packet Driver Protocol Filtering

An area fraught with problems is the use of a Packet Driver in a mixed protocol environment. An example would be TCP/IP and Novell Netware using its 802.3 protocol. The following gives details of the filtering algorithm used by the Packet Driver and applies to Versions 9 and 10 of the collection. It has been gleaned by examining the code (mostly HEAD.ASM) provided to standardise the implementation of the Packet Driver and will only apply to drivers that use this code. Drivers from earlier versions of the collection may work differently.

On receipt of a packet the Packet Driver examines the Packet Type field, which is the length field in IEEE 802.3, and checks if it is

If the -n (Novell) option has been given then Class 11 packets are checked to see if the DSAP/NSAP (the two bytes after the Packet Type/Length) is 0xFFFF. If so then this is assumed to be a Novell 802.3 packet and the Class is changed to 1 and the Packet Type is changed to 8137. Class 1 packets are also checked and if a Packet Type of 8137 is found this is changed to 8138. Similarily any packet sent by the Packet Driver that has a Type of 8137 is modified if the the -n option is given by replacing the Packet Type by the length thus making it IEEE802.3 conformant. Thus the -n option dictates the protocol being sent to and expected from the Novell server and in both cases the Novell application is using 8137.

The next test if to search the list of Packet Types that have been Accessed by the application. When such an Access is made the application specifies the Class and which Packet Type it wishes to receive. The Packet Type is either a specific one such as 0800 for IP, 0806 for ARP or 8137 for Novell or it can be for All Types implying that the application wishes to receive all packets of the given Class - whether it actually does so is dependent on whether it was the first Access or not as indicated in the rules below.

1. When an Access is made its Class and Packet Type are added as the next entry in the list unless it is a request for All Types.

2. If a request for All Types is the first Access for that Class it is added as the next entry in the list.

3. If the request for All Types is the second or subsequent Access for a Class then it is put at the end of the list and all subsequent Access entries added before it.

4. When the Packet Type list is searched to match an incoming packet the Class and Packet Type are checked against each entry until either a match is found or an All Types entry for any Class is found. Thus an All Types entry blocks any further entries in the table irrespective of their Class.

The rules have been slightly simplified in that they describe the Class 1 case where the Packet Type is fixed at two bytes. For Class 11 there is the likelyhood of longer Packet Types since this now selects on the DSAP/NSAP and following bytes so complications arise when different lengths are specified. Should the shorter be a subset of a longer Type then the former takes precedence and the latter Access is rejected.

These rules have several interesting consequences for programs, such as PC-NFS v3.0 (actually is the PKTD.SYS v3.0.2 program) and LLCPKT, which ask for All Types, and these are best illustrated by examples. Consider the case of PC-NFS 3.0 loaded first. As it asks for All Types of Class 1 then no other application irrespective of its Class will work alongside it.

The second case is where PC-NFS is not the first application to be loaded. If any Access was for the IP or ARP Packet Types then PC-NFS would not work since it needs to receive these Types. However if the first Access was for the Novell Packet Type 8137 then PC-NFS would run alongside it.

Note that the version of PKTD.SYS for PC-NFS v3.5 and later stopped asking for All Types so the problem effectively goes away and it can be run alongside Novell. But note that PC-NFS must be loaded before Novell for other reasons.

Tests have also been carried out on Version 5 Packet Drivers and suggest that they will accept just one All Types Access or several specific Accesses but will not accept any combination of the two. The official BICC card driver, MBDNDPD, is based on this version. This card's Version 10 driver, named ISOLINK, is probably a much better bet.

8. Performance Tips

PKTMUX has several features to improve its performance but they are not enabled by default because they could cause problems. However in most cases they work very well and where performance is an issue they should at least be tried out.

When only one channel is active PKTMUX and PKTDRV operate in a transparent manner so the performance of one application running over PKTMUX should be the same as when it is running directly over a Packet Driver. Thus any performance testing requires two channels to be active because at this point PKTMUX has to start copying data around, filtering it and routing it to the correct application. Conversely it is only when a second channel becomes active that performance problems become apparent.

The two features most likely to improve performance are the /b option on PKTMUX and the /w option on PKTDRV. The /b reduces the amount of data copying done but note its success if very dependent on the size of the data buffer within the Packet Driver. Some, such as NE2000.COM, are rather small so the improvement is small. A 64 bytes buffer in contrast can potentially remove all the copying. The /w speeds up the response because it activates PKTDRV when data is received by the ethernet card thus ensuring it is passed as soon as possible to the application. It does have the drawback that BAT files may have to be customised to a PC unless a standard interrupt number is used by all ethernet cards.

When running under Windows performance is significantly enhanced if the PKTDRVs are loaded before Windows rather than in a DOS session.

9. Problem Solving

This section attempts to suggest how problems with PKTMUX should be tackled. It is worthwhile reading the meaning of the various options and also the Technical Description above in order to ascertain if your problem and its solution is documented therein. Also the section on Bugs/Features and Problem Programs should be consulted.

One of the biggest difficulties with PKTMUX is sorting out why something is not working properly. To assist in this the utility PKTSTATS is provided which, when used with the /a option, gives details of what PKTMUX is up to and its various counts. Any count whose name is in CAPITAL letters indicates data being lost or discarded because there is a problem and the following attempts to explain what they mean. Note that such counts are usually only displayed when they have a value so their absence indicates all should be well.

The first class of problems is where PKTMUX simply does not work with an application. The first test is to run the application on its own having loaded PKTMUX with the /x option. Normally in this situation PKTMUX would pass data direct to the application but with the /x option (multiplex) it copies data to its buffers and uses its multiplexing facilities thus checking if they can cope with the application. If this fails then the application probably has some quirk that confuses PKTMUX. If it is a standard application that works elsewhere then you may have a networking set up that PKTMUX cannot cope with.

A second test is to add the /c (copy) option to each PKTDRV - PKTSTATS will show /TX_Copy for that channel. This causes it to copy the data sent to the network into its own buffers and this has been known to cure problems related to the use of upper and/or EMS/XMS memory.

Where an application works with PKTMUX as above but not in conjunction with other applications then it is worth trying different combinations and seeing what does and does not work. This may isolate one application as being the problem or show a certain loading order to be the cause. A major cause of problems in this area for programs running under Windows 3, especially DOS ones, is that the session does not have Execution in the Background enabled in its PIF. Other possible reasons are that a listener for a well known port is being usurped by another application (see PKTDRV /l option) or that one application simply prevents any other from running. Check the Bugs/Features and Problem Programs section for any indication of problems. It is also worthwhile checking if anyone else has a similar problem.

Another class of problems is where PKTDRV is marked Busy when it should be Free or there are no Free PKTMUX channels. This is usually due to applications failing in some manner and the means of recovery are described in the Channel Management part of the Technical Description section.

The final, and probably the largest, class of problems is where everything works for a while then things start going wrong. Using PKTSTATS can give an indication of the cause but in general it is only those cases where the problem can be reproduced that a solution can be found with any degree of certainty. Note that PKTMUX depends on probability for its successfull working and when the odds are wrong it will fail for no apparent reason. However for regular failures the suggestions below may help.

If the /b option is in use on PKTMUX it should be removed to see if this is the cause.

If you are running under a control program such as Windows 3 or DESQview then check that the application is getting a sufficient percentage of the processor especially if the application starts failing when running in the background. For Windows 3 a Background Priority of anything less than 50 can lead to problems and check that Execution in the Background is enabled. For Windows 3 applications run faster if their PKTDRV is loaded before Windows. For DESQview the reverse is true. So check whether applications running in a DOS box have their own PKTDRV or are using one loaded before the control program. This is shown by PKTSTATS and the PKTDRV entry at the start should have a /Win or /DV against it. If it has /DOS_to_Window then you are using a PKTDRV loaded before the control program which is correct for Windows 3.

A general technique is to run PKTSTATS /a and note the various counts. Then run the application(s) that cause the failure and subsequently run PKTSTATS /a again and note which counts have increased. This could give a clue about whats going wrong as detailed below.

A possible reason for an application not working properly is that it, or PKTMUX, has run out of buffers with which to receive data. This is especially prevalent under a control program such as Windows where applications do not get enough CPU time to process their data. Details of buffer usage are given in the "Buffers:" table and for the case of PKTMUX running out of buffers the count "PKTMUX NO BUFFER" is given in the "Recv ignored reasons" line. Increasing the number of buffers used via the /1 to /9 options on PKTMUX should solve this one.

Detecting that the application is running out of buffers is more difficult since PKTMUX may not be able to deliver the data for a variety of reasons. This can be isolated by running the application on its own over PKTMUX (without the /x option). As it has only one channel operative PKTMUX just passes all calls directly to the application. Any refusal by an application to supply a buffer causes PKTMUX to copy the data into its own buffers. This is shown in the Copied count on the "Recv total" line and and in the "Recv copy reasons" line. The PKTDRV channel counts also show the copied count for each channel. The only solution is to increase the applications buffers if this is possible.

When PKTMUX has more than one channel Busy, and has to wait to pass received data to an application, this is also recorded in the PKTDRV table. The wait reason is either no buffer available from application (Buff) or, under Windows 3 or DESQview only, the application was not in memory (App). For the Buff case if the /d option is given then the data is discarded and the "Dropped" counts increamented. The App case is especially prevalent for background processes paticularily when a foreground process requires a lot of CPU. When it has to wait PKTMUX has to decide whether to try again later or discard the buffer. The latter is only done when PKTMUX has several channels that are actually moving data at the same time and it has insufficient buffers to meet all their demands. The "LOST" count in the "PKTDRV channels:" table would be incremented in this instance and again increasing the number of PKTMUX and/or application buffers is a possible solution. The count in the "Queues" section "LOST DUE TO APPLICATION ....." sums this total for all channels. It may be that, especially under Windows 3 or DESQview, the PC has simply not enough horsepower to cope with the communications load as well as any processing in progress at the same time. Thus the application(s) are not processing the received data fast enough to cope with the incoming rate. Alternatively it may be an application, such as TRUMPET, that refuses to accept packets when it knows it is not expecting data. Note that the /d option can significantly increase this count since packets are throw away at the first refusal.

Another reason that data may be discarded is when there is no listener for the service that is being requested. The count "NO LISTENER" is incremented in the "Recv ignored reasons" list. There are several possible reasons for this but in general it is because the /1 and /!1 options on PKTDRV calls dont leave a suitable listener. Note that this count will not be incremented if there is a listener available but it does not support the service requested. In this case an application that supports the service must be run with a PKTDRV that routes requests for the service to it by using the /1 or /!1 options.

A final reason for discarding data is when an IP Fragment arrives out of order. If it arrives before the first fragment then PKTMUX has no way of knowing to which channel it belongs and so discards it and increments the count "NO FRAGMENT" in the "Recv ignored reasons" list. The subsequent retry should overcome this problem provided that this time the first fragment arrives first.

10. Bugs/Features and Problem Programs

The following list of situations that need special action. It is based upon limited experience so only covers a few cases at present.

The 16 bit BICC ethernet cards require the /c option on PKTDRV when running under a control program and dont work with WINPKT. The /c is not required when under DOS but is needed when a protocol stack such as PC/TCP is run under DOS for a Windows 3 application such as Vista eXceed X Windows.

When the X Window server Vista eXceed is running over PC/TCP it must have enough buffers allocated via the ETHDRV command otherwise the call will be reset at intervals and thus fail. An ETHDRV call similar to the following is recommended:

```
ethdrv -t 10 -p 20
```

Further details are given the Vista eXceed and PC/TCP manuals. It may also be necessary to increase the PKTMUX buffer allocation when using this or other X Windows servers.

PKTMUX will not work with the packet driver version of Novell IPX if the PKTDRV is using Interrupt 64. This is because Interrupt 64 is a Novell API and so from v1.1 onwards it is not allocated by default.

IEEE802.3 support can give problems with other protocols due to the overheads it imposes. In particular Novell running over Type 8137 protocol has been found to crash in this mode.

PKTMUX tends to operate a lot with interrupts disabled. This may cause problems with time critical communications methods such as asynchronous links using SLIP and ethernet communications via the PCs parallel port.

TRUMPET refuses to accept packets when it is waiting for user input and expects no more data. This can cause PKTMUX to run out of buffers under heavy loading. It is recommended that the /d option be added to the PKTDRV used by TRUMPET. A similar problem as been seen with PC Gopher running over a Packet Driver interface and /d is recommended in this case also.

When PC-NFS is in use alongside PC/TCP then a TSR, such as the MOS2 3270 emulator, is unable to run when the PC/TCP FTP program is waiting for a command. The problem does not occur with the CUTCP FTP so it appears to be related to the wait loop used by the PC/TCP FTP when waiting for a command.

11. Differences in PKTMUX versions

11.1 Version 1.0

First release to prove that the techniques worked. Note this version does not support IP Fragmentation.

11.2 Version 1.0a

PKTMUX now checks that it is loading on top of a real Packet Driver and aborts if it finds its actually a PKTDRV.

11.3 Version 1.1

The programs from this version must not be mixed with those from version 1.0 as they are incompatible.

In searching for a Packet Driver PKTMUX now checks the interrupts to see if PKTMUX or PKTDRV is already loaded and aborts if one is. Similarily if the Packet Driver interrupt is specified this is checked to see if it is a real Packet Driver. This is to prevent multiple loadings of the system. The option /o (override) has been added to PKTMUX to override this restriction.

PKTMUX now starts by default with 2 channels.

PKTDRV options /f and /!f have been added to force a PKTDRV to the Free or Busy state. The PKTSTATS output has been changed to reflect this.

PKTDRV no longer uses Interrupt 64 by default as this clashes with a Novell API.

IP Fragmentation was not supported in previous versions. It is now supported within limitations (see Technical Description).

The buffer management system has been improved especially with regard to discarding unwanted packets. The option /d (drop) has been added to tell PKTMUX to drop all packets for which the application has no buffer rather than keeping them until the application has space. The same option is available on PKTDRV which works on a per channel basis. The number base of buffers has been also been increased in some cases.

A bug in the Packet Driver handle mapping when PC-NFS was in use has been fixed as has one in the area of duplicate handles.

A bug in the mapping of ICMP packets onto channels has been fixed. The bug caused ICMP packets containing IP data to be sent all channels.

PKTMUX v1.0 used a time out mechanism to determine whether a PKTDRV and its application had been forcably terminated under a Windows or DESQview environment. Unfortunately this mechanism was also triggered when the window was Selected under Windows 3.0 for actions such as cut and paste and caused the channel to be closed down. This has been changed in v1.1 so that in these circumstances a channel will not be closed down. The option /r has been added to PKTMUX to reset such channels otherwise they are permanently busy and there is no PKTDRV to reset them. A third parameter has also been added to PKTMUX to reset such channels after a given time.

ARP Request Broadcasts are now sent to all channels. The /la option is therefore no longer needed. The handling of Broadcast packets has also been improved so that only those ARP requests that are not for this address are discarded.

BOOTP did not work for second and subsequent channels because it replies on a well known port and this only went to the first listener. This has now been changed and the response is sent to originator of the BOOTP provided no other channel has done a BOOTP in between. If this occurs then the timeout and retry mechanisms should recover the situation.

The problem solving section has been improved and the /x option (multiplex) added to PKTMUX to assist this process.

11.4 Version 1.2

The maximum number of channels has been increased to 8 in order to improve the flexibility under Windows 3. If a Channel is in a timed out state then further channels can still be opened without reaching the maximum.

The /q, /v, /e and PKTDRV /n options have been added and should enable a BAT file to work out the current state and load programs as required and this is illustrated by examples. Details of the options follows.

The option /n on PKTDRV only loads PKTDRV if it is needed, that is if there is not already a Free one available, and reports the result via the DOS ERRORLEVEL. It also resets the options on a Free PKTDRV.

The option /q has been introduced on PKTDRV, PKTMUX and PKTSTATS in order to query the current state and returns the reply in text and the DOS ERRORLEVEL. PKTSTATS indicates the presence of a Packet Driver, PKTMUX and PKTDRV (Free or Busy). PKTDRV and PKTMUX return the state of their own program.

The /v option causes the DOS Environment variable PKT_INT to be set to the hexadecimal value of the interrupt used or found by PKTDRV, PKTMUX or PKTSTATS when executing a command.

The /e option extends the search under a control program to outside the DOS session and helps in determining whether a PKTDRV is running within the DOS session or not.

The repeatable /s (silent) option has been introduced to reduce the output from PKTMUX, PKTDRV and PKTSTATS. /sss inhibits all output, /ss all but errors and /s lets through warnings as well.

The /b option to reduce packet copying by trying to locate the data in the Packet Driver has been added. This can may give improved performance which in the case of IEEE 802.3 can be dramatic.

Support for a channel using IEEE 802.3 over Packet Driver Class 1 is now included via the /i option on both PKTMUX and PKTDRV. This feature allows PKTMUX to support the RAL LLCPKT product instead of using LLCPKT2 and with the /b option can be very much more efficient. This is a test implementation for evaluation purposes.

A bug in PKTDRV whereby the /d option did not work has been fixed.

A bug in PKTDRV and PKTMUX whereby they could not be run by LOADHIGH when there was a limited amount of upper memory has been fixed. A bug in the correct allocation of buffer memory in such circumstances has also been fixed.

PKTDRV and PKTMUX now release all their file handles so frequent unloading does not cause you to run out.

A bug which caused WINQVT to crash the PC if it was run twice has been cured. This fix should cure problems with any Windows application which uses a DOS TSR (PKTINT in the case of WINQVT) to interface it to a Packet Driver.

The decision criteria for sending a packet direct to an application rather than copying it has been improved.

The algorithm whereby packets held in the buffer queue were dropped after a certain time has been improved. Packets are now held until the application requests them provided there are sufficient free buffers left. The numbers and sizes of buffers has also been revised in the light of experience.

The filtering of broadcasts and especially ARP requests has been improved.

Bugs in the handling of ICMP messages have been fixed.

PKTMUX now copes with features in both NCSA and B&W software where unusual Access commands are made.

PKTDRV actions on another PKTDRV such as /r, /u and /t are now checked under Windows and DESQview to see if they are being done on a PKTDRV that was loaded under DOS and refused if so.

PKTSTATS output now gives more information especially that relevent to the effect of the /b option.

11.5 Version 1.2a

This fixes a bug in the /i (IEEE802.3) code which caused it to crash the PC.

11.5 Version 1.2b

The programs from this version must not be mixed with those from other versions of PKTMUX as they are incompatible. This is now checked for.

The /w# option has been added to support those applications that use the ethernet board IRQ to drive them. Examples are B&W NFS, Wollongong NFS and PC/TCP v2.11 IDRIVE used under Windows, which all run very slowly over PKTMUX unless this option is given on their PKTDRV.

PKTMUX now works over the NDIS packet driver interface (DIS_PKT.GUP) provided by FTP Software Inc.

11.6 Version 1.2c

A bug whereby two applications that shared the same PKTDRV would probably crash the PC when one was closed down has been cured. This would usually have affected the use of Novell Netware using 8137 protocol alongside a TCP/IP application over the same PKTDRV. This bug may also have been the cause of other problems when an application closed down.

Support for Enhanced mode Windows 3 applications which require a PKTDRV to be loaded before Windows 3 has been improved and now uses the same techniques as WINPKT.

11.7 Version 1.2d

A feature of the PKTDRV /w option has been corrected. If the ethernet card used IRQ2 then the /w did not take effect unless it was done on IRQ9 which is where IRQ2 is redirected to. This is now automatically catered for.

The /b option to search for a buffer within a packet driver gave problems with DIS_PKT.xxx programs. This has now been fixed and /b succeeds with both DIS_PKT9.DOS and DIS_PKT.GUP.

A bug in the PKTDRV interface to Windows 3 whereby if an application crashed or was terminated without closing down then the next application to be loaded may also crash has been fixed.

The documentation has been changed, especially sections 4 and 6, to reflect the new recommendation that all PKTDRVs are loaded before Windows 3. This gives a significant performance improvement compared with loading the PKTDRV in the DOS session. Also a section on Performance Tips has been added.

11.7 Version 1.2e

A bug in NCSA software which caused PKTMUX to crash the PC when the /i option was used is now correctly handled.

PKTMUX with the /i option now works over the NDIS packet driver interface (DIS_PKT.GUP) provided by FTP Software Inc.

If two applications share a PKTDRV (eg Novell and a TCP/IP application) then both must be running under DOS otherwise this will crash the machine. A PKTDRV which is being used by a DOS application will be forced to Busy once Windows is loaded even if it had previously been made Free by a PKTDRV /F call.

11.8 Version 1.2f

PKTMUX now passes broadcast ARP reponses to an application instead of filtering them out.

The /b option to search for a buffer within a packet driver now has an improved algorithm and allows the Packet Driver buffer to be outside the main Packet Driver memory block in certain cases.

A bug in PKTDRV which caused it to give out messages when being unloaded despite the /s option has been fixed.

11.9 Version 1.2g

When running under a control program a Channel is always marked as Timed Out when it does not receive any polls from the PKTDRV irrespective of whether the PKTDRV is running under DOS or the Control program. In previous versions the DOS PKTDRV would have had the channel freed. This change is because applications using a communications stack such as PC/TCP or PC-NFS can be stopped, eg by pressing the Pause key, and this sometimes causes the PKTDRV to stop polling and would thus close down the call. A problem with PC Gopher 3 has also been cured by this fix.

11.10 Version 1.2h

There is a feature in PC/TCP v2.3 that causes it to fail to recognise the first PKTDRV it finds. It does however accept the second. PKTMUX now recognises this software and takes appropriate

action.

PKTMUX is now kept at a new location - details are below.

12. Support

PKTMUX is supplied free and is supported, within the limits of its specification, for all users at RAL on IBM PC and PS/2 computers and near clones. Note that support is confined to bugs in the programs and clarification in the documentation of the systems limitations.

Users outside RAL are requested in the first instance to obtain copies and help from their normal support sources.

Academic user support organisations may seek help from RAL but the latter will only be given on a 'best endeavours' basis.

There is no support for other organisations other than by private arrangement with the author.

Updates of the software may be obtained by anonymous FTP from FTP.CC.RL.AC.UK (currently 130.246.12.16). It is held in a self extracting binary file PKTMUXxx.EXE (xx being version number without a point - currently 12) in directory /pub/pcsupp/network/pktmux. Executing the file will produce the program and documentation.

Bug reports or problems should be reported, ideally by email, to Graham Robinson:

E-mail: G.Robinson@RL.AC.UK G W Robinson
Tel UK: 0235 44 5636 Atlas Centre
Int'l : +44 235 44 5636 Rutherford Appleton Laboratory
Fax : 0235 44 6626 Chilton, Didcot, Oxon, OX11 0QX, UK

13. Acknowledgements and References

The techniques used in interfacing a PKTDRV running under DOS to an application running under Windows 3 in Enhanced mode were learnt from the WINPKT code written by Russell Nelson & Roger F James.

The RAL LOADSYS system version 1.4 is held in file LOAD14.EXE in directory /pub/pcsupp/dos/loadsys as detailed above. It is a loader/unloader for both programs and device drivers.

The RAL LLCPKT and LLCPKT2 systems are held together in file LLCPKTS.EXE in directory /pub/pcsupp/dos/llcpkt. They map the BICC MPS ethernet interface onto a Packet Driver and are only of use to users of the Rainbow software.

The RAL MOS2 IBM 3270 emulator version 2.3 is held in files MOS23.EXE, MOS23X.EXE and MOS23Y.EXE in directory /pub/pcsupp/network/mos2. This TSR provides IBM 3270 emulation, EEHLLAPI and GDDM-PCLK support over asynchronous and ethernet communications.

Release 10.1 of the Crynwr 3C509 Packet Driver

For the copyright, see COPYING.DOC. In part it says that: Anyone with a copy of the drivers may give it away, and the source code for all modules must be available.

NO WARRANTY, see COPYING.DOC, but support for packet drivers is available, see SUPPORT.DOC.

This release contains only the 3C509 packet driver. Many more packet drivers (including 3c501, 3c503, 3c505, 3c507, and 3c523) are available in the complete packet driver release. See HOWTOGET.IT for availability.

Installation:

The packet driver has two parameters, one required and one optional.

Run the driver with one parameter, the software interrupt. A packet driver needs to have a software interrupt assigned to it for other programs to access it. I recommend 0x7f or 0x7e.

The driver needs an initial ID port to access the 3C509 hardware. The driver uses a default of 0x110 for the ID port. If you have a conflict at port 0x110 (very few machines do), the packet driver will be unable to initialize the adapter, so try specifying a different value as the second parameter. ID port must be 0x100, 0x110, 0x120, ..., 0x1f0.

Notes:

The 3C509 is a good, fast, cheap board. However, it requires close cooperation between the driver and the adapter. If your system has high interrupt latency, this cooperation is disrupted. Also, the decisions made by different driver authors will affect whether a driver/adapter combination will work in your system.

In particular, the bursty high packet rate typical of NFS implementations causes the 3C509 to overrun. You may have to reduce the maximum length of packets sent to the 3C509 to transfer large files.

This driver is ONLY supported and tested on the 3C509. This driver will be enhanced in the future to support the EISA-3C579 and MC-3C529 versions of the Etherlink III family. It currently contains code to also support the EISA-3C579 card. I have only been able to test the driver on my EISA 486, where it doesn't work. The problem shows itself by corrupting the contents of packets. But this could just be my system (which is of dubious origin). If you would like to beta test this driver on a 3C579, please tell me how your trial went.

Please report problems. I cannot guarantee that I will work on your problem if you are not a Crynwr customer. I CAN guarantee that I will not work on it if you don't report it to me. Written reports are preferred to phone calls.

Russell Nelson
nelson@crynwr.com
315-268-1925 voice
315-268-9201 FAX
Crynwr Software, 11 Grant St., Potsdam, NY 13676

from the clarkson distribution

Installation instructions for the packet driver collection

Document conventions

All numbers in this document are given in C-style representation. Decimal is expressed as 11, hexadecimal is expressed as 0x0B, octal is expressed as 013. All reference to network hardware addresses (source, destination and multicast) and demultiplexing information for the packet headers assumes they are represented as they would be in a MAC-level packet header being passed to the send_pkt() function.

Using the packet drivers

The packet driver must be installed prior to use. Since each packet driver takes only a few thousand bytes, this is best done in your autoexec.bat. Since the Ethernet boards typically have jumpers on board, the packet driver must be informed of the values of these jumpers (auto-configure is possible, but can disturb other boards). The first parameter is the software interrupt used to communicate with the packet driver. And again, because each board is different, the rest of the parameters will be different.

All parameters must be specified in C-style representation. Decimal is expressed as 11, hexadecimal is expressed as 0x0B, octal is expressed as 013. Any numbers that the packet driver prints will be in the same notation.

Before installing the packet driver, you must choose a software interrupt number in the range between 0x60 and 0x80. Some of these interrupts are used for other purposes, so your first choice may not work. See Appendix A for the section of Ralf Brown's interrupt list between 0x60 and 0x80.

Running a packet driver with no specifications will give a usage message. The parameters for each packet driver are documented below.

Most drivers can also be used in a PROM boot environment, see PROMBOOT.NOT for how to use -d and -n options for that purpose.

The -w switch is used for Windows. Install the packet driver before running MS-Windows. This switch does not prevent Windows from swapping your network application out of memory, it simply detects when that has happened, and drops the packets on the floor.

3Com 3C501

usage: 3C501 [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The 3c501 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

3Com 3C503

usage: 3C503 [-n] [-d] [-w] packet_int_no [int_level(2-5) [io_addr [cable_type]]]

The 3c503 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the cable type. The 3c503 can be attached to thick or thin Ethernet cables, and the selection is made in software. The cable type parameter should be zero for thick, and one for thin. The defaults are 2, 0x300, and 1 (thin). The 3c503 uses shared memory whose address is set by jumpers, but the software can ask the board what the address is.

3Com 3c505

usage: 3c505 [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The 3c505 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x300 and 0xd000.

3Com 3c507

usage: 3c507 [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The 3c507 will determine its parameters by reading the board. The only time you would need to specify the parameters is when you have multiple 3c507s in the same machine.

The 3c507 driver will use three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address.

3Com 3c523

usage: 3c523 [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The 3c523 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 3 and 0x300 and 0xc000.

ARCETHER

usage: arcether [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The ARCNET driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 5 and 0x2e0 and 0xd800. Note that a packet driver client must specifically support ARCNET. The only known client is Phil Karn's (KA9Q) networking package, NOS.

ARCNET

usage: arcnet [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The ARCNET driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 5 and 0x2e0 and 0xd800. Note that a packet driver client must specifically support ARCNET. The only known client is Phil Karn's (KA9Q) networking package, NOS.

AT&T

usage: at&t [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The AT&T driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x360 and 0xd000. This driver supports the StarLAN 1, StarLAN 10 NAU, EN100 and StarLAN Fiber NAU.

AT&TLP 0x62 2 0x360 0xd000 0 0

usage: at&t_lp [-n] [-d] [-w] <packet_int_no> [<int_no> [<io_addr> [<base_addr> [<media_sel> [<li_enabl>]]]]]

The AT&T_LP driver requires five additional parameters -- the hardware interrupt number, the I/O address, the memory base address, media select, and link integrity. The defaults are 2 and 0x360 and 0xd000. This driver supports the ATStarStation, ATStarLAN 10 LanPACER+ NAU, ATStarLAN 10 LanPACER NAU and AT Microelectronics T7231 evaluation board.

The final two numbers are new attributes.

0 0 --> This is for AUI setting
0 1 --> This is also for AUI setting
1 0 --> This is for TP setting, no link integrity
1 1 --> This is for TP setting, link integrity enabled

For the LP NAU only, "0 0" and "0 1" are invalid as there is no AUI port on that NIC.

David Systems Inc (DSI)

usage: davidsys [-n] [-d] [-w] <packet_int_no> <int_no> <io_addr> <delay_mult>

The DSI driver requires three additional parameters, the hardware interrupt number, the I/O port and the delay multiplier. Delaymult is a system dependent timing loop used for I/O to the card. A reasonable value is calculated during initialization, but on some fast systems it may need to be somewhat larger. The multipler is divided by ten, then multiplied by the calculated delay. The default multiplier is

10 (actually 1.0).

D-Link DE-600

usage: de600 [-n] [-d] [-w] packet_int_no

The D-Link Pocket Lan Adapter packet driver requires no additional parameters.

Digital Equipment Corporation DEPCA

usage: depca [-n] [-d] [-w] <packet_int_no> [<int_no> [<io_addr> [<mem_addr>]]]

The DEPCA packet driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 5 and 0x300 and 0xd000. The packet driver will resolve the io_addr automatically if io_addr is set to '?', e.g. depca 0x7e 5 ? 0xd000.

Multitech EN-301

usage: en301 [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The Multitech driver runs the EN-301 cards. The Multitech driver requires two additional parameters, the hardware interrupt number, and the I/O port.

Intel EtherExpress

usage: exp16 [-n] [-d] [-w] <packet_int_no> [<io_addr>]

The Intel EtherExpress packet driver has one optional parameter. The <io_addr> is only needed if there is more than one EtherExpress card in your system. Otherwise, the driver will search for adapter and get its parameters from it.

Mitel Express

usage: express [-n] [-d] [-w] <packet_int_no> [-n] [<driver_class> [<int_no>]]

The Mitel Express packet driver has one optional switch, and two optional parameters. The <driver_class> defaults to SLIP, and the <int_no> defaults to 7. The -n switch instructs card to be an NT. The <driver_class> should be SLIP or a number.

HP Ethertwist

usage: hppclan [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The hppclan driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

ICL EtherTeam16

usage: ETHIIE [-n] [-d] [-w] packetintno [intlevel [ioaddr [cabletype]]]

The ETHIIE driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the cable type. The interrupt levels supported by the adapter are 5, 9 (2), 12 and 15. The Ethenet IIe can be attached to thick or thin Ethernet cables, and the selection is made in software. The cable type parameter should be zero for thick, and one for thin. With the Twisted Pair (TP) version of the adapter, you must set interface to the value 1 (thin).

The defaults are 9 (2), 0x300 and 1 (thin).

Please note, that the adapter can be used only in a 16-bit slot of your computer.

IBM Token Ring

usage: ibmtoken [-n] [-d] [-w] packet_int_no [adapter_no]

The IBM Token Ring packet driver requires one additional parameters -- the adapter number. The default is zero. See IBMTOKEN.DOC for more information.

LocalTalk

usage: localtlk [-n] [-d] [-w] <packetintno> [<IP address>]

The LocalTalk packet driver requires atalk.sys to be installed. Because it is not an Ethernet class driver, it requires special code in the client. See LOCALTLK.NOT for more details.

NCR ET-105

usage: ncret105 [-n] [-d] [-w] <packet_int_no> <int_no> <base_addr> <Ethernet_address>

The NCR ET-105 driver requires four additional parameters -- the hardware interrupt number, the I/O address, the memory base address, and the Ethernet address. The Ethernet address assigned to any particular board is printed on sticky labels that come with the board.

Novell IPX

usage: ipxpkt [-n] [-d] [-w] packet_int_no [-n [no_bytes]]

The ipxpkt packet driver simulates Ethernet on Novell IPX protocol.

BICC Data Networks' ISOLAN 4110 ethernet

usage: isolan [-n] [-d] [-w] packet_int_no [int_no [base_addr]]

The BICC Isolan requires three additional parameters -- the hardware interrupt number and the memory base address. The defaults are 2 and 0xb800h.

Netbios

usage: nb [-n] [-d] [-w] packet_int_no ip.ad.dr.ess [receive queue size]

The netbios packet driver transports IP packets over NetBIOS.

Novell ne1000

usage: ne1000 [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The ne1000 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

Novell ne/2

usage: ne2 [-n] [-d] [-w] <packet_int_no>

The ne/2 driver requires no additional parameters.

Novell ne2000

usage: ne2000 [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The ne2000 driver requires two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 2 and 0x300.

Racal-Interlan (Formerly Interlan) NI5010

usage: NI5010 [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The NI5010 driver requires two additional parameters --

the hardware interrupt number and the I/O address. The defaults are 3 and 0x300.

Racal-Interlan (Formerly Micom-Interlan) NI5210

usage: ni5210 [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The NI5210 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x360 and 0xd000. Note that Interlan sets the default memory base to 0xa000, which is brain-damaged, because that area of memory is specifically reserved for video adapters, and in fact the EGA and VGA use it.

Racal-Interlan NI6510

usage: ni6510 [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The ni6510 driver has two additional parameters -- the hardware interrupt number and the I/O address. The defaults are 2 and auto-sense. These parameters do not need to be set unless the auto-sense routine fails, or otherwise disrupts operation of your PC.

Racal-Interlan (Formerly Micom-Interlan) NI9210

usage: ni9210 [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The ni9210 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x360 and 0xd000.

NTI 16

usage: nti16 [-n] [-d] [-w] packet_int_no [int_no [io_addr [base_addr]]]

The nti16 driver requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 3 and 0x338 and 0xd000.

SLIP8250

usage: SLIP8250 [-n] [-d] [-w] packet_int_no [-h] [driver_class] [int_no]
[io_addr] [baud_rate] [recv_buf_size]
The driver_class should be SLIP, KISS, AX.25, or a number.

The SLIP8250 driver is not strictly an Ethernet adapter, however some software packages (such as KA9Q's NET and NCSA

Telnet) support Serial Line IP (SLIP). SLIP must be specially supported because it doesn't use ARP and has no hardware addresses prepended to its packets. The PDS is not clear on this, but the packet driver does the SLIP encoding.

The parameters are as follows. The -h flag is included if you wish to use hardware handshaking (the packet driver will then suspend the transmission of characters while CTS is low). The driver_class is the class that is returned to a client of the packet driver spec in the driver_info call. The int_no is the hardware interrupt number, defaults to 4 (COM1). The io_addr is the hardware I/O address, defaults to 0x3f8 (COM1). The baud_rate defaults to 4800 baud. The recv_buf_size defaults to 3000.

EtherSLIP

usage: ethersl [-n] [-d] [-w] packet_int_no [-h] [int_no] [io_addr] [baud_rate] [send_buf_size] [recv_buf_size]

The EtherSLIP driver is a simulated Ethernet adapter. It appears to the application software to be an Ethernet driver, but it transmits and receives SLIP packets on the serial line.

The parameters are as follows. The -h flag is included if you wish to use hardware handshaking (the packet driver will then suspend the transmission of characters while CTS is low). The int_no is the hardware interrupt number, defaults to 4 (COM1). The io_addr is the hardware I/O address, defaults to 0x3f8 (COM1). The baud_rate defaults to 4800 baud. The send_buf_size and recv_buf_size default to 3000 each.

Ungermann-Bass NIC-PC

usage: ubnicpc [-n] [-d] [-w] <packet_int_no> <int_no> <base_addr>

The UB NIC-PC driver requires two additional parameters, the hardware interrupt number, and the memory base address.

Ungermann-Bass NIC-PS/2

usage: ubnicps2 [-n] [-d] [-w] <packet_int_no> <int_no> <io_addr> <base_addr>

The UB NIC-PS/2 requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are the contents of the POS registers, so the only time you would need to use the parameters is if you're using two NIC-PS/2 boards in one machine.

Tiara Lancard

usage: tiara [-n] [-d] [-w] packet_int_no [int_no [io_addr]]

The Tiara driver runs the Tiara LANCARD/E cards, both eight and sixteen bit cards. The Tiara driver requires two additional parameters, the hardware interrupt number, and the I/O port.

Western Digital WD8003 E EBT EB ET/A and E/A

usage: WD8003E [-n] [-d] [-w] packet_int_no [-o] [int_level [io_addr [mem_base]]]

The WD8003E driver runs the Western Digital E, EBT, EB, ET/A, and E/A Ethernet cards. The WD8003E requires three additional parameters -- the hardware interrupt number, the I/O address, and the memory base address. The defaults are 2 and 0x280 and 0xd000. The wd8003 cards do not enable their memory until configuration time. Some 386 memory mappers will map memory into the area that the card intends to use. You should be able to configure your software to leave this area of memory alone. Also driver will refuse to map memory into occupied memory. The occupied memory test fails on some machines, so the optional switch -o allows you to disable the check for occupied memory.

Errorlevels

Some of the packet drivers return error codes. Some of these error codes indicate fatal errors, and some are merely warnings. For the moment, you must consult the source to see what the errorcodes mean. For example, pktchk returns 0 if a packet driver exists at a given address, and 1 if not. You might use it in a batch file that only installs a packet driver if one is not found.

```
rem only install the packet driver if there isn't one
rem already.
pktchk 0x7e
if errorlevel 0 goto gotit
ni5210 0x7e

:gotit
```

The "errorlevel" test is true if the errorlevel is less than or equal to the parameter.

Utility Programs

There are also several utility programs for packet drivers:

PKTADDR

usage: pktaddr packet_int_no [ethernet_addr]

If the second argument is given, the Ethernet address of the given packet driver is set. The Ethernet address is printed out.

PKTALL

usage: pktall packet_int_no

All packets are received and discarded from the given packet driver. This program is of most use with PKTMODE and TRACE.

PKTCHK

usage: pktchk packet_int_no [packet_int_no]

Test for existance of a packet driver. Returns with errorlevel 0 if the specified interrupt has a packet driver. If the second argument is given, all interrupts in the range are checked for a packet driver. If no packet driver is found at all, errorlevel 1 is returned.

PKTMODE

usage: pktmode packet_int_no [receive_mode]

If the second argument is given, the receive mode of the given packet driver is set. A decimal number from the list of modes should be used. All the possible modes are printed out. Unimplemented modes are marked with "xx", and the current mode is marked with "->".

PKTMULTI

usage: pktmulti packet_int_no [-f filename | address ...]

The specified addresses are set as allowed multicast addresses. If no list of addresses is given, then the current list of addresses is printed. The
addresses may either be specified on the command line, or in a file using the -f option. When a file is used, any whitespace in the file is ignored.

PKTRING

usage: pktring [-m] dest_packet_int_no src_packet_int_no

Test two packet drivers loaded on two Ethernet cards in the same machine. Sends packets of every possible length using the driver located at src_packet_int_no to the driver located at dest_packet_int_no. If the -m switch is used, then the various

receive modes are tested. Note that the multicast tests (modes four and five) do not work properly yet.

PKTSTAT

usage: pktstat first_int_no [last_int_no]

The statistics for all packet drivers in the given range are printed. The default range is 0x60 through 0x80. The meanings of the columns are given below.

pkt_in is the number of packets ever received by this driver.
pkt_out is the number of packets ever transmitted by this driver.
byt_in is the number of bytes ever received by this driver.
byt_out is the number of bytes ever transmitted by this driver.
pk_drop Packets dropped because there was no handler for that Ethernet packet type.
err_in Dependent upon the packet driver.
err_out Dependent upon the packet driver.

PKTSEND

usage: pktsend packet_int_no [-r] [-f filename | packet]

The specified packet is sent using the specified packet driver. The -r option says to repeat sending as fast as possible. You shouldn't use this option very often. The packet may either be specified on the command line, or in a file using the -f option. When a file is used, any whitespace in the file is ignored.

PKTTRAF

usage: pkttraf packet_int_no

Graphically display traffic on an EGA or VGA screen. The first twenty Ethernet addresses encountered are assigned a node number. The traffic between each pair of nodes is displayed as a line of varying intensity. When any line reaches maximum intensity, the intensities of all lines are halved.

A cursor highlights one of the nodes. The Ethernet address of the highlighted node is printed in the lower-right corner. The cursor is moved using space and backspace.

TERMIN

usage: termin [-s] packet_int_no

The specified packet driver is terminated, and its memory recovered.

The s-option (stop) is used to prepare for termination. The in-use flag for all handles are cleared. This prevents upcalls to handlers that are to be removed and also makes it possible to later terminate the packet driver even though handles are not released. Actually, doing termin -s after prom

boot is like cutting the branch you are sitting on. Recipe for removing packet driver, IPX and NET:

```
pktdrvr 0x7c ....
MARKNET C:IPX&NET3.MRK
PDIPX
NET3
. . .
NET3 u                  ; unload netx to avoid
                          communication timeout
TERMIN -s 0x7c          ; pkt drvr no longer calls any
                          nonexistent rcvrs
RELNET ipxet3.mrk       ; IPX is "removed"
TERMIN 0x7c             ; It is now safe to terminate the
                          packet driver
```

TRACE

usage: trace packet_int_no [buffer_size]

Trace is very useful for debugging packet driver troubles. Trace lets you trace all transactions between a user program and the packet driver. The transactions are stored in a memory buffer whose size is set with buffer_size. The default size is 10,000 bytes.

When you run trace, it sets itself up and then spawns COMMAND.COM so that you can run a network program that uses the packet driver. After you quit your network session, you issue an "EXIT" command. This returns you to trace, which writes the transaction log to "TRACE.OUT". The following program, DUMP, interprets TRACE.OUT.

DUMP

usage: dump

Interprets the contents of TRACE.OUT as written by TRACE.

WINPKT

usage: winpkt <new_packet_int_no> <old_packet_int_no>

Provides a Packet Driver interface between Windows 3 Enhanced mode applications and a real Packet Driver. This attempts to solve the problem of Windows moving applications around in memory willy nilly. It replaces the -w flag hack.

Install WINPKT after the Packet Driver and before starting Windows.

Appendix A

Interrupt usage in the range 0x60 through 0x80, from Ralf Brown's interrupt list.

```
60 -- -- reserved for user interrupt
60 -- -- FTP Driver - PC/TCP Packet Driver Specification
60 01 FF FTP Driver - DRIVER INFO
60 02 -- FTP Driver - ACCESS TYPE
60 03 -- FTP Driver - RELEASE TYPE
60 04 -- FTP Driver - SEND PACKET
60 05 -- FTP Driver - TERMINATE DRIVER FOR HANDLE
60 06 -- FTP Driver - GET ADDRESS
60 07 -- FTP Driver - RESET INTERFACE
60 11 -- 10-NET - LOCK AND WAIT
60 12 -- 10-NET - LOCK
60 13 -- 10-NET - UNLOCK
60 20 -- FTP Driver - SET RECEIVE MODE
60 21 -- FTP Driver - GET RECEIVE MODE
60 24 -- FTP Driver - GET STATISTICS
61 -- -- reserved for user interrupt
62 -- -- reserved for user interrupt
63 -- -- reserved for user interrupt
64 -- -- reserved for user interrupt
65 -- -- reserved for user interrupt
66 -- -- reserved for user interrupt
67 -- -- LIM EMS
          ...
67 DE 00 Virtual Control Program Interface - INSTALLATION CHECK
          ...
68 01 -- APPC/PC
          ...
69 -- -- unused
6A -- -- unused
6B -- -- unused
6C -- -- system resume vector (CONVERTIBLE)
6C -- -- DOS 3.2 Realtime Clock update
6D -- -- VGA - internal
6E -- -- unused
6F -- -- Novell NetWare - PCOX API (3270 PC terminal interface)
6F 00 -- 10-NET - LOGIN
          ...
70 -- -- IRQ8 - AT/XT286/PS50+ - REAL-TIME CLOCK
71 -- -- IRQ9 - AT/XT286/PS50+ - LAN ADAPTER 1
72 -- -- IRQ10 - AT/XT286/PS50+ - RESERVED
73 -- -- IRQ11 - AT/XT286/PS50+ - RESERVED
74 -- -- IRQ12 - PS50+ - MOUSE INTERRUPT
75 -- -- IRQ13 - AT/XT286/PS50+ - 80287 ERROR
76 -- -- IRQ14 - AT/XT286/PS50+ - FIXED DISK
77 -- -- IRQ15 - AT/XT286/PS50+ - RESERVED
78 -- -- not used
79 -- -- not used
7A -- -- Novell NetWare - LOW-LEVEL API
7A -- -- AutoCAD Device Interface
7B -- -- not used
7C -- -- not used
7D -- -- not used
7E -- -- not used
```

7F -- -- HDILOAD.EXE - 8514/A VIDEO CONTROLLER INTERFACE
7F -- -- HLLAPI (High-Level Language API)
80 -- -- reserved for BASIC

~epcc24/cutcp on castle

The following note describes Clarkson Telnet, Packet Drivers, and Kermit 3.11 (or 3.12) used in conjuntion with PC-NFS. It may be of general interest.
None of the software is mine, but it seems to work for me. Please let me know if this note is wrong, or misleading.
People who should know say that the packet multiplexing software may be unreliable, but I have not found it so.

The Clarkson Telnet (also known as 'cutcp') is a freely available Telnet implementation. It has several interesting features including:

Allows several simultaneous interactive sessions.
Tektronix 4010 emulation.
Interesting use of colour.
Configurable keyboard mapping, to suit application.
Time displayed on screen.
Call-out to DOS.
Use of DNS name server.
Script files for automated sessions.

CUTCP - running stand-alone.

'cutcp' itself is in 'micros/ibmpc/dos/f/f265' on micros.hensa.ac.uk (name:hensa, password:hensa) This machine was formally known as 'pdsoft.lancs'. It is accessable via X25 as 'hensa.micros'.
The simplest way to try this is stand-alone. For this, the only files you need are 'telbin.exe', and a configuration file.
The release contains a configuration file, 'config.tel', which explains how to configure all the options. A minimum configuration file would look something like the following:

```
# Minimum cutcp configuration file
myname=noname
myip=129.215.200.74
netmask=255.255.255.0
hardware=wd8003
address=d000
ioaddr=280
# should have 'ftp=no'
# or 'passfile=' to enable password checking.
ftp=no
name=cancer
host=cancer.ucs.ed.ac.uk
hostip=129.215.200.7
nameserver=1
name=gw-EUCS-Offices
host=gw-EUCS-Offices.ucs.ed.ac.uk
hostip=129.215.200.254
gateway=1
# name=castle
# host=castle.edinburgh.ac.uk
# hostip=129.215.128.23
domaintime=1
domainslist="ucs.ed.ac.uk,ed.ac.uk"
```

The configuration file should contain details of the gateway machine. A DNS name server and domainslist may be specified, or the IP addresses of the machines to be accessed may be included.
For standalone use, the configuration file should contain lines such as:

```
hardware=wd8003
address=d000
ioaddr=280
```

To use 'cutcp', you need to do:

```
set configtel=c:\cutcp\config.001    - set up config file name
     - this can be in 'autoexec.bat'
telbin host
```

<Alt> h - gives help menu

CUTCP also include an FTP implementation, which uses the same configuration file.
It would be sensible for users to fetch the full Clarkson software for themselves from pdsoft.lancs.

PACKET DRIVERS

There are packet drivers in 'micros/ibmpc/dos/g/g224' on micros.hensa.ac.uk for 'research machines' ethernet cards.
Packet drivers may also be found on 'sun.soe.clarkson.edu' in 'pub/packet-drivers', or on 'omnigate.clarkson.edu'.
A convenient source of drivers is 'watsun.cc.columbia.edu', directory '/packet-drivers/bin'. Other sources can be found using 'archie'. Packet drivers have names such as 'ne2000.com', 'wd8003e.com', or '3c503.com'.
The '3c503' packet driver I have requires that the RAM is enabled on the board. The driver reads the RAM address from the hardware. There is no jumper for the interrupt address, as this is set by software.
If you have an NDIS driver for your board, but no packet driver, you can use the shim 'DIS_PKT9' to convert. I got file 'dis_pkt9.zip' from 'casbah.acns.nwu.edu', directory '/pub/nupop/ndis_pkt'.
Use archie (archie.doc.ic.ac.uk, user:archie) to locate other sources.

PACKET MULTIPLEXING SOFTWARE.

This can be obtained by:
hhcp -b ib.rl.ac.uk:pktmux12.exe pktmux12.exe
Word Length:8
Account:PCSOFT//192
No password.

It is also be available by anonymous ftp from 'ib.rl.ac.uk', directory pcsoft.192, file pktmux12.exe.
Use name 'anonymous' as 'ftp' is not recognized.
A previous version - pktmux10.exe - is said to be buggy, and should be replaced.

File 'pktmux12.exe' can be executed to give file pktmux.exe and pktdrv.exe.

CLARKSON WITH PACKET DRIVER.

To run CUTCP with packet drivers, you need to change the autoexec.bat to start the packet driver. E.g.
c:\cutcp\wd8003e.com 0x60 2 0x280 0xd000

The 'configtel' file should be changed to include
hardware=packet
ioaddr=0x60
address=0

PC-NFS WITH PACKET DRIVER.

To run PC-NFS with a packet driver, you need to change the 'config.sys' (for PC-NFS) to remove the specific hardware drivers and include:
device=c:\nfs\pcnfs.sys
device=c:\nfs\sockdrv.sys
device=c:\cutcp\pktd40a.sys

Also change the 'autoexec.bat' to include:
c:\cutcp\wd8003e.com 0x60 2 0x280 0xd000

'pktd40a.sys' is a PC-NFS driver which interfaces with packet drivers. The current recommended version is '/sun/pc-nfs/pktd40a.sys.Z' on 'src.doc.ic.ac.uk'. This file needs to be 'uncompress'ed on a unix host. 'src.doc.ic.ac.uk' is also known as 'puffin.doc.ic.ac.uk'.

The old version is file 'pktd35.sys', obtained from 'pub/packet-drivers/pnfspktd.35' on 'sun.soe.clarkson.edu'. This is also on 'src.doc.ic.ac.uk' as /sun/pc-nfs/pktd.sys.3.5.
Note: The 'pktd.sys' in 'pub/cutcp' on 'omnigate.clarkson.edu' does not work with PC-NFS 3.5.

PC-NFS TOGETHER WITH CLARKSON.
Modify 'config.sys' as for PC-NFS with packet driver.
'autoexec.bat' should contain:

```
c:\cutcp\wd8003e.com 0x60 2 0x280 0xd000
c:\cutcp\pktmux 2
c:\cutcp\pktdrv
c:\cutcp\pktdrv /lf
```

The 'pktdrv' line appears twice. The '/lf' option on the second 'pktdrv' allows Clarkson to act as an ftp listener, if this is required.
The CUTCP 'configtel' file should be changed to include

```
hardware=packet
ioaddr=0x64
address=0
```

When running windows, the 'pktdrv' should be loaded in a '.bat' script which also loads the application. Background execution should also be enabled, so communications can be kept alive when the window is not in use.

Note.
There is a PC-NFS version of CUTCP which does not require the packet multiplexing. This version has a bug in it, so it will not work with 'castle' at Edinburgh, which is a Sequent machine.
'CUTCP/CUTE 2.2D/NFS-A' is in the UKUUG archive at imperial college (src.doc.ic.ac.uk, by anonymous ftp)
as sun/pc-nfs/sezcutcp.exe (a self extracting archive).

KERMIT.
Kermit 3.11 or 3.12 may be used over the ethernet, in which case it acts as a Telnet implementation. It uses 'packet' software, as described above for Clarkson. See 'Clarkson with packet driver'.
Kermit will emulate VT100, VT102, VT220, VT320, Tek4010 amongst others. The VT320 will emulate a 132 column terminal.

*** Note.
The kermit release has changed since the following was written.
File msvibm.boo isn't there anymore. 'micros.hensa' say they supply it free of charge, but only on floppy disk. There is now a file msr312.upd, which indicates that there is a new release.
'msvgen.boo' is a '3.11', but does not do 'set port tcp/ip'.
The 'msvibm.exe' for 3.12 is available elsewhere - see below.
*** End Note.

The files were available from 'micros.hensa.ac.uk' in directory 'kermit/ms'. The files I found important were:

```
00read.me     - important documentation
msvibm.boo    - BOO coded executable file.
mskermit.pch - Patches, to fix certain known faults.
msvibm.pif    - PIF file, for windows (does not work for me!).
mskerm.doc    - User guide
mskerm.hlp    - Summary of commands and features
mskerm.bwr    - Restrictions and known bugs
msr311.upd    - 3.11 Release notes.
```

The 3.12 release is available from 'watsun.cc.columbia.edu' in directory /kermit/bin. The files you may need are:

```
msvibm.exe    - executable file.
   (No patch file is needed)
msvibm.pif    - PIF file, for windows
```

- NB Kermit requires a lot of memory.
From 'micros.hensa.ac.uk', directory 'kermit/ms', you can get
msr312.upd - 3.12 Release notes.

At start-up, kermit reads file 'mskermit.ini'. This may define useful macros. The directory containing this file must be in one of the directories in the current PATH.
My current 'mskermit.ini' reads:

```
comment -- file mskermit.ini executed at start-up
set tcp/ip address 129.215.200.74
comment -- set tcp/ip address rarp
set tcp/ip domain ucs.ed.ac.uk
set tcp/ip gateway 129.215.200.254
set tcp/ip broadcast 129.215.200.255
set tcp/ip primary-nameserver 129.215.200.22
set tcp/ip secondary-nameserver 129.215.200.7
set tcp/ip subnetmask 255.255.255.0
set port tcp/ip castle.ed.ac.uk
comment should find packet driver or odi automatically if omitted
set tcp/ip packet-driver-interrupt \x7e
comment set tcp/ip packet-driver-interrupt odi

comment -- defines
define quit       exit
define stop       exit
define castle     set tcp/ip host castle.ed.ac.uk, connect
define festival   set tcp/ip host festival.ed.ac.uk, connect
define hensa      set tcp/ip host micros.hensa.ac.uk, connect

comment -- do not overwrite my files
set warning on
comment -- use tcp/ip flow control
set flow none
end
```

I use this together with a file 'kermit.bat' which reads:

```
@echo off
REM WINDOWS BAT FILE to run Kermit
C:\F265\PKTMUX\PKTDRV.EXE 7e
g:\kermit\msvibm.exe %1 %2 %3
C:\F265\PKTMUX\PKTDRV.EXE /u
```

This batch file loads a packet driver, calls Kermit, and finally unloads the packet driver.

Kermit 3.11 is faulty, in that it is not able to react to re-routing ICMP messages. There are two gateways on the ethernet segment I use, so the only way to access hosts via the second gateway is to change the gateway definition. This fault is fixed in 3.12.

After a lot of trouble, I am finally able to run Kermit on my 486 using the supplied pif file. The necessary conditions seem to be:
1. Load the pc-nfs drivers high to make sufficient space.
2. Use the script as described to load the packet driver within windows.
3. Specify the particular packet driver interrupt number in 'mskermit.ini'

Anybody using kermit for file transfer should be aware that a faster way to operate kermit is by increasing the data transfer packet size up to the maximum. To do this set the two options in your Kermit initialisation
file (.kermrc on castle and festival):

```
set receive packet-length 1000
set send packet-length 1000
```

Gopher.
   There is a version of Gopher which runs on packet drivers, and can be run with PC-NFS in the same way as Kermit.
Nupop (see below) includes code for Gopher, Webster, Telnet, and FTP.

Mail Programs.
   Various mail programs run using packet drivers. These include POP, NUpop, ELM-PC ...

NUPOP Mailer.

From: pib@nwu.edu (Philip R. Burns)
Newsgroups: info.nupop
Subject: ***** NUPop v2.0 Beta Test Release Now Available *****
Date: 5 May 93 05:41:19 GMT

I've placed the first beta test release of NUPop v2.0 on ftp.acns.nwu.edu in /pub/nupop/nupop20beta/nupop200.zip. This test release contains ONLY the protected mode version. If you do not have an 80286 or better processor with 2 megabytes or more of memory, you cannot run this test release.

Nupop200.zip was created with the latest PKZIP utility. If you do not have a recent unzipper program, you can pick up /pub/nupop/unzip.exe to uncompress nupop200.zip.

The installation guide has not yet been updated to match this test release. If you are an experienced user of NUPop, the aareadme.200 file in nupop200.zip should be enough to get you started.

-- Phil "Pib" Burns
   Northwestern University
   Evanston, IL. USA
   pib@nwu.edu

Nupop includes code for Gopher, Webster, Telnet, and FTP.

Novell Netware.
   'pdether.com' will convert the packet interface to an odi interface. Latest version is PDE103. My copy of this came from a local novell server.
   I obtained pdether from 'gdr.bath.ac.uk', file '/simtel-cdrom/msdos/pktdrvr/pdether.zip', but this is obsolete and will not work with current versions of Netware.
   Netware's 'lsl' should be loaded before 'pdether'. This uses a file 'net.cfg' which is in the same directory as 'lsl'. It should contain a section for PDEther - mine reads:

```
Link Driver PDEther
        INT #1 0x65
        Frame Ethernet_II
        Protocol IP   800  Ethernet_II
        Protocol ARP  806  Ethernet_II
        Protocol RARP 8035 Ethernet_II
        Protocol IPX  8137 Ethernet_II
```

! Stephen Hayes, Edinburgh University Computing Service.
! S.T.Hayes @ edinburgh.ac.uk, Tel: 031-650 4990.

~a pcc24/cutcp on castle.

PC-NFS, NDIS, ODI, PACKET drivers explained.

This attempts to be a fairly simple explanation of the various ethernet drivers, and the way they interact.

Rather than all software to be written with code to drive all possible ethernet boards, it makes sense to divide the code into two parts, one of which is the 'driver' for the ethernet board. The driver is specific to the particular hardware, but the interface between the driver and the rest of the software is fairly fixed. Changes to the driver interface imply changes to all of the drivers, and so are not undertaken lightly.

The driver concept is such a good idea, that many people have done it, and done it in different ways. You could not expect different software companies to allow themselves to be dependent on drivers under the control of other companies, so everyone has their own! A quick list (apologies, rather vague, sorry if I have missed anyone) is:

Driver	Company	Application Software
PC-NFS	SUN	PC-NFS
ODI	Novell	NETWARE, LANWP, KERMIT
NDIS	3Com/Microsoft	LAN MANAGER
PACKET	NCSA, Clarkson	NCSA Telnet, Clarkson Telnet and FTP (CUTCP), NUPop, KERMIT, ELMPC ...

Of these, the Packet drivers seem to me to be most 'public domain'. There is source available for these.

The internals of the different drivers are different, and the interface between the driver and the rest of the software is also different. These differences can be important.

In principle, there could be several ethernet boards in one PC. These boards are addressed by a PORT number (e.g. 0x300) on AT bus machines, which distinguishes one board from another. Each board may have an interrupt address (usually 2,3,5,7,10). Some boards may use shared memory, or may use a DMA transfer channel.

SHIMS.

Eventually, the various companies seem to have realised that it does not make sense for them to have to write drivers for every single ethernet board that anyone makes. In order to simplify matters, there are various shims available which will convert one interface to another. For instance, there is a shim to allow PC-NFS to be used with an NDIS driver. This is a small piece of software, which interfaces with the NDIS driver software according to the NDIS interface, but which presents the PC-NFS driver interface to PC-NFS software. The manner of loading and linking shims varies greatly.

Some initialisation of drivers is done when they are loaded, but some is deferred until they are first called. This means that a shim may be loaded from 'config.sys', and when first called, may link into a driver loaded from 'autoexec.bat'.

Table of Known Shims.

Driver	Application Software Interface			
	PC-NFS	ODI	PACKET	NDIS
PC-NFS	<Not req>	<N/A>	<N/A>	<N/A>
ODI	nfsodi.sys	<Not req>	ODIPKT	ODINSUP.COM
PACKET	pktd40a.sys	PDE103.ZIP (PDETHER)	<Not req>	<N/A ??>
NDIS	nfs-ndis.sys	<Not Known>	DIS_PKT9.DOS	<Not req>

This table may be incomplete and out of date. I know little about

shims for NDIS drivers, so a shim to allow NDIS over PACKET could well exist. I suspect that there are no shims which allow other applications to use PC-NFS drivers, but I could be wrong.

I would expect there should be a shim which allows ODI to run over NDIS drivers, but I don't know what it is called.

More than one shim may be used in sequence if necessary, so you could run NDIS over ODI over PACKET, or PC-NFS over PACKET over NDIS.

PC-NFS driver.

Loaded by 'config.sys'. I suspect this provides a 'driver' interface in the same way as, say, a disk driver. I believe you can only have one PC-NFS driver, even if you have multiple ethernet boards.

Packet driver.

Loaded by 'autoexec.bat'. Communicates through a software 'vector' in the range 0x60 - 0x80. There may be many separate packet drivers, communicating through different vectors. Vector 0x61 is used by ODI, so can not be used by packet drivers if ODI is in use!

Packet multiplexors exploit this feature. The board driver is loaded, and may communicate through 0x60. PKTMUX drives this interface. Various PKTDRV programs then communicate with PKTMUX, and each PKTDRV offers a packet interface (almost) identical to the original, through a different vector.

NDIS driver.

Loaded by 'config.sys'. There is a 'protman.sys' driver component, which has a parameter which is the directory containing 'protocol.ini', a configuration file which contains all the different parameters which are needed, and specifies how the components link together. I suspect it also handles message passing between the different drivers. There is also a 'netbind' in 'autoexec.bat' which links all the software components together. The NDIS system allows linking of drivers for several ethernet boards, and possibly other components too. There is only ever one 'protman', one 'netbind', and one 'protocol.ini'.

ODI driver.

Loaded entirely from 'autoexec.bat'. The first component to be loaded is 'lsl.com'. Configuration information is obtained from the 'net.cfg' file, which should normally be in the base directory, or the directory containing 'lsl'. There can only be a single ODI driver. ODI drivers communicate with the applications program through a software 'vector' at location 0x61.

Disclaimer:

Sorry for any mistakes and inaccuracies, which are unintentional. Please let me know of any significant errors and omissions, of which I am sure there are many!

! Stephen Hayes, Edinburgh University Computing Service.
! S.T.Hayes @ edinburgh.ac.uk, Tel: 031-650 4990.

Mosaic

Mosaic via. SLIP

Mosaic will run over a SLIP or PPP connection. The most difficult aspect is establishing the SLIP/PPP connection. Below is a brief description for establishing a SLIP connection using a Shareware product called Trumpet Software International Winsock version 1.0. This particular product has an internal SLIP driver and an internal modem dialer. If you would like to obtain a copy of this product you can find it at the anonymous ftp site ftp.utas.edu.au. The file, twsk10a.zip, is located in the /pc/trumpet/winsock directory. For the convenience of our users we also keep a copy of this Shareware product on our anonymous ftp server, ftp.ncsa.uiuc.edu. The file, winsock.zip, is in the /PC/Mosaic/sockets directory.

After you have installed the Trumpet Winsock according to it's instructions, envoke tcpman.exe and select "Setup". Trumpet will need the following information about the SLIP server you are connecting to. Get this information from the company that is providing the SLIP service.

```
IP address      0.0.0.0
Name server     0.0.0.0
Time sever      0.0.0.0
Domain Suffix   the.name.of.your.domain
   ie. "ncsa.uiuc.edu" is our domain.
```

Check "Internal SLIP" and enter in the port number and baud rate of your modem. Save the information, click OK, and exit the program. Now, your system should be configured properly. Next, envoke Trumpet again and use the internal dialer option to place the call. Select the "Dialler" pull down menu. You will probably take notice there a few options available, however I will only discuss "Manual login" command. If you are interested in creating a script then please refer to the Trumpet documentation. Once you have selected Manual login you will need to enter the following command in the Trumpet window.

atdt xxx-xxxx (where the x's represent the phone number).

NOTE: There are two ways to gain access to the Internet via SLIP:

1) If you have a static slip account, that is, you were assigned an IP number that doesn't change.
- login to the server.
- Enter the command "slip".
- Hit the escape key.
- Double click on the Mosaic icon.

2) If you log into a server that assigns you a random IP number.
- login to the server.
- Enter the command "slip".
- Select the "Setup" pull down menu.
- Enter this IP in the appropriate field.

Note: After you enter the new IP number you will get a message that states you will have to close the application in order for this number to take affect.
- Hit the escape key.
- Close Trumpet.

Note: This action will not cause you to loose your connection.
- Double click on the Mosaic icon.

Here is a list of some of the companies that can provide nationwide SLIP and PPP access to the Internet. These companies are listed in alphabetical order and we do NOT endorse any particular provider.

Colorado Supernet (info@csn.org)
Colorado School of Mines
1500 Illinois Street
Golden, CO 80401
(800)748-0800

Institute for Global Communications (support@igc.apc.org)
PeaceNet/EcoNet/ConflictNet/LaborNet ** International Provider **
18 deBoom St.
San Francisco, CA 94107
tel: +1-415-442-0220
fax: +1-415-546-1794

JVNCnet (market@jvnc.net) ** SLIP only **
Global Enterprise Services
3 Independence Way
Princeton, NJ 08540
(609)897-7300

NetCom On-Line Communication Services (info@netcom.com)
4000 Moorpark Avenue
Suite 209
San Jose, CA 95117
(408)554-8649

Performance Systems International (info@psi.com)
11800 Sunrise Valley Drive ** International Provider **
Suite 1100
Reston, VA 22019
(703)620-6651

UUNET (info@uunet.uu.net) ** International Provider **
3110 Fairview Park DR
Suite 570
Falls Church, VA 22042
(703)204-8000

install.txt from the mosaic distribution

NCSA Mosaic for Microsoft Windows
Installation and Configuration Guide

Introduction

This is a step-by-step guide to installing and configuring NCSA Mosaic for Microsoft Windows. This guide assumes that the system on which you are installing NCSA Mosaic meets the following criteria:

- Microsoft Windows is properly installed and configured.
- Utilities such as ftp, pkunzip, and an ASCII editor are available.
- The system is connected to the Internet.

Acquiring the Software

Since you are reading this file, you probably already have the NCSA Mosaic for Microsoft Windows software. If that is the case, you can skip this section.

If you do not have the software, start by moving to a location where applications are normally installed on your system and create a subdirectory for NCSA Mosaic. The subdirectory name is arbitrary; the name mosaic is used here to clarify the following discussion. Move into the new subdirectory:

```
mkdir mosaic
cd mosaic
```

Now you are ready to log on to NCSA's FTP server and download the NCSA Mosaic files:

```
ftp ftp.ncsa.uiuc.edu
At the login prompt, enter anonymous
At the password prompt, enter your email address (e.g.,
        jdoe@business.com)
get README.FIRST
cd PC/Mosaic
ls              (To list the available files and directories)
bin             (To change to binary mode for the file transfer)
get wmos20a1.zip
```

The filename wmos20a1.zip will change with each release and update. It will always be in the format wmosversion.zip where version is the current version number. For example, wmos20a1.zip is the filename for NCSA Mosaic version 2.0, alpha release 1. The next alpha release will be wmos20a2.zip, and the first beta release of 2.0 will be wmos20b1.zip.

If you have never downloaded PC files from NCSA's server, read the file README.FIRST now. It provides useful information that may facilitate later steps in the installation process.

The file wmos20a1.zip is a compressed archive containing the NCSA Mosaic executable and several documents, including this guide and a tutorial when it is available. Execute the following command to retrieve the files from the compressed archive:

```
pkunzip wmos20a1.zip
```

Confirming the Files

Once you have downloaded and uncompressed the NCSA Mosaic files, the installation process is straightforward. Check to make sure all of the following files exist in the mosaic\ directory:

update.txt	Current list of this version's enhancements and bug fixes in ASCII format.
install.txt	This guide as an ASCII file
install.wri	This guide in Microsoft Write format
mosaic.exe	The NCSA Mosaic executable
mosaic.ini	The initialization and configuration file for Mosaic
readme.now	Last minute or emphasized information

Read the file readme.now. It may contain last minute information that

was not available when this document was prepared.

If you want to review the list of Mosaic enhancements and bug fixes since the last version, you should read the file update.txt. Or you can wait until you have NCSA Mosaic running and read this list online (follow the links from the Windows Mosaic home page.)

Checking the WinSock DLL

NCSA Mosaic is a WinSock 1.1-compliant program and requires that you have a WinSock 1.1-compliant sockets DLL (winsock.dll) installed to provide the TCP/IP networking under windows. Check your WinSock implementation's installation guide to see whether it is WinSock 1.1-compliant.

If you are using a commercial TCP/IP stack such as PC-NFS, or running a local area network such as Novell in addition to the TCP/IP, you must obtain the WinSock DLL directly from your network vendor.

If your WinSock DLL is not WinSock 1.1-compliant or you do not have a WinSock DLL installed, and if you are using a stand-alone system on the Internet, then you can use the shareware Trumpet WinSock. A compressed copy of this WinSock implementation can be found on NCSA's FTP server in the directory PC/Mosaic/sockets/. Download the files disclaim.txt and winsock.zip. (If you want to make sure you have the latest version of the Trumpet WinSock, it can be acquired via anonymous FTP from the server biochemistry.bioc.cwru.edu in the directory /pub/trumpwsk.)

Configuring NCSA Mosaic

To configure NCSA Mosaic, first copy the file mosaic.ini to the directory \windows. This is important because NCSA Mosaic will not otherwise recognize and save configuration changes. Be sure to leave a copy of mosaic.ini in the mosaic\ directory in case you need to restore any original entries.

If you are administrating a network site, and want to have one copy of Windows, and therefore can't put mosaic.ini in the Windows directory, use the environment variable MOSAIC.INI to specify the pathname of the INI file.

Edit the file \windows\mosaic.ini as follows using Notepad, edit, or any other ASCII editor. General users will not usually need to modify fields that are not mentioned in the following discussion.

Main section:

If you can be reached via Internet email, put your full email address in quotes in the E-mail field. This is used for annotations and for a return address when you select Mail to Developers. If your login ID is jdoe and you work at Business, Inc., your email entry might read as follows:

```
E-mail="jdoe@business.com"
```

If you do not want NCSA Mosaic to automatically load a document every time you run it, set Autoload Home Page to no:

```
Autoload Home Page=no
```

If you want to change the first document that is automatically loaded when NCSA Mosaic is run, change the Home Page entry to point to the document you want to load. As distributed, NCSA Mosaic points to a customized Home Page on NCSA's Web server:

Home Page =
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

If you will be using Mosaic over a slow network connection and do not want inline images to be automatically transferred, set Display Inline Images to no. Inline images will be replaced with an NCSA logo:

```
        Display Inline Images=no
```

The Grey Background entry allows you to specify whether the NCSA Mosaic window has a white (no) or grey (yes) background. Many documents posted to the Web are tuned to a grey background:

```
        Grey Background=yes
```

Fancy Rules toggles a different style of drawing horizontal rules in HTML documents (the <hr> tag.)

```
Fancy Rules=yes
```

List bullets slow down drawing documents. If you are a speed maven, you may wish to turn on simplified buillet drawing (line bullets instead of round bullets.)

```
Round List Bullets=yes
```

Settings section:

The anchor color is the color of the hyperlink anchors that appear in an NCSA Mosaic document. You may set the anchor color to any valid RGB (red, green, blue) combination. The RGB values must be separated by commas and can range from 0 to 255. As posted on the FTP server, the default color is blue (0,0,255):

```
        Anchor Color=0,0,255
```

In some instances, it is desirable to have NCSA Mosaic underline hyperlink anchors (e.g, when using a gray scale or monochrome monitor). As distributed, NCSA Mosaic is set to underline hyperlink anchors. Since underlining slows performance somewhat, you may wish to set Anchor Underline to no if your system has a color monitor:

```
        Anchor Underline=no
```

Mail section:

Edit the Default Title to contain the message you want to have appear in the subject line when you send email to the NCSA developers via the Mail to Developers selection on the Help menu. As distributed, NCSA Mosaic enters the phrase "WinMosaic auto-mail feedback":

```
        Default Title="WinMosaic auto-mail feedback"
```

Services section:

If you want to use NCSA Mosaic's news support, set your network news (NNTP) server here. As distributed, NCSA Mosaic specifies the University of Illinois' NNTP server:

```
        NNTP Server="news.cso.uiuc.edu"
```

NCSA Mosaic uses the SMTP server specified here to send mail back to the NCSA developers when you select Mail developers from the Help menu. As distributed, NCSA Mosaic specifies NCSA's FTP server because it is known and almost always available. If you have a local SMTP server that you would rather use, edit the entry accordingly:

```
        SMTP Server="ftp.ncsa.uiuc.edu"
```

Viewers section:

This section contains two subsections. The first subsection contains a list of file types in MIME (Multimedia mail) form. The second subsection specifies the viewer for each of the listed file types. This information is used to determine whether an external viewer must be launched to view a file and, if so, which viewer to launch.

If necessary, edit the */* entries in the second Viewers subsection to point to your viewers. If you do not have a viewer for a file type, leave the line alone. You may add arbitrary spawning of external viewers by defining a new TYPE#, specifying a viewer, and optionally providing a suffix list.

NCSA Mosaic must use telnet to connect to some information servers.

To do so, NCSA Mosaic needs to know where to find the telnet application on your system. Specify the full pathname for your telnet application in the telnet entry at the end of the Viewers section:

```
telnet="c:\trumpet\telw.exe"
```

Suffixes section:

This section lists the filename suffixes used to identify the file types of files retrieved via FTP or from HTTP version 0.9 servers. In such situations, NCSA Mosaic uses the information in this section and in the Viewers to determine whether an external viewer must be launched to view a file and, if so, which viewer to launch. Files that reside on HTTP version 1.0 servers are typed by the server and only the information in the Viewers section is used.

NOTE: If you are retrieving information from an HTTP/1.0 server (most World Wide Web servers are), the server automatically types data for you. The extensions you set up in the Suffixes section will not have any effect on what MIME type the document is assumed to be, because the data is already being typed by the server.

You may list any number of filename extensions for a given file type; simply separate them by commas.

The last suffix listed will be used when writing a file of that type to the local hard drive. If your external viewer requires a particular extension, make sure that it is listed last.

Consider the example of the JPEG file type. The line

```
image/jpeg=.jpeg,.jpe,.jpg
```

indicates that any file with the suffix .jpeg, .jpe, or .jpg is a JPEG image. Now look at the preceding section of the file. The lines

```
TYPE3="image/jpeg"
```

and

```
image/jpeg="c:\windows\apps\lview\lview30 %ls"
```

define a JPEG image as TYPE3 and identify the required external viewer. Using the information in the Suffixes and Viewers sections of this file, NCSA Mosaic will recognize files with the extensions .jpeg, .jpe, and .jpg as JPEG images and know that it must launch the external viewer lview31 when it encounters them.

Viewers specified in this section must be able to take a specified filename as a command line argument.

Annotations section:

Change the Directory entry to point to the directory on your local hard disk where you want to store personal annotations.

Change the Default Title to the title you want to use for your personal annotations.

User Menu sections:

This specifies the user-configured menus. NCSA Mosaic will accommodate up to ten user-configured menus in any combination of top level and pop-out menus. Top level menus (Menu_Type=TOPLEVEL) will show up in the main menu bar; otherwise, the menu must be listed as a pop-out from one of the top level menus.

YOU SHOULD REALLY USE THE MENU EDITOR IN MOSAIC TO EDIT THIS SECTION. Hand-editing can result in screwing up the menus.

User-configured menus are specified as follows. The first line indicates the number of the menu while the next one or two lines name the menu and specify whether it is a top level menu. If the second user-configured menu is to be a top level menu named Demos, the first three lines of the menu specification would be

```
        [User Menu2]
        Menu Type=TOPLEVEL
        Menu Name="Demos"
```

If the menu is to be a pop-out menu, omit the second line above.

These initial lines are followed by several lines specifying the items (Item#) in the menu. Each Item# line must begin with Item#= and the rest of the line must be in one of the following formats. A line specifying a hotlink must list a document and the URL with which it can be located separated by a comma:

```
        Item1=Vatican Exhibit,http://www.ncsa.uiuc.edu/SDG/
                Experimental/vatican.exhibit/Vatican.exhibit.html
```

A line specifying a menu separator must contain the word SEPARATOR:

```
        Item2=SEPARATOR
```

A line specifying a pop-out menu must contain the word MENU and the number of the pop-out menu, taken from the first line of another user-configured menu, separated by a comma:

```
        Item3=MENU,User Menu4
```

HotList section:

You can maintain a hotlist of files you wish to have conveniently available. The hotlist is maintained here and displayed when you select Open URL on the File menu and press the arrow button on the right side of the window.

YOU SHOULD REALLY USE THE MENU EDITOR IN MOSAIC TO EDIT THIS SECTION.
This "Hotlist" is also now know as the "QUICKLIST".

You can edit the hotlist directly in this section. If you want to delete a file from the hotlist, delete the corresponding line from this section. If you want to reorder the files, simply rearrange and renumber the URL# lines. Note that the entries must be numbered sequentially, starting with URL0. The easiest way to add a file to your hotlist is to bring the file up in NCSA Mosaic then select Add Current to Hotlist from the Hotlist menu; if you prefer, you can add files by editing this section directly.

The Hotlist menu will be merged with the user-configurable menus in a future release.

Document Caching section:

This sections tells NCSA Mosaic how many documents to cache so that you do not have to return to the network to retrieve a recently viewed document. As distributed, NCSA Mosaic caches five documents:

```
        Number=5
```

If you have lots of memory on your system, you can increase the cache number. If you have little memory, you may wish to decrease the number. If you want to turn caching off, set it to 0 (zero).

Do not modify the Type entry.

Font sections:

Do not edit this section directly; all font changes must be made from the Options menu.

Main Window section:

Do not edit this section directly; adjust the NCSA Mosaic window size with the mouse and select Save Window Size on the File menu to save the new dimensions.

Finding Viewers

NCSA Mosaic for Microsoft Windows uses external viewers and players to display certain types of files, such as JPEG images or MPEG movies. These viewers and players are separate applications and they are neither maintained nor formally distributed by NCSA.

However, NCSA is always watching for particularly suitable viewers and players. When one is located that NCSA can legally distribute, a copy is placed on NCSA's FTP server in the directory PC/Mosaic/viewers. If you do not have a good viewer or player for a particular file type, check this directory. If you find a viewer or player for a common data type that is not in this directory, or is significantly more useful than the one on our server, please let us know about it. (One way to communicate that information is to select Mail to Developers on the Help menu.)

Installing into Microsoft Windows

You are now ready to install NCSA Mosaic into the Microsoft Windows system. Select New... on the Program Manager's File menu. Select Program Item and click on OK to add NCSA Mosaic to a Program Group.

Executing and Testing NCSA Mosaic

Execute NCSA Mosaic now. If everything is properly configured and domain name serving is set up, NCSA Mosaic should be able to load documents successfully. Test this by selecting Windows Mosaic Home Page from the Demos menu.

If NCSA Mosaic hangs on execution or misbehaves in any other unexplained fashion, the most likely source of the problem is the WinSock DLL. See "Checking the WinSock DLL" near the beginning of this guide.

If you are able to execute NCSA Mosaic but selecting Windows Mosaic Home Page did not work, try the following exercises and include the results in your query to NCSA. The results will help NCSA technical support determine the nature of your problem:

- Select Open Local File on the File menu and try to open a file on your system.
- Select Open URL on the File menu and try to open an HTTP file on a remote Web server. The URL http://cs.indiana.edu/home-page.html will provide a good test.
- Select Open URL on the File menu and try to open an FTP file on a remote Web server. The URL file://cs.uwp.edu/pub/music/kurzweil will provide a good test.

Feedback to NCSA

Your comments on NCSA Mosaic are important; user feedback is an integral part of the Software Development Group's (SDG's) planning activities. Bug reports are particularly valuable because SDG's tests cannot duplicate all user environments and equipment configurations. Bug reports should include enough information to enable SDG developers to reproduce the problem. Please specify any information given by Mosaic, or give us a URL is your problem is reproducible.

To send bug reports, comments, and suggestions, select Mail to Developers on the Help menu, fill out the form that appears, and click on Send when it is ready to go. If Mail to Developers does not work, send an email message to the following address:

mosaic-win@ncsa.uiuc.edu

PC Speaker Driver

MICROSOFT SOFTWARE LICENSE

1. GRANT OF LICENSE. Microsoft grants to you the right to use and to make an unlimited number of copies of the Windows 3.1 Driver Library ("Software") provided that such copies are used only in connection with licensed copies of Microsoft's Windows 3.x products.

2. COPYRIGHT. The Software is owned by Microsoft or its suppliers and is protected by United States copyright laws and international treaty provisions any you may not remove the copyright notice form any copy of the Software.

3. OTHER RESTRICTIONS. This Microsoft License Agreement is your proof of license to exercise the rights granted herein and must be retained by you. You may not rent or lease the Software. You may not reverse engineer, decompile or disassemble the Software.

NO WARRANTY. ANY USE BY YOU OF THE SOFTWARE IS AT YOUR OWN RISK. THE SOFTWARE IS PROVIDED FOR USE ONLY WITH MICROSOFT WINDOWS 3.X AND RELATED SOFTWARE. THE SOFTWARE IS PROVIDED FOR USE "AS IS" WITHOUT WARRANTY OF ANY KIND. MICROSOFT AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall Microsoft or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use this Microsoft product, even if Microsoft has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

U.S. GOVERNMENT RESTRICTED RIGHTS

The Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227–7013 or subparagraphs (c)(1) and (2) of the Commercial Computer Software – Restricted Rights at 48 CFR 52.227–19, as applicable. Manufacturer is Microsoft Corporation/One Microsoft Way/Redmond, WA 98052–6399.

This Agreement is governed by the laws of the State of Washington.

Should you have any questions concerning this Agreement, or if you desire to contact Microsoft for any reason, please write: Microsoft Customer Sales and Service/One Microsoft Way/Redmond, WA 98052–6399.

04/06/92 filename LE911640.001

If you are interested in obtaining a copy of speak.exe then set Options... Load to Disk and click here. The file is 21,236 bytes.