Beginner's Guide to TCP/IP on the Amateur Packet Radio Network Using the KA9Q Internet Software

Version 1.0
May 9, 1990

Documenting
NET 890421.1
BM 3.3.1

by

Gary E. Ford, N6GF

Table of Contents

1. Introduction

TCP/IP is the common name for a set of protocols developed to allow cooperating computers to share resources across a network. A protocol is simply a set of conventions or rules which must be adhered to by the communicating computers on a network to ensure that the information being exchanged is received and interpreted correctly. In the amateur radio software implementation of TCP/IP networking, the command set of the TNC is replaced with a very basic set of commands and the protocols run in the user's computer. This opens up the real power of computer-to-computer networking via packet radio.

TCP/IP provides multiple connections (sessions), ASCII (text) and binary (executable programs and encoded text) file transfer, electronic mail, and keyboard-to-keyboard services. It was originally designed for the Department of Defense to connect unlike mainframe computers in the military, government, research institutions, private industry, and universities so that all could share resources on a common network. TCP/IP was originally designed so that it could be used with packet radio networks and it has since been modified so that it is both usable on amateur radio networks and complies with FCC rules on amateur radio digital transmissions.

While there are many commands in a TCP/IP software package, it is still easy to use. The user first sends a

few software commands to the TNC to substitute its limited command set with an even simpler command set (called "KISS," for "keep it simple stupid") and transfers control to the TCP/IP software in the user's computer. The user can then engage in TCP/IP communications, or, since the software package has the capability of communicating in normal AX.25 packet, the user can operate with AX.25 packet bulletin boards, keyboard connects, digipeaters, or NET/ROM nodes.

TCP/IP has several advantages over normal AX.25 packet. At its lower levels, its strategies for retransmission of packets, exponential backoff in the face of channel congestion, handling of lost and duplicated packets, and packetization of data to be transmitted often lead to better overall channel throughput. It is designed to be a multi-connect, store-and-forward system. With TCP/IP, your local switch (similar to a node on AX.25) will hold your mail, check every so often to see if your system is active and when it sees you on the air, will send your mail to a sub-directory on your computer's disk drive. Mail can also be forwarded from an AX.25 PBBS. There is never a "station busy" reply on TCP/IP, since the software provides for multiple sessions, with the switch passing out mail to connected stations much like a dealer dealing cards, while handling a file transfer at the same time. There is even a method to give emergency traffic a higher priority in the queue.

The most accurate name for the set of protocols we are describing is the "Internet protocol suite." This is a layered family of protocols. TCP (transmission control protocol) and IP (internet protocol) are two of the lower level protocols. While the end-user of the suite does not often interact with the TCP or IP protocols, they are

the best known of the protocols, and it has become common to use the term TCP/IP to refer to the whole family. It is probably not worth fighting this habit.

This document is a beginner's guide to use of the KA9Q Internet Software Package on the amateur radio packet network (AMPRNET). The KA9Q package is the result of several years of development by Phil Karn, KA9Q, and his "merry band of implementors." The "TCP group" has grown to include hundreds of individuals worldwide, many of whom have contributed ideas to this software. The software resulting from this collaboration is extremely versatile. It was written for the IBM PC and clones, but has been ported to the Apple Macintosh, Atari ST, Commodore Amiga, and to several versions of UNIX machines. It has drivers for several hardware interfaces, allowing communication on wire networks as well as packet radio networks.

The KA9Q Internet Package, in particular the program NET, provides the following services:

telnet The telnet protocol, as implemented in the KA9Q software, allows users to communicate by a keyboard connection. The end result is the same as doing an AX.25 connection in most cases, but you take advantage of the attributes of TCP/IP.

mail The simple mail transfer protocol (SMTP)

provides services for sending and receiving mail. The sending, receiving, and forwarding computers can run unattended during this transfer, so it is not necessary to log into a PBBS to pick up your mail. A separate program, Bdale's Mailer (BM), first written by Bdale Garbee, N3EUA, is available to compose and read mail messages. It is also documented in this guide.

file transfer The file transfer protocol (FTP) allows a user on any computer to get files from another computer, or to send files to another computer. Security is handled by requiring the user to specify a user name and password for the other computer.

ax25 services Regular AX.25 services are also provided, so that NET can be used for all of your packet radio activities. You can connect to a friend who is not running TCP/IP and conduct a keyboard chat. In addition, NET provides an AX.25 mailbox, allowing your friends to send you mail and to initiate a keyboard chat.

NET/ROM NET also allows your packet system to serve as a NET/ROM node, although this will not be documented in this guide.

An advantage to the TCP/IP protocols and software is that the routing of packets through several systems to the eventual destination is simpler for the user than that

required in AX.25 or NET/ROM forwarding. You do not need to know the full route to the destination. Rather, you set up a routing table to the stations that you can communicate with directly. If all other systems set up accurate routing tables, your packets will be forwarded properly to the desired destination.

There are many other TCP/IP services that may be made available for use on packet radio in the future. The routing tables can be set up and kept up-to-date automatically, without requiring the user to edit the table. When multi-tasking computers become commonplace hobbyist machines, you will be able to remotely login to these systems to execute programs on them. Network file systems may become available, so that you can store your files on a remote disk and access them through the packet radio network. Most of these potential services will require higher baud rate networks, as a 1200 baud radio link is just too slow to support them.

1.1. Objectives of This Guide

The objectives of this guide are intentionally limited. The guide is intended to encourage more hams to use the KA9Q NET software on the amateur packet radio network (AMPRNET) and thus, it is aimed at beginners. It provides information on installing and using version 890421.1 of NET and version 3.3.1 of BM on an IBM PC or clone, with a serial interface to a TNC running the KISS (Keep It Simple, Stupid)

firmware.

Installation and use of the software on other machines is similar to that on the PC, but no attempt has been made in this guide to deal with the software on these other machines.

This guide only documents the subset of the NET commands that are of most use to a beginning end-user of the software and the packet network. Short descriptions of all of the top-level NET commands are given in Appendix B. Further details are available in the manual "The KA9Q Internet Software Package," updated for the 890421.1 revision, dated May 8, 1989, by Bdale Garbee, N3EUA. This manual is often available on-line on the packet radio network as userman.doc, or in a compressed form, such as userman.arc or userman.zip. It should be consulted for details which are beyond the scope of this guide.

1.2. Acknowledgements

Much of the material in this guide was taken from "The KA9Q Internet Software Package," with permission from the author, Bdale Garbee. Thanks go to Shayne Hughes, N6SPE, who helped me with the "software archeology," as we read the source code and experimented with the programs to find out how they worked. Thanks also go to Jim Pearce, N6ESV, and Chuck Bland, N6DBT, who provided many useful comments on the manuscript.

I would appreciate receiving any comments you have about this guide. I can be contacted at the following addresses:

AX.25 PBBS: N6GF@WA6NWE.#NOCAL.CA
Internet: ford@iris.ucdavis.edu
U.S. Mail: 226 Diablo Ave., Davis, CA 95616

2. Necessary Resources

The purpose of this section of the guide is to describe the necessary resources you must have available to be able to set up an amateur packet radio system running TCP/IP. The hardware requirements of TCP/IP are nearly the same as any AX.25 packet station, although this guide assumes that the host computer is a PC or clone.

2.1. Computer

The computer required to run the version of NET described in this guide is an IBM PC or clone running the MS-DOS or PC-DOS operating system (to be referred to simply as DOS for the remainder of this guide). The PC can range from the original PC (8086) to the XT, AT (286), 386, or 486 machines. This computer must have at least one serial port and a floppy disk drive. A hard disk is preferred, but is not absolutely necessary.

It is assumed that the reader of this guide is familiar

with the basics of DOS. You should be able to set up directories, manage files, and have available and know how to use a text editor.

2.2. TNC

The TNC (terminal node controller) used for TCP/IP must run the KISS firmware. This includes the TAPR TNC-1, TNC-2 and clones produced by several manufacturers, equipped with a ROM running KISS. For the TNC-2 or clones, version 1.1.6 of the firmware, or later, is required. Most of the more recent TNCs also run the KISS firmware.

Attach your TNC to your PC serial port using an RS-232C cable, following your TNC manufacturer's instructions. Set the baud rate between the computer and the TNC as recommended in the TNC instructions. Verify that the TNC works properly in the AX.25 mode, again following the TNC instructions, and then enter the commands to run KISS.

For the TNC-2 and clones, at the "cmd:" prompt, first type "KISS ON" and you will receive the message "KISS WAS OFF" and another "cmd:" prompt. Type "RESTART" and you should note that the CON and STA LEDs will flash three times to indicate you have entered the KISS mode. This command set will then have effect for subsequent power on/off cycles. To return to normal operation,

enter the command "param ax0 255" at the "net>" prompt when you are running the KA9Q TCP/IP software package.

For Kantronics TNCs, typing "KISSMODE ON" while in command mode, followed by "RESET," will put the TNC in KISS mode. Turning the TNC off and then on will cause the TNC to return to command mode. If you first turn KISSMODE ON and then PERM the value in EEPROM, when the TNC is turned on, it will automatically be in KISS mode.

For other TNCs, follow the manufacturers instructions to enter the KISS mode.

2.3. Radio

The majority of the TCP/IP packet operations are on 2 meters, so you will need a 2m FM transceiver. The radio requirements for TCP/IP are the same as those for AX.25 packet. Follow the directions in your TNC manual to interface the transceiver to the TNC.

In most areas, TCP/IP packet operations are found in the frequency ranges 144.91-145.09 MHz and 145.71-145.79 MHz. You will have to ask around to find out what frequency is being used in your area. One way to find the TCP/IP operation is to operate your TNC in the AX.25 mode and monitor the frequencies with "MONITOR ON." The TCP/IP frequency is the one that causes the most "garbage" to be displayed on your

screen, although NET/ROM nodes also cause this problem as well. The reason for this is that AX.25 TNCs do not decode the TCP/IP or NET/ROM control information.

2.4. IP Address

IP addresses are 32 bit numbers that uniquely identify a given machine (or "host") running the TCP/IP protocol suite. All of the possible 32 bit numbers are coordinated by an entity known as the Network Information Center, or NIC. Amateur Radio operators are fortunate in that a "Class A Subnet" consisting of 24 bits of address, in the range 44.X.X.X, has been reserved for our use. By general consensus, Brian Kantor, WB6CYT, of San Diego, CA, now serves as the top level administrator of the 44.X.X.X address space, and assigns blocks of addresses to regional coordinators from around the world.

You need to have a unique address before you can link in with the rest of the networked world. The best way to get one is to ask around the local packet community and find out who your local address coordinator is. Your local coordinator will then assign you an address from the block for your area.

If you have not yet obtained your IP address and want to get on the air immediately, you may temporarily use [44.128.0.*], with '*' replaced by a number between 1 and 255. Try to be sure that no one else in your area

is using the same number.

Brian Kantor can be reached as brian@ucsd.edu on the Internet if you have access to this wire network and need help locating your local address coordinator.

2.5. KA9Q Software

The KA9Q TCP/IP program NET and the mailer BM are likely to found to be available in your local area. You should inquire about availability on your local packet BBS. This would not only provide you with the software, but also contact with someone who has used the software and could help you with its installation and use. Further, since the source code to these programs is available, many local versions are available and it is often to your advantage to use these local versions.

If you fail to locate the software locally, the Tucson Amateur Packet Radio association (TAPR) now provides floppy copies of the software on 360K PC floppies, and can provide KISS ROMs for various TNC's, at a nominal charge for duplication and shipping. Contact TAPR for more information.

TAPR
PO Box 12925
Tucson, AZ 85732
(602) 749-9479

The programs NET and BM must be installed and

configured on your computer. The easiest way to deal with this is to edit the sample configuration files that are included in most distributions of the software. Local distributions also include configuration information appropriate for your local network, so it is to your advantage to acquire the software locally. If your distribution does not include the sample files, detailed information on installation and configuration is given in Appendix A of this guide. The information in this appendix can also be used to understand the commands given in the sample files.

A new version of the KA9Q Internet Package is now under development. It is called "NOS" and is a major rewrite of the NET program. When an official release of this software is made, it is likely that most packet systems will convert to the new software.

3. Definitions, Conventions, and Notation

In this section, some terms used in TCP/IP networking are defined and the conventions and notation used in the guide are explained.

Each system on the amateur packet radio network is referred to as a "node" or "host," terms that are derived from wire networks. Since each TCP/IP node includes a computer, the term "machine" is also used interchangeably with "node" and "host." As a user, you employ a "local host" and you communicate with a

"remote host." The local host requests services from the remote host, and as a result, the remote host is known as a "server" and the local host is a "client." Actually, servers are provided for each of the supported protocols.

Some hosts are set up not for use by an end user, but rather to forward other packets, similar to the function of an AX.25 digipeater, and to serve as a file repository. These systems are generally operated 24 hours a day and are known as "switches" or "gateways." Some of these systems are set up to serve as mail gateways to and from AX.25 PBBS systems. Note, however, that end user hosts can also be used to forward TCP/IP packets.

Each host on the AMPRNET must be identified by an IP address, a 32 bit number that uniquely identifies a given machine. Hosts are also known by a symbolic name which is linked to the IP address in the configuration file, HOSTS.NET.

3.1. Conventions Used in this Guide

The conventions used in this guide are described below. The intent was to produce this guide in the form of a simple ASCII (plain text) file that could be distributed through the AMPRNET. Thus, it was not possible to use bold or italic fonts, changes in point size, or underlining to clarify meaning.

parameter Literal argument. A character string such as "parameter," with no surrounding brackets, is a required portion of a command and is to be typed exactly as shown.

<parameter> Variable argument. A command argument enclosed by arrow brackets, such as "<parameter>," is a variable. An appropriate value for the variable must be used in the command. Appropriate values to be substituted are described as needed in this guide.

[parameter] Optional argument. A command argument enclosed by square brackets, such as "[parameter]," is an optional argument. The effect of including, or not including, this argument in a command is described in this guide.

[<parameter>] Optional variable argument. A command argument enclosed by first by arrow brackets and then by square brackets, such as "[<parameter>]," is an optional variable argument.

| "OR" operator. Command arguments separated by "|" mean that either one or the other argument is to be used. For example, "<host>|clear" means that you are to either enter the variable <host> or the literal

argument "clear."

3.2. Notation Used in this Guide

The following is the notation to be used in describing the commands in this
guide.

<ip_addr> The numeric IP address of a host in dotted decimal notation enclosed by brackets, e.g. [44.2.0.100].

<host_name> The symbolic name of a host.

<host> Denotes a host, switch, or gateway, which may be specified either as a symbolic name (<host_name>), or as a numeric IP address (<ip_addr>). The mappings between IP addresses and symbolic names are defined in the file "HOSTS.NET" described in section A.4 of Appendix A.

<callsign> An amateur callsign, either in upper or lower case.

<directory> The name of an existing directory on the host computer. Directory references can either be relative to the current directory, or absolute, beginning at the root (\). To refer to the parent directory, ".." can be used.

<file> The name of a file, e.g. HOSTS.NET.

<value> An integer number.

<cr> A carriage return, usually marked "Return" on most keyboards. Note that all commands given in this guide must be followed by a carriage return, although the <cr> notation will not be used in this case. <cr> is only used when the "command" needed is a carriage return on an otherwise empty line.

<F10> The command key labeled "F10" or "f10" at the top or left side of the keyboard.

4. NET

The program that implements the Internet protocols is NET.EXE. In this section, information on executing NET, its command and converse modes, escaping back to DOS, NET utility commands, managing multiple sessions and exiting NET are provided. Information on the major NET commands is given in later sections.

4.1. Executing NET

NET is usually invoked by simply typing the following at a DOS prompt:

```
net>
```

If the software has been installed correctly, NET attempts to open the configuration file "AUTOEXEC.NET" in the root directory of the current drive. This file is described in section A.3 of Appendix A. You should then be presented with a banner including revision information and a copyright statement, followed by a prompt of "net>." If you don't get this, something is wrong. Check your installation to see if you missed something. If you still have troubles, find a NET user and ask for help.

NET can also be invoked by typing:

```
net <file>
```

NET will first attempt to open <file> as an alternate configuration file, which is read instead of AUTOEXEC.NET.

4.2. Command and Converse Modes

The program may be in one of two modes: command mode or converse mode. In command mode, the prompt "net>" is displayed and any of the NET

commands described in this guide may be entered. In converse mode, keyboard input is processed according to the "current session," which may be either a telnet, ftp, or AX.25 connection. In a telnet or AX.25 session, keyboard input is sent to the remote host and any output from the remote host is displayed on the console. In an ftp session, only ftp converse mode commands may be entered. In these sessions, the user remains in converse mode until either the session is terminated (described in the sections of this guide dealing with telnet, ftp, or AX.25) or by escaping back to command mode, as described below.

The user may escape back to command mode from converse mode by pressing <F10>. The command mode prompt "net>" will be displayed and any of the NET commands may be entered. The session that the user "escaped" from will remain active. By entering <cr> at a "net>" prompt, the user will return to converse mode in the "current session." Multiple sessions can be handled by NET. For more information on multiple sessions, see section 4.5 below.

4.3. Executing DOS Commands

While running NET, you may need to execute some DOS commands or to run a program such as BM. This can be done by suspending NET execution and then returning to NET when you are finished. To suspend NET, enter at a "net>" prompt:

```
!
```

You will be returned to your DOS prompt and you can execute any DOS command. Note that this will suspend all NET sessions. To return to NET, enter at an DOS prompt:

```
exit
```

You will be returned to the "net>" prompt.

Note that the command "shell" is the same as the "!" command.

4.4. Utility Commands

NET provides several utility commands to provide help information, manage directories, and to start and stop the servers. These commands are useful in a variety of sessions.

4.4.1. help

Displays a list of the main NET commands. The command "?" is equivalent to "help." Note that several commands are listed that are not described in this guide. Concise descriptions of the commands listed by help are given in Appendix B.

4.4.2. pwd

Displays the name of the current directory on the local machine.

4.4.3. cd <directory>

Changes the current directory to <directory>, which must be an existing directory on the local machine. The directory specified can be relative to the current directory, or absolute, with the name beginning at the root (\).

4.4.4. dir [<directory>]

List the contents of the specified directory on the console. If no argument is given, the current directory is listed.

4.4.5. start <server>

Starts the specified Internet server, allowing remote connection requests. Servers include: finger, ftp, remote, smtp, and telnet. Normally these servers are started in the AUTOEXEC.NET file. However, you may not want to start all of the servers automatically. For example, you may not want to start telnet until you know you will be available at the keyboard to respond. You can then enter "start telnet" from the "net>" prompt to activate the telnet server.

4.4.6. stop <server>

Stops the specified Internet server, rejecting any further remote connect requests. Existing connections are allowed to complete normally. For example, you could enter "stop telnet" if you expect to be unavailable to respond to a telnet request. Then, when anyone tries to telnet to you, they will get the message "Closed (Reset)." This is not the most informative message, but possibly better than waiting endlessly to see if you will respond to the telnet request.

4.5. Managing Multiple Sessions

The NET program can handle multiple sessions. For example, you can have an ftp file transfer running at the same time as a telnet session. However, you should limit your use of multiple sessions on a 1200

baud radio channel, as you will cause considerable congestion.

To start a second session, escape from the first session to the "net>" prompt by pressing <F10>. Start the second session as you would normally (as described in the sections of this guide dealing with telnet, ftp, or AX.25).

To monitor the multiple sessions, use the "session" command from the "net>" prompt. The syntax for this command is:

```
session [<session #>]
```

Without arguments, "session" displays a list of current sessions, including session number, remote TCP or AX.25 address and the address of the TCP or AX.25 control block. An asterisk (*) is shown next to the "current" session; entering <cr> at this point will put you in converse mode with that session. Entering a session number as an argument to the session command will put you in converse mode with that session. If the telnet server is enabled, the user is notified of an incoming request and a session number is automatically assigned. The user may then select the session normally to converse with the remote user as though the session had been locally initiated. An example of a session list:

```
#         &CB   Type    Rcv-Q   State          Remote
```

```
socket
0        8ac14     FTP          69   Established
eyolo:ftp
*1       8b3d4    AX25           0   Connected
n6spe-1
```

4.6. Abbreviating NET Command Names

Many of the NET command names can be abbreviated. However, the valid abbreviations have not been documented. Further, there are some cases where the abbreviated (or lengthened) name will appear to be accepted as a valid command name, but the command will not execute properly. Thus, it will be left to the user to experiment with the abbreviations. It is recommended that you use the command names given in this guide, as they have been tested and found to work as
described.

4.7. Exiting NET

Before you exit NET, you should check to see if you have any sessions active. This is done with the "tcp status" command, described in section 10.1.1.

When you are sure you want to exit the NET program and return to DOS, enter at a "net>" prompt:

```
exit
```

5. Telnet

The telnet command allows you to initiate a keyboard connection using the telnet protocol. The end result is the same as doing an AX.25 connect in most cases, but

you'll be taking advantage of the attributes of the TCP/IP protocols, as described in the introduction.

5.1. Initiating a Telnet Session

The command to initiate a telnet session with the specified host and enter telnet converse mode is:

```
telnet <host>
```

For example:

```
telnet n3eua                (talk to N3EUA,
address in HOSTS.NET)
telnet [44.32.0.4]      (use the numeric
address directly)
```

If the connection is made, you can type back and forth just as if you were connected with a normal TNC. When you're done, use the <F10> key to escape back to command mode, and then type 'close' to close the connection, as described in section 5.4 below.

5.2. Accepting a Telnet Session

If a remote host requests a telnet session, a message similar to the following will be displayed on your console:

```
Incoming Telnet session 0 from 44.2.0.96:1026
```

If you are in command mode, enter <cr> at a "net>" prompt and you will enter converse mode for the telnet session. If you are in converse mode, use the <F10> key to escape back to command mode, use the "session" command to list the active sessions, and then use the "session <session#>" command to enter converse mode on the desired telnet session.

5.3. File Upload and Download

The telnet session can be recorded to a file, or an ASCII file can be uploaded instead of entering the information at the keyboard.

5.3.1. record <file>|off

Opens <file> and appends to it all data received or sent on the current telnet session. If you are in telnet converse mode and want to initiate recording, you will need to use the <F10> key to escape back to command mode to issue the record command. The message "Recording into <file>" will be displayed and

another "net>" prompt will be issued. Enter <cr> on a blank line and you will return to the telnet converse mode with recording activated. The command "record off" stops recording and closes the file.

5.3.2. upload <file>

Opens <file> (must be an ASCII, file, not a binary file) and sends it on the current telnet session as though it were typed on the terminal. If you are in telnet converse mode and want to initiate uploading, you will need to use the <F10> key to escape back to command mode to issue the upload command. The uploading is initiated, but the file contents are not displayed on the screen during the uploading. When the uploading is complete, the message "Uploading off" is displayed. Enter <cr> on a blank line at the "net>" prompt and you will return to the telnet converse mode.

5.4. Closing a Telnet Session

To close a telnet session, the following command is used:

```
close [<session #>]
```

If you are in telnet converse mode, you will have to press <F10> to escape to the "net>" prompt to issue this command. If you are running only one session,

entering close without arguments will close the session. If you have multiple sessions, entering close without arguments will initiate a close on the current session. If you are running multiple sessions, the "session" command will display a list of these sessions. Entering close with a session number argument will initiate a close on the specified session. Only one of the hosts involved in the telnet session needs to initiate the close. "Disconnect" is functionally the same command as "close."

6. Mail

One of the most useful features of TCP/IP is electronic mail. Mail can be delivered to your unattended machine and you can read it at your leisure. There is no need to log into a PBBS to pick up your messages. Your messages can also be temporarily stored on your local switch and be delivered to your machine when you run NET.

Mail messages are composed and read and the mailbox is managed with the program BM. NET, using the protocol SMTP (simple mail transport protocol) sends and receives the mail. This section concentrates primarily on BM, but a few comments on SMTP are given in subsection 6.5.

The BM.EXE mail user interface program was created by Bdale Garbee, N3EUA, and despite popular belief, "BM" really stands for "Bdale's Mailer." It was later

extended by Gerard van der Grinten, PA0GRI, and Dave Trulli, NN2Z.

6.1. Executing BM

BM must be executed from a DOS prompt. If you are currently running NET, you will have to escape to a DOS subshell by entering "!" or "shell." You will be returned to your DOS prompt and you can execute BM. Note that this will suspend all NET sessions. When you are finished with BM, to return to NET, enter "exit" at a DOS prompt and you will be returned to the "net>" prompt.

BM is normally invoked by simply typing the following from a DOS prompt:

```
bm
```

BM will first read the mail configuration file, BM.RC, described in section A.5 of Appendix A, and then will read the default mailbox defined in this configuration file. Using the BM main menu commands described in section 6.2, you can then compose or read mail messages or manage your mailbox.

BM may also be invoked in two other ways. The first is:

```
bm -u <mailbox> | -f <file>
```

With the argument "-u <mailbox>," you can specify which mailbox to read, overriding the default from BM.RC. With the argument "-f <file>," BM will read the messages from <file> instead of a mailbox. This is useful if you have saved previously received messages to a file, using the "s" command described in section 6.2.6.

When you invoke BM, a banner will be displayed, followed by two lines of copyright information and then the mail header information, as described in section 6.2.1 below. Finally, there is a line reminding you to "Type ? for help" and then a prompt, consisting of your user name in quotes followed by an arrow (>).

If you are only interested in composing a mail message, you can invoke BM as:

```
bm <recip1> ..  ..  <recipN>
```

The arguments <recip1> to <recipN> are the mail addresses of your desired recipients. Information on mail addresses is given in section 6.4. You will be prompted to give a message subject and then to enter your message. When you have finished composing your message, you will be returned to your DOS prompt.

6.2. BM Main Menu Commands

All BM main menu commands are single letters

followed by optional arguments.

A <msglist> is a space-delimited list of message numbers, for example:

```
    1 3 4 5
```

The message numbers are given in the message headers.

6.2.1. h

Display message headers. The message headers contain the message number, the status indicating whether it has been read or deleted, the sender, size, date, and subject. For example:

Mailbox \spool\mail\n6gf.txt - 3 messages, 1 new

```
DY  1 n6dbt@n6dbt           02 Apr 16:31 666
Friday Night Pat Sajak
Y  2 N6SPE@n6spe.ampr.org 02 Apr 17:40 576 Re:
eyolo

> N  3  N6SPE@n6spe.ampr.org      02 Apr 17:42
942 Re: NET log
```

In the first line above, "D" indicates that the message has been marked for deletion and "Y" indicates that it has been read. The message number is 1, the sender is n6dbt@n6dbt, it was sent on April 2 at 16:31, is 666

characters long and the subject is

"Friday Night Pat Sajak." In the third line above, ">" indicates that this is the current message and "N" means that it has not yet been read.

6.2.2. ?

Display a help menu for BM commands.

6.2.3. [<msg#>]

Entering a message number from the header listing will cause the message text to be displayed. If a carriage return is entered on a blank line, the current message will be displayed.

6.2.4. d [<msglist>]

Mark messages for deletion. With no argument, the current message (indicated by ">" in the first column of the message header) is marked for deletion. Messages marked for deletion are removed when exiting BM via the 'q' command, when changing to an alternate mailbox with the 'n' command, or when updating with the '$' command.

6.2.5. u [<msglist>]

Undelete a message that is marked for deletion. The status of a message can be determined by looking at the status field of the message using the 'h' command.

6.2.6. s [<msglist>] [<file>]

The 's' command is used to save messages in a file. If no file name is given the default from the mbox variable in BM.RC is used. If no message number is supplied then the current message is saved. If <file> does not exist, it will be created. If <file> does exist, the messages will be appended to this file. The messages are stored in the same format as a mailbox file with all mail headers left intact. This file can later be read into BM by invoking BM with the "-f <file>" argument.

6.2.7. w [<msglist>] <file>

The "w" command is used to save messages in a file. Only the message body is saved. All mail headers are removed. If no message number is supplied then the current message is saved.

6.2.8. p [<msglist>]

The "p" command is used to send messages to the printer. This command uses the DOS device PRN for output. If no message number is supplied then the current message is printed.

6.2.9. m [<recipient_list>]

The "m" command is used to compose a message to be mailed to the list of recipients, specified by mail addresses or aliases, which are described in section 6.4. All local recipient addresses (those which don't contain an '@') are checked for possible aliases in the ALIAS file, described in section A.7 of Appendix A. Each alias that is found, is expanded into its list of addresses. If no recipient list is supplied with the "m" command, you will be prompted for a recipient list.

When you are composing a message, several commands are available such as invoking an editor or reading in text from other messages or files, as described in section 6.3. To end a message enter a line containing a single period.

It is important to remember that the input line buffer has a 128 character limit. You should format your text by entering a carriage return at the end of each line. Typing excessively long lines may cause data loss due to truncation when passing the message through other hosts. Keeping lines less than 80 characters is always a good idea.

6.2.10. r [<msg#>]

Reply to a message. Reply reads the header information in order to construct a reply to the sender.

The destination information is taken from the "From:" or the "Reply-To:" header, if included. If no message number is supplied the current message is used.

6.2.11. f [<msg#>]

The 'f' command is used to forward a mail message to another recipient. If no message number is supplied the current message is used. The user is prompted for the recipients and a subject. The mail header is added to the message text while retaining the complete original message in the body. Also see the ~m command.

6.2.12. b [<msg#>]

Bounce a message. Bounce is similar to forwarding but instead of your user information, the original sender information is maintained. If no message number is supplied the current message is used.

6.2.13. n [<mailbox>]

Display or change the mailbox. The 'n' command with no arguments will display a list of mailboxes containing mail. If an argument is supplied, then the current mailbox is closed and a new mailbox is opened.

6.2.14. l

List outbound messages. The job number, the sender, and the destination for each message is displayed. A status of "L" will appear if the SMTP sender has the file locked, meaning the message has begun to be sent to its destination. If you find that a message has been in the locked state for a long period of time, the corresponding SMTP session may be "stuck." See section 10.1 for a description of how to monitor the status of SMTP sessions and to reset them if they get stuck.

6.2.15. k [<msglist>]

Remove an outbound message from the send queue. A message can be removed from the send queue by specifying the job number obtained by the l command. If the message is locked you will be warned that you may be removing a file that is currently being sent by SMTP. It is best not to remove locked messages. They are better handled with the "tcp reset" command, described in section 10.1.2.

6.2.16. $

Update the mailbox, deleting messages marked for deletion. If you have a large number of messages in your mailbox and are cleaning it up by marking unwanted messages for deletion, updating the mailbox will shorten the header list.

6.2.17. x

Exit to DOS without changing the data in the mailbox. Messages marked for deletion will NOT be deleted.

6.2.18. q

Quit to DOS updating the mailbox. Messages marked for deletion WILL be deleted.

6.3. Text Input Commands

The following commands are available while composing a message. The tilde character (~) must be entered as the first character on a line.

~r <file> Read <file> into the message buffer.

~m <msg#> Read <msg#> into the message buffer.

~p Display the text in the message buffer.

~e Invoke the editor defined in BM.RC with a temporary file containing the text in the message buffer.

~q Abort the current message. No data is sent.

~~ Insert a single tilde character into the message.

~? Display the help menu of tilde escape commands.

6.4. Mail Addresses

Mail is addressed to a recipient, which is either a user name defined in a BM.RC file (described in section A.5 of Appendix A) or an "alias," which is an alternative name for one or more users. Aliases are defined in the ALIAS file, described in section A.7 of Appendix A. These recipients can be on the local host or a remote host.

Mail addressing varies from simple to mildly confusing. The simplest form is:

```
<recipient>[@<host>]
```

If @<host> is not included, BM will search to see if <recipient> is in the ALIAS file on the local host. If it is, it will be expanded to the recipient list given for the alias. If <recipient> is not given in the alias file, the message will be sent to <recipient> on the local host (this is probably not what you intended, unless you have more than one user on your host).

If the address <recipient>@<host> is used, the message will be sent to <host>, where <recipient> is first looked up in the ALIAS file on <host>. If <recipient> is found to be an alias, it will be expanded to the recipient list given for the alias, and the message will be forwarded to these recipients. If <recipient> is not found in the alias file, the message will be delivered to <recipient> on <host>.

Host names can be found in your HOSTS.NET file, described in section A.4 of Appendix A. Valid mail user names for a given host can be found using the finger command, as described in section 9.2, if the finger files have been set up on the this host. If an ALIAS file has been distributed for your area, mail addresses and aliases will be found there. Otherwise, you will have to contact the intended recipient and ask for his or her address.

If the remote host of the recipient is not on the air when you try to send the message, it will remain in your mail queue until some time when both hosts are on the air at the time you attempt to send the message. To avoid this delay, switches have been set up in many areas which run 24 hours a day and can be used for mail forwarding. If the switch your recipient communicates with is <switch> (a host name), then the mail can be addressed as:

```
<recipient>%<host>@<switch>
```

Your mail will be transferred to <switch> and then forwarded to <recipient>@<host>, using the stored route to <host>.

6.5. SMTP

NET sends and receives mail using the simple mail transport protocol (SMTP). This is handled automatically by NET, although, you may want to "kick" out your outgoing mail manually, as described below.

When mail is received, SMTP displays the message:

```
New mail arrived for <user>
```

where <user> is the addressee of the received mail.

If you have included the "smtp timer" command in your AUTOEXEC.NET file, SMTP will check your outbound mail queue at the time interval you have set to determine if there is any new outgoing mail that should be sent. If you haven't included this command, or you want to send out your mail before the next timer interval, you can manually "kick" out your outgoing mail, as described below.

6.5.1. smtp kick

This command will review the outgoing mail queue and attempt to deliver any pending mail. This

command allows the user to "kick" the mail system manually. This command can be entered at a "net>" prompt after you have composed mail messages.

7. File Transfer

The ftp command provides for the transfer of files using the file transport protocol. It enables you to do the following:

Transfer text and binary files between local and remote host
List directories on a remote host
Delete files on a remote host
Create and remove directories on a remote host

The remote host can be unattended and the ftp server on that host will provide the requested services.

File access privileges are defined in the FTPUSERS file, described in section A.8 of Appendix A. This file defines user login names, passwords, directories to be accessed and file access privileges. It is a common convention to allow arbitrary users limited access to files under the special user names "anonymous" or "guest."

7.1. Ftp Command

The command ftp is used to initiate an ftp session. It is invoked as:

```
ftp <host>
```

where <host> is the desired remote host. If the session is established, you will enter converse mode on the new ftp session. When in converse mode with an ftp server, only the ftp commands described below will be valid. This will remain true until the ftp "quit" command is issued, ending the ftp session, and returning you to the "net>" prompt.

When the connection between the two machines is opened, you'll get a banner from the remote machine, followed by a prompt for your user name and then your password. If you've negotiated with the person at the remote machine to have a special user name and password set up for you in his or her FTPUSERS file, use that. If not, use one of the special user names, "anonymous" or "guest," and in this case, use your call sign as your password. Your password is recorded in the log file on the remote host, allowing the manager of that host to keep track of ftp activity.

7.2. Ftp Converse Mode Commands

The following are the ftp commands that are valid in ftp converse mode, which you enter after your

password is accepted.

7.2.1. dir [<file>|<directory> [<localfile>]]

Without arguments, "dir" requests that a full directory listing of the remote server's current directory be sent to your display. If one argument is given, it is interpreted as a specific file or sub-directory on the remote file system that is to be listed. If two arguments are given, the second is taken as the local file into which the directory listing should be written (instead of being sent to the display). The full listing gives the file names, sizes, and creation dates. You should request a directory listing when you first log into an unfamiliar machine. There will often be a file named "README" or "whathere.txt" that will give some information about the files available on the remote machine. This file can then be acquired with a "get" command (described below), and read on your machine to learn more about the files available on the remote host.

7.2.2. ls [<file>|<directory> [<localfile>]]

ls is identical to the "dir" command except that an abbreviated directory listing is provided. This listing gives only the file names.

7.2.3. pwd

Displays the name of the current directory on the remote host.

7.2.4. cd <directory>

Changes the current directory on the remote host to the directory indicated by <directory>, which must be an existing directory on the remote host. The directory specified can be relative to the current directory, or absolute, with the name beginning at the root (\).

7.2.5. get <remote_file> [<local_file>]

Asks the remote host to send the file specified in the first argument and to write the file on the local machine. The second argument, if given, will be the name of the file on the local machine; otherwise it will have the same name as on the remote host. See the "type" command below if the file requested is other than an ASCII file. If the file is over 10,000 characters in size, you should only start the transfer when the radio channel is relatively quiet. Use the "abort" command below if you want to terminate the transfer before it has been completed.

7.2.6. put <local_file> [<remote_file>]

Asks the local host to send the file specified in the first

argument and to write the file on the remote machine. The second argument, if given, will be the name of the file on the remote host; otherwise it will have the same name as on the local machine. You must have write privilege on the remote host to use this command. Use the "abort" command below if you want to terminate the transfer before it has been completed. See the "type" command below if the file to be sent is other than an ASCII file.

7.2.7. quit

Terminates the ftp session and returns you to the "net>" prompt.

7.2.8. abort

Aborts a get, put, dir or ls operation in progress. This is the only acceptable command when these operations are in progress; all other commands will result in an error message. Abort is valid only when a transfer is in progress. When a get or put operation is aborted, a partial copy of the transferred file will be left on the destination machine, which must be removed manually if it is unwanted. This is also true for a dir or ls operation when the directory listing is written as a local file.

7.2.9. type [a|i]

Tells both the local and remote hosts the type of file that is to be transferred. Without arguments, the current mode is displayed. The default is "a", which means ASCII (i.e., a text file). In "i" mode, which means IMAGE, files are sent exactly as they appear in the file system. This mode must be used when exchanging raw binary files (executables, compressed archives, etc). The file type must be set before a "get" or "put" command is initiated. The file type remains in effect until it is changed by a subsequent "type" command.

7.2.10. dele <remote_file>

Deletes a file on the remote machine. You must have delete privilege on the remote host to use this command.

7.2.11. mkdir <remote_directory>

Creates a directory with the name <remote_directory> on the remote machine. You must have write privilege on the remote host to use this command.

7.2.12. rmdir <remote_directory>

Deletes <remote_directory> on the remote machine. The remote directory must be empty before you can

remove it. You must have delete privilege on the remote host to use this command.

7.3. Ftp Example

In the ftp example below, the user initiates an ftp session with the remote host "eyolo," logs in as the user "anonymous," requests a directory listing, changes to subdirectory, gets a binary file and terminates the session. The text given in parentheses to the right of the commands indicate what has been typed by the user.

```
net> ftp eyolo
(ftp eyolo)
SYN sent
Established
220 eyolo.ampr.org FTP version 89042.1 ready
at Mon Mar 26 16:16:54 1990
Enter user name: anonymous
(anonymous)
331 Enter PASS command
Password: n6gf
(n6gf)
230 Logged in
dir
(dir)
200 Port command okay
150 opening data connection for LIST \public
docs\                  16:42  3/02/90  lists\
16:40  3/02/90
```

```
programs\              14:40  3/02/90  utility\
16:42  3/02/90
whathere.txt   1,755  18:22  3/21/90
```

```
5 files. 6,959,104 bytes free.  Disk size
10,584,064 bytes.
Get complete, 265 bytes received
226 File sent OK
cd programs
(cd programs)
257 "\public\programs" is current directory
dir
(dir)
200 Port command okay
150 Opening data connection for LIST \public\
programs
bm.exe   41,225  17:55  2/25/90  net.exe
174,454  17:43  2/25/90
2 files. 6,959,104 bytes free.  Disk size
10,584,064 bytes.
Get complete, 135 bytes received
226 File sent OK
type i
(type i)
200 Type OK
get bm.exe
(get bm.exe)
200 Port command okay
150 opening data connection for RETR bm.exe
Get complete 41225 bytes received
226 File sent OK
quit
(quit)
221 Goodbye!
Close wait
Last ACK
```

```
Closed (Normal)
net>
```

The user's callsign was used as the password, which is shown in this example. However, the password is not echoed to the screen by the software. Note that there are no prompts for ftp in the converse mode. After you receive the message "230 Logged in" you can issue ftp commands. The display generated by the "dir" command in this example shows that the user was logged into the \public directory. The listing shows that there is one file, named "whathere.txt," of size 1755 bytes, created at 18:40 on 3/21/90. There are also four subdirectories, indicated by "\" at the end of their names: "docs\," "lists\," "programs\," and "utility\," all created on 3/2/90. The dir output is finished with the "226 File sent OK" message and the user can then issue another ftp command. The command "cd programs" is issued to change to the subdirectory programs. A dir command on this subdirectory shows that there are two files, "bm.exe" and "net.exe." These are executable programs since the file name extensions are "exe" and therefore, they are binary files. The "type i" command is issued so that a binary file can be transferred. The "get" command is issued and there will be a delay as the "bm.exe" file is retrieved. This is also finished when the message "226 File sent OK" is received.

8. AX.25 Services

NET provides AX.25 services, better known as the standard packet radio protocol that you probably used before you switched to TCP/IP. This allows you to use NET to move to another frequency and check into the local AX.25 PBBS, or to initiate a keyboard session with a friend who hasn't been convinced to switch to TCP/IP yet.

In addition, there is an AX.25 mailbox, allowing that same friend to connect to your system and initiate a keyboard session, or to send a message to anyone reachable on TCP/IP.

8.1. Initiating an AX.25 Connection

The connect command is used to initiate an AX.25 connection. The syntax is:

```
connect ax0 <callsign> [<digipeater> ...  ]
```

This initiates an AX.25 session to the specified call sign. Up to 7 optional digipeaters may be given; note that the word "via" is NOT needed. Data sent on this session goes out in conventional AX.25 packets with no upper layer protocol. The de-facto presentation standard format is used, in that each packet holds one line of text, terminated by a carriage return. Two examples are:

```
connect ax0 n3eua                  (connect direct
to N3EUA)
connect ax0 n3eua n1fed n0ccz  (conn to N3EUA
via N1FED and N0CCZ)
```

If all is well, you should get "Conn Pending" and then "Connected" messages. At this point, you're connected just like using a plain old TNC.

When you're ready to disconnect, use the <F10> key

to escape from the session back to the 'net>' prompt, and then type 'disconnect', as described in section 8.3.

8.2. File Upload and Download

AX.25 sessions can be recorded to a file and a file can be uploaded in place of typing the information on the keyboard.

8.2.1. record <file>|off

Opens <file> and appends to it all data received or sent on the current AX.25 session. If you are in AX.25 converse mode and want to initiate recording, you will need to use the <F10> key to escape back to command mode to issue the record command. The message "Recording into <file>" will be displayed and another "net>" prompt will be issued. Enter <cr> on a blank line and you will return to the AX.25 converse mode with recording activated. The command "record off" stops recording and closes the file.

8.2.2. upload <file>

Opens <file> and sends it on the current AX.25 session as though it were typed on the terminal. If you are in AX.25 converse mode and want to initiate uploading, you will need to use the <F10> key to escape back to command mode to issue the upload command. The uploading is initiated, but the file contents are not displayed

on the screen during the uploading. When the uploading is complete, the message "Uploading off" is displayed. Enter <cr> on a blank line at the "net>" prompt and you will return to the AX.25 converse mode.

8.3. Terminating an AX.25 Connection

To terminate an AX.25 connection, use the following command:

```
disconnect [<session #>]
```

If you are in AX.25 converse mode, press <F10> to escape back to the "net>" prompt to issue this command. If you are running only one session, entering disconnect without arguments will terminate the connection. If you have multiple sessions, entering disconnect without arguments will initiate a close on the current session. If you are running multiple sessions, the "session" command will display a list of these sessions. Entering disconnect with a session number argument will initiate a close on the specified session. After entering disconnect, you should get "Disc pending" and then "Disconnected" messages. Note that "disconnect" is the same as the "close" command and that the two command names can be used fully interchangeably.

8.4. AX.25 Mailbox

If your AUTOEXEC.NET file (described in section A.3 of Appendix A) contains the command "mbox on," then your AX.25 mailbox will be accessed when someone running the standard AX.25 packet protocol connects to you. When the connection is made, the remote user must first enter <cr> and then a banner and prompt similar to the following will be displayed:

```
Welcome to the n6gf.ampr.org TCP/IP Mailbox
(C)hat, (S)end, (B)ye >
```

If the user chooses (C)hat, an AX.25 keyboard connection with your system is requested. A message similar to the following will be displayed on your console:

```
Incoming AX25 session 0 from N6QGG
```

If you are in command mode, enter <cr> at a "net>" prompt and you will enter converse mode for the AX.25 keyboard session. If you are in converse mode, use the <F10> key to escape back to command mode, use the "session" command to list the active sessions, and then use the "session <session#>" command to enter converse mode on the desired AX.25 session. When you are finished with the chat, you can use the <F10> key to escape from the session back to the 'net>' prompt, and then type 'disconnect', as described above. Alternatively, the AX.25 user who initiated the session can terminate it by disconnecting in the standard way.

The syntax for the (S)end command is:

```
S <recipient>[@host] [< from_addr ] [$bulletin_id]
```

With one argument, this will send a message to the specified mail address. See sections 6.2.9 and 6.4 for a description of mail addresses. The user will be prompted for a subject, then asked to enter the message. Instructions are given for ending the message with ^Z or /EX beginning a line, the same as that for AX.25 PBBS's. With two arguments, the user can send a message into the TCP/IP network, using the addressing scheme described in Section 6.4 of this guide. The third and fourth arguments are primarily for use in PBBS forwarding and will not be described here. After sending a message, the mailbox command prompt will be displayed to the user again. Entering B (for Bye) will terminate the mailbox session.

9. Monitoring Activity

Several commands are available to monitor activity on the packet radio channel and to acquire information about a remote host.

9.1. ax25 heard [on|off|clear]

Works like the "mheard" function in many TNC's. The command "ax heard," with no parameters displays the list of callsigns heard and with options "on" and "off"

you control whether the list is updated or not. With the option "clear," you clear the list of callsigns. If you are interested in monitoring the channel with this command, include "ax25 heard on" in your AUTOEXEC.NET file, as described in section A.3.20 of Appendix A.

An example listing:

```
Heard list: Sat Mar 31 15:09:23 1990

KB6RIH     ARP  NETROM  IP    Sat Mar 31
15:08:07 1990
KA6FUB-3        NETROM        Sat Mar 31
15:04:57 1990
HIGH         Sat Mar 31 14:25:06 1990
N6VV-15      Sat Mar 31 14:05:16 1990
N6SPE-1    ARP           IP    Sat Mar 31
12:49:30 1990
N6QWS       (via HIGH)  Sat Mar 31 12:36:59
1990
N6QGG        Sat Mar 31 12:20:51 1990
```

KB6RIH is running NET, using the address resolution protocol (ARP), NETROM, and the internet protocol (IP). KA6FUB-3 is a NETROM node. HIGH is an AX.25 digipeater. N6VV-15 was being forwarded through the KA6FUB-3 NETROM node. The only clue from this listing that this was a NETROM forward is the SSID (-15) on N6VV's callsign. N6SPE-1 is also running NET, using ARP and IP. N6QWS was heard

digipeated through HIGH. Finally, N6QGG was heard direct on AX.25.

9.2. finger

This command allows you to "finger" information files on your host or on a remote host. Finger files are described in section A.9 of Appendix A.

The syntax for the finger command is:

```
finger [<finger_file>][@<host>]
```

where <finger_file> is the name of the finger information file you wish to query and <host> is the name of the remote host where the file resides.

If you issue the command in the following form:

```
finger <finger_file>
```

you can query information from a finger file on the local host, namely your own system. This is useful for testing finger on a system that you know is running.

The command in the following form:

```
finger @<host>
```

is used to acquire the names of the finger files available on <host>. This command returns a list of all finger files on the remote computer.

Finally, issuing the command as:

```
finger <finger_file>@<host>
```

will display <finger_file> from <host>.

When you have been fingered by a remote host, a message similar to the following will be displayed:

You're being fingered by 44.2.0.98:1001!

9.3. ping

This command is used to see if a remote host is on the air and if so, to determine the quality of the path between the local and remote host. The syntax for this command is:

```
ping [<host>|clear] [<interval>]
```

When the command is issued in the form:

```
ping <host>
```

the remote host given in the argument is queried once. If it returns an echo, the IP number of the host and the round trip time required are displayed. For example:

```
44.2.0.96: echo reply id 0 seq 30522, 5508
ms
```

In this case, the round trip to remote host [44.2.0.96] took 5508 ms, or about 5.5 seconds. If no echo is received, due to the host being off the air, a poor path to the host, or a packet collision, nothing is displayed.

Issuing the command in the form:

```
ping <host> <interval>
```

sets up a repetitive test, where the remote host is queried at a time interval given by the number in the second argument, interpreted in seconds. Repetitive queries can be set up for several hosts at once, by issuing a command for each host. Users should be careful not to overdo this testing, as the ping queries will add to channel congestion. The current results are displayed with the third form of the ping command, and the repetitive queries continue until the "ping clear" command is issued.

Entering the command "ping," without an argument,

displays a table of the current results of the repetitive queries, listing the IP numbers of the remote hosts, number of ping queries sent, number of ping echoes received, percent of queries echoed, average round trip time and the ping interval time. For example:

```
Host            Sent    Rcvd   %   Avg RTT
Interval
44.2.0.96         18      17  94      6596
60
44.2.0.98         18      18 100      3209
60
```

Finally, issuing the command in the form:

```
ping clear
```

cancels the repetitive ping requests and clears the table of ping query results.

9.4. trace

The trace command is used to monitor the activity on the channel. The syntax of this command is:

```
trace [ax0 [<flags>]|allmode|cmdmode]
```

The flags enable or disable tracing and determine the amount of information displayed. Without arguments, trace gives a list of all defined interfaces and their

tracing status. This guide only considers the use of a single interface, "ax0." The flags are given as a hexadecimal number which is interpreted as follows:

```
TIO
|||---  Enable tracing of output packets if 1,
       disable if 0
||----  Enable tracing of input packets if 1, disable
       if 0
|-----   Controls type of tracing:
```

0 - Protocol headers are decoded, but data is not displayed

1 - Protocol headers are decoded, and data (but not the headers themselves) are displayed as ASCII characters, 64 characters/line. Unprintable characters are displayed as periods.

2 - Protocol headers are decoded, and the entire packet (headers AND data) is also displayed in hexadecimal and ASCII, 16 characters per line.

There is an additional option for tracing that allows you to select whether traced packets are always displayed, or only displayed when you are in command mode. Having tracing only happen in command mode sometimes provides the right mix between "knowing what's going on," and "keeping the garbage off the screen" while you're typing. To select tracing all the time (the default mode), use 'trace allmode'. To

restrict tracing to command mode, use 'trace cmdmode'.

For example, to trace the activity on interface ax0, with an ASCII display:

```
trace ax0 111
```

To turn off this tracing:

```
trace ax0 000
```

10. Advanced Topics

While this document is intended to be a guide for beginners, there are a few advanced topics of interest to many users that should be mentioned. The presentation of these topics is concise and the reader should consult "The KA9Q Internet Software Package," by Bdale Garbee, for details.

Note that the commands included in the NET configuration file, AUTOEXEC.NET, and described in Appendix A, can also be used interactively. Further, these commands, when issued without their variable arguments, will report the current values of these arguments. For example, the "route" command, issued without arguments, will display the current routing table. These commands, issued with arguments, can be used to alter the configuration of NET while it is running. For example, "route add,"

described in section A.3.8., can be used to add entries to the routing table, possibly to experiment with alternative routes. To drop routes that are found to be unreliable, the following command can be used:

```
route drop <dest_host>
```

where <dest_host> is the IP address or host name for the destination of your packets. Note that if you find better routing methods with this experimentation, you will have to edit the "route add" commands in your AUTOEXEC.NET file for this routing table to be in effect the next time you run NET.

10.1. TCP

The TCP command can be used to monitor and control session status at a lower level than provided elsewhere in the KA9Q software package.

10.1.1. tcp status [<tcb_addr>]

Issued without the optional argument, this command displays the status table for TCP sessions. For example:

```
conout 102 conin 109 reset out 5 runt 0 chksum
err 3 bdcasts 0
  &TCB  Rcv-Q Snd-Q  Local socket      Remote
socket  State

2e90e444     0     0  44.2.100:79
0.0.0.0       Listen (S)
2e90e59c     0     0  44.2.100:25
0.0.0.0       Listen (S)
```

```
2e90e790     0     0  44.2.100:23
0.0.0.0       Listen (S)
2e90eafc     0   335  44.2.100:1001
44.2.0.32:25  Established
2e90e4fc     0     0  44.2.100:21
0.0.0.0       Listen (S)
```

The first line gives some TCP-level statistics, including the number of outbound and inbound connections to your host. The table below gives a summary of all existing TCP connections. "&TCB" is the TCP address, "Rcv-Q" and "Snd-Q" are the numbers of characters in the receiving and sending queues, "Local socket" and "Remote socket" give the IP address and port of the local and remote host, "State" gives the state of the session. Note that the remote socket IP address is given as 0.0.0.0 for the listening servers. Each session is assigned to a port. The server ports are 79 for finger, 25 for SMTP, 23 for Telnet, and 21 for FTP. In this example, there is an established outbound SMTP session, assigned to port 1001 on the local host, connected to the SMTP server on port 25 of the host with IP address 44.2.0.32. 335 characters are in the queue ready to be transmitted.

Issuing the command with the argument <tcb_addr>, taken from the "&TCB" column in the table, will provide a more detailed table of information on the specified TCP connection. Of particular interest is the last line of the table, which provides information about the retry timer. This is the timer that determines

when a packet retransmission will be attempted. It is expressed in the form "running time/threshold time," both given in milliseconds. When the running time reaches the threshold, the pending packet will be retransmitted. If receipt of this packet is not acknowledged, the threshold time will be increased and the running timer will

be restarted. When the channel becomes congested, the threshold time becomes very large, and the data throughput drops substantially. This is known as the "exponential backoff" strategy of TCP/IP.

You should display the TCP status before you exit NET. This will let you know if there are any active connections. For example, you should not exit NET if someone is running an FTP session with your host. Except for the clatter of your disk drive, you would not be aware of an active FTP session unless you check the TCP status. If there are some active sessions, it is best to leave NET running so that your TCP/IP node stays on the air. If you think you have "stuck" sessions in which there is no active packet transmission, see section 10.1.3 for information on resetting these sessions.

10.1.2. tcp kick <tcp_addr>

If there is data on the send queue of the TCP connection indicated by the argument <tcb_addr>, taken from the table generated by the "tcp status" command, this will force an immediate retransmission. This can be attempted if the connection appears to be "stuck." This can happen if the route is unreliable, or if there is considerable channel congestion. This command should be used sparingly, as it adds to channel congestion and defeats the TCP strategies to deal with this congestion.

10.1.3. tcp reset <tcp_addr>

Resets the TCP connection indicated by the argument <tcp_addr>, taken from the table generated by the "tcp status" command. This effectively terminates the connection. While this command should not be used indiscriminantly, there are situations in which it is useful, primarily when a TCP connection has gotten "stuck." The three common situations in which a TCP connection can get stuck are: remote host has crashed, propagation path of the route to the remote host has deteriorated, and there is extreme congestion on the channel. In each of these situations, tcp status for the corresponding TCP connection will show an increasingly large threshold time.

If you are in a telnet session or an AX.25 connection, you will find that you will wait nearly forever for a response from the remote host. If you want to give up on the session, escape to NET and close (or disconnect) the session. If there is no response from the remote host, the session can remain half open and should be reset using the tcp reset command.

Similarly, an FTP session can also become stuck. If you are in the midst of a file transfer (get or put operation), use the FTP converse command "abort" to terminate the transfer. Then the "quit" command will initiate a close on the session. If there is no response from the remote host, the half open session should be reset using tcp reset.

Another situation in which this command is appropriate for use is the "stuck" SMTP session. As discussed in section 6.2.14, when a mail message has begun to be sent, it is "locked." If the route to the destination host is unreliable, the exponential backoff strategy of TCP can cause the transfer to be delayed almost indefinitely. If you want to terminate the transfer, so that you can try again at another time, or using a different route, you can reset the corresponding TCP session. When this is done, the associated mail message is unlocked and can be resent, or killed from BM if you want to give up on the message entirely.

Appendix A. Installation of NET and BM

As with most software, some effort is needed to get NET and BM installed and properly configured on your system. This requires setting up the proper directory structure and editing some configuration files. While there are a number of details to attend to, none of this is very difficult.

The configuration files discussed below must be edited with a text editor. Any editor will do, as long as it writes ASCII (simple text) files. A word processing program will work as well, as long as you have it write an ASCII output file.

A.1. Installation Overview

It will be assumed that you have received a disk (or disks) containing at least the files NET.EXE and BM.EXE. You may also have the files AUTOEXEC.NET, HOSTS.NET, FTPUSERS, and ALIAS, although if these were not included on your distribution, don't panic, as you will be able to create them with an editor.

Here are the steps you will have to take to install the KA9Q Internet Package, where the root directory (\) is preferably on your C: hard disk drive, or otherwise on your A: floppy disk drive:

1. Copy NET.EXE and BM.EXE to the root directory.
2. Create the directory structure described in section

A.2.

3. Create the file AUTOEXEC.NET in the root directory and edit as described in section A.3.
4. Create the file HOSTS.NET in the root directory and edit as described in section A.4.
5. Create the file BM.RC in the root directory and edit as described in section A.5.
6. Edit your AUTOEXEC.BAT file in the root directory to add the time zone environment variable, as described in section A.6.
7. Create the ALIAS file in the root directory and edit as described in section A.7.
8. Create the FTPUSERS file in the root directory and edit as described in section A.8.
9. Create the finger file(s) in the directory \finger and edit as described in section A.9.

A.2. File Structure

Most of the required files will be kept in the root directory of your
primary disk drive, but several other directories must also be set up.

\spool The NET log file (described in section A.3.21) is normally stored in this directory.

\spool\mail This directory holds the individual mailboxes for each user name on your system. The extension .txt is added to the user name to form the mailbox name. Mail received by the SMTP server is appended to the mailbox file.

\spool\mqueue The directory holds the outbound mail jobs. Each job consists of 2 files: an xxx.txt and xxx.wrk file, where xxx is a unique numerical prefix. When the job is being sent, an xxx.lck file is also created. The file sequence.seq is used to keep track of the last message number. The mail transport protocol, SMTP, and the mail program, BM, manage the files in this directory.

\public This directory holds the files available for anonymous ftp, described in section A.8.

\finger This directory holds the finger files, described in sections 9.2 and A.9.

The files that will reside in the root directory include:

NET.EXE
BM.EXE
AUTOEXEC.NET
HOSTS.NET
BM.RC

ALIAS
FTPUSERS

A.3. NET Configuration File -- AUTOEXEC.NET

The AUTOEXEC.NET file, created in the root directory, has a function similar to that of the AUTOEXEC.BAT file in DOS, hence the name. When NET is executed, it reads AUTOEXEC.NET and executes all of the commands as if they had been typed in to the program from the keyboard. This provides an easy mechanism for setting up the initial system configuration, including setting the IP address, hostname, AX.25 parameters, routing table, servers to start, and protocol variables. This file is to be located in the root directory of the current disk drive on your system.

An example AUTOEXEC.NET file is usually distributed with the NET software. This file is fully commented and explains how the example file should be edited for your use. If this file is not available, use a text editor to create the file, following the instructions given in the subsections below. It is suggested that you put the commands into AUTOEXEC.NET in the order given below.

A.3.1.

Commands starting with the hash mark (#) are ignored. This is mainly useful for comments in the AUTOEXEC.NET file. The comments can appear anywhere in the file.

A.3.2. hostname <host_name>

Sets the local host's name (an ASCII string, NOT an IP address). This is usually chosen to be <call>.ampr.org, where <call> is your amateur call sign. The suffix 'ampr.org' is officially recognized as meaning an 'AMateur Packet Radio' station. Your hostname will show up in mail headers and in the greeting messages from the SMTP (mail), FTP (file transfer), and AX.25 mailbox servers.

For example:

```
hostname n6gf.ampr.org
```

A.3.3 ax25 mycall <callsign>[-<value>]

Set the local AX.25 address. It does the same thing that 'MYCALL' does in your AX.25 TNC. The standard format is used, e.g., KA9Q or WB6RQN-5. The optional dash (-) and <value> following the callsign is the SSID (substation ID), used when it is necessary to distinguish between two or more packet stations with the same callsign. The SSID will be 0 unless explicitly

set to another value, which must be a decimal number from 0 to 15. This command must be given before any attach command using the AX.25 mode is given. For example:

```
ax25 mycall n6gf
```

A.3.4 ip address <ip_addr>

Sets the local IP address. See section 2.4 for information on acquiring an IP address. For example:

```
ip address [44.2.0.100]
```

A.3.5. attach

The attach command configures and attaches a hardware interface to the system. While many interfaces can be handled by NET, in this guide, only a single serial interface to a TNC running the KISS protocol will be considered. The general form of the command for our purposes is:

```
attach asy <I/O address> <vector> ax25 ax0
<bufsize> <mtu> <baud>
```

"asy" refers to a standard PC asynchronous interface. Other hardware interfaces are supported by NET, but will not be covered in this guide.

<I/O address> is the base address of the control registers for the serial interface.

<vector> is the interrupt vector number. Both the I/O address and the vector must be in hexadecimal. (You may put "0x" in front of these two values if you wish, but note that they will be interpreted in hex even if you don't use it).

"ax25" forms IP datagrams to correspond to the AX.25 packet protocol. Other modes are supported by NET, but will not be covered in this guide.

"ax0" is the name by which the interface will be known to the various commands, such as "connect," "route" and "trace".

For asynchronous ports, <bufsize> specifies the size of the ring buffer in bytes to be statically allocated to the receiver; incoming bursts larger than this may (but not necessarily) cause data to be lost. The suggested value is 1024.

<mtu> is the maximum transmission unit size, in bytes. Datagrams larger than this limit will be fragmented at the IP layer into smaller pieces. The suggested value is 256.

<baud> is the baud rate of the serial communication between the computer and the TNC. It must be chosen to be the same value as the baud rate selected on the TNC, as discussed in section 2.3. The suggested value is 4800 for an XT, and 9600 for an AT.

Example 1 -- Attach the PC serial card normally known as "com1" (the first controller) to operate in AX.25 mode at 4800 baud with a KISS TNC.

```
attach asy 0x3f8 4 ax25 ax0 1024 256 4800
```

Example 2 -- Attach the secondary PC serial card ("com2") to operate in AX.25 mode at 4800 baud with a KISS TNC.

```
attach asy 0x2f8 3 ax25 ax0 1024 256 4800
```

A.3.6. ip ttl <value>

Sets the default time-to-live value placed in each outgoing IP datagram. This limits the number of switch hops the datagram will be allowed to take. The idea is to bound the lifetime of the packet should it become caught in a routing loop, so the value should be somewhat larger than the diameter of the loop. A suggested value is 16. For example:

```
ip ttl 16
```

A.3.7. param ax0 <value1> <value2>

Param invokes a device-specific control routine. On a KISS TNC interface, this sends control packets to the TNC. Data values are treated as decimal. This command is used to change TNC parameters such as TXDELAY, TXTAIL, PERSIST, and SLOTTIME. This command is TNC-specific, so you must read the documentation for the KISS implementation for your TNC. Most KISS implementations include good default values, so you shouldn't have to use this feature, but if things don't work, you can use the "param" command to try tweaking the TNC.

On a TNC-2, <value1>=1 will set TXDELAY (the time after key down when packet information is transmitted) to <value2> X 0.01 seconds. To set TXDELAY to 0.5 seconds:

```
param ax0 1 50
```

On Kantronics TNCs, <value1>=1 sets TXDELAY to <value2> X 0.01 seconds, as with the TNC-2. For <value1>=2, PERSIST is set to <value2>. For <value1>=3, SLOTTIME is set to <value2> X 0.01 seconds. The default values for these parameters are generally acceptable, so it is usually not necessary to include these commands in AUTOEXEC.NET.

A.3.8. route add

The "route add" command adds an entry to the routing table, defining how your outgoing packets should be routed. Routing can be a complicated issue, so it would be best for you to get some help on this from an experienced local TCP/IP user.

The command syntax is:

```
route add <dest_host>[/bits]|default ax0
[<gateway_host>]
```

Basically what you are trying to do is to route your packets directly to those hosts that you can "hear" and route your packets to the remaining hosts through hosts that can serve as gateways (similar to AX.25 digipeaters). The destination host for your packets is <dest_host>, which is either an IP address or a host name, as defined in the file HOSTS.NET, described in section A.4. The gateway is <gateway_host>,

also an IP address or host name. Note that it is not necessary to specify the entire sequence of hosts from your system to the destination, but rather only the destination and the first stop on the way. If the routing table on the other hosts has been set up properly, your packets will get to the desired destination. Thus, everyone has to cooperate in keeping the packets moving.

In my area, N6RQR is over 40 miles away, but he is up in the foothills above me and has a good station, so I can communicate with him directly. His IP address is [44.2.0.54], so I route packets directly to him with the statement:

```
route add [44.2.0.54] ax0
```

The option "/bits" can be used to avoid having to include a route add statement for each and every host you can communicate with directly. To understand this, more details on IP addresses are needed. The IP address is a 32-bit number, with four 8-bit numbers separated with periods (.). Each 8-bit number can range from 0 to 255. The optional "/bits" suffix to the destination host id specifies how many leading bits in the host IP address are to be considered significant in the routing comparisons. If not specified, 32 bits (i.e., full significance) is assumed. With this option, a single routing table entry may refer to many hosts all sharing a common bit string prefix in their IP addresses. For example, [44.0.0.0]/8 would match all addresses in the

form [44.*.*.*], where '*' is any number between 0 and 255; the remaining 24 bits are "don't-cares". When an IP address to be routed matches more than one entry in the routing table, the entry with largest "bits" parameter (i.e., the "best" match) is used. This allows individual hosts or blocks of hosts to be exceptions to a more general rule for a larger block of hosts.

The "/bits" option can be used to route packets directly to all the hams on TCP/IP in my town. We have been assigned IP addresses in the range [44.2.0.96] to [44.2.0.111], so our addresses agree in the first 24 bits. Further, 96 decimal is 01100000 binary and 111 decimal is 01101111 binary. Thus, our addresses agree in an additional 4 bits, for a total of 28 bits. The command used is:

```
route add [44.2.0.96]/28 ax0
```

Hams in a neighboring region have been assigned IP addresses of the form [44.4.*.*]. N6RQR can forward packets to them, so we use him as a gateway with the command:

```
route add [44.4.0.0]/16 ax0 n6rqr
```

The special destination "default" is used to route datagrams to addresses not in the routing table; it is equivalent to specifying a /bits suffix of /0 to any destination host. Care must be taken with default entries since two nodes with default entries pointing

at each other will route packets to unknown addresses back and forth in a

loop until their time-to-live (TTL) fields expire. (Routing loops for specific addresses can also be created, but this is less likely to occur accidentally).

In my area, we route packets to all other destinations through our local switch, which has the host name "eyolo." The command used is:

route add default ax0 eyolo

A.3.9. smtp timer <value>

Sets the interval to <value>, in seconds, between scans of the outbound mail queue to determine if there is any new outgoing mail which should be sent. For example, "smtp timer 600" will cause the system to check for outgoing mail every 10 minutes and attempt to deliver anything it finds. For an end-user system that is not normally intended as a mail forwarder, you do not want to set the interval to be too short, as you will be frequently accessing disk needlessly. An interval of 30 minutes (<value>=1800) is probably reasonable. You can also "kick" out the mail manually, as described in section 6.5.1. The suggested command is:

```
smtp timer 1800
```

A.3.10. smtp gateway <host>

Defines the host to be used as a "smart" mail relay.

Any mail sent to a host not defined in the file "HOSTS.NET" will instead be sent to the gateway for forwarding. You will have to ask locally if there is a host that is used as a mail gateway. If so, include this command in AUTOEXEC.NET.

A.3.11. tcp mss <value>

Set the TCP maximum segment size in bytes that will be sent on all outgoing TCP connect requests (SYN segments). This tells the remote end the size of the largest segment (packet) it may send. An mss of 216 will force folks to send you packets of 256 characters or less (counting the overhead). Suggested command:

```
tcp mss 216
```

A.3.12. tcp window <value>

Sets the window parameter, which establishes the maximum number of bytes that may be outstanding before your system expects an ack. If the window is twice as big as mss, for example, there will be two active packets on the channel at any given time. Large values of window are a problem on the air. Keep mss <= window <= 2*mss. Suggested command:

```
tcp window 432
```

A.3.13. start <server>

Starts the specified Internet server, allowing remote connection requests. Suggested commands:

```
start ftp
start telnet
start smtp
start finger
```

A.3.14. ax25 digipeat [on|off]

Controls whether AX.25 packets addressed to this station as a digipeater will be repeated or not. If you want to operate as a digipeater (for those poor souls not operating TCP/IP), include the following command:

```
ax25 digipeat on
```

A.3.15. ax25 heard [on|off]

Controls whether the list of call signs heard is updated or not. Suggested command:

```
ax25 heard on
```

A.3.16. ax25 maxframe <value>

Establishes the maximum number of frames that will be allowed to remain unacknowledged at one time on AX.25 connections. This number cannot be greater than 7. Suggested command:

```
ax25 maxframe 1
```

A.3.17. ax25 paclen <value>

Limits the AX.25 packet length. This parameter should be less than or equal to the <mtu> defined in the attach command. Suggested command:

```
ax25 paclen 256
```

A.3.18. ax25 retry <value>

Limits the number of successive unsuccessful retransmission attempts on AX.25 connections. If this limit is exceeded, link re-establishment is attempted. If this fails "retry" times, then the connection is abandoned and all queued data is deleted. Suggested command:

```
ax25 retry 10
```

A.3.19. ax25 window <value>

Sets the number of bytes that can be pending on an AX.25 receive queue. Suggested command:

```
ax25 window 2048
```

A.3.20. mbox [on|off]

Establishes whether or not the AX.25 mailbox is on. The mailbox allows AX.25 packet users to leave a message for you or to establish a keyboard

conversation. Suggested command:

```
mbox on
```

A.3.21. log <file>

Defines the name of the file for server session log entries. Don't include this command if you don't want to keep a log. The suggested command is:

```
log \spool\net.log
```

You should read this file occasionally and then discard it, as it can grow to be quite large.

A.4. The Hosts File -- HOSTS.NET

The file HOSTS.NET, created in the root directory, provides a mapping between an IP addresses and symbolic hostnames. It is used by NET to look up a hostname to figure out the correct IP address to use. These hostsnames may be used in establishing a TCP/IP connection, e.g. it is not necessary to enter 'telnet [44.2.0.98]',

but merely 'telnet n6spe'. It is kept in the root directory. At a minimimum each entry should contain the IP address and callsign. Subsequent aliases in the same entry should be separated by a single space. Each entry (IP address) represents a separate and distinct computer address.

The form of an entry:

```
<ip_addr> <host_name1> <host_name2> ...
```

Note that this is the one case where the IP address does not have to be enclosed in brackets. A host can have more than one symbolic name. A Tab is recommended between the IP address and host name.

Here are some examples of HOSTS.NET entries:

```
44.2.0.98 n6spe shayne
44.2.0.100 n6gf
44.96.0.2 wb2sef xt.wb2sef
```

Note that the domain name .AMPR.ORG has been assigned for amateur radio. By default, we assume that the hostname is the user's callsign in the case where a user has one system online, and so <callsign>.AMPR.ORG is the implied official hostname. If you have more than one machine on the air, distinguish them by prefixing a familiar name followed by a period, as in "winfree.n3eua" or "at.n0ccz".

Note that the use of a callsign as a host name has nothing to do with the "mycall" parameter. It is convenient to use the callsign as a hostname, and required to use the callsign for "mycall" to properly identify a station according to FCC rules.

If there is an established group of TCP/IP users in your area, they probably maintain an up-to-date HOSTS.NET table that is made available by an ftp file transfer.

A.5. Mail Configuration File -- BM.RC

The BM.RC file, created in the root directory, provides BM with the configuration needed for the operation of the mailer.

The format for each line in the BM.RC file is:

```
<variable> <value>
```

The variables described below are valid in the BM.RC file.

A.5.1. host <host_name>

Sets the local hostname for use in the mail headers. This is a required field. This should match the hostname definition in AUTOEXEC.NET.

A.5.2. user <username>

Defines the user name of the person who is sending mail. This is also used as the default mailbox for reading mail. On the AMPRNET this is usually set to your call. There is a DOS limit of 8 characters for the user name.

A.5.3. edit <directory>\<editor>

Defines the name of your favorite editor which can be used to construct and edit the text of outgoing messages. <editor> is the name of the editor and <directory> is the name of the directory where it is stored on the host computer. The use of edit is optional.

A.5.4. fullname <your full name>

Provides your full name to the mailer for use in the comment portion of the "From:" header line. The use of fullname is optional.

A.5.5. reply <return address>

Defines the address where you wish to receive replies to messages sent. This option is useful if you are operating your pc on a local area network and would like your mail replies sent to a more "well known host," for exampe, your local switch. The address

specified by reply is used to generate a "Reply-To:" header in outbound mail. The "Reply-To:" header overrides the "From:" header which is the address normally used to reply to mail. This field is optional.

A.5.6. mbox <file>

Specifies the default file to be used for the "save" command, for saving received messages. This file is in the same format as a mailbox and may later be viewed using the -f option of BM. If this option is not used then the default file name is set to mbox.

A.5.7. record <file>

If defined, a copy of each message sent will be saved in <file>.

A.5.8. folder <directory>

If defined folder contains the directory used by the save command. If not defined, files will be saved in the current directory.

A.5.9. smtp <directory>

Defines the directory containing the mailbox files. The default directory is \spool\mail on the current drive.

A.5.10. Example BM.RC File

Here is an example of a BM.RC file:

```
host n6gf.ampr.org
user n6gf
reply n6gf%n6gf@eyolo
edit \bin\vi
fullname Gary Ford
mbox rcv.txt
folder \ford\packet
smtp \spool\mail
```

A.6. Time Zone Environment Variable

The time zone used in mail headers is obtained from the DOS environment variable TZ. An example TZ setting is:

```
set TZ=EDT4
```

It should be added to your AUTOEXEC.BAT file. The first 3 characters are the time zone and the fourth character is the number of hours from GMT time. If TZ is not set, GMT is assumed.

A.7. The Alias File -- ALIAS

The ALIAS file, created in the root directory, provides an easy way to maintain mailing lists. It allows you to send mail to an easily-remembered name, instead of a complicated address. An alias can be any string of characters not containing the "@" symbol. The format for an entry in the alias file is:

```
<alias> <recip1> <recip2> <recip3>
<TAB> <recip4> ...
```

Note that a long list of aliases can be continued on an additional line by placing a tab or space in the first position of the continuation line.

Some example aliases are:

```
spe n6spe%n6spe@eyolo
dave nn2z@nn2z
# mail to local eyolo users
ey-gang n6gf%n6gf@eyolo n6spe%n6spe@eyolo
```

In the above example, when specifying ey-gang as the recipient, BM will expand the alias into the list of recipients from the alias file.

An alias may not contain another alias from the same file. However, an alias may contain a recipient name that is an alias on the local host or a remote host. For example, if the host "eyolo" contains the following alias in addition to the aliases above:

```
# gang mail
gang   se-gang@sesac   nc-gang@ncsac   ey-
gang@eyolo
```

then mail addressed to "gang" will be forwarded to the alias "se-gang" at the host "sesac," to the alias "nc-gang" at the host "ncsac" and to the alias "ey- gang" on "eyolo" itself. The mail forwarded to "ey-gang" will then be forwarded again, as indicated by the example alias above for "ey-gang."

The use of an ALIAS file is optional. If you are just getting started with TCP/IP, you might want to wait until you are more familiar with mail addressing before you establish this file. More information on mail addressing is given in section 6.4.

Your local TCP/IP user group may maintain an alias file, so you should inquire as to its availability.

A.8. The FTP Users File -- FTPUSERS

Since MS-DOS was designed as a single-user operating system, it provides no access control; all files can be read, written or deleted by the local user. It is usually undesirable to give such open access to a system to remote network users. The FTP server therefore provides its own access control mechanism.

The file "FTPUSERS," created in the root directory, is used to control remote FTP access. The default is NO access; if this file does not exist, the FTP server will be unusable. A remote user must first "log in" to the system, giving a valid name and password listed in FTPUSERS, before he or she can transfer files.

Each entry in FTPUSERS consists of a single line of the form

```
<username> <password> <directory1> <perm1>
<directory2> <perm2>
```

There must be exactly one space between each field. Comment lines are begun with "#" in column one. <username> is the user's assigned login name. "password" is the required password. Note that this is transmitted in plain text; therefore it is not a good idea to give general write permission to the root directory. A password of "*" (a single asterisk) means that any password is acceptable.

<directoryN> is the name of a directory that may be accessed by the remote user. The remote user will also have access to the subdirectories of this directory. The directory name must be absolute, i.e. it must begin from the root directory (\).

<permN> is a decimal number granting permission for read, write, and overwrite and delete operations for <directoryN> and its subdirectories. For a permission of 1, the user is allowed to read a file subject to the directory access restrictions. A permission of 2 allows a user to write a new file if it does not overwrite an existing file. A permission of 4 allows a user to write a file even if it overwrites an existing file, and in addition he or she may delete files. Again, all operations are allowed subject to the directory access restrictions. Permissions may be combined by adding permission values. For example, 3 (= 2 + 1) means that the user is given read and create permission, but not overwrite/delete permission. Similarly, 7 (= 3 + 2 + 1) means that the user is given read, write, overwrite and delete privileges.

It is a standard convention to keep a repository of downloadable files in the directory \public and to allow users to logon with the username "anonymous" with no password to access these files. In some areas, the public access username is "guest." Every system providing an FTP server is encouraged to provide restricted access to "anonymous" and "guest" users.

The appropriate FTPUSERS entries allowing the users "anonymous" and "guest" to read files under \public and subdirectories, but not to write, overwrite or delete any files are:

```
anonymous * \public 1
guest * \public 1
```

If you want to allow these users to write files as well, but not to overwrite or delete files, change the permission to 3.

You might want to give a friend access to both his or her own directory as well as the public access directory. For example:

```
n6spe test \users\n6spe 7 \public 3
```

This gives user "n6spe," with password "test," read, write, overwrite and delete privileges for files under \users\n6spe and read and write privileges for files under \public; he may not access files in other directories.

A.9. Finger Files

Finger files are information files having the name <file>.txt, stored in the \finger directory. It is common practice to set up a file for each of the users on the host. Thus, if n6gf is a user, he should have an information file named "n6gf.txt." This file is edited

with a text editor and should include information about the user such as his or her full name, address, and phone number. Additionally, information about the packet system, such as computer, TCP/IP software, TNC, radio, and

antenna could be included. Using the finger command, as described in section 9.2, the remote user can access the names of the finger files and then have them displayed on his or her console. Thus, you could have a separate information file for your system and include general information about the files available for downloading and your usual hours of operation. There is no specified format for these files. They will be displayed on the remote host just as they appear in the file.

Appendix B. NET Command Descriptions

Given below are all of the commands listed by the "help" command in NET, with concise descriptions of these commands. If the command is described in more detail in this guide, the section(s) in which it is referenced is given in parentheses. An asterisk (*) after the section reference indicates that not all of the subcommands have been described in this guide.

!	Suspend NET, return to DOS. (4.3)
arp	Address Resolution Protocol. Connects IP addresses with callsigns.
attach	Configure and attach a hardware interface. (A.3.5)*
ax25	AX.25 (normal packet) services. (4.2, 8., 9.1, A.3.3, A.3.14-A.3.19)
cd	Change directory. (4.4.3)
close	Close a session. (5.4, 8.3)
connect	AX.25 connect request. (8.1)
dir	List contents of a directory. (4.4.4)
disconnect	Close a session (alias for close). (5.4, 8.3)

echo	Controls telnet keyboard echo.
eol	Controls telnet end of line behavior.
escape	Controls command-mode escape command (not available on PC).
exit	Exit NET and return to DOS. (4.3, 4.7)
finger	Finger information files on remote host. (9.2, A.9)
forward	Forward traffic to another hardware interface.
ftp	File Transfer Protocol. (4.2, 7.)
help	List NET commands. (4.4.1)
hostname	Display or set hostname. (A.3.2)
kick	Force an immediate retransmission on a session.

log Controls logging of server sessions. (A.3.21)

ip Internet Protocol. (A.3.4, A.3.6)*

memstat Displays internal free memory list.

mbox AX.25 mailbox. (8.4, A.3.20)

mode Controls transmission mode on AX.25 interfaces.

mulport Controls routing of data between interfaces.

netrom Controls NET/ROM services.

nrstat Displays NET/ROM statistics.

param Invokes a device-specific control routine. (A.3.7)*

ping Query a remote host. (9.3)

pwd Display name of current directory. (4.4.2)

record Record telnet or AX.25 session to a disk file. (5.3.1, 8.2.1)

reset Reset a session.

route Controls the routing table. (10., A.3.8)

session Controls sessions selection. (4.5, 5.2, 8.4)

shell Suspend NET, return to DOS. Alias for ! (4.3)

smtp Simple Mail Transport Protocol. (6.5, A.3.9, A.3.10)*

start Start a server. (4.4.5, A.3.13)

stop Stop a server. (4.4.6)

tcp Transport Control Protocol. (10.1, A.3.11, A.3.12)

telnet Telnet keyboard-to-keyboard protocol. (4.2, 5.)

trace Monitor packet traffic. (9.4)

udp User datagram protocol.

upload Upload ASCII file to telnet or AX.25 session. (5.3.1, 8.2.2)

? List NET commands. Alias for help. (4.4.1)