CONVEX Networking Concepts

Document No. 710-000103-201

First Edition
December 1989

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX Networking Concepts
Order No. DSW-128
First Edition

Printed in the United States of America

Revision Information for
CONVEX Networking Concepts

Edition	Document No.	Description
First	710-000103-201	Released with CONVEX UltraNet Interface V1.0 and ConvexOS V8.0, December 1989.

Table of Contents

Appendices

List of Figures

Preface

Purpose and Audience

This document creates a context in which to discuss the details of network operation, management, and programming. It is intended for use by users, system managers, and programmers who need a basic understanding of CONVEX Networking before going on to the more detailed information found in the *CONVEX Networking Utilities System Manager's Guide, CONVEX Networking Utilities User's Guide,* and *CONVEX Interprocess Communication (IPC) Programming Guide.*

Organization

This manual is organized as follows:

- Chapter 1, "Introduction," defines networking in general terms and introduces CONVEX Networking.
- Chapter 2, "Host Names and Addresses," describes how hosts on a network are identified.
- Chapter 3, "Routing and Subnets," discusses the processes involved in determining paths to hosts on a network.
- Chapter 4, "Optional CONVEX Networking Products," discusses the CONVEX UltraNet Interface.
- Appendix A, "Bibliography," provides sources for further information on the concepts presented in this book.
- Appendix B provides instructions for reporting software or documentation problems to the CONVEX Technical Assistance Center (TAC).

Notational Conventions

The following conventions are used in this document:

- Words enclosed in rounded rectangles indicate keyboard keys that you press. For example, (RETURN) refers to the carriage return key. Words separated by a hyphen and enclosed in rounded rectangles indicate two keys that you must press simultaneously. For example, (CTRL-X) indicates that you must press the (CTRL) key while simultaneously pressing the keyboard ⊗ character key.
- The word "enter" in a phrase such as "enter a command" means that you type the command and press the carriage return key. In contrast, the word "type" (for example, "type a line of text") means that you do not press the carriage return key.

- Within command sequences

Boldface	indicates characters that must by typed just as they appear.
Italics	indicate filenames or arguments for which you must substitute actual filenames or arguments.
Brackets ([ ])	designate optional entries.
Horizontal ellipsis (...)	shows repetition of the preceding item(s)

Consider the following example:

COMMAND *input_file* [, . . .] [*output_file*]

where **COMMAND** must be typed as it appears; *input_file* indicates a filename that must be supplied by the user; the horizontal ellipsis in brackets indicates that additional input filenames, separated by commas, may be supplied; and *output_file* indicates an optional filename.

- References to the *CONVEX UNIX Programmer's Manual* appear in the form *adb*(1), where the name of the manual page is followed by its section number enclosed in parentheses.

- `Constant-width font` is used for examples of computer-generated output.

Associated Documents

Using network software successfully may require information not within the scope of this guide.

The following documents are provided by CONVEX Computer Corporation to help you with ConvexOS:

- *CONVEX UNIX Primer* provides an introduction for users who have not previously used ConvexOS.

- *CONVEX UNIX Programmer's Manual*, Parts I and II, is the standard reference for the ConvexOS operating system.

- *ConvexOS System Manager's Guide* contains information needed to manage and maintain a CONVEX supercomputer.

- *CONVEX UNIX Tutorial Papers* is a collection of previously published papers that provides instruction in document preparation, programming, text editing, supporting tools and languages, system maintenance, and system implementation.

- *CONVEX UNIX Utilities User's Guide* provides detailed information on the CONVEX Assembler, Loader, text editors, and *adb* debugger.

The following documents are provided by CONVEX Computer Corporation to help you use CONVEX Networking products (which are all optional products).

- *CONVEX Networking Utilities User's Guide* explains how to use the CONVEX network to get work done on a day-to-day basis. In particular, it describes the network utilities that enable you to log in to remote systems, execute commands on different machines, and transfer files across the network.

- *CONVEX Networking Utilities System Manager's Guide* describes how to configure, test, and maintain the software used by CONVEX Networking and the CONVEX UltraNet Interface.
- *CONVEX Interprocess Communication (IPC) Programming Guide* provides programmers with the necessary information to develop network applications.

Ordering Documentation

To order CONVEX documentation, contact your local CONVEX office.

To order the current edition of this or any other CONVEX document, you need to know the exact title or the six-character order number. See the copyright page of this document for its six-character order number. See the *CONVEX COMPUTER Price Book* or call your local CONVEX office for the order numbers of other CONVEX documents.

To order an edition other than the current edition, you need to know the 12-digit document number. See the title page of this document for its 12-digit document number. Call your local CONVEX office for the document numbers of other CONVEX documents.

Technical Assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC). Use the following phone numbers:

Within the continental U.S.	1(800)952-0379
From locations in Alaska, Hawaii, & Canada	1(214)497-4379
From all other locations	Contact nearest CONVEX office

1

Introduction to Networking

1.1 Introduction

This chapter discusses the concept of networking and describes CONVEX Networking software. It provides basic information for persons unfamiliar with networking concepts and serves as an introduction to CONVEX Networking.

The network software described here is optional. To acquire it, you must order a CONVEX Networking or CONVEX UltraNet Interface License which includes both the network software and a current set of release notes.

1.2 Computer Networks

In the simplest sense, a network is a collection of autonomous computers interconnected by communications lines. These lines may be fiber-optic or coaxial cables; they may also be telephone lines or satellite links. Some networks are made up of computers and workstations in close proximity, such as those located within a single corporate office building or throughout several buildings on a community college campus. Called local area networks, or LANs, they connect machines located within a few kilometers of each other. LANs operate at high speeds, usually in the range of 3 to 100 Mbits per second.

Other networks are vast, providing communication between computers at major universities or government installations spread throughout the country or around the world. Such networks are called long haul or wide area networks and employ microwave and satellite technology to transfer information over great distances. Long haul networks operate at speeds in the range of 2 Kbits per second to 3 Mbits per second and usually involve the use of telephone lines and special packet switching computers. Well-known wide area networks include UUCP, developed by AT&T Bell Laboratories to copy files from one UNIX system to another, and X.25, a wide area network popular in Europe.

An emerging networking technology, the Metropolitan Area Network (MAN), is beginning to compete with telephone companies for providing large corporate customers with the ability to transfer massive amounts of voice and data within an area the size of a city. MANs use hundreds of miles of fiber-optic cable to connect installations spread across several miles. Like LANs, MANs use dedicated communication lines to transfer data faster than wide area networks which rely on regular telephone switching technology.

1.3 Internetworking

Local and wide area networks may operate as self-contained entities or they may be interconnected to form what is called an internet. Though composed of diverse physical networks, an internet operates as a single, virtual network. Networking enables computers to share information and resources; Internetworking extends the service area of a network, permitting machines of differing types, as well as greater numbers, to participate.

The benefit of internetworking is that it allows an installation to select the networking products most suited to its own needs; the burden is that it requires each computer on the internet to adhere to a set of standards.

Internetworking standards in widespread use today were developed for the ARPANET, now called the DARPA Internet, or more often, simply "the Internet." Funded by the Defense Advanced Research Projects Agency (DARPA), the Internet was developed to serve as a testbed for internetworking technology. From that research came a set of standard formats and procedures, collectively called protocols, used for local area networking. Included in those standards is a family of communication protocols known as TCP/IP and a philosophy for structuring network software in logical layers. This layered arrangement of protocols and software is called a network software architecture.

NOTE

The lowercase terms "internet" and "an internet" refer to any interconnection of networks. When capitalized, "Internet" and "the Internet" refer specifically to networks that use the standards developed for the DARPA Internet. Designation of a network as an "Internet network" does not mean the network is actually connected to the DARPA Internet, only that it adheres to standards that would allow it to be. The capitalized "Internet" is also used here to refer to the standards, as in "Internet addresses."

1.4 The CONVEX Network

CONVEX offers a wide range of networking products, including Ethernet, COVUEnet, CONVEX UltraNet, and CX.25. This document covers a subset of available CONVEX networking products, those that use TCP/IP protocols and interface to application programs through the "socket" facility (sockets are discussed briefly in section 1.4.1 and at length in *CONVEX Interprocess Communication (IPC) Programming Guide*). CONVEX networking products that use other protocols or application program interfaces are documented elsewhere. See *Introduction to CONVEX Documentation* to find out where to locate information on other CONVEX networking products.

The terms "CONVEX Networking" and "the CONVEX Network" refer to optional LAN products that use Internet TCP/IP protocols to communicate between machines, and the socket abstraction to communicate between application programs and network software. CONVEX Networking operates over Ethernet, HYPERchannel, or UltraNet hardware, which consists of coaxial or fiber-optic cables, controllers, and host adapters. CONVEX Networks can be interconnected to form self-contained Internets. For example, an installation might have a network composed of several interconnected Ethernets. Because CONVEX Networking conforms to Internet standards, CONVEX LANs can also communicate with large external networks.

NOTE	For the sake of simplicity, this discussion addresses a generic CONVEX Network based upon CONVEX's Ethernet TCP/IP implementation. Details specific to other CONVEX Networking products are discussed in Chapter 4, "Optional CONVEX Networking Products."

1.4.1 CONVEX Network Software Architecture

Though CONVEX Networking uses a variety of hardware and software components, network-specific details are hidden under layers of software. Figure 1-1 presents a simplified picture of the layered software architecture that provides for network independence.

Figure 1–1: Layered Network Architecture

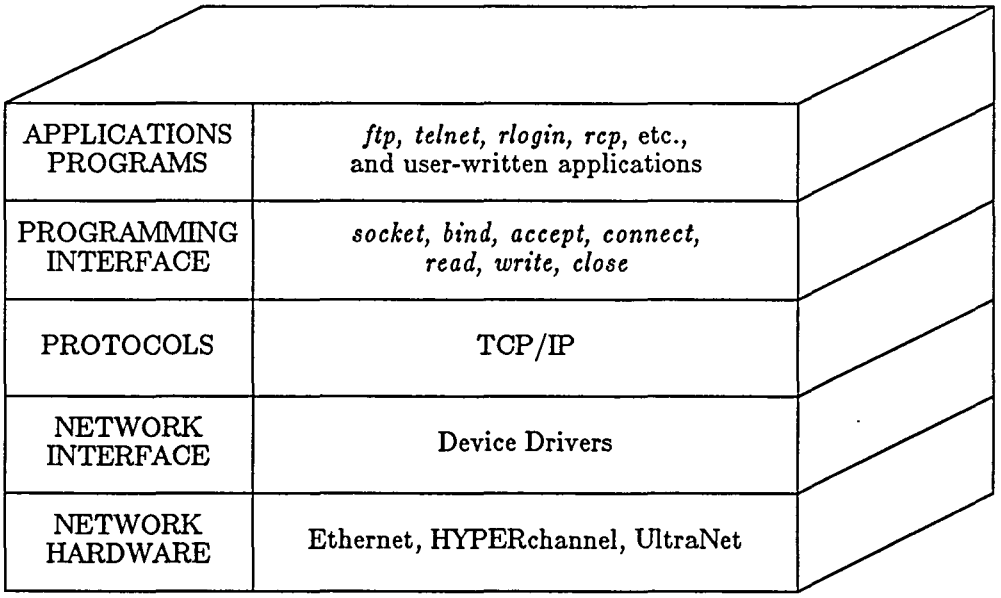


Figure 1-1 shows four logical layers of network software.

- Application programs make system calls to create and use sockets for sending and receiving data.
- Sockets provide the interface between application programs and network protocols, serving as endpoints for network communication. A process creates a socket, associates it with a particular service on a specific destination machine, and either initiates or waits for a connection over which to send or receive data.

- On the sending side, TCP/IP breaks the data flowing through a socket into manageable units called packets, labels those packets with address and routing information, and passes them along to the device driver for transmission. On the receiving side, TCP/IP strips off the address and routing information, reassembles the packets, and passes the data through a socket to the awaiting application program.

- Device drivers handle the details specific to reading or writing over the particular network interface.

This layered approach to network software allows application programs and most of the operating system to remain network-independent. Only device drivers need to know details about the network hardware. The inclusion of TCP/IP among the layers allows the CONVEX Network to interoperate with other networks that run TCP/IP. Besides using TCP/IP as a common mechanism for network communication, Internet networks also recognize the same addressing and routing information. The following chapters discuss the concepts of Internet addressing and routing.

2

Host Names and Addresses

2.1 Introduction

In the previous chapter, a packet was defined as a unit of data sent over a network. Packets are labeled with information about their origin and destination. For packets sent across the CONVEX Network, this information takes the form of Internet addresses.

This chapter discusses the concepts of host naming and addressing as the means of identifying participants on a network. The latter part of the chapter points out issues to consider when choosing an addressing structure.

2.2 Internet Names and Addresses

An Internet address is a number assigned to each host on an Internet network. CONVEX Networking uses Internet naming and addressing for the same reason it uses TCP/IP protocols, so that it can interoperate with other Internet networks. Just as TCP/IP provides the common mechanism for sending packets across the network, Internet addressing provides the common means for specifying *where* to send them.

Host names, Internet addresses, and physical addresses are three different ways of identifying destinations. While names are easier for people to work with, computers handle numbers more efficiently, so network software converts host names to Internet addresses before using them internally. To allow network software to remain ignorant of network hardware, Internet addresses must be network-independent. Therefore, another number, the device-dependent physical address, is needed to identify the destination host in a way that the hardware understands.

There must, of course, be some way to associate host names, Internet addresses, and physical addresses. The mechanism for making these associations is called "mapping." Host names are mapped to Internet addresses and Internet addresses are mapped to physical addresses. The first form of mapping is simple. It uses the host name data base contained in the file */etc/hosts*. (Another facility for mapping host names to Internet addresses, Yellow Pages (*yp*), is a network lookup service provided by the CONVEX Network File System (NFS) product. For more information on using *yp*, consult your CONVEX sales representative or the *CONVEX Network File System* documentation set.)

The partial */etc/hosts* file shown in Figure 2-1 illustrates how host names are associated with Internet addresses.

Figure 2–1: Sample */etc/hosts* File

```
#
# Host Database
#
# The format is:
#  Internet-address  official-hostname aliases...
#
127.1    localhost
# (used by some utilities and daemons)
#
# Local Net -- 10Mb/s Ethernet
#
190.55.10.2    ariel  air
190.55.10.3    puck
190.55.11.1    hamlet son
```

When a user specifies a host name on the command line of a network utility, the software calls the library routine *gethostbyname*(3), which looks up the name in the host database and returns its Internet address. Thus, the user need only remember the host name "ariel," or its alias, "air," not its unwieldy Internet address, 190.55.10.2.

The procedure for mapping Internet addresses to physical addresses, called address resolution, requires more work and is network-dependent. Each CONVEX Networking product has its own method of address resolution. The Address Resolution Protocol (ARP) runs on hosts that have Ethernet controllers. Similar facilities exist for resolving addresses on HYPERchannel and UltraNet networks, though they require you to explicitly create the associations. For the purpose of this discussion, it is not necessary that you understand how this mapping is done, just that some method exists for translating Internet addresses into physical addresses.

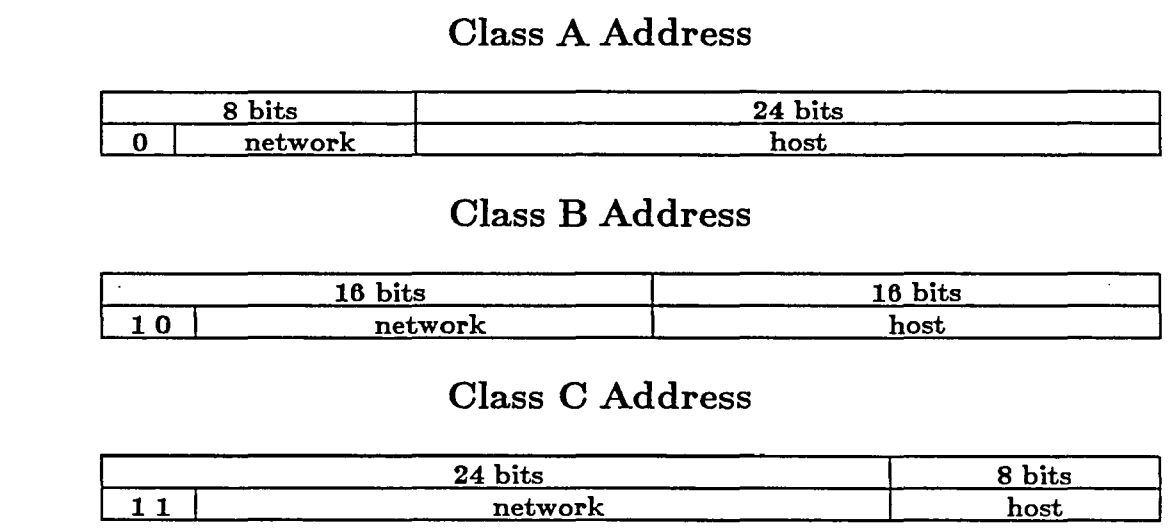
The *CONVEX Networking Utilities System Manager's Guide* contains a section on resolving UltraNet addresses with the */etc/unetcf madd* command and an appendix on using ARP.

2.2.1 Address Classes

Internet addresses consist of 32-bit values representing three fields: the address class, the network number, and the host number. The address class indicates the size of the network in terms of how many hosts it can support. The network number refers to the interface used to communicate with the host. Host numbers refer to individual machines connected to the physical interface.

As illustrated in Figure 2-2, distribution of the 32 bits of address depends upon the address class defined in the high-order bits.

Figure 2–2: Internet Addresses By Class



2.2.2 Dot Notation Addresses

In situations where you must identify a host by its Internet address, as in the */etc/hosts* file, "dot notation" is used. Addresses are specified as a series of numbers, usually decimal, separated by periods. Dot notation addresses take on many forms; some of them help to visually separate the fields.

When the addresss is of the form:

a.b.c.d

each 8-bit number (a,b,c,d) is assigned to the corresponding octet of the Internet address. This format can be used to specify addresses of any class.

The form:

a.b.cd

specifies a three-part address which is interpreted as having 16 bits of host number (cd). The three-part format conveniently represents class B network addresses as *net.net.host*

In the two-part representation

a.bcd

the high-order bit identifies the address class, the network portion occupies 7 bits, and the host portion occupies the remaining 24 bits. The large host number field allows class A networks to support as many as 2^{24} hosts. The two-part format provides a convenient way of specifying class A addresses as *net.host.*

2.3 Multi-homed Hosts

Thus far, to keep matters simple, this discussion has allowed you to believe that each Internet address identifies a particular host computer. This is only partially true, because some hosts communicate over more than one network interface. In such cases, a single Internet address is not enough.

Termed "multi-homed hosts," machines connected to multiple network interfaces must be assigned multiple network addresses, one for each interface. This situation makes it apparent that an Internet address does not really refer to a host computer, but to a connection—a pathname that specifies a particular network interface on a particular machine.

Naming conventions can help minimize the confusion caused by multi-homed hosts. One such convention consists of a root name identifying the host computer (for example, "hamlet") followed by suffixes for each network connection (for example, "hamlet-ex" for accessing "hamlet" over an Ethernet; and "hamlet-hy" for accessing it via a HYPERchannel connection). An example of this naming convention is given in Figure 2-3.

Figure 2–3: Multi-homed Host Naming

```
#
# Host Database
#
127.1    localhost
#
# Local Net -- 10 Mbits/sec Ethernet
#
190.55.10.2    ariel-ex  air
190.55.10.3    puck-ex
190.55.11.1    hamlet-ex son
#
# HYPERchannel -- 50 Mbits/sec
#
190.56.10.2    ariel-hy  wind
190.56.11.1    hamlet-hy pop
```

Unfortunately, this naming convention creates names that are less than user friendly. You can compensate for this awkwardness by assigning aliases that are both meaningful and friendly, as in the example above.

Assigning names that identify both the host and the network interface becomes even more helpful when one of the interfaces is to an UltraNet network. See Chapter 4, "Optional CONVEX Networking Products," for further discussion on the multi-homed host problem.

2.4 Choosing Network Addresses

So why should you care that names and addresses actually specify an interface rather than a host? Because not all interfaces are created equally. In particular, some transfer data many times faster than others. For example, if a host has both an Ethernet and a HYPERchannel interface, the faster HYPERchannel address is preferable, provided that an efficient route over that interface exists.

2.4.1 Obtaining Official Network Numbers

If you were setting up a LAN that you knew you would never want to connect to any other network, you could pick host addresses at random, or invent your own numbering scheme. However, if you choose your addresses randomly and later, after your small, simple network has grown in size and complexity, you find it desirable to connect your LAN to other networks, you would face the difficult task of changing your address structure to conform to the expectations of the external world.

Internet protocols require each host to have a unique address. Unless you use "official" Internet addresses, you cannot guarantee that your addresses are unique. If they are not, the Internet to which you connect your LAN will be confused. Therefore, it pays to obtain official addresses before you set up your network. To do so costs nothing, and it affords you the ability to expand the service area of your network as your needs grow.

An organization called the Defense Data Network (DDN) Information Center assigns official network numbers. Once you have determined what class network numbers you want, you can write to them at DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, or call their toll-free number, 800-235-3155, and they will provide you with network numbers.

2.4.2 Address Numbering Conventions

Hosts on the Internet are expected to adhere to certain address numbering conventions. One is the interpretation of a host number field of all ones as meaning "broadcast." A broadcast is a message sent to all hosts on the network. One example of where broadcasts are used is in routing. Internet hosts share routing information by broadcasting their routing tables to each other. Because some network interfaces (HYPERchannel, for instance) do not support broadcasts, simply using an Internet broadcast address does not guarantee broadcasts will take place.

A similar convention is the interpretation of a host number of all zeros as referring to the network itself, rather than to a particular host. This convention is used for, among other things, routing-related messages.

2.4.3 Other Considerations

The expected size of your network will influence how you choose addresses. Because your choice of network class sets a hard limit on the number of hosts your network can support, you should choose a large enough class to meet your needs as your network grows. Few networks require the capacity of class A addresses (16,777,214 hosts), but many need more than the 254 (host numbers of all zeros and all ones are reserved, as mentioned above) available on class C networks. Choosing an address class is a relatively easy decision to make before the network is set up, and a difficult one to change afterward.

Before you assign addresses, you should consider the issue of whether you will want to use subnets. Subnetting requires you to take bits away from the host portion of the Internet address, so you should choose small host numbers if you plan to use subnets. How you structure your addresses also affects routing, because routing decisions are based upon the network number portion of the Internet address. All hosts connected by a common network interface should have the same network number.

The topics of routing and subnetting are discussed in the following chapter.

3

Routing and Subnets

3.1 Introduction

The previous chapter discussed Internet addresses as the means by which hosts on a network specify the destination for packets. However, simply saying *where* to send a packet is often not enough. Sometimes the sender does not know exactly where to find the destination host because the two machines are not connected to the same physical network. In such cases, the sender needs a way of determining *how* to get the packet to its destination. This chapter discusses the concept of routing, the task of finding the path over which to send packets to their destination.

3.2 Routing

The simplest route connects one host to another on the same LAN. Packets never leave the physical network where they originate; they are sent to their destination directly over the LAN hardware. The process of routing becomes complicated when a host needs to send a packet to a machine on a different physical network. Transferring packets across physical network boundaries requires the services provided by gateways and bridges.

3.2.1 Gateways and Bridges

Gateways are computers used to connect networks. They may be special purpose packet-switching computers or simply hosts with more than one network interface, which transfer packets from one network to another in addition to serving as general purpose computers. Gateways can be used to connect similar or different types of networks. For example, you might use a gateway to connect an Ethernet to a HYPERchannel.

Gateways that exchange routing information are called "active" gateways; those that do not are called "passive" gateways. Active gateways are responsible for keeping dynamically-created routing tables up to date.

Bridges also connect networks, but at the physical level. They are normally used to connect networks of the same type. For example, a bridge can be used to concatenate Ethernet cables so that the resultant network appears to network software as a single physical network. Unlike gateways, bridges are transparent to network software.

3.2.2 Managing Routing Tables

If you want your network to successfully route packets that originate within it, each machine on your network needs to know the routes to all possible destinations. Network software bases its routing decisions upon information stored in a pair of routing tables. One table contains entries for routes to specific hosts and the other contains entries for routes to other networks. Each entry specifies the destination Internet address, the address of the gateway used to reach the destination network, a pointer to the network interface, and other information about the route.

You can manage your routing tables either manually or dynamically. The simplest method is to create the tables manually by adding */etc/route* commands to the startup file */etc/rc.local*. This method is practical only if your network remains relatively small and unchanging.

The *CONVEX Networking Utilities System Manager's Guide* explains how to use */etc/route*. In general, you can add and delete routes to specific hosts or to networks. Figure 3-1 illustrates typical */etc/route* commands. Entries specify the Internet address of the gateway used to reach the host or network. You can also identify a default gateway to send packets to if no other route is known. If you decide to manage your routes manually, be aware that as your network grows, so will the task of keeping the routing tables up to date on all computers it serves.

Figure 3–1: Sample */etc/route* Commands

```
#
#  Internet routes
#
#  the format is:
#       /etc/route add|delete [net|host] destination gateway [metric]
#
/etc/route add host 128.70.50.1 129.50.1.1  2
/etc/route add net  128.68.1    128.68.4.1  1
/etc/route add default 129.50.1.2
#
#
```

You can more effectively manage a large, complex network, or one that changes frequently, by letting the network software create and maintain the routing tables for you. The routing daemon, */etc/routed*(5), runs in active gateways. It receives messages from other routing daemons informing it of changes in routes and recommending better ones. It uses these "routing redirect" messages to keep its tables current.

The routing daemon also eavesdrops on the network, listening for messages directed to routing daemons on other machines. If it detects that a route has been inactive for a long time (currently defined as four minutes), it deletes the route from its tables.

The system initialization file, */etc/rc.local*(8), normally starts the routing daemon. Upon startup, */etc/routed* reads the */etc/gateways* file to initialize its routing tables. It then takes a quick inventory of the physical network interfaces to which it is directly connected. Once it has determined its own network configuration, it transmits requests to routing daemons running on other machines, asking them for their routes and updating its tables with the responses it receives.

3.2.3 Table-Driven Routing

The previous discussion described the ways in which routing tables are created and maintained. The rest of the story lies in how network software uses the routes stored in the tables.

Though gateways have the primary responsibility for routing, the task of routing begins with the sender. Software in the sender uses the destination Internet address to determine whether a packet it has ready to transmit is destined for its own local network. If the network number portion of the destination address matches the local network number, the packet is sent directly over the LAN hardware.

Otherwise, the sender searches the routing tables for a route to the specific destination host. Failing to find one, the sender again looks in its routing tables, this time searching for a gateway to the destination network. If one exists, the packet is forwarded to it. A default gateway, specified with the */etc/route*(8) command at startup, is used if no explicit route to the network is found. If no default gateway exists, the host is considered unreachable.

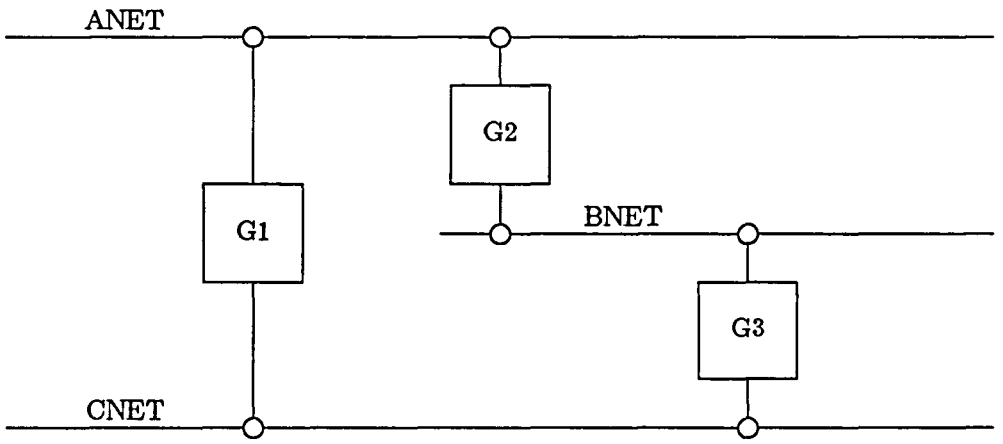
Each gateway through which the packet passes performs a similar algorithm. By comparing the network number against its own, the gateway determines whether the host specified in the destination address is connected to any of its own network interfaces. If it is, the gateway accepts the packet and sends it over the appropriate interface.

Otherwise, the gateway searches the routing tables for a route to the specific host, then for a route to the host's network, then for a default route to use as a wild card. If it finds no further route over which to send the packet, the gateway either ignores the packet or returns a routing redirect message back to the sender, depending upon whether the gateway is active or passive, i.e., whether or not it has a routing daemon running in it.

3.2.4 Performance Considerations

How you set up routing on your network can greatly affect its performance. Generally speaking, packets travel quickest over the shortest routes. Often, you have no choice of routes; there is only one path to the destination. However, when multiple network interfaces connect the sending and the receiving hosts, the right choice can make a difference. Take, for example, the situation in Figure 3-2.

Figure 3–2: Hosts Connected by Multiple Network Interfaces



Two routes exist between "ANET" and "CNET." One route passes through a single gateway, "G1," while the other must be processed by both "G2" and "G3." If "BNET" is a high-speed network such as UltraNet, the longer route might be the quickest one. Otherwise, the more direct route would be preferable.

3.3 Subnets

Subnetting is the partitioning of the network defined by a single network number into multiple virtual networks. Only the network itself knows that it uses subnets. To the rest of the world, it is one network, addressed by a single network number.

Sometimes subnetting is used to divide a single physical network into several virtual networks. For example, a system administrator with foresight might choose to logically divide a single Ethernet LAN into subnets so that, if the network grows beyond the capacity of a single Ethernet cable, the switch to multiple Ethernets is relatively painless.

Subnet numbers may also correspond to separate network interfaces. For example, an installation may have one Ethernet to connect the machines used by engineering, another to serve the accounting department's machines, another for marketing, and so forth, each one assigned its own subnet number.

Another way to achieve the same logical partitioning of a large local network would be to assign a different network number to each department's LAN. This method, however, has definite disadvantages. To illustrate, take the situation where you have an installation consisting of several LANs with only a single connection to the outside world.

You can obtain one official network number and partition it internally into subnets, or you can request a different network number for each LAN. The machine that connects to the external network and to one or more of your LANs serves as the gateway in and out.

Though the decision of whether to use subnets or multiple network numbers is yours to make, it has an impact on systems beyond your own. First of all, each network address class has a fixed number of networks it can support, as defined by the number of bits in the network number field.

There can be only 127 class A networks, 16,383 class B networks, and a little over 4 million class C networks. While these limits may have seemed generous when the classes were defined, today they seem frugal. If you choose to use multiple network numbers, you will unnecessarily deplete the pool of available numbers.

More importantly, because all machines that need to communicate with yours must know all your network numbers, your decision to use multiple network numbers causes space to be taken up in the routing tables of those other systems. If you use subnets, the burden of keeping track of your subnets stays within your network. Other systems need know only the route to your gateway.

3.3.1 Dividing Your Network into Subnets

Once you have made the decision to divide your network space into subnets, you must tell the network software which bits to use as the network+subnet number. You specify this with the *netmask* parameter when you configure the network interface with the *ifconfig*(8) command in your */etc/rc.local* file.

Network software uses the subnet mask to divide the host number portion of an Internet address into two parts, subnet number and host id (as distinguished from "host number"). The network mask can be specified as a single hexadecimal number with a leading 0x or with a dot-notation Internet address. For example,

0xffffff00 and 255.255.255.0

mean the same thing, that the host number field is to be divided into eight bits of subnet and eight bits of host id.

For a class B network, the above mask has the effect of partitioning the Internet address as shown in Figure 3-3.

Figure 3–3: Subnet Address Partitioning

16 bits		8 bits	8 bits
1 0	network	subnet	host

While this division into 8 bits for the subnet and 8 bits for the host id is common for class B networks, you can partition the address any way you choose. The number of bits you give each field depends upon your needs. Will you have just a few large subnets, or is your installation more suited to having many small subnets? Just as you need to give careful consideration to choosing an address class, you also need to choose your subnet mask carefully.

Subnetted class B addresses are commonly written in four-part dot notation as:

128.net.subnet.host

The use of the subnet field is very different for Internet addresses assigned to hosts on an UltraNet network. Chapter 4, "Optional CONVEX Networking Products," discusses these differences as well as other details specific to the CONVEX UltraNet Interface.

4

Optional CONVEX Networking Products

4.1 Introduction

Thus far, this book has discussed CONVEX Networking in general terms. Though CONVEX Networking products provide similar functionality, they differ in some of the details of implementation. This chapter focuses on the differences between standard CONVEX Networking and the high-performance networking product, CONVEX UltraNet.

4.2 CONVEX UltraNet Interface

CONVEX UltraNet uses high-speed circuitry and large data buffers to provide a network that can transfer data at speeds of up to 7 Mbytes/sec. It includes an intelligent protocol processor to relieve the host computer of time-consuming network processing functions. The protocol processor runs in the interface adapter, performing much of the work associated with end-to-end data delivery, error processing, and routing.

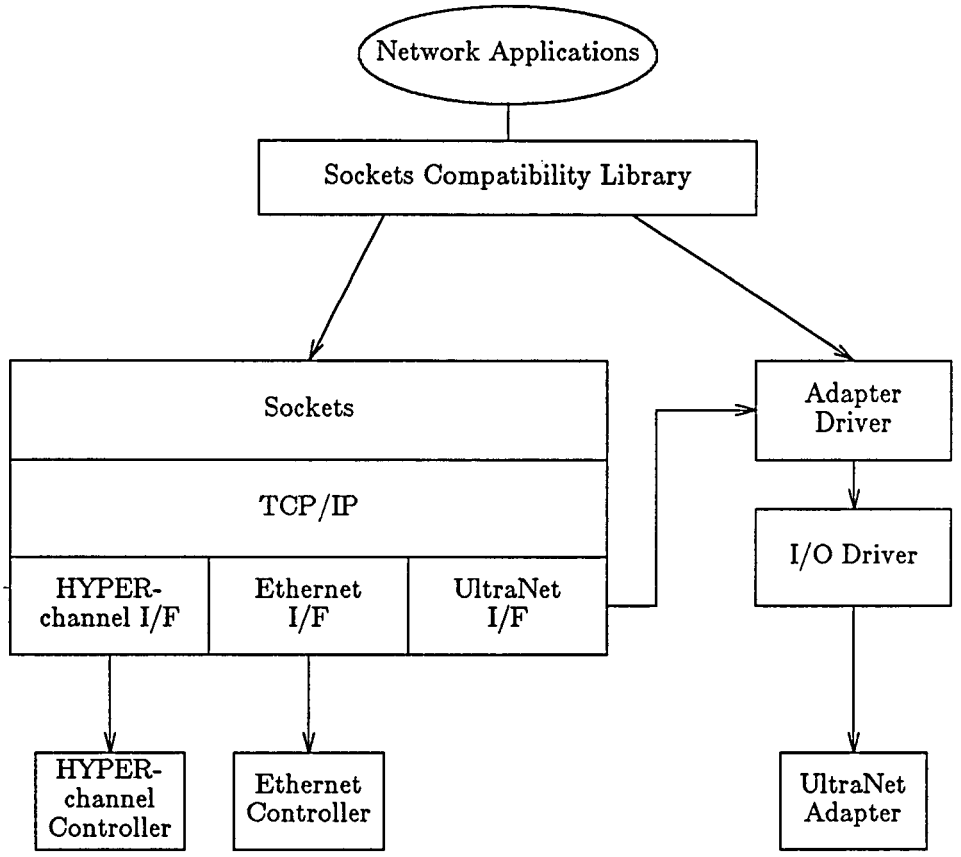
4.2.1 Network Software Architecture

Like the generic network discussed in this book, CONVEX UltraNet uses Internet standards such as layered network architecture, Internet naming and addressing, and the socket abstraction. Unlike the generic network, however, UltraNet uses its own "native" protocols for communicating within the local UltraNet network.

While CONVEX UltraNet gains performance by using its own protocols, it loses interoperability with other networks. Because it uses protocols no other network understands, UltraNet cannot broadcast to the outside world, nor can it take advantage of Internet routing. Because most installations require connections to the outside world, UltraNet runs TCP/IP protocols when communicating with other networks and uses native protocols to communicate within the local network.

Application programs interface to both TCP/IP and native UltraNet protocols through the socket abstraction. Because CONVEX UltraNet is typically installed in sites with other networks already in place, network software allows a process to use standard sockets, UltraNet sockets, or both. Figure 4-1 illustrates the multiple paths through UltraNet network software.

Figure 4–1: CONVEX UltraNet Network Software Architecture



UltraNet's multiple paths add to the multi-homed host confusion discussed in Chapter 2. In the example given in Chapter 2, the machine "hamlet" had two host names: one for an Ethernet interface ("hamlet-ex"), and the other for a HYPERchannel interface ("hamlet-hy"). Now "hamlet" needs a third and a fourth name: one for accessing native UltraNet ("hamlet-ul"), and one for using TCP/IP over the UltraNet interface ("hamlet-un"). The four host names indicate which of three paths the data will take.

- The path via the normal TCP/IP software across the Ethernet or HYPERchannel. Example host names are "hamlet-ex" and "hamlet-hy."
- The path running TCP/IP over UltraNet. Example host name: "hamlet-un."
- The native UltraNet path, which bypasses TCP/IP entirely. Example host name: "hamlet-ul."

Application programs are linked with either the standard socket calls in *libc.a* or with the UltraNet Sockets Compatibility Library, *libulsock.a*. When a program is linked with UltraNet sockets, socket-related calls (*socket, bind, connect, accept,* and so forth) are intercepted by CONVEX UltraNet software which then uses the destination host address to determine the type of socket you need for the path you want to take. For this reason, you must choose host names carefully. The wrong name will get you the wrong type of socket and the wrong network interface.

The sample */etc/hosts* file presented in Chapter 2, revised to include an UltraNet network, might look like the example shown in Figure 4-2.

Figure 4–2: Host Database with UltraNet

```
#
# Host Database
#
127.1    localhost
#
# Local Net -- 10 Mbits/sec Ethernet
#
190.55.10.2    ariel-ex  air
190.55.10.3    puck-ex
190.55.11.1    hamlet-ex son
#
# Hyperchannel -- 50 Mbits/sec
#
190.56.10.2    ariel-hy  wind
190.56.11.1    hamlet-hy pop
#
# Native UltraNet -- 7Mbytes/sec
#
190.57.11.1    hamlet-ul ultrapop
190.57.12.1    romeo-ul romeu
#
# UltraNet Internet Interface
#
190.58.11.1    hamlet-un ultraip
190.58.12.1    romeo-un  romeip
#
```

4.2.2 Internet Addressing

CONVEX UltraNet network names and addresses are essentially the same as Internet host and network addresses, with the following exceptions:

- UltraNet uses 16-bit station (host) addresses internally, rather than using the number of bits indicated by the network address class. This poses no problem if you are using class A or B network addresses, because the host number portion of the Internet address is 16 bits or greater; mapping between Internet addresses and UltraNet station addresses is one to one. However, class C addresses and subnetted network addresses with fewer than 16 bits of host number must be explicitly mapped to UltraNet hardware addresses by using the *unetcf madd* command. *CONVEX Networking Utilities System Manager's Guide* explains how to use *unetcf madd.*

- Every UltraNet interface must have its own network number. The network number of the native UltraNet interface need not be officially assigned, because there is no forwarding between it and the Internet. It is a good idea to use official numbers anyway, however, because packets occasionally escape into the Internet, as they would if a user specified the wrong interface on a command line.

- UltraNet via TCP/IP must also have its own network number. Because the standard Internet routing facilities are supported for this path, this should be an official network number.

- Internet addresses assigned to UltraNet hosts must be of the form

  net.net.subnet.1

  where *subnet* is unique for each UltraNet host. The addresses shown in Figure 4-2 conform to this convention.

4.2.3 Internet Routing

Because it cannot broadcast requests to other networks to ask for routing information, native UltraNet does not perform dynamic routing. Therefore, you must create UltraNet routing tables manually. To do this, you use the *unetcf radd* utility, similarly to the way in which you use *route add* to create routes for your Ethernet. See the *CONVEX Networking Utilities System Manager's Guide* for more information on using *unetcf radd*.

4.2.4 The Effect of Buffer Size on Network Performance

CONVEX UltraNet performance is closely tied to the size of the buffers used to transfer packets. In general, using larger buffers yields better performance. The increase in performance begins to drop off around 384K; performance ceases to improve with buffer sizes greater than 512K.

The benefit to network performance from using large buffers must be balanced against main memory size. Because network software locks the buffer used for UltraNet packets, choosing too large a buffer can greatly affect system performance (not just network performance). Application programs should take this into consideration.

A

Bibliography

Comer, Douglas. *Internetworking with TCP/IP.* New Jersey: Prentice-Hall, 1988.

An easy-to-read overview and introduction to TCP/IP. The book discusses in depth such topics as network software architecture, Internet addressing, routing, and the more commonly used protocols.

Hendrick, Charles L. "Introduction to Administration of an Internet-based Local Network." Rutgers University, New Jersey, 1988.

This "reader-friendly" paper provides a practical introduction to configuration or administration of a network based on TCP/IP protocols.

Tanenbaum, Andrew S. *Computer Networks.* 2d ed. New Jersey: Prentice-Hall, 1988.

A comprehensive, technical discussion using the OSI Reference Model as a framework.

B

Reporting Problems

This appendix introduces the CONVEX Technical Assistance Center (TAC) and the *contact* utility. The *contact* utility is an online system for reporting problems to the TAC. To learn *contact* by using it, enter **contact** at the system prompt and then answer the questions as they appear on the screen. To find out more about using *contact*, read through this appendix. It describes prerequisites and tips for using *contact* and the step-by-step process *contact* takes you through.

B.1 Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address the diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation question, contact the TAC. This group stands ready to solve such problems.

B.2 The *contact* Utility

The TAC recommends using the *contact* utility to report a hardware, software, or documentation problem. The *contact* utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix them.

After you invoke *contact*, it prompts you for information about the problem. When you finish your report, *contact* electronically mails it to the TAC. You are notified within 48 hours that the TAC has received your report.

B.3 Prerequisites

To use *contact* requires

- a UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- the full path name of the program or utility in question
- the version number of the program or utility in question

B.3.1 UUCP Connection

Before using *contact*, check with your system administrator to be sure there is a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX system to another. The *uucp* (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

B.3.2 Finding the Program Path Name

To determine the full path name of the program or utility in question, use the *which* command. The next screen illustrates using the *which* command to find the full path name of the loader (*ld*) utility.

```
>which ld
/bin/ld
>
```

In this example, the full path name of the loader is */bin/ld*.

For more information on the *which* command, refer to the *which*(1) man page. You can also use the *info* online information system. Enter **info which** at the system prompt.

If you use the C shell (*csh*), you can also use the *whence* command to find the program path name. The *whence* command works like *which*, only faster.

B.3.3 Finding the Program Version Number

To determine the version number of the program or utility in question, use the *vers* command. The next screen illustrates using the *vers* command to find the version number of the loader (*ld*) utility. Enter **vers**, then the path name of the program or utility.

```
>vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader version number is 7.0.

For more information on the *vers* command, refer to the *vers*(1) man page. You can also use the *info* online information system. To do so, enter **info vers** at the system prompt.

B.4 Tips on Using the *contact* Utility

The *contact* utility is interactive and easy to use. This section lists tips to help use it efficiently. In particular, this section tells how to

- use a *.contact* file
- abort a contact session
- resubmit an aborted report
- suspend a contact session
- move from one prompt to another
- use tilde-escape sequences in the *contact* utility

B.4.1 Using a *.contact* File

When you invoke *contact*, it prompts for information regarding the problem. The first prompt is for your name, title, phone number, and company name. You can, however, create a *.contact* file to skip this first prompt. Follow these steps:

1. Create a *.contact* file in your home directory.

2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke *contact*, it automatically includes the *.contact* file as input for the first prompt and proceeds to the next prompt.

B.4.2 Aborting the Report

To abort a contact report, either press the interrupt key (usually CTRL-C) or choose the abort option when prompted by the *contact* utility. Using CTRL-C to abort does not save the contents of the report. Using the abort option saves the contents of the report in a file named *dead.report* in your home directory.

B.4.3 Submitting the *dead.report* File

After you abort a contact session, the *contact* utility saves the report in a file named *dead.report* in your home directory. Using the *contact* command with the *-r* option automatically merges the contents of the *dead.report* file into the new contact session. Enter

contact -r

and *contact* finds the *dead.report* file in your home directory and merges it into the contact report. You can then edit the report. When you end the editing session, *contact* returns to the final prompt, which asks you to review, edit, submit, or abort the report.

B.4.4 Suspending a Report

Sometimes it is necessary to stop in the middle of a contact report and return to the shell (for instance, to suspend the contact session to find the program path name or version number). To suspend the contact session, press CTRL-Z. To return to the contact session, press **fg**. Using CTRL-Z and the *fg* (foreground) command lets you toggle back and forth between the *contact* utility and the shell. You cannot, however, use CTRL-Z and *fg* to toggle back and forth if you are using the Bourne shell (*sh*).

B.4.5 Ending a Response

The *contact* utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press RETURN. Other prompts require more than a one-line response; to move to the next prompt, press CTRL-D.

B.4.6 Tilde-Escape Sequences

The *contact* utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. The following tilde sequences are recognized by *contact*:

~e	start the text editor (defined in the EDITOR environment variable)
~h	display a list of available tilde-escape sequences
~p	print the contact report to the terminal screen
~r *filename*	read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence works only for prompts that allow more than a one-line response.
~~	insert a single tilde as the first character in the line

B.5 Using the *contact* Utility

The *contact* utility prompts for the following information:

- your name, title, phone number, and corporate name
- the name and version of the product
- a one-line summary of the problem
- a detailed description of the problem
- the priority of the problem
- instructions on how to reproduce the problem
- comments about the problem
- comments about the documentation supporting the problem
- files to include in the contact report

The following is a step-by-step discussion of these prompts.

Step 1a To invoke the *contact* utility, enter **contact** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software or documentation question. The next screen illustrates the *contact* command and the system response.

```
>contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

Step 1b If there is a *.contact* file in your home directory, *contact* skips the first prompt. The next screen illustrates the *contact* command and the system response when a *.contact* file is in your home directory.

```
>contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

Step 2 The *contact* utility prompts for the version number of the product. If you do not know the version number, use CTRL-Z to suspend the session. Use the *which* (or *whence* if you use *csh*) and *vers* commands to find the version number of the product. Use the *fg* command to return to the session and enter the version number in the form X.X or X.X.X.X.

Step 3 The *contact* utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Please make this summary as descriptive as possible in one line.

Step 4 The *contact* utility prompts for a detailed description of the problem. Please make this description as complete as possible. Include source code and a stack backtrace when possible. (Refer to the *adb*(1) or *csd*(1) man page for information on obtaining a stack backtrace.) The more information you provide, the quicker the TAC can isolate and solve the problem.

Step 5 The *contact* utility prompts for the priority of the problem. The next screen illustrates this prompt and the priority levels from which to choose; you must enter a priority number.

```
Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.
>
```

Step 6 The *contact* utility prompts for an explanation of how to reproduce the problem. Please include the command syntax and options you used and anything else you did to make the program run.

Step 7 The *contact* utility prompts for any other pertinent comments. Please include all relevant information.

Step 8 The *contact* utility prompts for suggestions regarding the documentation supporting the product. Indicate if the documentation could be revised to address the question.

Step 9 The *contact* utility asks for the names of files necessary to reproduce the problem. The next screen illustrates the *contact* prompt and sample user response.

```
Are there any files that should be included in this report (yes | no)?
>yes
Please enter the names of the files, one to a line (^D to terminate)
>test.f
>~/subroutines/sub.f
>
```

> **NOTE** Tilde-escape sequences are not recognized in responses to this prompt. In *contact*, a tilde in this section means your home directory. This convention is based on use of the tilde for expanding file names in *csh*.

If the files specified are small text files, they are automatically included in the contact report. If the files are too big to be included in this report, *contact* gives further instructions on how to submit these files.

To specify a directory, combine the directory files into a single file using the *tar* command (refer to the *tar*(1) man page for further information) or enter each file name in the directory on a single line in the contact report.

Step 10 The *contact* utility prompts you to review, edit, submit, or abort the contact report. The next screen illustrates this prompt.

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

Review — review the text of the contact report. You are then prompted again to select an option.

Edit — edit the text of the contact report. If you choose to edit the report, *contact* puts you in your default text editor.

Submit — sends the report to the CONVEX TAC. You are notified within 48 hours that the TAC has received the report. This option exits the *contact* utility and returns you to the shell environment.

Abort — saves the text of the report in a file named *dead.report* in your home directory. This option exits the *contact* utility and returns you to the shell environment.

Index

CONVEX Networking Concepts
Document No. 710-000103-201, First Edition

Reader's Forum

Please use this form to submit comments or questions concerning the clarity and service of this manual. Constructive critical comments are most welcome and help us continue in our efforts to generate quality customer documentation. Please list the page number for questions or comments.

_____

_____

_____

_____

_____

_____

_____

_____

From:

Name_____ Title_____

Company_____ Date_____

Address and Phone No._____

FOR ADDITIONAL INFORMATION OR DOCUMENTATION:

Location	Phone Number
From all locations in continental U.S.	1(800)952-0379
From locations in Alaska, Hawaii, & Canada	1(214)497-4379
From all other locations	Contact nearest CONVEX office

Direct mail orders to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851 USA

(Fold Here First)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1046 RICHARDSON, TEXAS

POSTAGE WILL BE PAID BY ADDRESSEE

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson TX 75083-3851

(Fold Here Second)

(Tape or Staple)