UniFLEX News

Technical Systems Consultants, Inc. **SUMMER 1989**

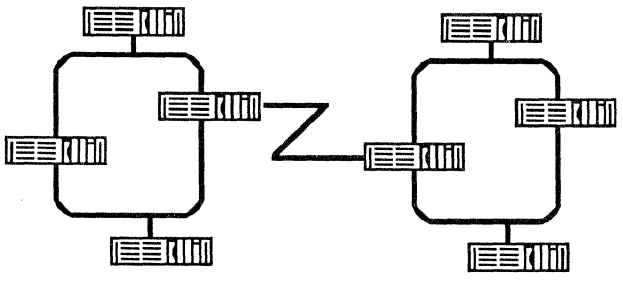
TCP/IP Networking

Technical Systems Consultants, Inc. is pleased to announced complete TCP/IP networking support for its popular UniFLEX® real-time operating system. The networked version of UniFLEX runs on all GMX computers based on the 68020 and 68030 processors. It is fully compatible with the new GMX Ethernet board (SBC-EN) as well as the Arcnet board (SBC-AN).

A full range of network utility programs or "services" come with the system, including:

FTP - File transfer program
Telnet - Remote login utility
Sendmail - Remote Mail utility
RSH - Remote program execution
Talk - Interactive network conversations

Built into the basic system is the ability to automatically route data between networks. This means that a installations with multiple local networks can be logically connected. And because auto-routing is part of the underlying network system, network users do not need to execute any special commands to utilze this feature.



The figure above illustrates two Ethernet local networks connected by an Arcnet which can run upto 8 Km in length without a repeater.

Also included with the network package is a library of C functions based on the BSD *socket* mechanism. This greatly simplifies the task of writing new user programs which interact directly with the network.

UniFLEX: The Total Solution

A variety of solutions exists for system designers looking for off-the-shelf system software for real-time microcomputer applications.

One approach is to use a full-featured operating system for normal tasks in conjunction with a more limited real-time "executive" to monitor real-time tasks. With this arrangement, however, the designer is often forced to do application coding and compilation under one system then migrate to a different system for test and debug. Needless complexity is added to the design process.

UniFLEX, however, provides the total solution in one operating system. UniFLEX is a multitasking, multiuser, Unix-style operating system designed specifically for real-time applications. Features such as demand-paged virtual memory, hierarchical file system, TCP/IP and NSF networking, plus System V compatible C libraries make UniFLEX the perfect choice for a multiuser software development enviroment. Special features such as the separate real-time task list, shared data pages, intertask message exchanges, and contiguous file support provide a solid base on which to build any high-performance real-time application.

UniFLEX comes complete with a command shell, file-system, and device independent I/O, as well as editor, relocating 68020/030 macro assembler, quick debugger, and Make-style utility. Options include a C compiler, source-level debugger, several third-party languages and public-domain editors and programming tools.

UniFLEX has been ported to all GMX 68020 and 68030 based systems. It works with most GMX add-in boards including the DMA SCSI, Ethernet, Arcnet, and serial I/O boards. For more information on making UniFLEX the solution for your next high-performance system contact either GMX, Inc. or Technical Systems Consultants, Inc.

TSC
Your Systems Software Partner

The UniFLEX® operating system was developed by Technical Systems Consultants, Inc. in 1981. Located in Chapel Hill, North Carolina, TSC is a group of technical experts who are dedicated to satisfying the need for a full-featured, high-performance real-time operating system.

Originally ported to the 6809 CPU as a follow-on to the popular FLEX system, UniFLEX has since been ported to a full range of 680x0 systems. For those who need real-time performance in an off-the-shelf computer system, GMX hardware delivered with UniFLEX software has been the choice of many intelligent users for years. Also, businesses with any of Apple's Macintosh II family of computers can now run UniFLEX as an alternative operating system.

For users who build sophisticated custom systems, UniFLEX is has become a very popular choice for VMEbus sytems. It has been ported to a variety of single board VMEbus computers from verdors such as Motorola, Force Computers, and Ironics. In addition, off-the-shelf drivers exist for more than 35 VMEbus boards including SCSI, ESDI, and SMD disk drives, serial ports, streaming-tape and 9-track tape drives, array processors, analog I/O boards, graphics controllers, and Ethernet cards.

For increased performance system designers may opt to take advantage of one of the muliprocessor versions of UniFLEX. Now available on a number of VMEbus computers, muliprocessing gives users an easy way to add power to a system without recoding their programs. UniFLEX performs dynamic load balancing of tasks to allow maximum throughput for complex multitask applications.

Finally, it should be noted that, while many other software vendors often sell only a single configuration, TSC is willing to work closely with its customers to meet their specific needs. In fact, UniFLEX is designed from the bottom up to be fully configurable. For specialized device drivers, TSC can provide the training, tools, and support for users to write their own. Alternatively, users may choose to contract TSC for custom driver development. We can even port the entire UniFLEX system to your custom 680x0 board or system, most likely at a fraction of the cost to develop you own OS from scatch.

File Sharing with NFS
the Network File System

With the release of full network support for the UniFLEX operating system, users have a powerful new tool at their disposal: the Network File System. What is NFS and what how can it be used to your advantage?

NFS was developed by Sun Microsystems but has been released for public use in order to promote the open systems concept. NFS is a mechanism that allows a program running on one computer to access the file system on another computer across the network. The remote file system, once mounted on the local file system, can be manipulated just as if it resided on the local disk. This technique allows almost transparent sharing of files and directories across all computers in a given network.

Using the NFS capablity is extremely easy. Users need only specify a host name in the mount command. For example, suppose you wanted to attach the directory `/X11/Release3` on a remote computer with a host name of `Saturn`. The following command would mount this remote directory under your local file system as the directory `/usr0`:

```
++ mount  Saturn:/X11/Release3  /usr0
```

After this command is executed, any reference to the directory `/usr0` and the files within it, would automatically operate on files under the `/X11/Release3` directory on the computer named `Saturn`.

UniFLEX News

UniFLEX News is published by:

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514

Phone: (919) 493-1451
Fax: (919) 490-2903

technical systems consultants, inc.

May 15, 1989

RE: TCP/IP Support for GMX Computers

Technical Systems Consultants, Inc. is pleased to announce immediate availability of two networking packages for GMX computers running the UniFLEX Operating System. One package supports the GMX Arcnet controller, the other supports the new GMX Ethernet interface. Both are based on the TCP/IP network protocol. Once two or more GMX systems are networked with UniFLEX, the following user level applications are available:

Telnet

This program allows a user to login onto any system connected to the network. The result is exactly as if a direct terminal connection had been made.

Ftp

This is a file transfer program which allows the user to effortlessly send or receive files from any system connected to the network.

Rsh

This application allows a user to execute a single shell command or script on any system connected to the network. This command also allows I/O redirection and pipes between machines.

Rlogin

This program is similar to Telnet, but will automatically login the user on the requested machine under that user's name.

Sendmail

This application provides a very powerful electronic mail facility between all systems connected to the network. Mail sending, receiving, collection, and reviewing facilities are all provided.

Talk

Talk allows an interactive terminal link between two users anywhere on the network. This is more efficient than mail if the desired user is known to be on the system at the time.

111 Providence Road • Chapel Hill, North Carolina 27514 • (919) 493-1451 • TWX 510-920-0540

Our TCP/IP implementation offers two advantages over most others. The first is automatic routing capabilities. This allows users to communicate with systems on the network without knowing their exact location in the network, or the path from their local machine. The second advantage is speed. We have one of the fastest TCP/IP implementations available (up to three times faster than others). There is one bonus with our implementation as well. Because of the automatic routing capability, a system can support both Ethernet and Arcnet in the same machine, allowing data from the Arcnet to reach systems on the Ethernet, and vice versa.

The TCP/IP Networking Package is only available for the real-time UniFLEX Operating System. The price of the option is $400.00 when ordered with the operating system. Current users may upgrade for $450.00 and must return their original disks. For users currently owning non-realtime systems, an operating system upgrade to the realtime version, plus the TCP/IP option, is $650.00. Only systems purchased prior to May 15, 1989 are qualified for this special upgrade price.

UniFLEX Networking Overview

1. What is networking?

Before looking into the many useful capabilities available on the UniFLEX networking system, we will introduce some important network related terms. *Networking* is the process of connecting together two or more computers. Each computer in a network is connected to an electronic device known as a *network interface*. All the network interfaces within a network are connected to a common transmission medium, such as a cable. The interconnection of network interfaces supports bi-directional communication among all computers in the network.

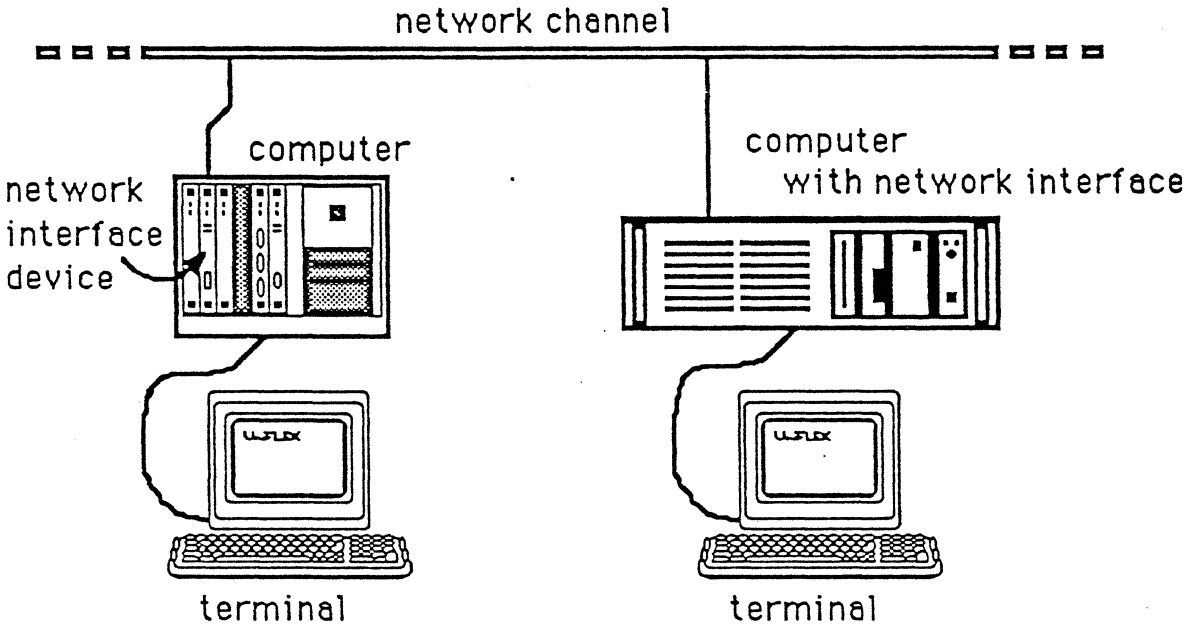


Figure 1. Typical Network Configuration.

Each network interface unit is known as a network *node*. The computer connected to a node is known as the *host*. Due to the close association between the network node and its attached computer host, the host is also often referred to as a node. Specialized software executes on the network interface devices and their attached host computers that support the efficient use of the physical network. This combination of network hardware and software enable a user logged in on any one of the computers to efficiently transfer files, exchange electronic mail, execute commands, and login to any of the other computers.

Information is exchanged over the network using standardized *protocols*. These protocols specify the rules for how nodes can reliably exchange information. Protocols allow the nodes of the network to make sense of all the data passing through the network at any one time, such as which piece of information goes to which node, etc. The protocols used in the UniFLEX system were developed by DARPA, a government agency responsible for creating a large network of computers known as the Internet. This network is comprised of many thousands of computers all over the country, connected in many small [local] networks. These local networks are then connected together in a wide-area network to compose the global Internet. Most networking in use today has its roots in the Internet system.

The UniFLEX networking system (UniFLEX/RN) supports wide-area and local networks using a variety of network interfaces. The most common is Ethernet, developed by XEROX PARC, which transmits data between systems at 10 Mbps (10 million bits per second). Another common network interface supported by the UniFLEX system is Arcnet, developed by DataPoint Inc., which normally transmits data at 2.5 Mbps.

Networking in the UniFLEX system is based on a system of functional sub-units known as *layers*. Each layer in the system supports certain functions which are used by the next higher layer and utilizes functions defined in the next lower layer. These layers can be depicted in a table as:

Layer Name	Examples		
Network Services	FTP	TELNET	SMTP
High Level Protocols	TCP	UDP	
Low Level Protocols	IP	ICMP	ARP
Physical Interfaces	ETHERNET	ARCNET	

For example, a computer user might run FTP to send a file from one computer on the network to another. FTP is a *network service*, or user-level program, for doing file transfers. FTP, like many other common network services, is based on the TCP (Transmission Control Protocol) protocol. TCP is in turn based on the IP (Internet Protocol) protocol, quite often referred to as TCP/IP. Finally, at the lowest level, TCP/IP may use ETHERNET to accomplish the file transfer at the physical level.

2. What's your network address?

Every node in a network must be uniquely identifiable so that information can be sent to the proper node. This is accomplished by assigning a unique address to every node. There are two main types of addresses associated with each node: the hardware address and the Internet address. Most network users are only concerned with the Internet address which is how nodes on the network are accessed using the upper level protocols, such as TCP.

An *Internet address* is a thirty-two (32) bit number which describes both the node and the network it is part of. Internet addresses come in 3 types known as Class A, Class B and Class C addresses. In each type, a portion of the address specifies the node and the remainder specifies the network the node is a part of. The latter is also known as the sub-net because these addresses are considered to be part of the global Internet. A typical Class C address would be **88.0.0.4** which specifies the sub-net **88.0.0** and the node **4**. Classes A and B are rarely used and therefore are not discussed here.

The node's hardware address is just that; a number which uniquely addresses (at least within the network) the network interface unit for the node. How this works is different for each type of network interface. In the case of Ethernet, the hardware address is a 48-bit quantity, normally assigned by a group within the IEEE for every Ethernet node in the world. In the case of Arcnet, the address is an 8 bit value which is chosen when the system is installed.

The correspondence between the hardware address and the Internet address is handled automatically by a protocol layer known as ARP (Address Resolution Protocol).

Symbolic names are normally given to Internet addresses to make them easier to use. Thus the address **88.0.0.4** might also be known as **Jupiter** or **george**. The correspondence between names and the Internet address is kept in a data-base, quite often in the file "/etc/hosts".

3. What can I do with networked UniFLEX?

UniFLEX/RN supports a full range of network services. Network services are network-based applications which allow users to access the computer systems on the network. Two services, **telnet** and **rlogin,** enable a user logged in on one computer to remotely log in onto any of the other computers on the network. When you need only to execute a single shell command on another computer, the **rsh** "remote shell" program provides this function. The **ftp** program supports a complete set of commands which allow users to interactively inspect the file structure on any of the remote computers and to exchange files with them. The **mail** program makes it easy for users to exchange electronic mail with other users throughout the network. For those times when interactive user-to-user communication would be more appropriate, the **talk** program provides an on-line split-screen interface for two-way terminal-based conversations. The **netstat** program displays the contents of various network-related data structures which show the current state of your network and statistics on network activity.

3.1 Remote Login (TELNET and RLOGIN)

Using the **telnet** program, you can login on another computer system on the network. This is just like having your terminal attached to the other system. Most anything that could be done using a terminal which is physically attached to the other machine can be done using **telnet.** Here's a sample **telnet** session (computer output is in **bold**):

```
++ telnet jupiter
Connected to Jupiter
Escape char is '^]'

Telnet (Jupiter)

Technical Systems Consultants, Inc.
     UniFLEX Operating System
pty00 12:35 Mon March 27, 1989
Jupiter/Login: guest

     *** Welcome to Jupiter ***
     *** The system will go down at 6 PM
     *** for maintenance

Terminal type: ansi

++ prompt '#H++ '
jupiter++ who
bob             tty00     10:12am
harry           tty03     09:32am
guest           pty00     12:35pm
jupiter++ log
```

In this example, we used the **telnet** program to access the computer system known as "Jupiter". Once connected, the session is nearly indistinguishable from a normal login session. One difference is that login prompts for the type of terminal you are using. Because you could have accessed the machine from anywhere in the network, the system has no idea of what type of terminal you might be using. This is important whenever you do any screen oriented program such

as **more** or the screen editor **use**. Notice the first operation performed was to adjust the shell prompt to include the host name. This causes the prompt to show on what computer the command will execute . Of course, the remote computer accessed does not need to be a UniFLEX system. In fact, in most cases **telnet** is used for access between dissimilar computers.

Another network service, the **rlogin** program, also supports remote login over the network. The **rlogin** program is very useful when you have the same user account on various hosts on the network. In fact, if you have the same password and default terminal type for both your local and remote accounts, logging onto any host is very simple using **rlogin** Just type:

```
++ rlogin <remote-host>
```

Then you continue with your login session as normal, except that you may type "~." at any time to disconnect from the remote host.

3.2 File Transfer Protocol (FTP and TFTP)

The **ftp** program makes it easy for files to be transferred over the network. Any sort of file may be transferred (binary or text) and the nodes can use different systems. A typical session might transfer data files from a PC to a UniFLEX development system. Here's what it might look like:

```
++ ftp pcbox2
Connected to 'pcbox2'
File Transfer Protocol/PC Server
Name: ftp
Password: <anything>
ftp> dir
ANALYZE.EXE   GATHER.EXE    CONTROL.DAT
SAMPLE0.DAT   SAMPLE1.DAT   SAMPLES.DAT
ftp> bin
ftp> get samples.dat
ftp> bye
```

In this simple session, we connected to the node "pcbox2" using the **ftp** program. After the connection, we had to login to the node. Note: this is similar to the normal login process. The **ftp** program uses the same names and passwords as does **login**. In this example, we chose the special name **ftp**. It is special in that it is supported on most systems and no password is required. Also, on most systems, the files which can be accessed are restricted when you login as **ftp**. In this session, we first got a listing of the available files. We then selected the binary transfer mode. This is important: if you are sending binary data, you must select binary transfer mode or the data may be transferred incorrectly. The "text" transfer mode transfers files with generic end-of-line indications (a carriage-return followed by a line-feed character) which are mapped to the local end-of-line (carriage-return) character. A binary file transferred in text mode risks these transformations which would most likely disturb the data. The next command **get** copies a file from the remote system to the local system. This is known as "getting" a file. (There is also a **put** command for "putting" a file which is not in the example above.) After the file transfer has completed, the session is terminated with the **bye** command.

Whereas **ftp** uses the TCP protocol, the program **tftp** is a version of the same program but transfers data using the UDP protocol. It is included to allow communication with nodes which understand UDP but not TCP.

3.3 Remote Execution (RSH)

The **rsh** program allows a single shell command to be executed on another computer. This is very much like logging into the computer and executing the command except that the input/output of the program being executed can come from the local computer. Here's an example:

```
++ page letter | rsh Jupiter nec
```

This command indicates to run two programs. The first program **page** is a standard UniFLEX utility which formats a text file for printing. The output of this program is then passed along (via the pipe '|') to the program **nec** which is to be executed on the computer "Jupiter". In this example, the program is a printer spooler program. The net effect of the command is to format a document and send it to a spooler on another computer for printing. Without remote execution, the document would have to be formatted and saved in a file, then transferred to the other computer, perhaps using **ftp**, and finally spooled on the other computer, perhaps using **telnet**. The remote shell operation makes all this possible in a single command.

3.4 Electronic Mail (SMTP)

Sending mail using the network is the simplest operation of all. All you need do is execute the **mail** program providing the user name and the host name for the recipient. An example should make this clear:

```
++ mail james@tsc1
... Type the first line of mail here...[RETURN]
... Type the second line of mail here...[RETURN]
...
... Type the last line of mail here...[RETURN]
^D
EOT
```

That's it! Notice that each line ends with a RETURN and the entire mail message ends when the RETURN is followed by '^D' (control-D). The mail will be packaged up and sent to the computer whose node name is "tsc1". The mail will be delivered to "james" on that computer. Depending on how your system is set up, the mail may be delivered immediately, or will be held in a queue and sent in a batch with other pending mail at a later time.

Receiving mail is just as easy. If electronic mail that is addressed to your user account has been delivered to your system you will see the following message when you log on:

```
You have mail.
```

Whenever you wish, you may read your mail by executing the **mail** program as follows:

```
++ mail
  U   1 sharon            Fri May 12 16:49   9/111
>N   2 pat                Fri May 23 11:04   8/357
```

Mail will display a list of all electronic mail files sent to you showing who sent it, when it arrived, and how large the file is. The leading uppercase character indicates whether the mail file is unread '**U**' or new '**N**' since you last looked at your mail. The arrow '**>**' points to which mail file the

program will display next if you press the [RETURN] key. After you read a mail file it is copied to the file *.mbox* in your home directory It will not be displayed in the mail list next time you run **mail**.

3.5 Interactive Conversation (TALK)

Next, let's assume that you need to converse with a user on another system across the network. You don't want to compose a letter to be sent via **mail** but instead you would like to get some quick feedback on a few short questions. This is the perfect time to use the **talk** program. If you want to talk with someone on the host system "saturn" who's username is "tom" you would enter the command:

```
++ talk  tom@saturn
```

talk attempts to establish a network connection with the user `tom` on the host `saturn`. If `tom` is logged onto `saturn`, a message will appear on his screen indicating that a **talk** connection is requested. Otherwise, you are notified that your requested party is not currently logged on and the program exits. If the user `tom` is logged on he will be asked to invoke **talk**, specifying your username and host name. Once this is done, two windows will be displayed on each user's terminal. As you enter characters on your terminal the top window displays your message while the bottom window displays the remote user's messages. Each character is processed by **talk** and sent across the network one at a time, including backspace characters. This interaction continues until either you or the other user enters a control-C.

3.6 Monitoring the Network (NETSTAT)

The program **netstat** can be used to display information about the state of the networking system. Based on options, you can find out information about the hardware, current connections, how data can be routed between separate networks and how the network system is using memory. One very useful option for **netstat** is the **-a** option. This option shows the state of all sockets. A sample output might be:

```
Active Internet connections (including servers)
Proto  Recv-Q  Send-Q  Local Address    Foreign Address      (state)
tcp    0       0       Apollo.telnet    Zeus.1036            ESTABLISHED
tcp    0       0       *.smtp           *.*                  LISTEN
tcp    0       0       *.telnet         *.*                  LISTEN
tcp    0       0       *.ftp            *.*                  LISTEN
tcp    0       0       *.login          *.*                  LISTEN
tcp    0       0       *.shell          *.*                  LISTEN
udp    0       0       *.router         *.*
```

In the example above, the local host "Apollo" has a telnet connection established with the foreign host "Zeus". The remaining items in the list such as "*.smtp" show sockets which are listening but have no connection active.

Another important option for **netstat** is the **-i** option. It displays which hardware interfaces are present on the system and how they have been initialized. Also, you will see the Internet address being used by each active interface.which selects information about the hardware interfaces. Calling **netstat** with the **-r** option will show the routing table. The option **-m** tells **netstat** to show statistics recorded by the memory management routines. The network manages a private section of memory so it is sometimes necessary to monitor these statistics

4. Automatic Routing

When data is addressed to a host which resides on a network other than your local network, an appropriate delivery route must be determined. Packets are allowed to cross over from one network to another via an interface commonly known as a *gateway*. When the two networks use dissimilar protocols, in addition to transferring the packets to the new network, the gateway must also perform protocol translation. The UniFLEX/RN system supports automatic routing of network information, even across networks with dissimilar protocols. Because the auto-routing is built right into the network driver, network users do not need to execute any special commands to utilize this feature. In fact, it is completely transparent from the user's point of view.

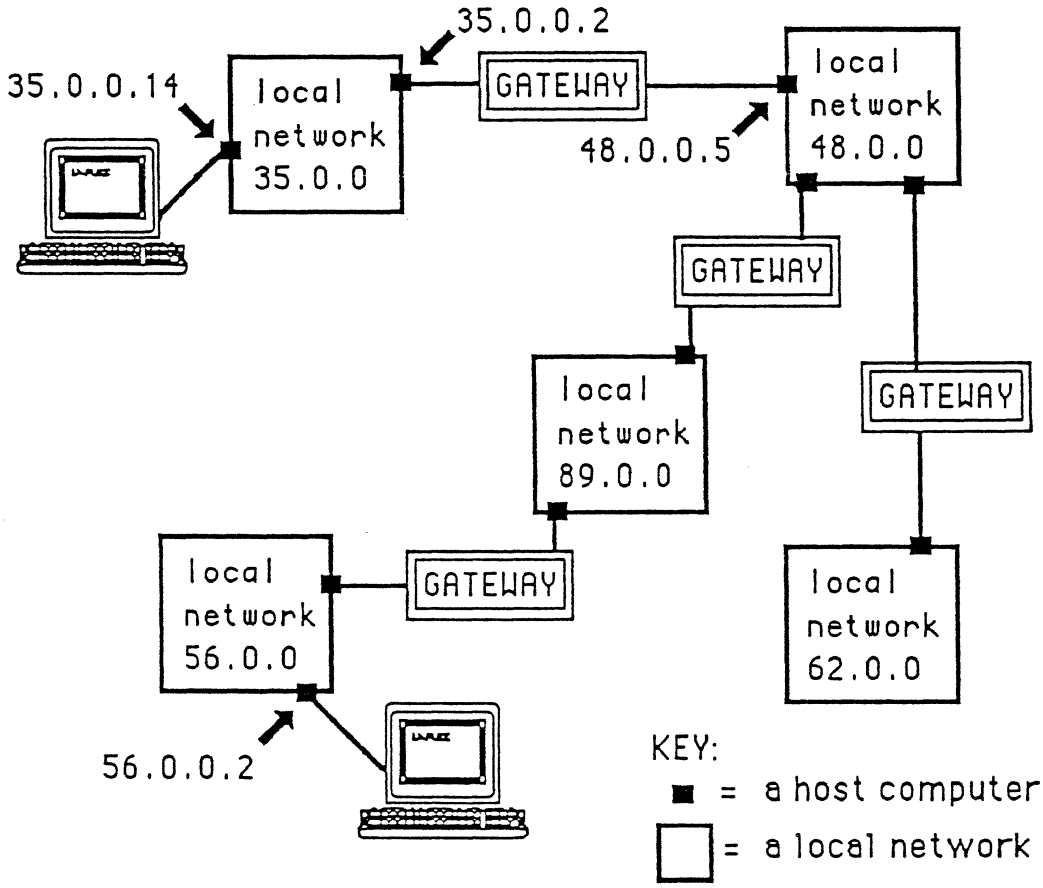


Figure 2: Interconnected Local Networks

Figure 2 illustrates five local networks connected by various gateways. The network 35.0.0, for example, is connected to network 48.0.0 via a single host computer system which acts as a gateway. This single host has two network interface devices, one with Internet address 35.0.0.2 and the other with address 48.0.0.5. When data is sent out onto local network 35.0.0 yet it is addressed to one of the hosts on local network 48.0.0, the gateway automatically causes node 35.0.0.2 to consume the data and pass it along to the appropriate network. Looking at a more complex case, suppose a user logged onto host 35.0.0.14 wants to communicate with the host 56.0.0.2. If the UniFLEX/RN auto-routing software is running on all of the connected gateways, the network data will find its way to network 56.0.0 passing through the shortest intermediate path of local networks and gateways.

5. The Network Services

The following is a complete list of the network services included with UniFLEX/RN.

ftp	User interface for ARPANET file transfer protocol.
hostname	Set or show name of current host system.
mail	Interactive electronic mail processing system.
netstat	Show network status.
ping	Test and measure low-level network connection.
rlogin	Remote login program.
rsh	Remote shell command execution.
talk	Interactive two-way user-to-user communications program.
tcp_test	Test and measure the TCP connection between two hosts.
telnet	User interface to the TELNET protocol for remote login.
tftp	Trivial file transfer program.

6. The Network C Library

The following is a complete list of the network C Library included with UniFLEX/RN.

accept	Accept a connection on a socket.
bind	Bind a name to a socket.
connect	Initiate a connection on a socket.
endhostent	End network host entry handling.
endnetent	End network file handling.
endprotoent	End protocol file handling.
endservent	End service file handling.
gethostbyaddr	Get the first entry in the network host file containing the specified host address.
gethostbyname	Get the first entry in the network host file containing the specified host name.
gethostent	Get the next entry in the network host file.
gethostid	Get the unique identifier of the current host.
gethostname	Get the name of the current host.
getnetbyaddr	Get the first entry in the network data base containing the given network address.
getnetbyname	Get the first entry in the network data base containing the given network name.
getnetent	Get the next entry in the network data base.
getpeername	Get name of connected peer.
getprotobyname	Get the first entry in the network protocol data base whose name matches the specified name.
getprotobynumber	Get the first entry in the network protocol data base whose number matches the specified number.
getprotoent	Get the next entry from the protocol data base.
getservbyname	Get the first entry in the network data base which contains the specified service name.
getservbyport	Get the first entry in the network data base which contains the specified network number.
getservent	Get the next entry from the protocol data base.

getsockname	Get the name of the specified socket.
getsockopt	Get the options on the specified socket.
htonl	Convert a long value from host byte-order to network byte-order.
htons	Convert a short value from host byte-order to network byte-order.
inet_addr	Convert a character string to an Internet address.
inet_lnaof	Extract the local network address from an Internet address.
inet_makeaddr	Create an Internet address from a network number and a local network address.
inet_netof	Extract the network number from an Internet address.
inet_network	Convert a character string to an Internet network number.
inet_ntoa	Convert an Internet address to a character string.
ioctl	Low level control function for socket and network functions.
listen	Listen for connections on a socket.
ntohl	Convert a long value from network byte-order to host byte-order.
ntohs	Convert a short value from network byte-order to host byte-order.
rcmd	Execute a command on a remote machine.
recv	Receive a message from a connected socket.
recvfrom	Receive a message from a socket.
recvmsg	Receive a message from a socket.
rexec	Return a stream to a remote command.
rresvport	Obtain a socket with a privileged address bound to it.
ruserok	Authenticate client wishing to use 'rcmd'.
send	Send a message to a connected socket.
sendmsg	Send a message to a socket.
sendto	Send a message to a socket.
sethostent	Reset host file handling.
sethostid	Set unique identifier of current host.
sethostname	Set name of current host.
setnetent	Reset network data base handling.
setprotoent	Reset network protocol data base handling.
setservent	Reset network service data base handling.
setsockopt	Set options on sockets.
shutdown	Shut down part of a full-duplex connection.
socket	Create an endpoint for network communication.