ICL PERQ

Establishing IPA under PNX

ICL endeavours to ensure that the information in this document is correct, but does not accept liability for any error or omission.

Any procedures described in this document for operating ICL equipment should be read and understood by the operator before the equipment is used. To ensure that ICL equipment functions without risk to safety or health, such procedures should be strictly observed by the operator.

The development of ICL products and services is continuous and published information may not be up to date. Any particular issue of a product may contain part only of the facilities described in this document or may contain facilities not described here. It is important to check the current position with ICL.

Specifications and statements as to performance in this document are ICL estimates intended for general guidance. They may require adjustment in particular circumstances and are therefore not formal offers or undertakings.

Statements in this document are not part of a contract or program product licence save insofar as they are incorporated into a contract or licence by express reference. Issue of this document does not entitle the recipient to access to or use of the products described, and such access or use may be subject to separate contracts or licences.

R10131/00

First Edition January 1984

ICL will be pleased to receive readers' views on the contents and organisation, etc. of this publication. Please write to:

The Registry (Readership Survey)
UK Software and Literature Supply Sector
International Computers Limited
60 Portman Road
Reading
Berks RG3 1NR

Distributed by UK Software and Literature Supply Sector
International Computers Limited
Registered Office: ICL House, Putney, London SW15 1SW
Printed by ICL Printing Services
Engineering Training Centre, Icknield Way West
Letchworth, Herts SG6 4AS

Preface

This publication describes how to establish and use the PERQ in an IPA network under the PNX operating system, release 2.0. PNX is ICL's version of the UNIX operating system for the PERQ. IPA, Information Processing Architecture, is ICL's framework for developing distributed processing systems.

Under PNX, the PERQ can be connected to an Open Systems Local Area Network (LAN). The user has access to this at two levels, these are level 3 and 4 of the ISO 7 (seven) layer model (see Fig 1.1). An interface is provided for use of the transport service, which conforms to the EMCA-72 standard. This is described in Chapter 2. When communicating with another PERQ on an Open Systems LAN, the other PERQ does not need to be running under PNX. This publication does not cover Wide Area Network (WAN) communications on the PERQ. Facilities for this are not available in IPA under PNX. Definitions of a Wide Area Network (WAN) and an Open Systems Local Area Network (LAN) are given in the glossary at the back of this publication.

However, there are non-IPA utilities available for use on the PERQ which do permit WAN communication. These are the *chatter*(1) and *mftp*(1) utilities.

Details of these facilities can be found in the publication:

ICL PERQ: Guide to PNX (Edition 2, R10139/00)

There is a special applications package for communications with, or simulation of, IBM 2780s and 3780s. Details of this application can be found in the publication:

ICL PERQ: 2780/3780 Communications under PNX (Edition 1, R10133/00)

The reader is assumed to be familiar with the PERQ, and with the relevant parts of IPA and PNX, although an introduction to the use of PERQ in an IPA network is given in Chapter 1. For more details of IPA and the facilities that can be offered under it, see the publications:

Guide to IPA (R10000/00 Issue 3)

Plannng Distributed Applications (TP90000, Edition 2)

To use the Open Systems LAN transport service, you will need to write programs in the language C, and should refer to the standard text:

Kernighan, B.W., and Ritchie, D.M., *The C Programming Language*, Prentice Hall, Englewood Cliffs, 1978.

The ECMA-72 standard is given in the following publication of the European Computer Manufacturers Association:

Local Area Networks Layers 1 to 4 Architecture and Protocols (TR/14 ECMA 1982)

To set up a connection to a remote system which is a PERQ running under POS, see the publication:

Establishing IPA under POS (Edition 1, R10120/00)

To set up an IPA connection to a remote ICL system that is not another PERQ, you should consult the relevant IPA publication for the remote regime. For connection to a system from another supplier, the documentation for that system must be consulted.

The following publications are relevant to the use of PERQ under PNX:

ICL PERQ: UNIX Programmer's Manual, Volume 1 (Edition 7, R10125/00)

ICL PERQ: UNIX Programmer's Manual, Volume 2A (Edition 7, R10126/00)

ICL PERQ: UNIX Programmer's Manual, Volume 2B (Edition 7, R10127/00)

An up-to-date guide to data communications is provided by the series *The Principles of Data Communications*, prepared by ICL and published by Heinemann Computers in Education Ltd. The four volumes in the series, for which no specialised knowledge beyond a general appreciation of data processing has been assumed, are:

Basic Concepts in Data Communications — a general introduction to the subject. Topics covered include: the development of computer networks, advances in technology and various aspects of distributed systems.

Communications Network — deals with a number of aspects of networking including characteristics, services, techniques, components and implementation.

Techniques in Data Communications — deals in some depth with a number of technical aspects of communications such as the theory and techniques of data transmission, protocols and data communication standards.

Structure in Data Communications — concerned with communications architectures and related topics.

These books are available through booksellers or direct from the publishers,

Heinemann Computers in Education
Windmill Press
Kingswood
TADWORTH
Surrey

Typographical convention

This publication follows certain typographic conventions established in the UNIX Programmers Manual (UPM). When a program name appears in the text, it will be written in lower case italics, with a number in brackets to indicate the section in the UPM in which a specification of that program may be found.

Examples of the format used are *mknet*(1), *read*(2) and *ioctl*(2). The specification of *mknet* would therefore be found in UPM Volume 1, section 1, and the specification of *read* and *ioctl* would be found in UPM Volume 1, section 2.

Contents

The text of this publication is divided into chapters in the normal way, and each chapter is subdivided into sections. A section's level in the hierarchy is indicated by its number. Therefore, within Chapter *n*, first level section headings are numbered *n*.1, *n*.2 and so on; second level headings are numbered *n*.1.1, n.1.2 ... *n*.2.1 and so on.

The contents list and index, and cross-references in the text, all refer to section numbers.

Pages are numbered within chapteapters, in the form *c-p*, where *c* is the chapter number and *p* the page number within that chapter. Figures and tables, where they appear, are also numbered within chapters, so that Figure *n*.2 is the second figure in Chapter *n*, and Table *n*.2 is the second table in that chapter.

Section numbers, page numbers and figure and table numbers in appendices are preceded by the letter A.

Introduction

Chapter 1

1.1 Information Processing Architecture

IPA, Information Processing Architecture, is ICL's framework for developing distributed processing systems. PERQ under PNX can be part of an Open System as described in the ICL publication: *Guide to IPA*.

As a matter of ICL policy, IPA is consistent with the relevant international and national standards, which are described in detail in *Guide to IPA*.

The next three sections give a brief guide to these standards and the parts of IPA relevant to the networking facilities available on PERQ under PNX.

1.1.1 ISO 7-layer model

The highest level standard to which IPA conforms is the 7-layer model of the ISO, International Standards Organisation. This model separates the functions of communication across a network into 7 distinct levels, from the physical aspects of the electrical signals at layer 1 up to the activities in the application program which uses the communications facilities in layer 7.

Figure 1.1 shows the layers in this model, and the corresponding terms in IPA, with an example function at each level.

This standard does not say anything about how the activities at each layer are to be implemented, but simply that activities at these levels must be kept separate. Each layer can depend on the ones below it to supply the stated aspect of the connection.

In some implementations, there may be no activity at some of the intermediate levels.

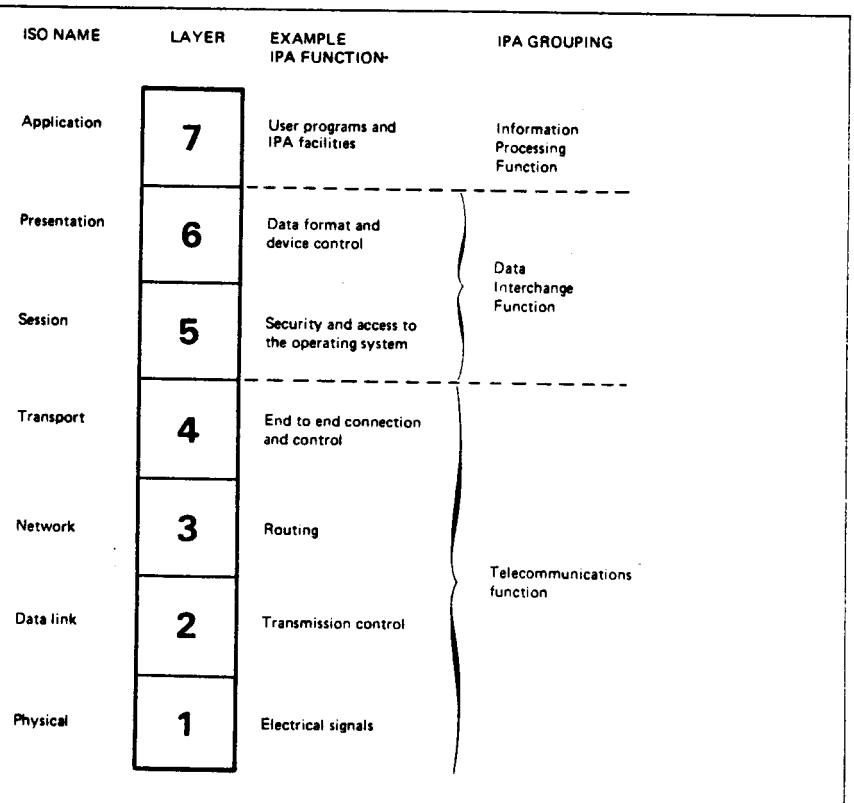

Figure 1.1 IPA grouping and functions related to the ISO 7-layer model

For more details of the ISO model and its relation to IPA see *Guide to IPA.*

1.1.2 ECMA standards

The rules and procedures that control the activities at a particular level are called a protocol. The ISO model does not say anything about what protocols should be used.

The major standards for protocols in local area networks are those published by ECMA, the European Computer Manufacturers Association. In particular, the ECMA-72 standard specifies the protocol for level 4 (the transport layer) of the ISO model. For details of the LAN standard see *Local Area Networks Layers 1 to 4 Architecture and Protocols* published by ECMA in 1982 as Technical Report 14.

This standard allows a number of variants as part of a particular open systems LAN, and details of the values used on the PERQ under PNX are given in section 2.9.

1.1.3 Open Systems Local Area Network

The IPA networking facilities available on the PERQ under PNX are for connection to an Open Systems LAN (Local Area Network).

A LAN is a network of limited size, usually within one building, or several buildings on the same site. An Open Systems LAN is one to which equipment from a variety of suppliers can be connected as long as the equipment supports the IPA networking standards. See also Glossary: "Open systems interconnection".

1.2 IPA under PNX

PNX supports connection to an Open Systems LAN. The transport service conforms with the ECMA-72 standard. It makes use of the Open Systems LAN interface, which has been provided to give the user access to the Open Systems LAN. See Chapter 2 for details of the transport service and the Open Systems LAN.

PNX provides a further facility for file transfer between PERQs which use this transport service.

Using the Open Systems LAN transport service

Chapter 2

This chapter explains how you can write programs to enable PERQs operating under PNX to communicate with each other using an Open Systems LAN. This is done using the transport service running on the PERQs under PNX.

Sections 2.1 to 2.3 explain some general aspects of the transport service, and sections 2.4 to 2.7 take you through the steps needed to set up and use a link.

2.1 Standards

The transport service corresponds to layers 1 to 4 of the ISO 7-layer model which are together known in IPA as telecommunications control. The protocols used to implement the transport service conform to the ECMA-72 standard. These standards are introduced in Chapter 1.

A program using the transport service must itself provide facilities that are needed at the higher levels 5 to 7 in the ISO Model, covering dialogue control and the application. See figure 2.1.

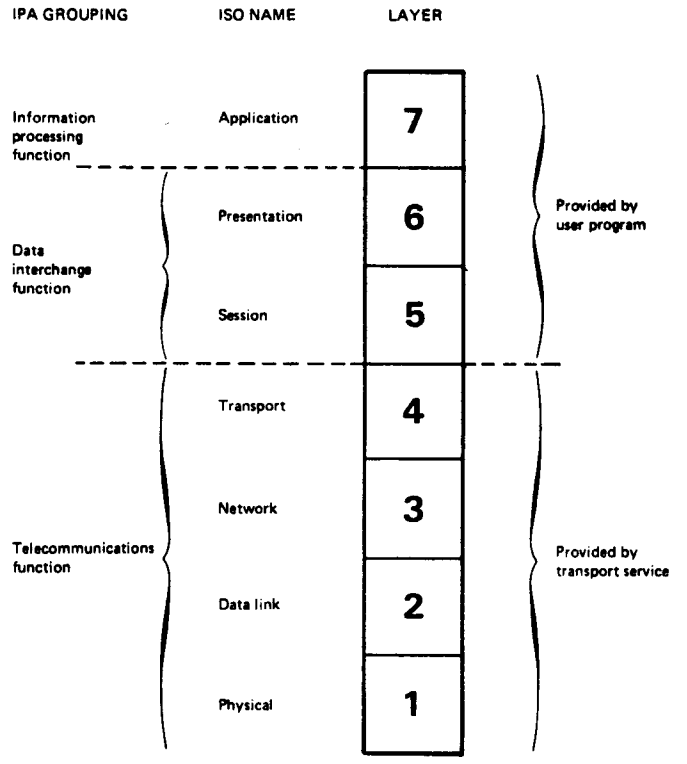

Figure 2.1 Relations between ISO layers, IPA grouping and the transport service

2.2 Agents and servers

In any communication using the transport service between two programs or two PERQs, one of them provides a service and is known as the server for that communication, while the other program or PERQ uses the service and is known as the agent. Such communications will be referred to, in this publication, as links.

A maximum of 256 PERQs may be connected together to form a network, and up to 12 links may use the transport service concurrently in any one PERQ. For each link the PERQ or program may be the server or the agent. The limit of 12 links is set in the Kernel of the PNX software.

A service can be accessed by several PERQs or programs acting as agents at the same time. A separate link is needed to connect each agent to the server. A service is set up as a special device file in the transport address file by running the *mknet*(1) program, see section 2.6. Each service or agent has its own unique filename invented by the user.

As an example, it may be that only one PERQ in a network has a printer. This PERQ can act as server for a central print service offered to other PERQs acting as agents, or to other users of the PERQ to which it is connected. This is illustrated in Figure 2.2.

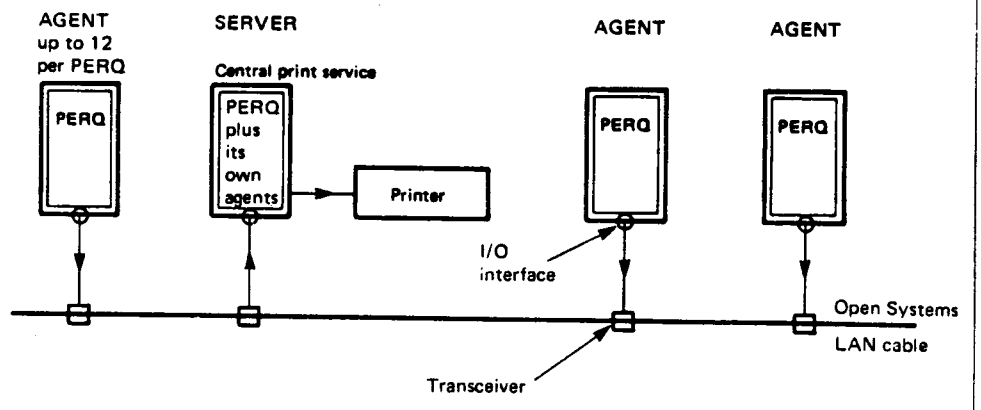

Figure 2.2 An example of three agents accessing one server for a print service

2.3 Addresses in a network

2.3.1 Network and transport addresses

Each PERQ attached to a network has a unique address built into the hardware to identify it in the network. This is known simply as the network address of the PERQ, and must be known by the network administrator. Since each PERQ, or user program, can act as agent or server for several services at once, each such server or agency has a local address to identify it within the PERQ. This local address is known as the transport selector, usually abbreviated to TSEL. The TSEL must be unique within a PERQ, but may be repeated in another PERQ for an unrelated agent or server. It is recommended however, that TSEL should be used as an identifier for the user. This means that the same TSEL implies the same facility.

These addresses are analogous to street name and house number within a street. There is no particular relation between houses with the same number in different streets. The transport selection address and the network address together are called the transport address of the server or agent.

In establishing a link only the agent need be aware of the transport address of the other end of the link.

Figure 2.3 shows the relation of the transport selection and network addresses of the agent and server in a link.

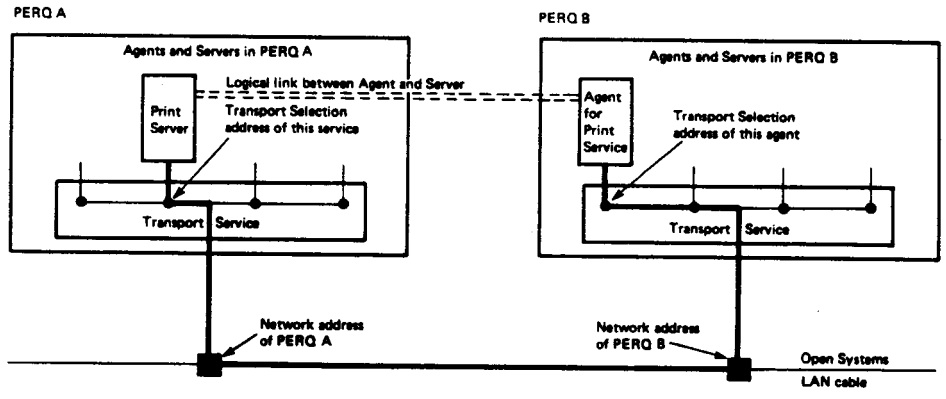

Figure 2.3 Network and Transport Selection addresses in a link between an agent and a server

2.3.2 Addresses and filenames

The network and transport selection addresses of the agent and server are used in setting up a link between two PERQs. How to do this is explained in section 2.6.

The programs which provide and use a link, however, do not use these explicit addresses. At the program level, each service appears as a character special device file, with a filename obeying the conventions for such filenames, and similarly each agent appears as a file with its own filename. Defining the correspondence between transport address and the filename is part of the set up procedures described in section 2.6. The character special device file is one part of the transport address file which is set up when the *mknet*(1) program is run. Both files may be found in the directory /lan as can the Transport address file. A list of character devices can be found in the file /lan/names. The file /lan/names is also set up by running *mknet*(1). /lan is a directory holding all the files that are relevant to the local area network to which the PERQ is connected. It is set up by running the utility *mkdir*(1).

2.4 Steps to use the transport service

The main steps in setting up and using the transport service are:

1 Connect each PERQ to the network. See section 2.5

2 Set up a transport address file in each PERQ specifying the names and addresses of the services offered as a server and those requested as an agent. This is done using the program *mknet*(1). See section 2.6 and the ICL publication *ICL PERQ: Guide to PNX* R10139/00 Edition 2 for details of how to run the *mknet*(1) program

3 Set up the ability to use a link by writing your own program in the C language using the standard UNIX *open*(2) command for the corresponding file in each PERQ. See section 2.7

4 Use the link by making calls in your program, to *read*(2), *write*(2) and *ioctl*(2) commands to the corresponding files. See section 2.7

2.5 Physical connection

Each PERQ connects to the network via a special I/O interface socket called the Open Systems LAN interface. It is labelled OSLAN or OSLAN A on the processor cabinet. For PERQ 1, the connection to the Open Systems LAN is made via the optional I/O board, which is available on order from ICL. For PERQ 2 the connection to the open systems LAN is made via the standard EIO board using the optional OSLAN facility, which is available on order from ICL.

The cables and junction boxes forming the network can also be ordered from ICL. Details of these hardware components are given in *ICL PERQ: User Guide*. Consult your ICL support staff for further advice and information about ordering the hardware parts.

2.6 Establishing and Maintaining the transport address file and running *mknet*(1)

There must be a transport address file in each PERQ you wish to connect to the network. It can be created before or after the PERQ has been connected to the network. This file defines the name and address of each service to be offered or requested by this PERQ. For a link to be made the entries for the service in the transport address files of the server and the agent must match.

The next two sections define the form of the transport address file and give example files for a small network. Subsequent sections explain use of the interactive programs *mknet*(1) to set up and edit transport address files, *mftp*(1) to use the PERQ as a master or agent on the network and *slaveft*(1) to use the PERQ as a server or slave. When *mknet*(1) is run on a PERQ it will list all the facilities that are available on that machine for communicating with other PERQs. Details of all these programs can be found in the publication: *ICL PERQ: Guide to PNX* R10139/00 Edition 2.

2.6.1 Format of the transport address files

Each line in the transport file relates either to:

1 An agent in the local PERQ requesting access to a server which may be in the local PERQ or in a remote machine

or

2 A server in the local PERQ offering a service to agents in remote machines or in the local PERQ

For example, the form of the line relating to an agent in the local PERQ is:

filename service TSEL dummy network address

filename is the name of the agent file in the local PERQ. This name is determined by the user of the PERQ, and may be any valid *filename.*

service TSEL, abbreviated to srvTSEL, is the transport selection address of the service in the local PERQ or in a remote machine. This address is determined by the user of the local PERQ or the remote machine offering the service, and is an integer in the range 1 to 65535.

dummy is not significant in this implementation, but is reserved for the TSEL of the agent in the local PERQ. It must be an integer in the range 1 to 65535, and so can simply be set to 1 in every case.

network address is a three part number with the parts titled high, medium or low and is analogous to the town, street and house number in a real life address. Each part is an integer in the range 1 to 65535 and is fixed and unique, worldwide, for each particular PERQ. If the high and medium part of the address are the same for both communicating PERQs they need not be specifically given. However, if they are different then it is mandatory to give all three parts of the address. The low part of the address must always be given in any case, so to avoid confusion it is advisable to make a standard practice of giving all three parts on each occasion a network address is required to be input.

The form of the line relating to a server in the local PERQ is:

filename service TSEL

filename is the name of the file in the local PERQ offering the service. It is determined by the user and can be any valid *filename.*

service TSEL is the transport selection address of the service in the local PERQ. It is determined by the user and is an integer in the range 1 to 65535.

For a link to be made between an agent and a server, the TSEL parameter and network address in this line for the server in its PERQ must be the same as the TSEL and network address in the line for this agent in its PERQ.

The filenames can be, and usually are, different. Everytime *mknet*(1) is run for any machine on the network, the same srvTSEL must be used. This means that a standard TSEL should be invented by the user for the service on each machine. For example, if the machine is a server, or slave, the program *slaveft*(1) will provide an address of say: 105.

If the machine is an agent, or master, the program *mftp*(1) will provide an address of say: 50.

The digits 105 and 50 are invented by the user. They can be any number from 1 to the local machine maximum, for example, less than 65535 for a 64K address buffer.

2.6.2 Example transport selection address files

Suppose you wish to set up the network shown in Figure 2.4. This network shows how printers can be shared between PERQs in the network.

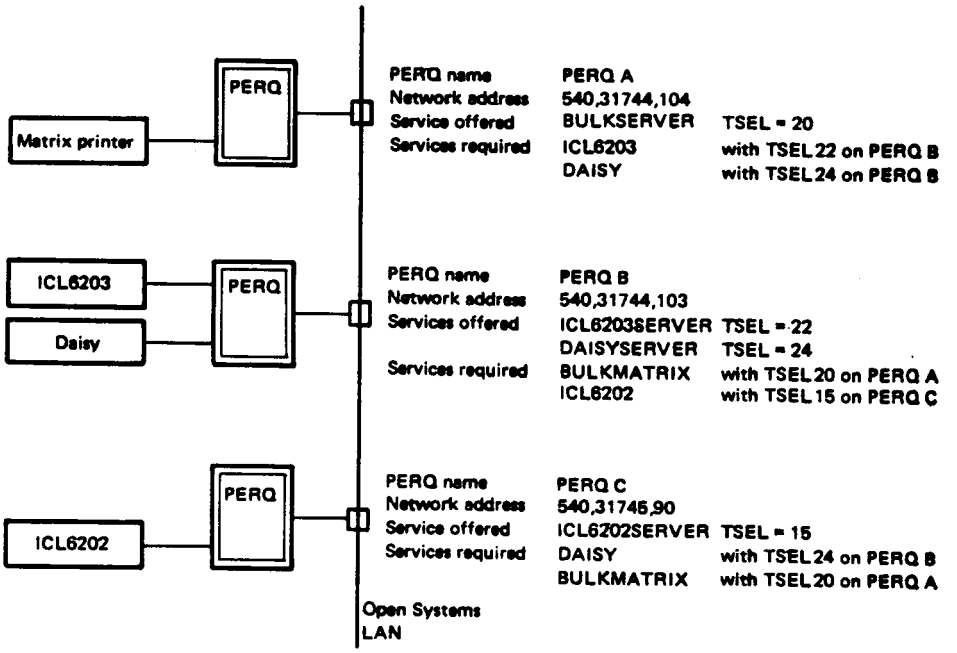

Figure 2.4 Example network showing names and addresses

The transport address files for the PERQ could be set up with the following entries:

For PERQ A, network address = 540, 31744, 104:

ICL6203	22	1	103
daisy	24	1	103
bulkserver	20		

For PERQ B, network address = 540, 31744, 103:

bulkmatrix	20	1	104
ICL6202	15	1	540, 31745, 90
ICL6203 server	22		
daisyserver	24		

For PERQ C, network address = 540, 31745, 90:

daisy	24	1	540, 31744, 103
bulkmatrix	20	1	540, 31744, 104
ICL6202 server	15		

2.6.3 Creating the transport address file

Creating and editing a transport address file is done by running the program *mknet*(1) in *superuser* mode.

mknet(1) is an interactive program which takes the user through the process with prompts and messages. *mknet*(1) makes each *filename* a type of special device in a similar manner to *mkwind*(1). In any response from the user determining a choice of alternatives, only the first character is checked, and any further characters between the first and the return are ignored. Whether the first character is upper or lower case is also ignored. In the following description only the first character is given, in upper case. If any response is given not starting with one of the allowed characters in that context, then the prompt is repeated.

If no transport address file exists for this PERQ when *mknet*(1) is run, it will give the following prompt on the screen:

Transport address file does not exist — create a new (empty) file?

If the user does not wish to create a new file, type: N which causes no action to be taken, except the message:

mknet terminates without creating the transport address file

It will not be possible to set up a link to this PERQ.

Otherwise, type: Y which causes a new, empty file to be formatted, closed and reopened. If this is done successfully the following message appears:

new (empty) transport address file formatted

Error messages which can occur during this phase are:

creating new transport address file fails - mknet abandoned

file formatting error — mknet abandoned

re-open of newly formatted file fails — mknet abandoned

In the absence of a transport address file it will not be possible to set up a link to the PERQ. If any failures occur *do not* run *mknet*(1) again. Carry out the normal investigation procedure into the state of the system. In practice this means running the *fsck*(1) utility.

2.6.4 Consistency check on the transport address file

If a transport address file already exists when *mknet*(1) is run, or following creation of a file, a consistency check is made of all special device files in the directory /lan against the transport address file

For each special device file which exists but for which there is no entry in the transport address file, the following prompt is given:

no entry for special filenames <names>, c to create entry, d to delete

Typing: D deletes the special device file and gives the message:

special device file deleted

Typing: C leads to a dialogue to set up a line in the transport address file. This starts with the prompt:

Insert filename of server or press <RET> to abort

If RET is pressed *mknet*(1) is abandoned and the user is returned to the operating system. Otherwise, if the filename is acceptable, that is it conforms to the correct criteria, the next prompt is:

TSEL-ID of service (which may be a different file)?

The user should type in an integer in the range: 1 to 65535 which is the transport service identifier for the service corresponding to the special file. This is the second parameter in the line in the transport address file.

An invalid response gives the message:

parameter should be positive decimal — please retype

After a valid reply, mknet gives the prompt:

does the file offer a service — (reply y or n)?

If the required response is Y, enough information has now been obtained, and the consistency check continues with the next special filename, if any.

Typing: N indicates that the PERQ is acting as an agent to access a service in another machine, so the address details in the remote machine must be obtained. The next prompt is:

TSEL-ID to identify caller when requesting service?

A decimal integer in the range 1 to 65535 is required, but since this value is not significant in the current implementation, the response can always be 1.

The last prompt in this phase of the consistency check is:

address of m/c providing service

Type in the three part network address, which is the fourth parameter in the line of the transport address file, as descibed in section 2.6.1. It is advisable to give all three parts of this address every time to avoid any confusion as to which agent or server is being checked. Each part is an integer in the range 1 to 65535 and is separated from the next part by single commas.

This line of the transport address file is now complete, and the consistency check continues with the next special device file, if any.

After checking all special device files, any entries in the transport address file for which no corresponding special file exists are removed and the following message is displayed:

spurious entry <num> srvTSEL = xxxx has been removed

This is repeated until all spurious entries have been removed.

An open call is then made which will force the transport service to update the network address entry in the transport address file. Should no EIO board or optional Open System LAN facility be present then the following message will be displayed:

no EIO board present on this machine (or an equivalent message for OSLAN)

When the consistency check is complete, there is a one to one correspondence between special device files in the directory /lan and entries in the transport address file.

2.6.5 Updating the transport address file

When the consistency check of the transport address file is complete, there is an opportunity to inspect the file and make changes to it. *mknet*(1) puts out the prompt:

supply special filename <RET> to terminate, l to list

Typing: l (which must be lower case) gives a list of the transport address file in this PERQ. Applied to the PERQ A example shown in section 2.6.2, together with headings and account of entries, it comes out as

filename	srvTSEL	usrTSEL	mc address	
ICL6203	22	1	540, 31744, 103	(agent)
daisy	24	1	540, 31744, 103	(agent)
bulkserver	20			(service offered)

3 entries found/local network address is 540, 31744, 103

The prompt is then repeated.

If however this prompt is responded to with the name of a special device file, this should be given without the prefix /lan/

Either the special device file already exists in the directory /lan (see the publication *ICL PERQ: Guide to PNX R10139/00 (Edition 2)*, or it does not. If it does not exist, no local network address information will be displayed by the l command and an attempt is made to find space in the transport address file for a new entry. If there is no space, *mknet*(1) puts out the message:

unable to create <*filename*> due to full transport address file

and the program returns to the prompt to supply a special filename. In this circumstance it is only sensible to respond with the name of an existing file, or with one of the other types of response.

Provided there is space in the transport address file, the following prompt appears:

new entry, c to create entry, otherwise no change

Typing: C causes a special device file with the given name to be created, and an interaction follows as described in section 2.6.4 to get details for the entry in the transport address file. If the special device file cannot be created messages output are:

filename cannot be used for service name

or

failure to create special device file

When this is complete, *mknet*(1) returns to the prompt to input another filename given at the head of this section.

Any response not starting with C will cause no action. This is in case the *filename* has been given wrongly and the user intended to give the name of an existing file. *mknet*(1) returns to the prompt to input a *filename.*

If the *filename* given is of an existing special device file, the user is given the opportunity to amend or delete the corresponding entry in the transport address file: following the consistency check, such an entry must exist. First the program gives details of the current entry. If this is for a service offered by this PERQ, *mknet*(1) displays the message:

TSEL-ID of service is <n>
file offers a service
d to delete, c or u to update — otherwise no change

If the entry in the transport address file is for this PERQ to act as an agent to access a service in a remote machine, the middle line of this message is replaced by:

file uses TSEL-ID <n>
to call machine address <n>

Typing: C or U allows the entry in the transport address file to be amended, with the interaction as described in section 2.6.4. The program then returns to the prompt to input filename.

Typing: D causes the special device file and the corresponding entry in the transport address file to be deleted. The program outputs the message:

special device file deleted

and returns to the prompt to input a *filename.*

Any response not starting with D, C or U, for example pressing the return key, causes no change. The device file and the transport address file are left unaltered, and the program returns to the prompt to input a *filename.*

If the prompt to input a *filename*, described at the head of this section, is given a null response by pressing the return key only, the *mknet*(1) program terminates.

The error messages that can appear during *mknet*(1), in addition to those given in section 2.6.3, are:

transport address file read failure — mknet abandoned

transport address file update failure — mknet abandoned

error on read from standard file — mknet abandoned

In each case do not attempt to run *mknet*(1) again. Carry out the normal investigation procedure into the state of the system. In practice this means running the *fsck*(1) utility: *mknet*(1) may then be rerun prefixed by use of the I command.

2.7 Writing programs to use the transport service

When the *mknet*(1) program has been run successfully in each PERQ connected to the network, all the information is in place ready for links to be made across the network. To establish and use a link a program must be written in a suitable language, for example the C language, for each PERQ, with calls to the standard functions *open*(2), *close*(2), *read*(2), *write*(2) and *ioctl*(2), as described in the following sections.

2.7.1 Establishing a link

For each service offered by your PERQ acting as server, the service is made available across the network by opening the corresponding special device file. If the call to open fails it returns a value -1, otherwise it returns a positive integer as the file descriptor. Further information concerning any *open*(2) failure, can be obtained using the standard C mechanisms.

Your PERQ or program acting as agent establishes a link to such a service offered in another PERQ or program by opening the corresponding special device file. Thus, when the special device files in this and/or the remote PERQ are open, the link is established.

The PERQ or program acting as server is not automatically told that the link has been made, and the program in that PERQ should periodically make a call to *read*(2) or to *ioctl*(2) with the argument ECMAIPSTATE for each server to see if a link has been made to it (see section 2.7.5).

If an attempt is made to open a special device file which does not exist, and which therefore has no entry in the transport address file, the *open*(2) fails.

The *filename* of the special device file should consist of the characters ft prefixing the name of the most common user of that device. For example, ftsteve.

2.7.2 Using a link

Once a link is established, your program can read and write over the link. The maximum data size that can be transmitted over a link or held in a queue is 522 characters.

There is an input queue and an output queue for each end of the link. When data is sent across a link to your program, it is normally added to the end of the input queue for that link in your PERQ. Data is placed at the head of the queue if it was issued with an expedited write in the sending PERQ or program, as explained below.

Data remains in the input queue until your program issues a read call to use it.

When your program writes data across a link, it is placed in the output queue for that link in your PERQ. The data stays in the queue until the transport service sends it across the network to the input queue in the receiving PERQ. The transport service does not tell your program when data has been received correctly, but successful transmission in sequence can be assumed provided that calls to read, write and ioctl return values indicating correct transmission.

Data is normally put at the end of the output queue for the link in your PERQ. If the queue is full, your program will be suspended until space becomes available, or the link is closed or broken due to an error.

Data can be put at the head of the output queue by preceding the write call by a call to *ioctl*(2) with the parameter ECMAXNEXT. This is called an expedited transfer, and when the data is received at the other end of the link it is put at the head of the input queue, and so it is the next data to be read by the receiving program. Again if the queue is full, your program will be suspended until space becomes available, or the link is closed or broken due to an error.

If your program issues a read call when there is no data in the input queue, your program will be suspended until data is received, or the link is closed or broken due to an error. To enquire whether any data is available in the queue, call *ioctl*(2) with parameter ECMAIPSTATE, and the first character of the parameter area will show the state of the queue, as explained in section 2.7.5.

Cooperating programs in the sending and receiving machines may take further precautions to ensure the safe transfer of data. For example, many application programs transfer data on a handshake basis, that is, the agent program sends a message indicating that it is ready to send data, and the server program responds with a message that it is ready to receive data. The data is then sent, and the server responds with a message to show that the data has been received.

The *mftp*(1) program is used on the PERQ to manage and control the transfer of files to another PERQ or program. The *slaveft*(1) program, which is transparent to the user, is the program in the server machine that receives, and acknowledges receipt of files to the *mftp*(1) program in the agent machine. For details of *mftp*(1) and *slaveft*(1), see the publication *ICL PERQ: Guide to PNX* R10139/00 Edition 2.

2.7.3 Closing a link

To close a link that was established by the program in your PERQ acting as an agent, the program issues a call to close for the appropriate special device file. This file is referred to by using the file descriptor given when the file was opened, as described in section 2.7.1.

The service remains available in the other PERQ, and a link can be made to it again by opening the special device file.

2.7.4 Withdrawing a service

To withdraw a service offered by your PERQ acting as server, the program issues a call to close for the appropriate special device file. This file is referred to by using the file descriptor given when the file was opened, as described in section 2.7.1.

Any link open to the service will be closed. The service can be made available again by opening the special device file. Links to it will have to be made again in just the same way as when the service was first offered (see section 2.7.1).

2.7.5 Parameters to *ioctl*(2)

Only the following parameters to *ioctl*(2) are unique to the transport service:

ECMAIPSTATE defined as (('e'<<8 | 0)

If the call to *ioctl*(2) with this parameter is successful, it will return in the first character of the parameter area addressed by the third argument:

0 if no data is available

1 if data is available

If the call is unsuccessful then the connection has been broken.

ECMAXNEXT defined as (('e'<<8 | 1)

This sets the next write to be issued as an expedited write.

2.8 Open Systems LAN transport service specification

The transport service uses the following constants in its implementation of the EMCA-72 standard:

Retransmission count	= 4 retries
T1	= 2 seconds
Acknowledge to verify an idle connection	= every 5 seconds

Disconnect request for a connection is sent if 20 seconds elapse and no acknowledge is received.

See also section 1.1.2.

Glossary

Appendix 1

This glossary contains definitions of communications terms used in this publication. Where helpful, terms used in definitions that are themselves defined elsewhere in the glossary are printed in *italics.*

Agent/Master

A computer that provides access over a *network* to a service on another computer called the *server* or *slave.*

Data transfer phase

The second phase in the *IPA* Transfer Utility (*mftp*(1)) where data is actually transmitted along a communications line. The line *protocol* controls this phase.

ECMA-72

The European Computer Manufacturer's Association (ECMA) has established a set of standards for computer interconnection. The ECMA-72 standard defines the *transport service* level.

Initiator

The part of the *mftp*(1) program that initiates a file transfer. Either the local or remote machine can provide the initiator.

IPA

Information Processing Architecture, ICL's framework for developing distributed processing systems, that is *networks*, their *links* and the facilities that use them.

ISO

The International Organisation for Standardisation (ISO) have defined a Reference Model for Open Systems Interconnection: the 7-layer model. The model forms an architectural framework, within which standards can be defined. ECMA-72 is an example of a standard which conforms to level 4 of the ISO 7-layer model, with other standards for other levels.

Link

The connection between two systems attached to a *network.*

Local area network (LAN)

A *network* for computers and similar devices located within a small area, usually a single building, or a complex of buildings all on one site.

Master File transfer Utility (*mftp*(1))

The PNX facility in *IPA* for the transfer of files from one PERQ to another in a *network.*

Negotiation phase

The first phase of the PNX *IPA* facility *mftp*(1), where the remote and local machines agree the type of transfer that is to occur.

Index

READER'S COMMENTS
ON PERQ SYSTEMS CORPORATION DOCUMENTS

PERQ Systems Corporation wishes to provide you with documents that are clear, complete, and accurate. To the extent it is possible, our documents are user-tested before release; however, errors can still occur. You can help us provide you with excellent documentation by taking a few minutes to report any inaccuracies you discover, improvements you feel are needed, or features you find especially helpful. Please send this form to us any time you encounter a feature you'd like to comment on.

Name of Document _____ Release Date _____

Section Title, if applicable _____ Page ____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

(over)