Laboratory for Concurrent Computing Systems
Technical Report 31-041

Version 1.0 February 1993

Remote Control of MS-DOS hosted devices from UNIX

S Zeng
Swinburne University of Technology
Australia

Professor G. K. Egan
Swinburne University of Technology
Australia

A. Sekercioglu
Swinburne University of Technology
Australia

LABORATORY FOR CONCURRENT COMPUTING SYSTEMS
COMPUTER SYSTEMS ENGINEERING
School of Electrical Engineering
Swinburne University of Technology
John Street, Hawthorn 3122, Victoria, Australia.

Remote Control of MS-DOS hosted devices from UNIX

S. Zeng G. K. Egan A. Şekercioğlu

February 1993

1 Introduction

Main objective of this project is to develop the Application layer software for setting up the communication between an MS-DOS hosted device and a UNIX hosted controlling program. For example, the device to be controlled can be a robot arm, or an autonomous experimental vehicle. The main reasons for developing this package can be summarized as:

- A numerically intensive control algorithm can be executed on a networked host such as Cray Y-MP-EL of the LCCS, and generated control commands can be sent via network link to the device to be controlled.
- A user can perform the experiments anywhere in the network remotely.
- Several users can share the same controlled device for their experiments.

The **NetComm** package has two main modules: **PCServer** running on the MS-DOS side and **UNIXClient** running on the UNIX side. The original **PCServer**, which is developed by Luigi Rizzo (luigi@iet.unipi.iet), is an MS-DOS hosted transputer to network interface program. It works with the IServer originally written by Inmos and modified by Luigi Rizzo. The **PCServer** and **UNIXClient** are the modified versions of these programs which are adapted to our specific application goals.

This Technical Report introduces the general structure of these programs and contains the information related to the use of them.

2 PCServer

The flow chart of the simplified PCSERVER is indicated in Figure 1

Other two important functions in PCSERVER are described as below:

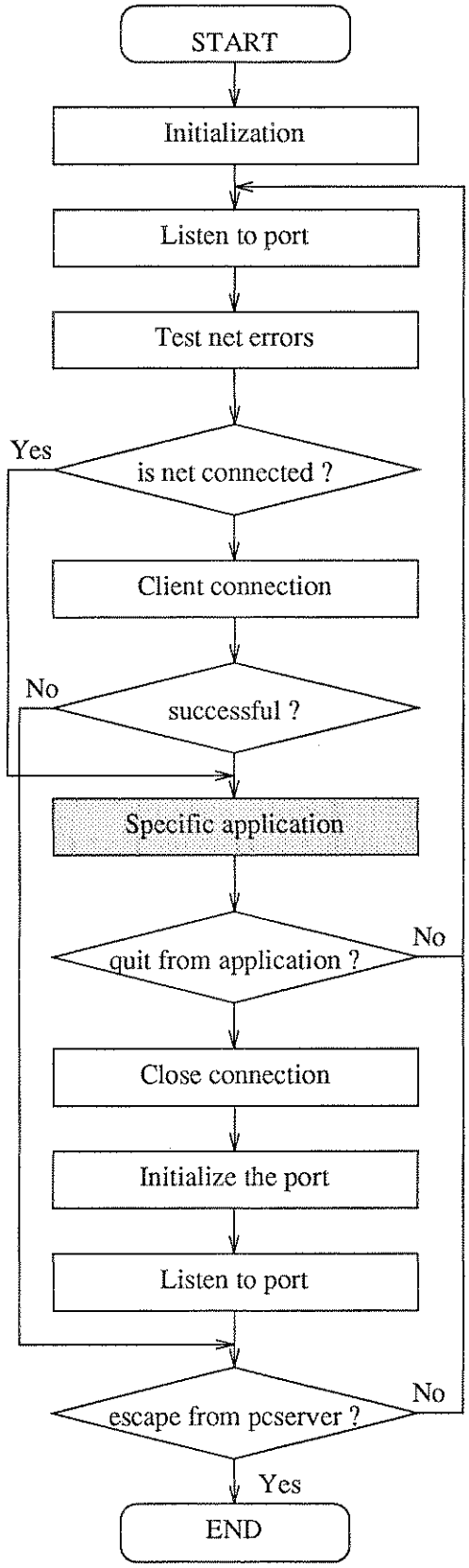

Figure 1 Flow chart of PCSERVER

2.1 NAME

write_to_net - write an string from a MSDOS machine to a network machine.

read_from_net - read a string from a network machine to an MSDOS machine.

ncsa_net_init - initialize the port.

listen_port - listen to the port.

test_net_error - test the network error.

client_connect - wait for a connection from a network machine.

close_net - close the connection between an MSDOS machine and a network machine.

2.2 SYNOPSIS

```
\#include<tcp>
\#include"glob.h"

int write\_to\_net(buf)
char buf[IOBUFSIZE]

int read\_to\_net()

void ncsa\_net\_init()

void listen\_to\_port()

void test\_net\_error()

int client\_connect()

void close\_net()
```

2.3 DESCRIPTION

read_to_net() - read a string from a network machine to an MSDOS machine. It returns 1 on success and 0 on failure. The string written is in the array liobuf1 which is definited in the header file glob.h.

write_to_net() - write a string from an MSDOS machine to a network machine. It returns 1 on success and 0 on failure.

ncsa_net_init() - network initializations.

listen_to_port() - listen to the TCP port. It returns 1 on success and 0 on failure.

test_net_error() - test the network errors. If there are errors, then it displays a message on the screen.

client_connect() - wait for the connection signal.from the network side. It returns 1 on success and 0 on failure.

close_net() - close the connection between an MSDOS machine and a network machine.

3 UNIXClient

On the network machine side, several functions are provided for communication. They are described as follow

3.1 NAME

Open_pc - open the connection between an MSDOS machine and a network machine

Close_pc - close the connection between an MSDOS machine and a network machine

read_from_net- read a string from an MSDOS machine to a network machine

write_to_net - write a string from a network machine to an MSDOS machine

3.2 SYNOPSIS

```
\#include <sys/time.h>
\#include "net.h"
\#include "glob.h"

int Open\_pc(devicename)
char *devicename

int Close\_pc(TheDevice)
int TheDevice

int read\_from\_net(t)
int t

Int write\_to\_net(buf,size)
char buf
int size
```

3.3 DESCRIPTION

Open_pc() attempts to open the connection between an MSDOS machine and a network machine. The parameter devicename specifies the name of the MSDOS machine. It returns a descriptor on success and 0 on failure.

Close_pc() closes the connection established by Open_pc. The parameter TheDevice is the descriptor returned by Open_pc(). It returns 1 on success and 0 on failure

read_from_net() reads a string from an MSDOS machine to a network machine within time t. If there is no string being read in before time t elapses, it will give a message. It returns n bytes actual read in on success and 0 if time t expires and -1 on failure.

write_to_net() attempts to write a size bytes to a MSDOS. It returns size bytes written on success and 0 on failure.

4 An Example

A example of the specific application program here is turtle control. On the network machine side, the flow chart of the turtle control program can be described as:

On an MSDOS side, the specific application program is the turtle_control, the flow chart is given in Figure 3

PCSERVER can be accessed on the network machine donald under the directory /home/research/lccs/projects/netcomm/RCS_pcserver, and UNIXCLIENT under the directory /home/reseearch/lccs/projects/netcomm/RCS_unixclient.

To get the files for PCSERVER side, you should make a temporary directory, use RCS co command to check out all the files, ie. type:

```
donald> mkdir tmp
donald> cd tmp
donald> ln -s /home/research/lccs/projects/netcomm/RCS\_pcserver RCS
donald> co-lccs -l filename
```

then use ftp download these files to the MSDOS machine, follow the instructions in the README file to compile the files and generate the executable file pcserver;

To get the files for UNIXCLIENT, you can type:

```
donald> mkdir tmp
donald>cd tmp
donald>ln -s /home/research/lccs/projects/netcomm/RCS\_unixclient RCS
donald>make
```

The make command automatically checks out all the file, compiles the files and produces a executable file called turtle.

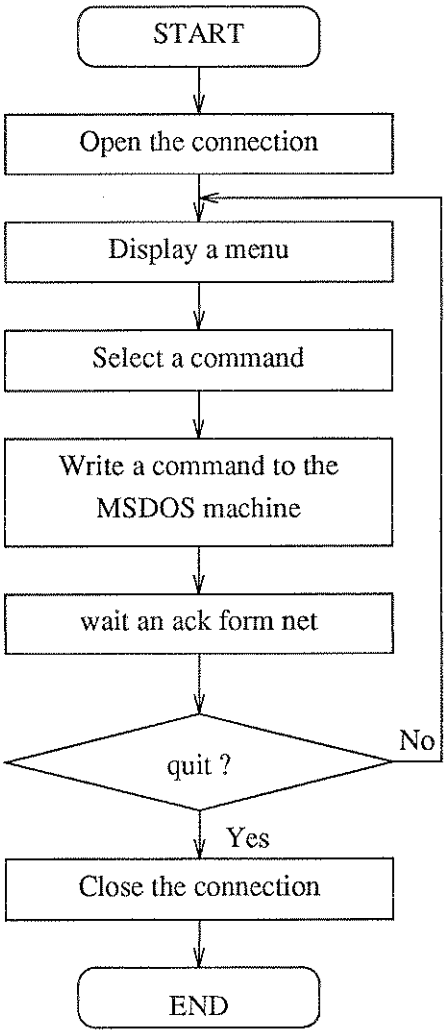

Figure 2 Flow chart of the turtle control

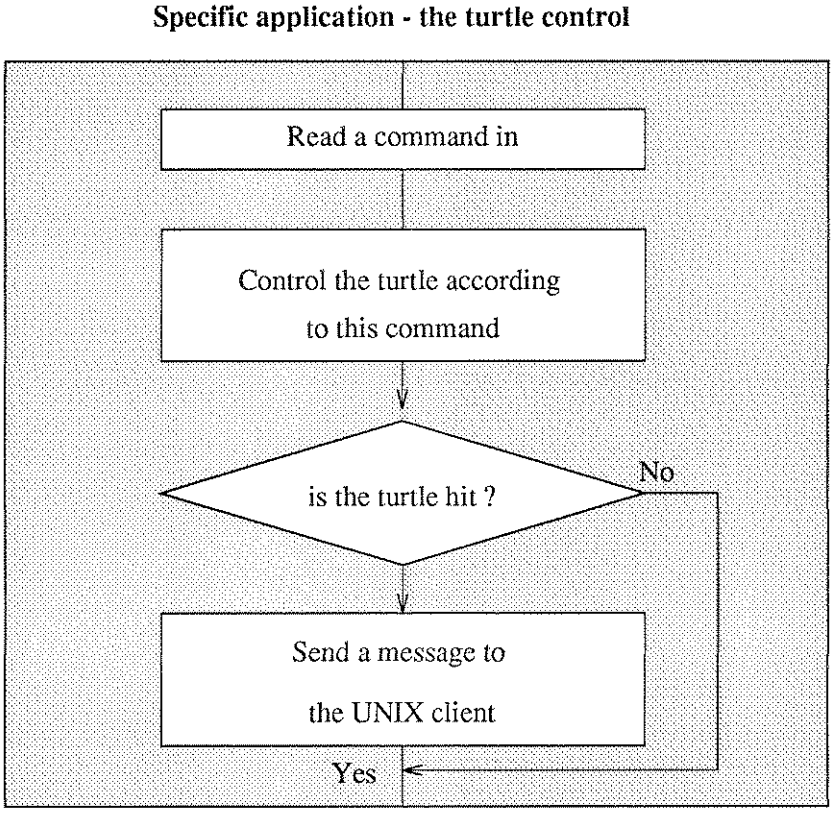

Figure 3 The flow chart of the turtle_control

5 Conclusion

The network communication software package and its usage has been described. A example of application has been given. The text files and Makefiles for both of PCSERVER and UNIXCLIENT are given in the appendix. It still needs to be consummated through the applications. If there are any bug being found, please send E-mail to ssz@stan.xx.swin.oz.au.

6 Acknowledgements

S. Zeng would like to thank:

Luigi Rizzo (luigi@iet.unipi.it) for providing the original PCSERVER and ISERVER codes;

Marc DeBruyn (mdb@stan.xx.swin.oz.au) for providing the original TURTLE control program;