TCP/IP

Connect, Inc.
1701 Quincy Avenue, Suites 5 & 6, Naperville, IL 60540
Ph: (630) 717-7200 Fax: (630) 717-7243
www.connectrf.com

Table of Contents

Streams Failure

Problem Description

The system "Stops" at some interval on TCP/IP Enabled NCU's. This is defined as the inability for RF terminals to make an external Telnet connection to a host and the inability to Ping or Telnet into the NCU.

Cause

On some TCP/IP networks there is unusually high TCP/IP traffic. This traffic can overwhelm the SCO TCP/IP stack, which can run out of a required resource, "streams buffers".

How To Identify

There are two UNIX commands you can use to detemine if this failure has occurred. These are:

netstat -m
llistat -l

Samples of failed reports are below:

netstat -m

streams allocation:

config alloc free total max fail

streams 512 81 431 1726476 106 0
queues 3368 398 2970 3699090 538 0
mblks 7250 202 704884310293 2371 0
dblks 5800 202 559875883137 2152 263
class 0, 4 bytes 496 40 456 667546 151 0

class 1, 16 bytes 2048 6 204229700021 373 0
class 2, 64 bytes 1424 49 137536497583 1424 0
class 3, 128 bytes 912 80 832 6053054 319 0
class 4, 256 bytes 456 27 429 898726 153 0
class 5, 512 bytes 128 0 128 1484879 9 0
class 6, 1024 bytes 92 0 92 484538 42 0
class 7, 2048 bytes 212 0 212 35519 25 150
class 8, 4096 bytes 32 0 32 61271 20 67
total configured streams memory: 1463.89KB
streams memory in use: 41.32KB
maximum streams memory used: 463.54KB

Note: The failures are in the last column and should be under the 2048 and 4096 buffer sizes that TCP/IP uses.

llistat -l

Look to the center of this report for the following:

Frames Deferred 0
Total Collisions 0
Frames Involved in a Collision 0
Out of Frame Collisions 0
Frames Dropped due to Excessive Collisions 0
Frames Dropped due to No Streams 24 <<<<<<<<<< This One
Frames Dropped due to No Resources 24 <<<<<< and This One
Bad Checksums Received 0
Bad Alignment Received 0

How To Fix

1. The first method is to identify the source of the excessive traffic and eliminate it.

2. If this is not possible, retune the Kernel and allow more buffers for the 2048 and 4096 buffers. To do this, use our tune utility and try increasing by 50% and monitoring.

Caution: It is recommended that this be attempted only by those technicians that have been CWNE certified. It is possible to tune the system and make it unusable.

Upgrade Considerations

Introduction

The TCP/IP CS (Client Streaming) host kit is a general purpose Client/Server interface that can offer tremendous performance increases in properly designed systems.

It has never mated with a standard piece of host software, as is the case with our STEP TCP/IP kit (Symbol STEP ENABLERS). For each installation, it talks to a unique piece of customer-authored or integrator-authored host code.

Over the years, we have had requests from customers and integrators to alter the way this product works. We have handled these requests by adding them to the standard product and making them command line options to the TCP/IP CS program.

One of the product improvements in R6 is to offer an interface for setting options, which previously might have required a hand edit of a configuration file.

When we worked on R6, we looked at the many ways this kit could be configured and decided to bring out only the two most common methods. This covers 99.99% of its use and simplifies the setup by not making the setup overly complicated.

As a result, when an upgrade is done from a pre R6 release, it is possible that you may need to run in a mode, that we do not give you a method of setting through the standard interface.

This section provides details on determining how your old system is running and how to configure R6 to match this.

Note: This Tech Note is authored for certified PowerNet personnel (CWNE). It requires good PowerNet and UNIX knowledge. If unsure of this procedure, please locate a CWNE for assistance.

How Do I Check My Old System?

The process that provides the CS interface to the host application is titled "tcphost". Find the options set by doing a process check and looking at the command line options for tcphost.

The command for this is:

ps -ef | grep tcphost

It will look similar to this:

0:00 ./tcphost 0 r 206.183.67.155 1028 -d 0 -crh
0:00 ./tcphost 0 w 206.183.67.151 1028 -d 0 -crh

In this example, notice that tcphost is running with just the -crh option.

How Does R6 Set Up TCPHOST?

To view the options available under our R6 version of tcphost, type the program name (tcphost) without any options.

It will cause this to be displayed:

```
PowerNet(tm) Release 6.0.0
Protected by U.S.Patent 5,513,328
Copyright (C) 1992-1999 Connect, Inc. All Rights Reserved.
Usage: tcphost N r|w IPaddr IPport [options]
N ..... host number (0-3)
r|w ..... read or write host
-d N .... debug level (9 for hex dump)
-cr ..... assemble delimited records from host
-crh .... append delimiter to records for host
-er N ... set record delimiter to N (default: |)
-of N ... adjust TCP send/rcv ports by N
-pad N .. pad all host records to length N
-b ..... use binary data instead of ASCII
-ka ..... disable keepalive socket option
```

Two setup options are available: DELIMITED and BINARY. Below are examples of how these are set up.

R6 "DELIMITED" SETUP

```
0:00 ./tcphost 0 w 206.183.67.151 1028 -d 0 -cr -crh -er 128
```

R6 "BINARY" SETUP

```
0:00 ./tcphost 0 w 206.183.67.151 1028 -d 0 -b
```

How Do I Configure R6 If I Need Some Other Option Setting?

The setup program for TCP/CS is "tcpset 2". It will write out the configuration files that are required for operation. The command line options for tcphost are contained in the following configuration files:

```
TCPHOST0.cf
TCPHOST1.cf
TCPHOST2.cf
TCPHOST3.cf
```

Once the addresses and ports are set up, edit these files and place the required options there.

Note: Once these files are edited, any re-save done through the menus will overwrite the changes. The ability to run the setup from the menus may be defeated by editing the file PCP.cf and removing the following lines:

```
%process=Setup;
menu=TCP/IP-CS;
type=utility;
startby=tcpset 2;
```

If changes are required later, run this option by typing "tcpset 2" from the Linux command line. Changes made after this should be made to the corresponding TCPHOSTn.cf files.

Direct TCP/IP

The following provides a product description for the "Direct TCP/IP" option. This product is a standard feature of System Release 5.0 only.

Revision Dates:

DATE	REASON	AUTHOR
July 16th, 1996	Initial Draft	GWK
October 16th, 1997	Added: -Lantronics Setup -Sample R5 Setup	GWK

Introduction

The Symbol Spectrum One network uses a standard RS232 connection to its network. There are many commercial devices that can be used to either move RS232 data or convert from one medium to another.

The direct TCP/IP option uses a standard RS232 to TCP/IP converter commonly called "Terminal Servers" or "Print Servers". The intent of these devices is to allow single serial devices (Printers, terminals etc.) to be connected to a TCP/IP network, without the expense of a dedicated workstation.

"Direct TCP/IP" capitalizes on this industry standard equipment and an enhancement to the Connect/RF software, to allow Spectrum One to be run over TCP/IP networks.

This scheme works best where there is an existing TCP/IP network to tie into. It is seldom cost effective to propose this system from the beginning, as the normal cabling methods for Spectrum One are generally more cost effective.

In existing systems with TCP/IP installed, it can allow you to manage multiple sites using one controller centrally located. This allows each remote site that is covered by TCP/IP to have only a low cost terminal server and a network of Spectrum One transceivers.

Caution: You are replacing something extremely reliable (point to point wire) with something less reliable (network connection). Maintenance of this wire (TCP/IP network) and the effect on loading and response times is outside our control.

During the BETA program, we uncovered no product flaws, but have spent a fair amount of time troubleshooting the core TCP/IP networks we are tying into.

It is recommended that before any attempt to install this product, a network performance test be done prior to installation to gauge health and reserve capacity of the TCP/IP network.

The effect on loading of the TCP/IP network that the Spectrum One protocol will cause can be factored using normal TCP/IP modeling techniques.

Description

Using TCP/IP Terminal Servers With Connect/RF Gateways

Connect has developed a new communications interface program that allows using TCP/IP terminal servers instead of high speed serial connections to interface from the NCU to spectrum 1 RF bases. This interface may be used alongside the current serial interface or by itself, allowing the user to mix and match as needed to match their environment.

There are no restrictions beyond normal TCP/IP rules as far as routers or bridges are concerned, allowing the customer to integrate the interface without having to worry about abnormal restrictions. The interface program is available with SCO, AIX and HP-UX NCUs.

The TCP/IP interface works by establishing a raw-socket connection between the NCU and a port on the terminal server.

A "Raw Socket" connection is defined as the ability for the host to initiate a socket connection to a specific serial port and transparently pass data in and out of the port. Xyplex and Lantronix both support this capability by assigning a specific socket number to each port. The user connects to the appropriate socket (i.e. socket 2200 is port 2 on Xyplex) and can then freely pass data in and out of the serial connection.

From the terminal server, a serial cable is run to the interface port on the RF base. In order for this to work, the terminal server must support transparent raw sockets (which most do). Currently, the Xyplex 1608 and the Lantronix mss1 have been validated as working in this manner. The Lantronix mss1 is a relatively low-cost single port terminal server, which allows adding dedicated terminal servers to accomplish the TCP/IP interface function.

As far as the bases and NCU are concerned, the TCP/IP interface is a transparent replacement for the serial connection.

What this means for the user is that bases can be connected to terminal servers on a one-to-one basis or a terminal server can be used to drive a coax connected RF LAN just as if it had been a serial connected base. For example, if a user had a campus-wide ethernet and needed to run RF in several different warehouses, the ethernet could be used to distribute the RF network to each of the warehouses, at which point the bases in each warehouse could be wired as traditional coax connected LANs.

Data Flow

GATEWAY - TCP/IP - TERMINAL SERVER - SPECTRUM ONE

1. The Gateway ties directly to the TCP/IP network through an Ethernet or Token Ring connection. This requires that the Gateway be configured with the proper TCP/IP host kit to match the target TCP/IP Topology.

2. The TCP/IP Network transfers Spectrum One packets wrapped inside TCP/IP packets.

3. The remote terminal server unwraps the spectrum One packets and delivers them or receives them from the Symbol Spectrum One Network.

Lantronics Setup

Password for the Lantronix device is "SYSTEM".

The following is the setup for a Lantronics mss1 terminal server.

Lantronix MSS1 Configuration

Note: Use the "SET PRIV" command first to enter privileged mode.

```
Local_3> show ports

Port 1: Username:                       Physical Port 1 (Job Service)

  Char Size/Stop Bits:  8/1             Baud Rate:            57600
  Flow Ctrl:            None            Session Limit:        4
  Parity:               None            Modem Control:        None
  Access:               Remote          Local Switch:         None
  Forward:              None            Backward:             None
  Break Ctrl:           Local           Terminal Type:        None
  Preferred Service:

  Characteristics:  Verify Telnet Pad

  Sessions:                     0       Current Session:          146.217.164.245
  Input/Output Flow Ctrl:       N/N     DSR/DTR/CTS/RTS/CD:       N/Y/N/Y/N

  Seconds Since Zeroed:         183358          Framing Errors:       0
  Accesses Local/Rem:           0/231           Parity Errors:        0
  Flow Control Violations:      0               Overrun Errors:       0
  Bytes Input:                  31404           Bytes Output:         14891
  Input Flow On/Off:            0/0             Output Flow           On/Off: 0/0

Local_3> show services

 MSS1 Version V3.4/5(961028)            Uptime:                 2 Days  03:02

  Hardware Addr: 00-80-a3-0f-0a-9a      Name/Nodenum:           MSS_0F0A9A/ 0
  Ident String: Micro Serial Server

  Inactive Timer (min):  30     Serial Delay (msec):    30
  Password Limit:        3      Session Limit:          4
  Queue Limit:           32     Node/Host Limits:       50/20

LAT  Circuit Timer (msec):      80    Keepalive Timer (sec):  20
  Multicast Timer (sec):        30    Retrans Limit:          10

TCP/IP Address:         146.217.28.221          Subnet Mask:            255.255.255.0
  Nameserver:           (undefined)             Backup Nameserver:      (undefined)
  TCP/IP Gateway:       146.217.28.200          Backup Gateway:         146.217.28.200
  Domain Name:          (undefined)             Daytime Queries:        Enabled

  Load Address:   00-00-00-00-00-00     Prompt:         Local_%n%P>

  Characteristics:
  Incoming Logins:  Telnet   (No Passwords Required)
  LAT Groups: 0

Local_3> show version

MSS1 Version V3.4/5(961028) Platform: 5b
```

MSS100

The command set for both the MSS1 and the MSS100 is different from the command language used in the tech note. For example, use "change bootp disable", "change rarp disable", "change dhcp disable", "change ipaddress xxx.xxx.xxx.xxx", "change subnet mask xxx.xxx.xxx.xxx", "change gateway xxx.xxx.xxx.xxx", "change speed 57600", and "change flow control none".

Example: Release 5.0 Direct TCP/IP Setup

Under "Port Setup", toggle "Type" until TCP is selected. In the "Device" section, key in the Pseudo name or IP address followed by port number.

Once this is done, a normal P port will be available in the RF setup form. In this example, it is P5.

Port Setup Page 1

Port	Type	Device	Port	Type	Device
P1	[DIGI]	[/dev/ttyi1a]	P17	[DIGI]	[none]
P2	[DIGI]	[/dev/ttyi1b]	P18	[DIGI]	[none]
P3	[DIGI]	[/dev/ttyi1c]	P19	[DIGI]	[none]
P4	[DIGI]	[/dev/ttyi1d]	P20	[DIGI]	[none]
P5	[TCP]	[mss 3001]	P21	[DIGI]	[none]
P6	[DIGI]	[none]	P22	[DIGI]	[none]
P7	[DIGI]	[none]	P23	[DIGI]	[none]
P8	[DIGI]	[none]	P24	[DIGI]	[none]
P9	[DIGI]	[none]	P25	[DIGI]	[none]
P10	[DIGI]	[none]	P26	[DIGI]	[none]
P11	[DIGI]	[none]	P27	[DIGI]	[none]
P12	[DIGI]	[none]	P28	[DIGI]	[none]
P13	[DIGI]	[none]	P29	[DIGI]	[none]
P14	[DIGI]	[none]	P30	[DIGI]	[none]
P15	[DIGI]	[none]	P31	[DIGI]	[none]
P16	[DIGI]	[none]	P32	[DIGI]	[none]

<ESC>=QUIT <SPACE>=CHANGE <ARROWS>=MOVE <CTRL><B>=PGUP <CTRL><F>=PGDN

Client Streaming Application

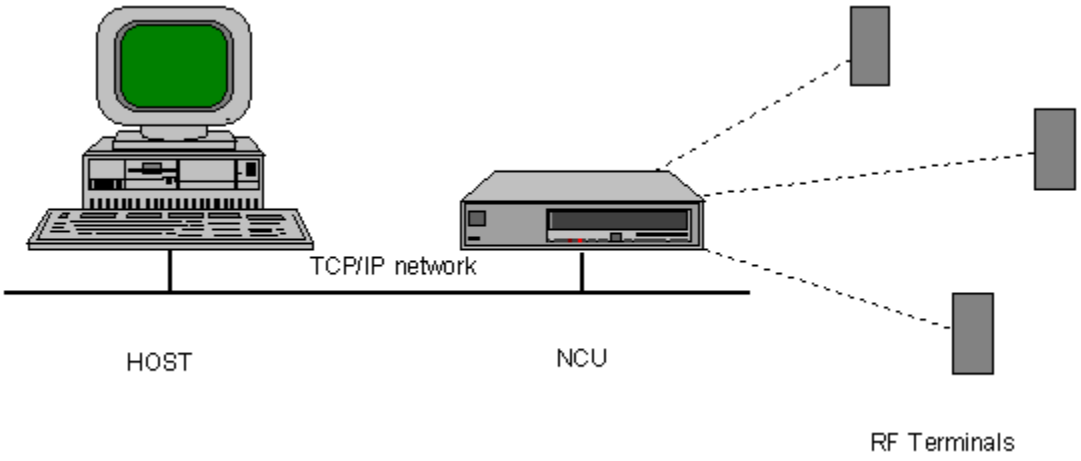
1. Introduction

The following are guidelines for developing Client Streaming applications for RF systems using the TCP/IP NCU.

In Client Streaming applications, the host application is design specifically for RF terminals. Instead of sending video terminal commands, these applications exchange specialized packets with the RF terminals.

When correctly implemented, Client Streaming allows support to a greater number of terminals and higher transaction rates at the cost of greater development effort (when compared to video terminal emulation). Client Streaming can also result in greater battery life.

The figure below shows a TCP/IP Client Streaming configuration.



2. RF Terminal Software

There are two basic options for RF terminal software in a Client Streaming environment:

- STEP - This a Symbol standard software, with commands to display messages, create input forms, send data to a printer connected to the terminal, etc. Since STEP is ready and tested, it can reduce the development costs of a Client Streaming application. On the other hand, it is a generic software, with little option for doing special data processing in the terminal (like input validation).

- Custom Software - In this option, the developer can optimize the RF communication for a specific application and implement specific data processing in the terminal. On the other hand, this development requires learning the RF terminal software architecture and development tools.

On most applications, STEP is recommended, unless it is unable to fulfill some basic application requirement.

Should you decide to roll your own RF terminal software, keep in mind that:

- A decisive factor in realizing the Client Streaming potential is the specification of the packets' exchange between the host application and the RF terminal. The packets' sizes should be kept minimum. Do not leave wasted positions and code the data in binary.

- Battery life is significantly affected by how long the radio stays active. A lower battery consume is achieved by working in an exclusive transactional form, alternating transmission and reception, instead of allowing unsolicited messages from the host.

- The RF terminal can communicate in two modes: Datagram and Session. Datagram is more efficient, but there is no delivery guaranty; Session gives reliable message delivery, at the cost of greater overhead. For most applications, Session mode is preferred.

3. Host Software

3.1. How the NCU Communicates with the Host Application

TCP/IP allows communication between two applications, be they in the same or in different computers. Each side in a TCP/IP communication is defined by two values:

- The IP address, which defines the machine the location of the application. On most computers, there is a table to associate names to these addresses.
- The port number, which selects the application in a machine. Some of the port numbers are reserved for standard tasks, like remote terminal connection (Telnet), communication testing (Echo), etc.

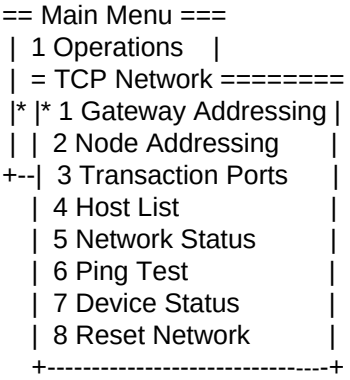
TCP/IP defines two protocols for inter-application communication: the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). UDP allows connection-less Datagram exchange, with no delivery guaranty. TCP gives a reliable message delivering service.

The communication between the NCU and the host application uses two TCP connections, one for transmission and one for reception. The messages for all the RF terminals are multiplexed over these two connections.

3.2. NCU Configuration

Before connecting the NCU to the TCP/IP network, it is necessary to make a few configurations.

All TCP/IP NCU configurations are in the TCP Network menu, found in the Main Menu.



The first step is to configure the Node Addressing option with the name and address of the involved machines:

TCP-IP Node Addresses

Name	Address	Name	Address
[ncu]	[90.0.0.1]	[ ]	[ ]
[winnt]	[90.0.0.2]	[ ]	[ ]
[ ]	[ ]	[ ]	[ ]
[ ]	[ ]	[ ]	[ ]
[ ]	[ ]	[ ]	[ ]
[ ]	[ ]	[ ]	[ ]

In the above screen, we defined two machines:

- The NCU, named ncu with IP address 90.0.0.1
- The Host, named winnt with IP address 90.0.0.2

Next, it is necessary to configure the NCU parameters in the Gateway Addressing option:

Gateway TCP-IP Setup

State	[on]	[off]
Adapter Number	[0]	[1]
Media	[ENET-sme]	[NONE]
Node Name	[ncu]	[ ]
Netmask	[255.0.0.0]	[ ]
Broadcast	[90.255.255.255]	[ ]
Router Name	[ ]	[ ]

The next step is to define the TCP connections that will be used. The NCU allows up to four TCP connections, defined in the Transaction Ports option:

Transaction Port Setup

	State	Host Node Name	Host IP Port	Gateway IP Port	Log Level
Transaction Host 0	[on]	[winnt]	[1029]	[1028]	[0]
Transaction Host 1	[off]	[ncu]	[1028]	[1029]	[0]
Transaction Host 2	[off]	[ncu]	[1028]	[1030]	[0]
Transaction Host 3	[off]	[ncu]	[1028]	[1031]	[0]

In the above screen, there is only one TCP connection active (state = on), with the following characteristics:

- The host application will be in the winnt machine and will wait for calls from the NCU on port 1029.
- The NCU will wait for call from the host application on port 1028.

Finally, it is necessary to create a Host List entry for the application:

Host List Setup

Menu Name	Handler	Active	Custom Options
[TIP-CHECK]	[CHECK]	[no]	[ ]
[STEP-CHECK]	[CHECK]	[yes]	[ ]
[tcp-cs]	[TCP/IP]	[yes]	[-cm -slp -test]
[ ]	[NONE]	[no]	[ ]
[ ]	[NONE]	[no]	[ ]

In the above screen, the entry tcp-cs will be used to connect to the host application. The options in the Custom Options field have the following meanings:

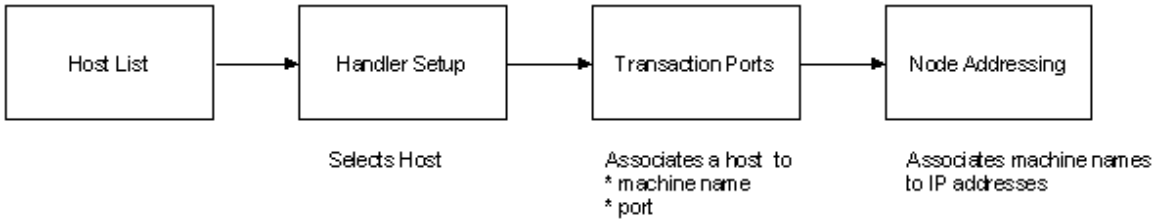
- -cm: obligatory with TCP/IP Client Streaming, assures that some commands from the RF driver do not go to the TCP/IP driver.
- -slp: stops SLP headers from being sent to the host application. The SLP protocol is used by the RF terminal when in Session mode.
- -test: makes the NCU respond to the “host ready” test made by the SLP protocol, speeding up the RF terminal connection to the host application.

In the Handler Setup for the TCP/IP entry, select which host, defined in the Transaction Ports screen, will be used:

Handler Setup

Handler [TCP/IP]
Menu Name [tcp-cs]
Host Number [0]
Monitor Level [0]

The diagram below shows the relationship between all these configurations.



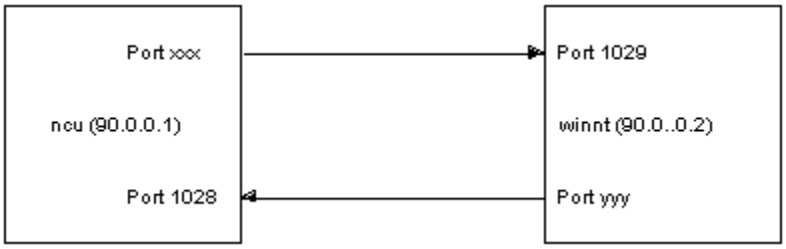
3.3. How the Communication between NCU and Host Application is Established

The communication between the NCU and the host application uses two TCP connections, one for transmission and one for reception. The establishment of these two TCP connections is as follows:

- The NCU acquires the host name from the Transaction Port Setup, and converts it to an IP address, using the Node Addressing. The host application port number is also obtained from the Transaction Port Setup.
- The NCU tries to establish a connection to the host application, with this IP address (that selects the host machine) and port number (that selects the application in the host).

- The host application, when it detects the first connection, will then establish the second connection, using the NCU's IP address and port number. The first connection will be used to send messages from the NCU to the host application and the second connection will be used the other way.

The diagram below shows the connections for the configuration screens shown before. Note that, just like in a phone connection, the one who calls needs to know the "number" (IP address and port number, in our case) of whom it is calling, but not the other way. So, the port numbers indicated by xxx and yyy are completely irrelevant.



3.4. The Format of the Messages Exchanged by the NCU and Host Application

The messages exchanged by the host application and the many RF terminals are multiplexed over the two TCP connections. This is done by adding a small header at the start of each message:

- An 'M' ASCII character
- A three-digit number that identifies the RF terminal that sent or will receive the message. The NCU generates this number by subtracting 49 from the RF address of the terminal.

At the end of each message, a delimiter must be appended. The default value for this delimiter is '|' (ASCII code 124) and it cannot be used inside the message. This delimiter can be changed to any other ASCII character by the inclusion of the option -er xxx (where xxx is the decimal ASCII code of the character) in the Custom Options field in the Host List.

The application in the RF terminal receives and sends messages without header and delimiter.

For example, to send the message ABC to a RF terminal with address 65, the following message should be sent to the NCU:

M 0 1 6 A B C |

It is important to note that the RF terminal address, and how it is converted in the three-digit number, is irrelevant to the application. When a terminal connects to the RF network, it should send a Logon Ready message to the application. When it receives this message, the application will store the number generated by the NCU and use it to send messages to the terminal and to recognize the messages it sent.

3.5. Using the Sockets Interface in the Host Application

The Sockets interface was created in the Berkeley University as an abstraction level for developing C applications that do TCP/IP communications in the UNIX environment. This interface was popularized through the years and expanded to other environments and networks.

When using the Sockets interface to develop a Client Streaming host application, these steps should be followed:

- Create (using the Socket function) two sockets, one to wait for the NCU call and the other to establish the transmission connection.
- Associate (using the Bind function) the first socket to the local address (IP address + port number).
- Associate a queue (using the Listen function) to this socket and wait a connection (using the accept function) from the NCU.
- When a connection from the NCU is received, the accept function returns a new socket that will be used to receive messages from the NCU.
- Associate (using the Bind function) the second socket to the local address (IP address + port number) and connect (using the Connect function) to the NCU.
- From this point, the application can start to receive (using the Read function) and send (using the Write function) messages.
- Before exiting, the application should close (using the Close function) all the sockets.

3.6. Example

This example was developed under Windows/NT, using the WinSock interface. It assumes that the terminal is running STEP.

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <stdlib.h>
#include <winsock.h>

void main (int argc, char *argv[]);

// STEP command buffers
// & will be replaced by ESC
char buf_gret[] = "&C00&D0401020Greating from TCP/IP&I0510001|";
char buf_loff[] = "&L|";

void main (int argc, char *argv[])
{
  WSADATA wsaData;
  WORD wVersionReq = 0x0101;
  SOCKET s_listen, s_snd, s_rcv;
  SOCKADDR_IN addr, ncuaddr;
  IN_ADDR ncuIn;
  int sizncuaddr;
  int i;
  int nrec;
  char buf_rx [512];
  char buf_tx [512];

  // Prepares STEP command buffers
  for (i = 0; i < sizeof(buf_saud); i++)
    if (buf_gret[i] == '&')
            buf_gret [i] = 0x1B;
  for (i = 0; i < sizeof(buf_loff); i++)
```

```
  if (buf_loff[i] == '&')
          buf_loff [i] = 0x1B;

// Tests if arguments where given
if (argc != 3)
{
  printf ("Usage: tcpcs ncu_port host_port\n");
        exit (1);
}

// Starts WinSock
if (WSAStartup (wVersionReq, &wsaData) != 0)
{
  printf ("Error %d in startup!\n", WSAGetLastError ());
        exit (2);
}
printf ("WinSock %s initialized.\n", wsaData.szDescription);

// Creates two sockets
s_listen = socket (AF_INET, SOCK_STREAM, 0);
s_snd = socket (AF_INET, SOCK_STREAM, 0);
if ((s_listen == INVALID_SOCKET) || (s_snd == INVALID_SOCKET))
{
  printf ("Error in socket creation!\n");
        if (s_listen != INVALID_SOCKET)
          closesocket (s_listen);
        if (s_snd != INVALID_SOCKET)
          closesocket (s_snd);
  WSACleanup ();
  exit (3);
}
printf ("Sockets were successfully created.\n");

// waits for NCU call
addr.sin_family = AF_INET;
addr.sin_port = htons ((u_short) atoi (argv[2]));
addr.sin_addr.s_addr = htonl (INADDR_ANY);
if (bind (s_listen, (LPSOCKADDR) &addr, sizeof(addr)) == SOCKET_ERROR)
{
  printf ("Error %d in bind!\n", WSAGetLastError());
  closesocket (s_listen);
  closesocket (s_snd);
  WSACleanup ();
  exit (4);
}
printf ("Bind OK.\n");
if (listen (s_listen, 1) == SOCKET_ERROR)
{
  printf ("Error %d in listen!\n", WSAGetLastError());
  closesocket (s_listen);
  closesocket (s_snd);
  WSACleanup ();
  exit (5);
}
```

```
sizncuaddr = sizeof (ncuaddr);
s_rcv = accept (s_listen, (LPSOCKADDR) &ncuaddr, &sizncuaddr);
if (s_rcv == INVALID_SOCKET)
{
  printf ("Error %d in accept!\n", WSAGetLastError());
  closesocket (s_listen);
  closesocket (s_snd);
  WSACleanup ();
  exit (6);
}
memcpy (&ncuIn, &ncuaddr.sin_addr.s_addr,4);
printf ("Received call from NCU (IP=%s, port=%d).\n",
                    inet_ntoa(ncuIn), ntohs(ncuaddr.sin_port));

// makes second conection with the NCU
addr.sin_family = AF_INET;
addr.sin_port = 0;
addr.sin_addr.s_addr = htonl (INADDR_ANY);
if (bind (s_snd, (LPSOCKADDR) &addr, sizeof(addr)) == SOCKET_ERROR)
{
  printf ("Error %d in second bind!\n", WSAGetLastError());
  closesocket (s_listen);
  closesocket (s_snd);
  closesocket (s_rcv);
  WSACleanup ();
  exit (7);
}
printf ("Second bind OK.\n");
addr.sin_family = AF_INET;
addr.sin_port = htons ((u_short) atoi (argv[1]));
addr.sin_addr.s_addr = inet_addr (inet_ntoa (ncuIn));
if (connect (s_snd, (LPSOCKADDR) &addr, sizeof(addr)) == SOCKET_ERROR)
{
  printf ("Error %d in connect!\n", WSAGetLastError());
  closesocket (s_listen);
  closesocket (s_snd);
  closesocket (s_rcv);
  WSACleanup ();
  exit (8);
}
printf ("Connect OK.\n");

// Loop to talk with the RF terminals
while (!_kbhit())
{
  nrec = recv (s_rcv, buf_rx, sizeof(buf_rx), 0);
  if (nrec == SOCKET_ERROR)
  {
    printf ("Error %d in reception!\n", WSAGetLastError());
    closesocket (s_listen);
    closesocket (s_snd);
    closesocket (s_rcv);
    WSACleanup ();
    exit (8);
```

```
  }
  else if (nrec == 0)
  {
    printf ("Conection closed!\n");
    closesocket (s_listen);
    closesocket (s_snd);
    closesocket (s_rcv);
    WSACleanup ();
    exit (8);
  }

  else
  {
          printf ("Message received:\n");
          // list buffer
          for (i = 0; i < nrec; i++)
          {
            if (isprint (buf_rx [i]))
                    printf ("%c  ", buf_rx [i]);
                  else
                    printf ("%2.2X ", buf_rx [i]);
                  if ((i & 15) == 15)
                    printf ("\n");
          }
          if (i != 0)
            printf ("\n");

          printf ("Sending answer: ");
          memcpy (buf_tx, buf_rx, 4);                  // Mxxx
          if (buf_rx [5] == 'L')                  // Logon Ready
          {
            printf ("Greatings.\n");
                  memcpy (buf_tx+4, buf_saud, sizeof (buf_saud)-1);
                  send (s_snd, buf_tx, 3+sizeof(buf_saud), 0);
          }
          else
          {
            printf ("Log off.\n");
                  memcpy (buf_tx+4, buf_loff, sizeof (buf_loff)-1);
                  send (s_snd, buf_tx, 3+sizeof(buf_loff), 0);
          }
  }
 }


 // final clean-up
 getch ();
 closesocket (s_listen);
 closesocket (s_snd);
 closesocket (s_rcv);
 WSACleanup ();
 printf ("Normal End.\n");
 exit (0);
}
```

Note: In this example, WinSock blocking functions were used. In a practical application, the use of the Asynchronous (non-blocking) functions are recommended, which will not block the application until the communication operation is concluded.

4. Conclusion

With Client Streaming, it is possible to create high performance RF applications that are able to support high terminal count and high transaction rate. Even thought Client Streaming application developing is more complex than simple video terminal emulation, it requires only some knowledge about the RF terminals and TCP/IP and the following of some simple guidelines.

5. References

- Gateway NCU System Manual, Symbol Technologies, 1995
- TCP/IP Connectivity Manual, Symbol Technologies, 1995
- Spectrum One Development System - Application Programmer's Guide, Symbol Technologies, 1992
- Series 3000 STEP - Application Programmer's Guide, Symbol Technologies, 1992
- Network Controller Interface - Programmer's Guide, Symbol Technologies, 1991
- Programming WinSock, Arthur Dumas, Sams Publishing, 1995

Troubleshooting Slow Hosts and/or Networks

Introduction

The following is a reference for troubleshooting slow hosts and/or networks.

Problem Description

Often times, you cannot tell where the slow down is occurring. The following should help you identify if it is application, host load or network related.

Resolution

For Unix type hosts, view the document at http://www.connectrf.com/Documents/loopback.

It is a shell that can be made executable at the host and/or OPEN-AIR server side.
Once that is done, the RF terminal can be directed to that host where the shell scripts reside.
Have the user login and run loopback.

Two parameters are requested:
1. Number of times to loop if zero it is continuous until the session is ended.
2. Sleep interval in seconds within the loop.

When things are expected to be slow, start a terminal running this shell and observe the display. Is the loop counter increasing at the specified sleep interval?

If yes, then you have eliminated the RF network, customers network, and host load as being the problem. At this point, the customer should contact the application and database providers for assistance.

If no, then you should move the shell to a local UNIX machine on the same subnet as the RF terminals and run the same test.

When things are expected to be slow, start a terminal running this shell and observe the display. Is the loop counter increasing at the specified sleep interval?

If yes, then the problem is network and/or host load related. Contact the customer's systems group for assistance with running SAR (System Accounting Reports) report on the host and diagnosing the network delays.

If no, verify the configuration of all Access Points and RF terminal radio drivers.

Are Your IP Addresses Correct?

Introduction

The following helps you determine if your IP addresses are correct.

Problem Description

It is sometimes difficult to tell if you are using correct and valid IP address combinations.

Resolution

Use the following address to verify your network addresses:

http://www.software-energy.com/Network/CalcNetmask.htm#Len2Mask

Or launch the document at:

http://www.connectrf.com/Documents/CalcNetmask.html

About This Document

This document is based on the following Technical Documents in our Notes Database that have been made obsolete: A1010, A1025, T1075, T1077, T1161, and T1187.

Please let us know about any errors in this document at: http://207.241.78.223/isoxpert/calltrak.nsf/WebTracking?OpenForm.