Using the RangeLAN2 791x Serial Adapter in Network Mode

Introduction

Proxim has developed a new mode of operation for the RangeLAN2 791x Serial Adapter that allows a serial device, connected to a RangeLAN2 Serial Adapter, to communicate wirelessly with an Ethernet host. This new feature is called **Network Mode**.

A Serial Adapter operating in Network Mode may communicate directly with a computer that has either a RangeLAN2 7100 ISA Card, RangeLAN2 7110 PCI Card, or a RangeLAN2 7400 Series PC Card installed or is on the same Ethernet segment as a RangeLAN2 Access Point. Figure 1 below illustrates one possible configuration for the RangeLAN2 Serial Adapter in Network Mode.

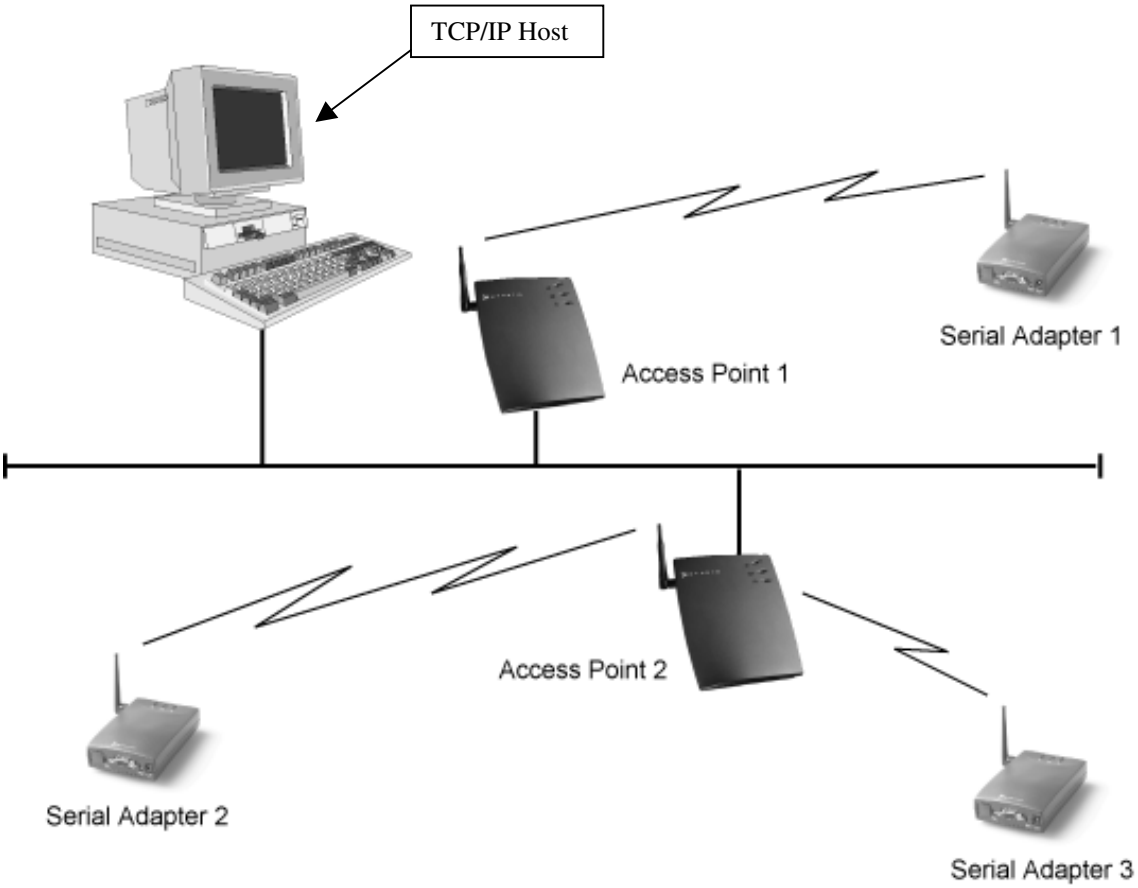


Figure 1
Serial Adapters communicating with an Ethernet Host

In the figure above, three Serial Adapters communicate with a host computer on the Ethernet network through RangeLAN2 Access Points. Note that this configuration assumes that the host computer is running the TCP/IP protocol.

© 2001 Proxim, Inc.

How Network Mode Works

Understanding Pass-Through Mode

When configured to the default setting, Pass-through Mode, a pair of RangeLAN2 Serial Adapters acts as a wireless replacement for a 3-wire RS-232 cable. Each Serial Adapter is connected to a serial device, such as a serial printer, or to a computer's serial port with an RS-232 cable. The two devices connected to the Serial Adapters communicate with each other as if they were physically cabled together, as shown in Figure 2 below.

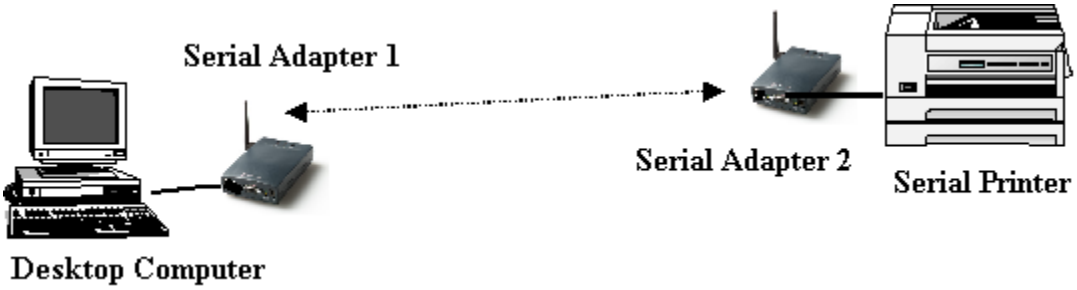


Figure 2
Pass-through Mode

In Pass-through Mode, a RangeLAN2 Serial Adapter receives characters on its serial port over the RS-232 cable from the attached device. The Serial Adapter then buffers these characters until one of three conditions (i.e., Input Timeout, Maximum Line Length, or Delimiter) is fulfilled. (Refer to the RangeLAN2 Serial Adapter User's Manual for more information on these serial parameters.) Once one of the three conditions is met, the Serial Adapter encapsulates the serial data into a TCP/IP packet and transmits the packet to the remote Serial Adapter.

This TCP/IP packet includes a custom header that contains information about serial pin status and packet size; essentially, this custom header allows the Serial Adapter to mimic RS-232 connectivity so that the RangeLAN2 Serial Adapters are transparent to the serial devices connected to them.

When the remote Serial Adapter receives the TCP/IP packet, it processes the information contained in the custom header and removes the serial data from the TCP/IP packet. The remote Serial Adapter then sends the characters out its serial port to the attached device over an RS-232 cable.

Differences Between Network Mode and Pass-through Mode

When configured to operate in Network Mode the Serial Adapter behaves exactly as described above for Pass-through Mode except it no longer appends a custom header to the TCP/IP packets it generates. This means that the packet's recipient will not be another Serial Adapter but a TCP/IP-based device. This TCP/IP device may be a stand-alone computer with a RangeLAN2 PC Card installed or it may be an Ethernet node connected to the same network as a RangeLAN2 Access Point. For example, the Serial Adapter can send a packet containing accumulated serial

© 2001 Proxim, Inc.

characters wirelessly to an Access Point and the Access Point will forward the packet to a host on the Ethernet network, as shown in Figure 3 below.

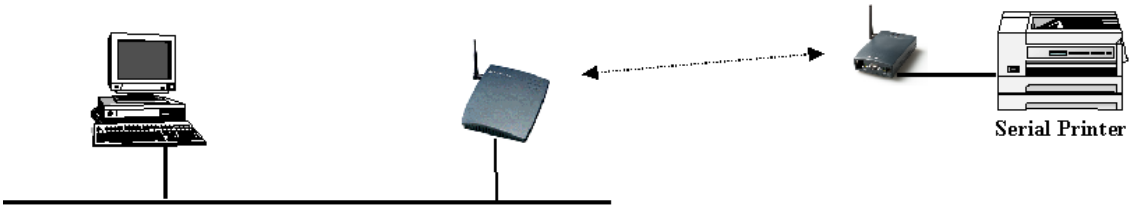


Figure 3
TCP/IP Host Communicating With Serial Printer Wirelessly

Using Network Mode to Allow Serial Devices And a TCP/IP Host to Communicate

There are two steps to establishing communication between serial devices and a TCP/IP host:

Step 1: A device converts RS-232 data into TCP/IP packets.
Step 2: The TCP/IP host interprets the packet it receives and processes the data.

The RangeLAN2 Serial Adapter's Network Mode provides only Step 1. The RangeLAN2 Serial Adapter will encapsulate accumulated serial characters into a TCP/IP packet.

Step 2 above requires a software program that runs on the TCP/IP host and processes the incoming data from the Serial Adapter. Proxim does not provide this software.

If you already have an existing RS-232 to TCP/IP connection over an Ethernet network, the RangeLAN2 Serial Adapter can act as a "drop-in" solution to make the RS-232 to TCP/IP connection wireless. If you do not have an existing RS-232 to TCP/IP solution, then you may need to write a custom TCP/IP-based software program to suit your particular application. This requirement is discussed in greater detail in the section entitled "Understanding RS-232 to TCP/IP Communication," which appears later in this technical note.

Configuring the Serial Adapter to Operate in Network Mode

Network Mode Requirements

To allow wireless communication between a TCP/IP host and a serial device, the following items are required:

1. A RangeLAN2 Serial Adapter that has flash code version 1.2-B9 installed.[1]
2. A second RangeLAN2 product that will allow the Serial Adapter to communicate with the TCP/IP host.
3. An existing software solution or a custom TCP/IP application to allow the TCP/IP host to process the data received from the RS-232 device.

[1] The Serial Adapter's software version 1.2-B9 is available for download on Proxim's Web site at http://www.proxim.com/support/.

© 2001 Proxim, Inc.

Configuring the Serial Adapter

Below is a broad overview of the steps necessary to establish communication between an RS-232 device connected to a Serial Adapter and a TCP/IP host. These steps assume that you have either an existing software solution that allows the TCP/IP host to communicate with an RS-232 device or have written a custom TCP/IP application.

This procedure is written for the case where one Serial Adapter is talking to a TCP/IP host on an Ethernet network; it can easily be expanded to have multiple Serial Adapters communicating with a single wired TCP/IP host.

1. Connect a RangeLAN2 Access Point to the same Ethernet network as the TCP/IP host.

2. Using a straight through RS-232 cable, connect the Serial Adapter to a computer running a terminal emulation program, such as HyperTerminal.

3. Type "$$$" within the terminal screen to call up the configuration menu. Alternately, change the Domain switch to 9 (if you change the Domain switch to 9, be sure to return the Domain switch to its original position after you have finished configuring the device).

4. Within the configuration menu, confirm that the RangeLAN2 Serial Adapter is running software version 1.2-B9 or greater.

5. Within the Advanced Configuration menu, configure the Operating Mode parameter to "Network".

6. Within the Radio Configuration menu, configure the Serial Adapter as a Station with the same Domain and Security ID as the Access Point you connected to the network in Step 1 above.

7. Within the Network Configuration menu, configure the Serial Adapter's Send Mode.

   Note:
   The Send Mode of the Serial Adapter must match the protocol that your TCP/IP host is using to receive messages (TCP or UDP).

8. Configure the Destination Address. The Destination Address is the IP address of the TCP/IP host on the network.

9. Configure the IP Address parameter to assign the Serial Adapter an IP address that is valid on your network.

   Note:
   The IP address you assign the Serial Adapter must match the IP address to which the TCP/IP Host will send packets.

© 2001 Proxim, Inc.

10. Configure the Subnet Mask and Default Gateway Address as required by your network. See your Network Administrator for details.

    Note:
    When using the Serial Adapters in conjunction with an Ethernet backbone, you cannot send a broadcast message through a router.

11. Configure the Local Port Number and Remote Port Number parameters as determined by your application. By default, both parameters are set to 5000. Your application may require you to change these parameters.

    Note:
    The Local Port Number must match the port on which your TCP/IP Host will listen, and the Remote Port Number must match the port to which the TCP/IP host will send packets.

12. Reset the Serial Adapter in order for these changes to take effect.

Demonstrating RS-232 to TCP/IP Connectivity With Network Mode

This section describes a simple test to confirm that the Serial Adapter has been configured correctly using two computers, Telnet, and a terminal emulation program. Follow these steps to confirm that Network Mode is operational:

1. Using a straight through RS-232 cable, connect the Serial Adapter to a dumb terminal or to a computer running a terminal emulation program, like HyperTerminal.
2. Type "$$$" within the terminal screen to call up the configuration menu. Alternately, change the Domain switch to 9 (if you change the Domain switch to 9, be sure to return the Domain switch to its original position after you have finished configuring the device).
3. Under the Network Configuration menu, change the Local Port Number to 23. This is the TCP port number reserved for Telnet.
4. Under the Network Configuration menu, configure the Destination Address so that it matches the IP address of the TCP/IP host from which you will telnet into the Serial Adapter.
5. Exit the configuration menu and launch the Serial Adapter into operating mode. Leave the terminal emulation window open and connected to the Serial Adapter.
6. Open an MS-DOS window on the TCP/IP host and ping the Serial Adapter. The Serial Adapter should respond to the ping request if it has been configured correctly and is successfully synchronized to the Access Point connected to the Ethernet network.
7. Launch a Telnet application on the TCP/IP host.
8. Type in the IP address of the Serial Adapter in the Telnet application's Host Name field. Also, confirm that the Port is set to "telnet" or "23".
9. Within the Telnet application, connect to the Serial Adapter.

© 2001 Proxim, Inc.

10. Once a session has been established with the Serial Adapter's IP address, type characters on the telnet screen. These characters should appear within the terminal emulation program's window.
11. Type characters within the terminal emulation window on the remote computer connected to the Serial Adapter. These characters should appear in the TCP/IP host's Telnet window.

Understanding RS-232 to TCP/IP Communication

When Is a Custom TCP/IP Program Required?

As described earlier in this technical note, there are two steps to establishing communication between serial devices and a TCP/IP host:

Step 1: A device converts RS-232 data into TCP/IP packets.
Step 2: The TCP/IP host interprets the packet it receives and processes the data.

The RangeLAN2 Serial Adapter's Network Mode provides only Step 1; Step 2 requires a software program that runs on the TCP/IP host and processes the incoming data from the Serial Adapter. In the demonstration described above, the Telnet program provides this capability.

If you already have a solution in place that allows RS-232 serial devices to communicate with TCP/IP devices on an Ethernet network, then the RangeLAN2 Serial Adapter will be a "drop-in" replacement for your current solution. Basically, the RangeLAN2 family will make the existing RS-232 to TCP/IP solution wireless.

For example, the Serial Adapter may be used to replace an existing RS-232 to TCP/IP connection that uses a terminal server. In this configuration, prior to the installation of the Serial Adapter, the customer already has serial devices communicating with Ethernet devices. The serial devices connect to ports on a terminal server that in turn connects to the Ethernet backbone.

Network mode allows you to replace an existing RS-232 to TCP/IP connection with a RangeLAN2 Serial Adapter. Theoretically, if you have an existing solution where a serial device can communicate to a server through a terminal server or RS-232 to TCP/IP converter, then you should be able to remove the wired RS-232 to TCP/IP converter and replace it with a RangeLAN2 Serial Adapter configured to operate in Network Mode. This is the "drop-in" solution because the software on the Ethernet devices is already in place to accept and process the converted serial data.

If you do not have existing RS-232 to TCP/IP communication, then one option is to use available software, such as Telnet and a terminal emulation program, to communicate with a Serial Adapter. This option is only applicable if your solution requires that your TCP/IP host initiates communication and connect to the Serial Adapter first. In this case, you may use Telnet on the TCP/IP host to establish a connection with the Serial Adapter and a terminal emulation program on the end of the connection to send and receive data through the serial port on the Serial Adapter. However, you may need write a custom TCP/IP program if your solution requires a

© 2001 Proxim, Inc.

more detailed communication mechanism that is beyond the scope of Telnet or a standard terminal emulation program.

If your solution requires that the Serial Adapter establish the TCP/IP connection, then you must write a custom TCP/IP program so the TCP/IP host listens on the appropriate port for incoming packets from the Serial Adapter. A sample program of this type is included later in this document.

Writing a Custom TCP/IP Program

If your solution requires a custom TCP/IP program, then this program must use a library that interfaces with the TCP/IP protocol to send and receive data.

For example, Windows Sockets is the most commonly used interface on a PC and is a standard part of Windows 98. Note that Windows Sockets programs require a compiler, such as Visual C++, in order to be executed. On UNIX machines, Berkeley Sockets is available on most of the common platforms. Many reference books are available on Sockets programming, both for Windows and UNIX. Suggested reference books include:

Comer, Douglas E., *Internetworking with TCP/IP*, Vol. 1-3, 3rd ed. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

Stevens, W. Richard, *TCP/IP Illustrated*, Vol. 1-3, Addison-Wesley, Reading, Mass., 1994.

Stevens, W. Richard, *UNIX Network Programming*: *Networking APIs: Sockets and XTI*, 2nd ed., Prentice-Hall, Englewood Cliffs, New Jersey, 1997.

Note that Proxim Technical Support cannot assist you with a custom TCP/IP application. However, Proxim has provided a sample program, shown below, to help you get started.

Sample TCP/IP Communication Program

Below is a sample TCP/IP communication program that allows a RangeLAN2 Serial Adapter to initiate a connection with a TCP/IP host. This program runs on the TCP/IP host and the host listens on the specified port for packets from the Serial Adapter.

After the TCP/IP host accepts a connection from the Serial Adapter, this program spawns a process that accepts data on that connection and logs all of that data to a file called "proxlink.a.b.c.d-n" and then continues to listen for more connections.

Note that in the file name "proxlink.a.b.c.d-n", "a.b.c.d" is the IP address that made the connection and "n" is the socket that that IP address connected from.

Note:
This code is known to compile with Microsoft C++ version 6.0. It should compile with earlier versions. You'll need to modify the "#INCLUDE" statements to compile on UNIX platforms. You can download the source at ftp://ftp.proxim.com/pub/791x/tcp.zip.

© 2001 Proxim, Inc.

```
/*************************************************************
 * tcprecv.c
 * Receives messages from one or more Serial Adapters using TCP
 *
 *************************************************************/

// link with wsock32.lib libcmt.lib

#include <sys/types.h>
#include <winsock2.h>
#include <stdio.h>
#include <conio.h>
#include <Windows.h>
#include <process.h>    /* _beginthread, _endthread */

#define TRUE  1

int acceptfunc (void *sock);

main()
{
   int sock,length;
   int msgsock;
   struct sockaddr_in server;
   struct sockaddr_in acceptedSocket;
   struct sockaddr_in *acceptedSocketp;
   void *parameters[10];
   WSADATA wsaData;
      acceptedSocketp = &acceptedSocket;

   /* Start the socket services DLLs */
   WSAStartup(0x0101, &wsaData);

   /* Create socket */
   sock= socket(AF_INET, SOCK_STREAM, 0);
   if (sock < 0)
   {
      perror("opening stream socket");
      exit(2);
   }

   /* Create a name with wildcards */
   server.sin_family= AF_INET;
   server.sin_addr.s_addr= INADDR_ANY;
   /* server.sin_port= 0; */
   server.sin_port = htons(45300);
   if (bind(sock, (struct sockaddr *)&server, sizeof(server)) < 0)
   {
      perror("binding stream socket");
      exit(3);
   }
```


© 2001 Proxim, Inc.

```
/* Find assigned port number and print it out */
   length= sizeof(server);
   if (getsockname(sock, (struct sockaddr *)&server, &length) < 0)
   {
      perror("getting socket name");
      exit(4);
   }

   printf("Socket port #%d\n", ntohs(server.sin_port));

   /* Start accepting connections. Listen() will return when it
      determines that a remote machine is trying to open a
      connection. The second parameter to listen() is a queue
      length, which is the maximum number of pending connections
      on that socket at any given time.
    */
   do
   {
      listen(sock, 5);
      /* Establish the TCP connection. Accept() returns a new
         socket number for the connection, allowing listen() to
         (if desired) continue to listen on the old socket.

         Parameters to accept() are: the socket that the
         previous listen() call was using, a sockaddr structure,
         and the length of the sockaddr structure. The second
         and third parameters are zero here, but can be used to
         get the IP address and port number of the remote end of
         the connection.
      */

         msgsock= accept(sock, (struct sockaddr *) &acceptedSocket, &length);

         parameters[0]=(void*) msgsock;
         parameters[1]=(void*) acceptedSocketp->sin_addr.S_un.S_un_b.s_b1;
         parameters[2]=(void*) acceptedSocketp->sin_addr.S_un.S_un_b.s_b2;
         parameters[3]=(void*) acceptedSocketp->sin_addr.S_un.S_un_b.s_b3;
         parameters[4]=(void*) acceptedSocketp->sin_addr.S_un.S_un_b.s_b4;
      _beginthread( acceptfunc, 0, parameters);

      } while (TRUE);
   return 1;
}


int acceptfunc (void **sock)
{
   int i;
   int rval;
   char buf[1524];
   char filename[32];
   FILE *logfile;
   int insock;
   unsigned char oct1, oct2, oct3, oct4;
```


© 2001 Proxim, Inc.

```
insock = (int) *sock;
   oct1 = (unsigned char) *(sock + 1);
   oct2 = (unsigned char) *(sock + 2);
   oct3 = (unsigned char) *(sock + 3);
   oct4 = (unsigned char) *(sock + 4);

   /* Open the log file */
   sprintf (filename,"proxlink.%i.%i.%i.%i-%i", oct1, oct2, oct3, oct4,
insock);

   logfile= fopen(filename, "wb");
   if (!logfile) {
      printf("Error: Couldn't open log file\n");
      exit(1);
   }

   printf ("Accepted from %i.%i.%i.%i\n", oct1, oct2, oct3, oct4);
   printf ("%s %i\n","Accept socket",insock);

   if (insock==-1)
   {
      perror("accept");
   }
   else do
   {
      memset(buf, 0, sizeof(buf));
         if ((rval= recv(insock, buf, 1024,0)) < 0)
         {
         perror("reading stream message");
         exit(5);
         }
         else if (rval==0)
      {
         printf("Read zero bytes. Exiting...\n");
         exit(6);
         }
         else
         {
         /* Print out data on screen and log it to the logfile.
            Note we are in Network mode (no Serial Adapter
               header bytes). */
         printf("Read %d bytes\n", rval);
         buf[rval]= '\0';
         printf("Data(%i): --> %s\n",insock, buf);
               for (i= 0; i< rval; i++)
                  fputc(*(buf+i), logfile);
         fclose(logfile);
         logfile= fopen(filename, "ab");

//              npackets++;
//              if (npackets > maxpackets)
//              {
//                 brkflag= 1;
//                 break;
//              }
      } // else
```


© 2001 Proxim, Inc.

```
   } while (TRUE); //else do

//       if (brkflag) break;

   fclose(logfile);
   closesocket(insock);
   WSACleanup();
   return (1);
//   exit(0);
}
```

Contacting Proxim Technical Support

If you have any additional questions concerning the information presented above, please contact Proxim Technical Support.

You can reach Proxim Technical Support by voice, fax, e-mail, mail, or over the Internet:

Tel: 1-800-477-6946 or 408-731-2640
Fax: 408-731-3676
Web: http://www.proxim.com/support/
E-mail: support@proxim.com
Mail: send correspondence to:

Proxim, Inc.
Attn: Proxim Technical Support
510 DeGuigne Drive
Sunnyvale, CA 94085

© 2001 Proxim, Inc.