Cross-platform TCP/IP Socket programming in REXX

Abstract:
TCP/IP is the key modern network technology, and the various REXX implementations have useful, if incompatible interfaces to it. In this session, we cover writing TCP/IP clients and servers in REXX, on MVS, Win32, and UNIX platforms, and propose a workable solution to the incompatibility issues.

Bob Stark bstark@protechtraining.com 412-445-8072

Contents
- TCP/IP Networking Review
- Socket Programming Overview
- Sample TCP/IP Socket Application
TCP/IP Application Functionality

Via TCP/IP users may:
- Use terminal access to other systems (Telnet)
- Send commands to other systems and receive responses
- Transfer and share files between systems (FTP and NFS)
- Use devices on other systems (Printing)
- Enable cross-platform communication between cooperating applications

Two Similar Packet Delivery Systems

FROM: ProTech
One Monroeville Center, Suite 622
Monroeville, PA 15146

TO: You
Your Company
Your street address
Your city, State, zipcode

Ethernet frame:
TO: 00:23:44:31:A1:19
FROM: 00:23:44:38:F1:D4
DATA: Your data goes here
Packet Routing

Packet Routing
Network Physical Layer, IP Layer

- Physical layer
  - Ethernet is the most common physical network type in use today (others include ATM, FDDI, Token Ring)
  - Each Ethernet card contains a unique hardware address, burned into it by the manufacturer
- IP Addressing
  - Every network interface card (NIC) gets an IP Address
  - Forms a network of interconnected networks
  - Example:
    - Host: 209.161.92.101 (rexxdog.protechpts.com)
    - Net Mask: 255.255.255.0 (Class C network, 254 hosts)
    - Network: 209.161.92.0

OS/390 TCP/IP Sockets

Server

- 21 ftpd
- 23 telnetd
- 53 dns
- 80 http

Client

- 1281 ftp
- 1282 HTTP
- 1953 nslookup
- 1288 telnet
## Well Known UDP & TCP Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo</td>
<td>7/udp</td>
<td>Echo Datagram back to sender</td>
</tr>
<tr>
<td>FTP-Data</td>
<td>20/tcp</td>
<td>File Transfer data transfer</td>
</tr>
<tr>
<td>FTP</td>
<td>21/tcp</td>
<td>File Transfer dialogue</td>
</tr>
<tr>
<td>Telnet</td>
<td>23/tcp</td>
<td>Telnet remote login port</td>
</tr>
<tr>
<td>SMTP</td>
<td>25/tcp</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>DNS</td>
<td>53/udp</td>
<td>Domain name server</td>
</tr>
<tr>
<td>HTTP</td>
<td>80/tcp</td>
<td>Default Web server port</td>
</tr>
<tr>
<td>POP3</td>
<td>110/tcp</td>
<td>PC mail retrieval service</td>
</tr>
<tr>
<td>NNTP</td>
<td>119/tcp</td>
<td>Network news access</td>
</tr>
<tr>
<td>NTP</td>
<td>123/udp</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>SNMP</td>
<td>161/udp</td>
<td>Simple Network Management queries</td>
</tr>
<tr>
<td>SNMP-trap</td>
<td>162/udp</td>
<td>Simple Network Management alerts</td>
</tr>
</tbody>
</table>

## TCP/IP Services

- **ARP - Address Resolution Protocol**
  - Translates between hardware address & IP address
  - Routers use arp to find out which machine on its network is to receive a given IP data packet
- **DNS - Domain Name Service**
  - Translates between hostnames & IP addresses
  - Essential for "human" use of the Internet
  - Each domain is locally managed, i.e. ProTech gets to name the machines in protechpts.com
  - Slow TCP/IP response often related to DNS problems
  - nslookup - UNIX command used to query the domain name server
TCP/IP Applications

- **telnet - Terminal emulation**
  - Allows terminal -> host connection w/o dedicated wire
  - Terminal type and capabilities are negotiated
  - TN3270 is telnet for 3270 terminals

- **ftp - File Transfer Protocol**
  - Allows files to be transferred between hosts
  - Supports transfer between dissimilar machines
    - OS/390 (EBCDIC) to PC (ASCII) or other UNIX (ASCII)
    - Dataset name <-> UNIX filename

- **sendmail - e-mail transfer program**
  - Stores and forwards mail to other users, machines

TCP/IP Applications

- **NFS - Network File System**
  - Allows remote disks on another system to appear local
  - Slower, more convenient for accessing common data
  - Has several cousins w/ additional features, complexity
    - AFS - Andrew File System
    - DFS - Distributed File System
    - OSF DCE - Distributed Computing Environment

- **WWW - World Wide Web**
  - Client-server protocol with generic client, the *browser*
  - Provides interconnected Hypertext documents, forms
  - Supports application development via CGI scripts, Java
Telnet Access

- The telnet Command
- The telnet client is a TCP/IP utility that enables terminal access
- telnet is normally invoked from the command line in the format:
  
  \$ telnet hostid

- telnet also supports an interactive mode
- Upon connection you will be prompted for Userid and Password

TCP/IP Applications: rsh, rexec

- rsh - Remote Shell
  - Permits you to issue a command on another machine and receive the response
  - Sending host, userid must be pre-authorized
    - .rhosts file contains permitted machines, usernames
  - Userid and cmd text traverses network, but not passwd
    
    \$ rsh rexxdog ls -al

- rexec - Remote Execution
  - Issue a command on remote host & receive response
  - Userid, password, and cmd text traverses the network
    
    \$ rexec -l bstark -p pswd rexxdog ls -al
  - Userid and password from .netrc file in home directory
    machine rexxdog login bstark password pswd
    
    \$ chmod go-rwx .netrc  
    (Remove others access)
FTP (File Transfer Protocol)

- File Transfer Protocol (ftp)
- ftp allows users to transfer files between systems
- Any machine (UNIX or otherwise) may make an ftp connection to any other machine if both support the protocol
- ftp has a command line mode and an interactive or command mode
- From the command line, ftp operates like telnet:
  
  $ ftp rexxdog.protechtraining.com
  
  -or-
  
  $ ftp 209.161.92.101

Socket Programming Overview

- TCP/IP implementations provide a C language programming API for socket functions
- Sockets are like file handles:
  - Open a socket, read from the socket
  - Write to the socket, close the socket
- You can use TCP or UDP protocols, but most applications use TCP, since it is easier
- Data sent and received on a socket is not translated
  - The standard is to send ASCII characters, so you must provide your own EBCDIC translation on mainframes
  - You must also work out the byte ordering for numbers (big-endian or little-endian)
- Sockets are full duplex: a program can both read and write to the same socket
Socket Programming Overview

REXX Programming interfaces are provided for:

- z/OS & OS/390 TSO/E REXX via IBM TCP/IP stack
  - Documented in SC31-8788 z/OS Communications Server
    IP Application Programming Interface Guide
  - Uses the SOCKET() function
  - Compatible with VM and VSE
- Windows, AIX, Linux, OS/2 via RxSock function library
  - Documented in Object REXX for Windows REXX TCP/IP
    Socket Library Functions (RxSock)
  - Provides 27 REXX functions, all starting w/ Sock…()
- Regina REXX (many platforms) provides an RxSock compatible version matching the IBM Object REXX

Basic Socket Send/Receive Functions

- `SockSocket(domain, type, protocol)` PC
  - Creates a new socket and returns its number
  - Typical parms: 'AF_INET', 'SOCK_STREAM', 'IPPROTO_TCP'
- `SockRecv(socket, buffer, length, flags)` PC
  - Reads data from socket, up to length bytes (may return less)
- `SockSend(socket, dataToSend, flags)` PC
  - Writes dataToSend to socket
- `SockClose(socket)` PC
  - Closes socket (makes it unusable on both sides)
Resolver, Port number

- Socket functions deal with IP addresses, not hostnames
- SockGetHostByName()
  - Converts a hostname to an IP address, by interfacing to the resolver on the local machine
  - Generally, the resolver communicates with one or more DNS servers
  - Use this to accept both IP addresses and/or hostnames
- Multiple servers on a machine are distinguished by port number
  - Client and server MUST agree on the port number
  - Hard-code the port number in your program, or accept it as a parameter in both client and server

Basic Socket Connection Functions

- SockBind(socket, address) PC/UNIX
  Socket('Bind', socket, address) MVS
  - Used by servers to reserve a port
- SockAccept(socket, [address] ) PC/UNIX
  Socket('Accept', socket) MVS
  - Used by servers to wait for a client to make a connection
  - socket must have been previously bound
  - When a connection occurs, a new socket is returned, which is used for further communication with the client (original socket remains open, waiting for new connections)
- SockConnect(socket, address) PC/UNIX
  Socket('Connect', socket) MVS
  - Used by clients to connect to a server
Basic Socket Connection Functions

Client
- SockSocket()
- SockConnect()
- loop ...
  - SockSend()
  - SockRecv()
- SockClose()

Server
- SockSocket()
- SockBind()
- loop ...
  - SockAccept()
  - loop ...
  - SockRecv()
  - SockSend()
- SockClose()

Sample TCP/IP Socket Application
- Echo client and server, written by Thorsten Schaper of IBM
  - Client sends a string to the server
  - Server appends text to the string and sends it back
- Contains both Mainframe and PC/UNIX implementations
- Server can handle multiple clients, Host or workstation
  - EBCDIC <-> ASCII translation is handled
- Available for download at:
Sample TCP/IP Socket Application

```
D:\>rexx serverw.rex
-> trying to bind service to:
   host = 127.0.0.1
   port = 5000
...listening for clients on socket: 300

-> trying to accept a new client
...OK, new client is connected on socket: 308
...clientInfo: WindowsNT COMMAND
D:\download\rexx\rxsocket\clientw.rex
-> got a request from client connected at socket: 308
...received: 'TXTHello, this is a test message'
...answering: 'server has received your text:'TXTHello, this is a test message'
-> got a request from client connected at socket: 308
...received: 'CMDshutDownService'
-> shutting down all sockets...
308: is down.
300: had problems -> RC = -1
-> Socket REXX-API functions NOT dropped to avoid crashing other REXX programs that are using the Socket API at the moment.
good bye

D:\>rexx clientw.rex
...trying to connect to service at:
   host = 127.0.0.1
   port = 5000
--> established on socket: 300
1.) send 'TXTHello World!' text message
2.) enter text for sending a text message
3.) send 'CMDshutDownService' command message
4.) close connection & exit
2
Hello, this is a test message
=====================================...sent text was  : 'TXTHello World!' text message
...server replied  : 'server has received your text
   :CMDshutDownService'
=====================================1.) send 'TXTHello World!' text message
2.) enter text for sending a text message
3.) send 'CMDshutDownService' command message
4.) close connection & exit
3
=====================================...sent text was  : 'CMDshutDownService'
```

xSocket() Internal Function

- Internal REXX function used to map calls to MVS socket() into Windows/Linux/OS/2 Socket API
- Pasted in via rxcopy edit macro
- On MVS, it is a straight pass-thru
- Other platforms, it calls the appropriate socket function, and places the data into MVS format
- Sets variables:
  - xsr = 0 (no error), or mnemonic, e.g. EWOULDDBLOCK
  - xsrmsg = " (no error), or error msg, e.g. Operation would block
Practical Applications from the Field

- Client connection between CA-OPS/MVS (Automated Operations) & BMC Patrol Enterprise Manager (PEM)
  - Client replaced BMC host product but kept PEM
  - Interface enabled MVS REXX code to open alerts on PEM
  - Implemented as an MVS Started Task written in REXX
- Trouble Ticket interface between MVS and Expert Advisor running on Windows
  - Expert Advisor has a command line interface for integration with HP-OpenView and NetView for AIX
  - Installed Denicomp systems rexec daemon
    www.denicomp.com - 800-2424-PSL
  - Implemented rxrexecl() external function, which implements REXEC protocol but stores encrypted password elsewhere

This slide intentionally left blank