VSE E06 - TCP/IP for VSE/ESA
Socket Programming

VM/VSE Technical Conference, Orlando

Ingo Adlung
VSE Development
IBM Lab Böblingen

adlung@de.ibm.com

RETURN TO INDEX

TCP/IP Socket Programming

Contents

VSE/ESA

TCP/IP Socket Programming

Overview

TCP/IP provides different access methods connecting to/from a VSE/ESA host and interchanging data with this system :

- Telnet
  - connect to VTAM applications (inbound)
  - connect to remote hosts : 3270 mode, UNIX, PCs, etc.
- File Transfer Protocol (FTP)
- Web Server
  - static page contents : HTML only
  - dynamic page contents : HTML, Javascript, CGIs, Server Side Includes (SSI), Java applets, etc.
- User written Client/Server applications

TCP/IP Socket Programming

What is a TCP/IP socket ?

A socket programming interface provides the routines required for inter-process communication, either on the local system or spread in a TCP/IP based network.

- Internet address
  - e.g. 9.164.178.140
- Communication protocol
  - User datagram protocol (UDP)
  - transmission control protocol (TCP)
- Port
  - "well known" ports, e.g. port 23 for Telnet
  - user defined ports

TCP/IP Socket Programming

What is a TCP/IP socket ? (cont)

Socket applications were usually C or C++ based, using a variation of the socket API defined by the Berkley System Distribution (BSD). Today Java provides socket classes too.

- UNIX considerations
  Beside TCP/IP sockets, UNIX systems provide socket interfaces for inter-process communication.
- VSE/ESA considerations
  VSE/ESA providesTCP/IP based sockets only. They can be used for inter-process communtication too, although primarily aimed for network communication.

TCP/IP Socket Programming

Socket interfaces in VSE/ESA

- TCP/IP for VSE/ESA
  - Assembler SOCKET macro interface
    - Beside traditional socket applications it also supports to exploit buit-in Telnet, FTP and LPR protocol support. Requires to specify VSE batch or CICS environment.
  - COBOL and PL/I pre-processor interface
    - It needs to be specified if used in batch or CICS
  - REXX
  - C socket call interface
    - dynamically determines program environment

TCP/IP Socket Programming

Socket interfaces in VSE/ESA (cont)

- Language Environment for VSE/ESA
  - LE/VSE 1.4 C socket interface
    - Dynamically determines program environment
    - Industry standard programming interface,compatible with OS/390, VM/ESA C APIs, as well as with non-S/390 operating system environments.

TCP/IP Socket Programming

Portability Aspects

- Assembler
  - Most efficient way of programming.
  - TCP/IP SOCKET macro isn't portable, but VSE specific.
  - VSE/ESA 2.3/2.4 - OS/390, VM/ESA portability when calling LE socket services only.
  - New : VSE/ESA 2.5 - OS/390 source compatibility
    - EZASMI macro interface, and
    - EZASOKET callable interface

TCP/IP Socket Programming

Portability Aspects (cont)

- COBOL and PL/I
  - Dominant programming languages for VSE/ESA.
  - No "standard" for socket programming.
  - Pre-processor API is not portable but VSE specific.
  - OS/390 and VM/ESA portability when calling LE socket services only.
  - New : VSE/ESA 2.5 - OS/390 source compatibility with EZASOCKET callable soutet interface routines.

TCP/IP Socket Programming

Portability Aspects (cont)

- REXX
  - TCP/IP provided SOCKET support
    - similar rich functionality as assembler SOCKET macro
    - VSE proprietary interface
  - VSE/ESA REXX socket support
    - compatibility with OS/390 and VM/ESA REXX socket support

TCP/IP Socket Programming

Portability Aspects (cont)

- C language
  - Dominant TCP/IP programming language.
  - "Standard" for socket programming.
  - TCP/IP C interfaces differ from BSD standard.
  - Best cross-platform portability when calling LE C socket services.

TCP/IP Socket Programming

Portability Aspects (cont)

- VSE Language Environment
  - Beside C, LE provides portability to OS/390 and VM/ESA for Assembler, COBOL, and PL/I too, if exploiting the LE interlanguage communication capabilities.
  - Programs using socket interafaces, written in anything but C and Java are by definition not portable to non-S/390 operating system environments.

TCP/IP Socket Programming

Which API to use ?

Consider :

- Portability
  - Ease of cross-platform development (single source).
- Compatibility
- Serviceability
  - Decoupling the socket application from the TCP/IP product to maintain, service both parts independently.

TCP/IP Socket Programming

Which API to use ? (cont)

- Assembler
  - TCP/IP SOCKET macro most powerful socket interface.
    - example :

```
SOCKET OPEN,TCP,
         ACTIVE=YES,
         FOIP=IPADDR,
         FOPORT=65,
         DESC=SOCKDESC,
         ECB=RESULTS
```

  - Support for applications basing on Telnet, FTP and LPR.
  - Little risk for TCP/IP service level dependencies.

TCP/IP Socket Programming

Which API to use ? (cont)

- COBOL and PL/I
  - Language specific stub routine is link-edited with the application when using the TCP/IP pre-processor API :
    - COBOL - IPNETXCO.OBJ, and PL/I - IPNETXP.OBJ

```
EXEC TCP OPEN FOREIGNPORT(2000)
FOREIGNIP(IPADDRESS)
LOCALPORT(0)
RESULTAREA(RESULTS)
DESCRIPTOR(MY-DESC)
ACTIVE
WAIT(YES)
```

  - Support for applications basing on Telnet, FTP and LPR.
  - Little risk for TCP/IP service level dependencies.

TCP/IP Socket Programming

Which API to use ? (cont)

- REXX
  - easy-to-use VSE Batch script language
  - TCP/IP REXX socket support grants similar functionality as the assembler SOCKET macro
    - enabled with SET REXX_SUPPORT=ON
    - example:

      s=SOCKET('TCP','OPEN',loport,foip,foport,sysid,timeout,async,mode)
  - VSE/ESA REXX socket support grants compatibility to OS/390 and VM/ESA environments
    - example :

      s=Socket('socket')
      call Socket 'Connect',s,'AF_INET' foport foip

TCP/IP Socket Programming

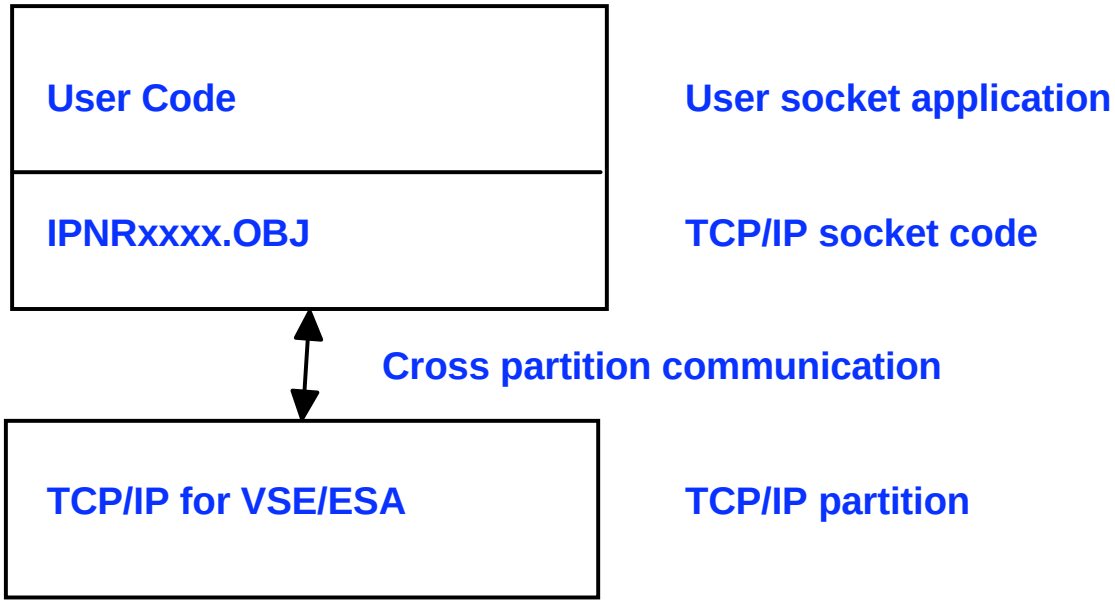
Which API to use ? (cont)

- C language
  - TCP/IP BSD alike interfaces
    - Close to the de-facto BSD standard
    - A stubroutine IPNRxxxx.OBJ matching a specific C socket call is link-edited with the application.
    - Portability with affordable effort possible
    - High risk or TCP/IP service dependencies

TCP/IP Socket Programming

Which API to use ? (cont)

- C language (cont)



TCP/IP Socket Programming

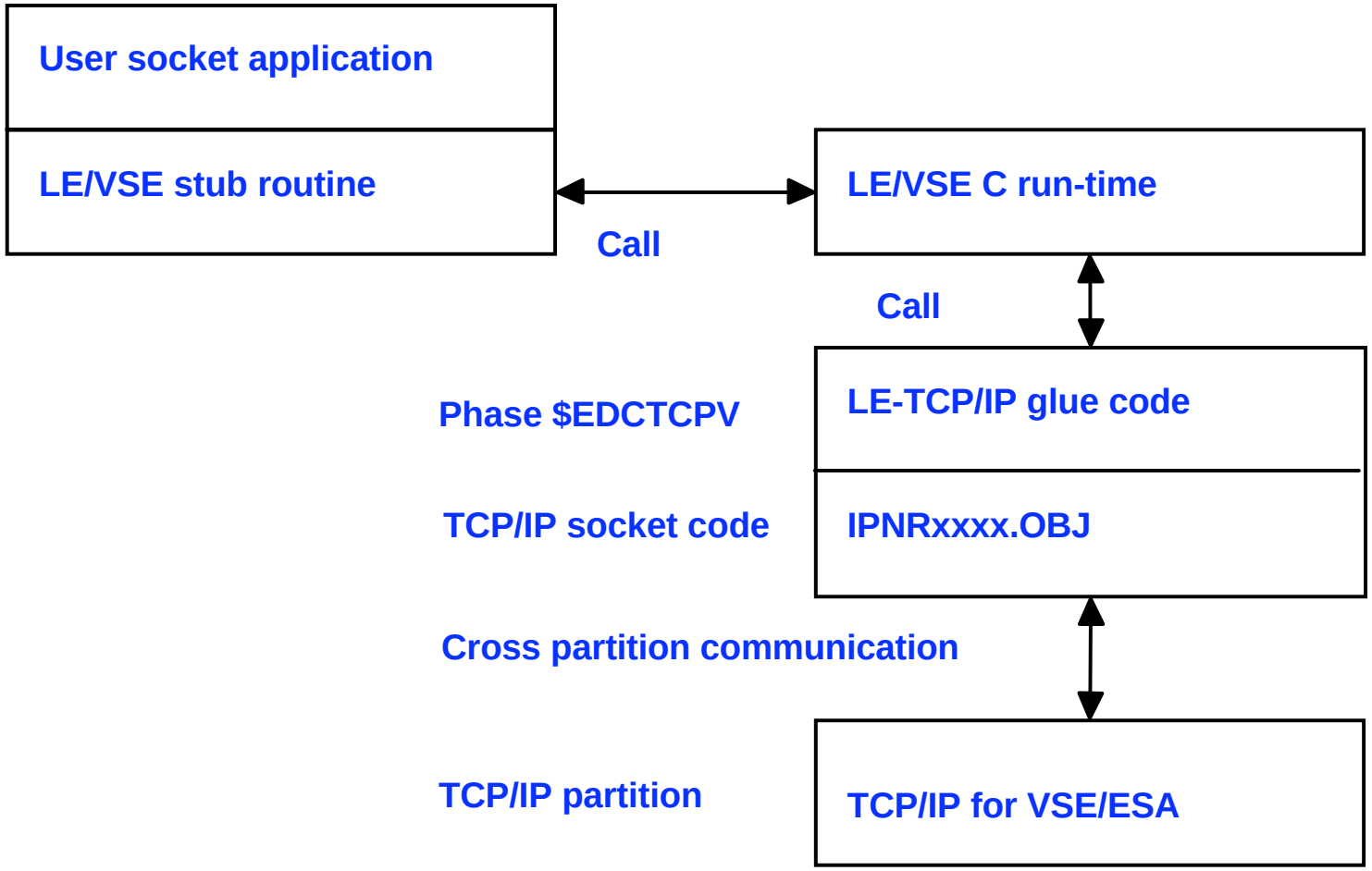
Which API to use ? (cont)

- C language (cont)
  - Language Environment C socket interfaces
    - Following the OpenGroup CAE standard XPG4.2
    - TCP/IP is tied to the LE interfaces by special "glue" code provided with phase $EDCTCPV. Current service level is APAR PQ34615.
    - Portability with little/no effort possible
    - No risk for TCP/IP service dependencies.
    - No risk for LE service dependencies

TCP/IP Socket Programming

Which API to use ? (cont)

- C language (cont)



TCP/IP Socket Programming

Exploiting the LE/VSE socket API

VSE/ESA

IBM

Applications using LE run-time services (C, COBOL,and PL/I) or LE enabled Assembler programs can use the LE C socket routines, either directly or using the LE interlanguage communication (ILC) support.

Remarks :

- C Language
  - LE doesn't implement TCP/IP but bases on the C interfaces TCP/IP provides.
  - TCP/IP for VSE/ESA provides glue code to map the LE calls to native TCP/IP calls.
  - The glue code implements some enhancements, not found in the native TCP/IP C routines.

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

Remarks (cont) :

- COBOL
  - LE/VSE ILC support between C and COBOL is provided for
    - COBOL for VSE
    - VS COBOL II Release 3 and later

- PL/I
  - LE/VSE ILC support between C and PL/I is provided for
    - PL/I for VSE
  - NULL in C is x'00000000' where NULL in PL/I is x'FF000000'. Therefore PL/I programs should check for SYSNULL where appropriate.
  - A character string in C is logically unbound with a x'00' end indicator (last byte).

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- C Language

A simple client :

```
socket()              - create a socket
   |
   v
connect()             - bind and connect to server
   |
   v
send() / recv()       - data interchange
   |
   v
close()               - close communication
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- C Language (cont) - A simple server :

```
s=socket()              - create a socket
    ↓
bind(s)                 - bind socket to port
    ↓
listen(s)               - make it a passive socket
    ↓
c=accept(s)             - wait for incoming requests
    ↓
send(c) / recv(c)       - data interchange
    ↓
close(c)                - close communication
    ↓
close(s)                - destroy listening socket
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- Assembler

  A LE conforming Assembler program calling C

```
GETHOSTN  CEEENTRY PPA=MAINPPA,MAIN=YES
*
          LA     1,PARMSTR
          CALL   GETHNAM
*
          LTR    15,15
          BZ     RETOK
          WTO    'Calling gethostname() failed'
RETOK     DS     0H
          CEETERM
*
CBUFLEN   EQU    20
PARMSTR   DC     A(HNAME)
          DC     F(CBUFLEN)
HNAME     DS     CL(CBUFLEN)
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- Assembler (cont)

  C for VSE routine with OS linkage

```
#include <types.h>
#include <unistd.h>

#pragma linkage( os_gethostname, OS )
#pragma map( os_gethostname, "GETHNAM" )

int os_gethostname( char  *buffer,
                    size_t size)
{
   int ret;

   ret = gethostname( buffer, size)

   return( ret);
}
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- COBOL

COBOL calling C for VSE socket routine

```
* Create a TCP stream socket. The socket value will be
*  returned in variable RVALUE if not -1
*
* Domain type AF_INET is 2
* Socket type SOCK_STREAM is 1
* Protocol IPPROTO_TCP is 6
*
MOVE 2 TO DOMAIN
MOVE 1 TO SOCKTYPE
MOVE 6 TO PROTOCOL

DISPLAY 'Calling C socket() routine'

CALL 'CSOCKET' USING BY CONTENT DOMAIN, SOCKTYPE, PROTOCOL
                     BY REFERENCE RVALUE, RDETAIL
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- COBOL (cont)

  C for VSE socket routine with COBOL linkage

```
#include <types.h>
#include <socket.h>
#include <errno.h>

#pragma linkage( cobol_socket, COBOL )
#pragma map( cobol_socket, "CSOCKET" )

void cobol_socket( int  domain,
                   int  type,
                   int  protocol,
                   int *psocket,
                   int *perr)
{
   *psocket = socket( domain, type, protocol);
   *perr    = errno;
}
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- PL/I

  PL/I calling a C socket routine :

```
DCL GETHNAM EXTERNAL ENTRY
         RETURNS( FIXED BIN(31));
DCL HOSTNAME CHAR(20);
DCL HNSIZE FIXED BIN(31);
DCL CRETURN FIXED BIN(31);

HNSIZE=20;
CRETURN = GETHNAM(ADDR(HOSTNAME),(HNSIZE));
```

TCP/IP Socket Programming

Exploiting the LE/VSE socket API (cont)

- PL/I (cont)

  C for VSE socket routine with PL/I linkage :

```
#include <types.h>
#include <unistd.h>

#pragma linkage( pl1_gethostname, PLI)
#pragma map( pl1_gethostname, "GETHNAM")

int pl1_gethostname( char **buffer,
                     size_t size)
{
   return( gethostname( *buffer, size));
}
```

TCP/IP Socket Programming

What's new for VSE/ESA 2.3/2.4 ?

- TCP/IP for VSE/ESA 1.4 (PQ29053/PQ34615) introduces
  - getclientid(),givesocket(), and takesocket()
    - passing sockets between 2 programs running in the same partition (CICS) or different partitions (batch).
  - asynchronous C socket routines : aio_read(), aio_write(), aio_cancel(), aio_error(), aio_return(), and aio_suspend()
    - allow for asynchronous socket processing based on user ECBs.

TCP/IP Socket Programming

What's new for VSE/ESA 2.5 ?

- New TCP/IP for VSE/ESA 1.4 add-ons :
  - EZASMI Assembler Macro
    - allows for source compatibility with OS/390
    - example :

```
EZASMI TYPE=SOCKET,
       AF='INET',
       SOCKTYPE='STREAM',
       PROTO=PROTOCOL,
       ERRNO=ERRNO,
       RETCODE=RETCODE
```

TCP/IP Socket Programming

What's new for VSE/ESA 2.5 ? (cont)

- EZASOKET callable socket interface
  - allows for source compatibility with OS/390
  - no need to call LE C socket routines (VSE/ESA 2.3/2.4)

```
WORKING STORAGE
    01  SOC-FUNCTION    PIC X(16)  VALUE IS 'SOCKET'.
    01  AF              PIC 9(8) COMP VALUE 2.
    01  SOCTYPE         PIC 9(8) BINARY.
       88  STREAM        VALUE  1.
       88  DATAGRAM      VALUE  2.

    01  PROTO           PIC 9(8) BINARY.
    01  ERRNO           PIC 9(8) BINARY.
    01  RETCODE         PIC S9(8) BINARY.

 PROCEDURE
     CALL 'EZASOKET' USING SOC-FUNCTION AF SOCTYPE
                  PROTO ERRNO RETCODE.
```

TCP/IP Socket Programming

What's new for VSE/ESA 2.5 ? (cont)

- EZASOKET callable socket interface (cont)
  - special interface routines :

    => EZACIC04 : EBCDIC-to-ASCII
    => EZACIC05 : ASCII-to-EBCDIC
    => EZACIC06 : bitstring manipulations for SELECT
    => EZACIC08 : GETHOSTID, GETHOSTNAME, GETHOSTBYNAME

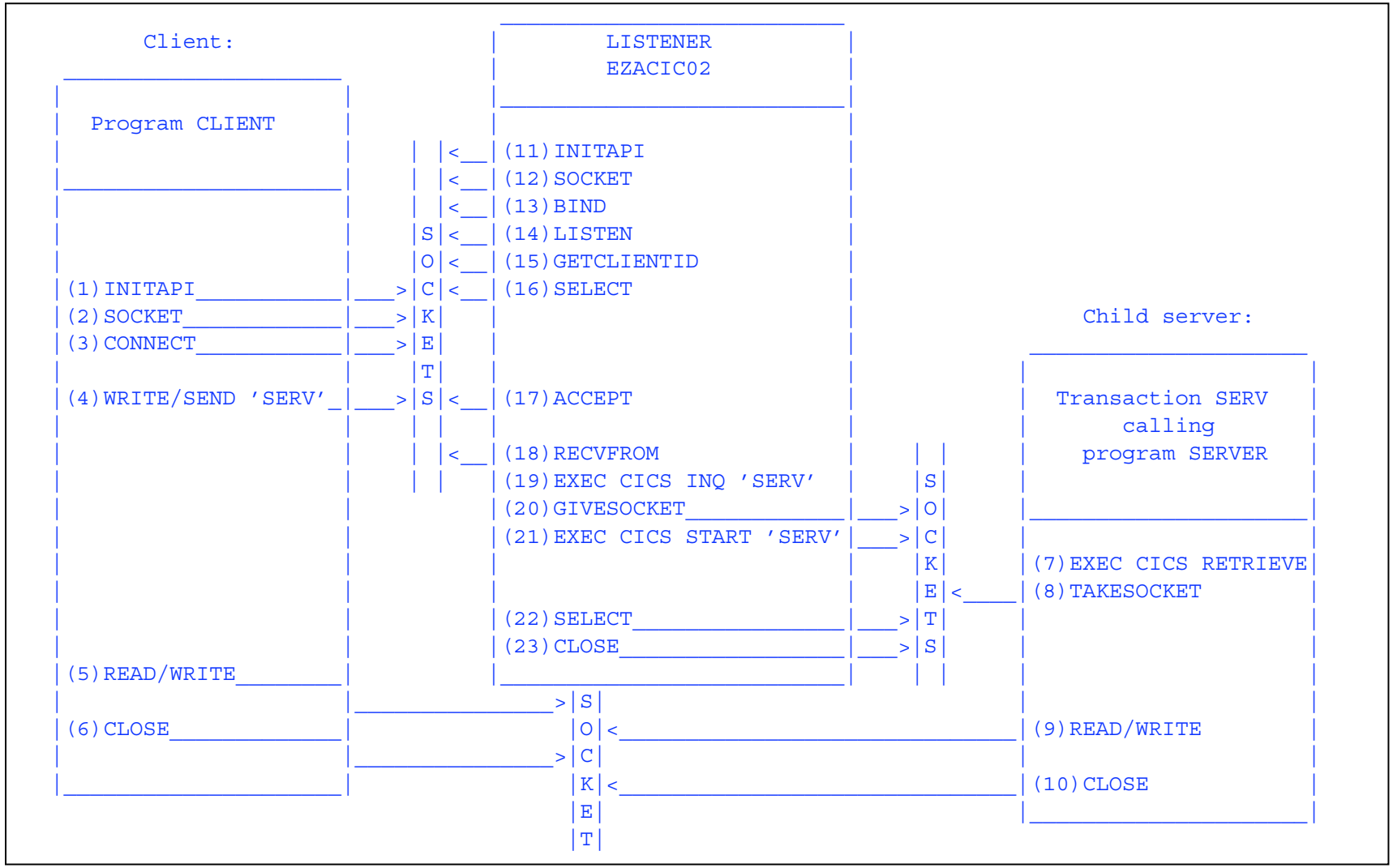
TCP/IP Socket Programming

What's new for VSE/ESA 2.5 ? (cont)

- CICS Listener Transaction
  - provides a generic long running CICS TCP/IP Listener transaction to kick off CICS server transactions on client request.
  - supported with CICS Transaction Server only

TCP/IP Socket Programming

What's new for VSE/ESA 2.5 ? (cont)



TCP/IP Socket Programming

Pitfalls and restrictions

- LE/VSE doesn't allow for multiple LE enabled VSE tasks in 1 partition. It does allow for multiple concurrent LE programs in CICS only.
- Don't attach LE enabled programs to the TCP/IP partition, but run them in their own private partition.
- LE C socket routines restrictions
  - recv() and recvfrom() don't support MSG_PEEK and MSG_OOB options.
  - send() and sendto() don't support MSG_DONTROUTE and MSG_OOB options.

TCP/IP Socket Programming

Pitfalls and restrictions (cont)

- LE C socket routines restrictions (cont)
  - shutdown() doesn't support SHUT_RD and SHUT_WR options.
  - setsockopt() supports SO_KEEPALIVE for source compatibility only. The function itself is covered by the general TCP/IP setting SET PULSE_TIME=nnn.
- Multiple TCP/IP partitions / TCP/IP partition ID is not "00"
  - // OPTION SYSPARM='nn'

    is required to access the proper TCP/IP partition.

TCP/IP Socket Programming

LE/VSE C socket routine messages

- Messages issued by LE/VSE C run-time
  - EDCT001I Unable to load phase $EDCTCPV
    - Application is cancelled with message CEE3322C
  - EDCT002I xxxxxxxx implementation not found
    - Application is cancelled with message CEE3322C
  - EDCT003I Unsupported C run-time function called
    - Application is cancelled with message CEE3322C

TCP/IP Socket Programming

LE/VSE C socket routine messages (cont)

- Messages issued by LE/VSE C run-time (cont)
  - EDCV001I TCP/IP function xxxxxxxx not implemented
    - function is not supported by TCP/IP. A function specific return code is given.
    - LIBDEF search chain of the executing partition has PRD2.SCEEBASE ahead of PRD1.BASE. LE provides a dummy phase $EDCTCPV which does nothing but produce those messages and give function specific return codes.
    - EDCV002I Unexpected TCP/IP return code :nnnn TCP/IP returned an unexpected error code 'nnnn'. Value EOPNOTSUPP is returned to the application.

TCP/IP Socket Programming

Book References

- TCP/IP for VSE Programmer's Reference
- LE/VSE socket interfaces
  - LE/VSE Enhancements (SC33-6778-00)
- TCP/IP for VSE/ESA socket interfaces
  - TCP/IP for VSE/ESA IBM Program Setup and Supplementary Information (SC33-6601-02)
- LE/VSE interlanguage communication
  - Writing Interlanguage Communication Applications (SC33-6686-00)
- new API with VSE/ESA 2.5
  - eNetwork CS IP API Guide (SC31-8516-02)
  - IP CICS Socket Guide (SC31-8516-00)

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets

- Reference list

Function	Description	LE	TCP/IP
accept	accept a new connection	yes	yes
bind	bind a name to a socket	yes	yes
close	close a socket	yes	yes
connect	connect to socket (remote/local)	yes	yes
endhostent	close the host information data set	no	no
endservent	close the network service data set	no	no
fcntl	control characteristics of sockets	yes (1)	no
gethostbyaddr	get a host entry by address	yes	yes (2)

(1) Only options F_GETFL and F_SETFL are supported
(2) Proprierary, non-standard interface implementation

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
gethostbyname	get a host entry by name	yes	yes (2)
gethostent	get the next host entry	no	no
gethostid	get unique identifier of current host	yes	yes
gethostname	get the name of host processor	yes	yes
getnetbyaddr	get a network entry by address	no	no
getnetbyname	get a network entry by name	no	no
getnetent	get the next network entry	no	no
getpeername	get the name of the peer connected to a socket	no	no

(2) Proprierary, non-standard interface implementation

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
getprotobyname	get a protocol entry by name	no	no
getprotobynumber	get a protocol entry by number	no	no
getprotoent	get the next protocol entry	no	no
getservbyname	get a network service entry by name	no	no
getservbyport	get a network service entry by port	no	no
getservent	get the next network service entry	no	no
getsockname	get the name of a socket	yes	yes
getsockopt	get the options associated with a socket	yes (3)	no

(3) Only SO_LINGER data can be retrieved

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
htonl	translate address host to network long	yes	yes
htons	translate address host to network short	yes	yes
inet_addr	translate internet address to network byte order	yes	yes
inet_lnaof	translate a local network address into host byte order	yes	yes
inet_makeaddr	create an internet host address	yes	yes
inet_netof	get the network number from the internet host address	yes	yes
inet_network	get the network number from the decimal host address	yes	yes
inet_ntoa	get the decimal internet host address	yes	yes

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
ioctl	specify the socket operating characteristics	yes (4)	no
listen	prepare for incoming client requests	yes	yes
ntohl	translate long integer into host byte order	yes	yes
ntohs	translate a short integer into host byte order	yes	yes
read	read data from a socket into a buffer	yes	no
readv	read data on a socket and store it into a set of buffers	no	no
recv	receive data on a socket	yes (5)	yes (5)
recvfrom	receive messages on a socket	yes (5)	yes (5)

(4) Only option FIONBIO supported

(5) Functional restrictions apply

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
recvmsg	receive messages on a socket and store it into an array of messages	no	no
select	monitor activity on socket	yes	yes
selectex	monitor activity on sockets	yes	yes (6)
send	send data on a socket	yes (7)	yes (7)
sendmsg	send messages on a socket	no	no
sendto	send data on a socket	yes	yes
sethostent	open the host information data set	no	no
setnetent	open the network information data set	no	no

(6) Proprietary, not conforming to OS/390 implementation

(7) Functional restrictions apply

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
setprotoent	open the protocol information data set	no	no
setservent	open the services information data set	no	no
setsockopt	set options associated with a socket	yes (8)	no
shutdown	shutdown all or part of a duplex connection	yes (9)	yes (9)
socket	create a socket	yes	yes
write	write data from buffer to connected socket	yes	no
writev	write data to a socket from an array of buffers	no	no

(8) Only option SO_LINGER, SO_REUSEADDR, and SO_KEEPALIVE supported

(9) Functional restrictions apply

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
aio_read	async. receive data	yes	yes
aio_write	async. send data	yes	yes
aio_cancel	abort outstanding async. request	yes	yes
aio_return	retrieve the return code associated with an async. request	yes	yes
aio_error	retrieve the error code associated with an async request	yes	yes
aio_suspend	wait for an async. request to complete	yes	yes

TCP/IP Socket Programming

LE/VSE C vs. TCP/IP C sockets (cont)

- Reference list (cont)

Function	Description	LE	TCP/IP
getclientid	get client id info for current process	yes	yes
givesocket	give a socket connection to another process	yes	yes
takesocket	take a socket connection from another process	yes	yes