IBM System z

Technical University 2011

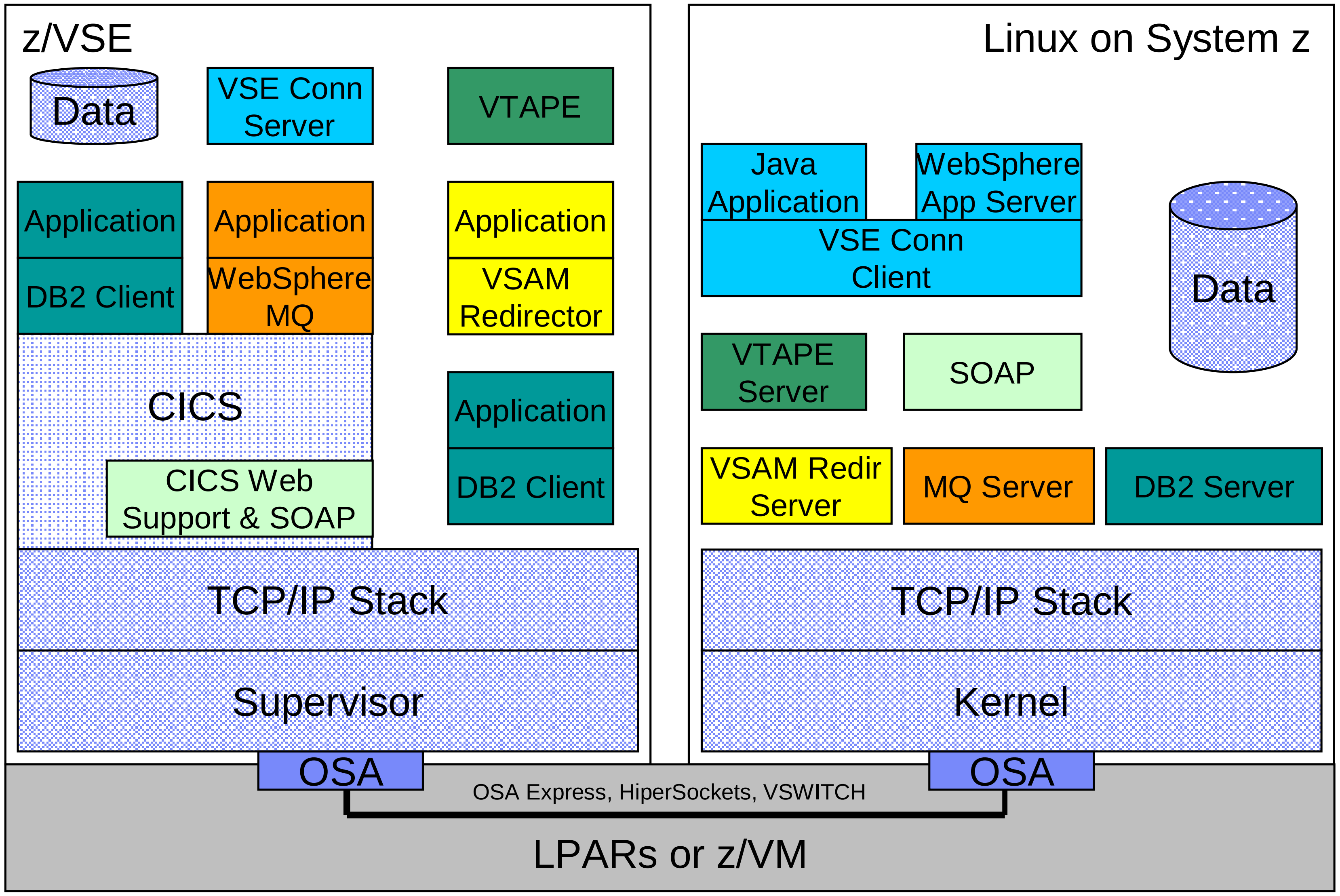
z/VSE Fast Path to Linux on System z

zDL01

Ingo Franzki

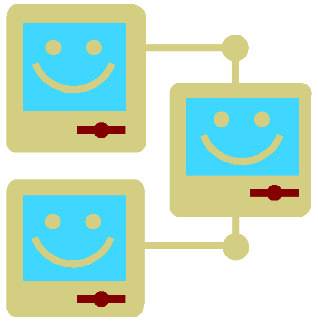
Trademarks

z/VSE Applications communicating with Applications on Linux



z/VSE Applications communicating with Applications on Linux

§ **Communication is mostly based on TCP/IP**
- Although z/VSE and Linux run on the same box



§ **TCP/IP**
- Allow reliable communication over a non-reliable network
- Uses sequence numbers, acknowledges, checksums
  - To protect against packet loss, duplicate packets, packet sequence errors, damaged or incomplete packets, etc.

à **Time consuming processing**

§ **When z/VSE and Linux run side by side on the same box**
- Why do we need all this expensive processing?
- There should be a more direct communication method !

à **z/VSE Fast Path to Linux on System z**

(for short: Linux Fast Path or just LFP)

Fast Path to Linux on System z (LFP)

NEW!

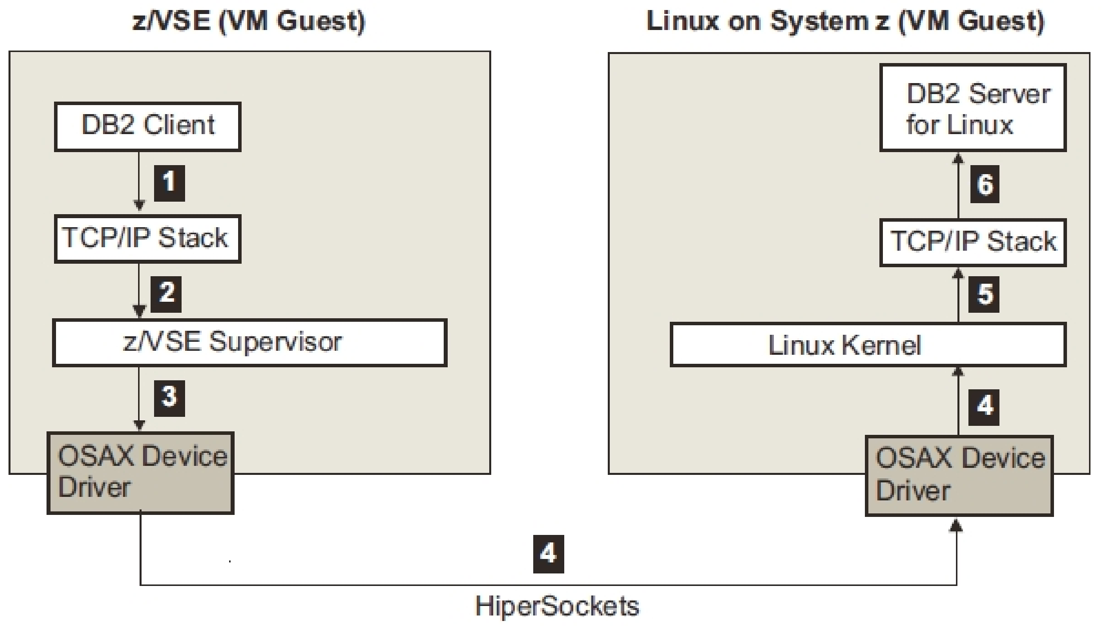
- **The Linux Fast Path uses an IUCV connection between z/VSE and Linux, where both systems run in the same z/VM-mode LPAR on IBM z10, z196 or z114 servers**
- **It allows selected TCP/IP applications to communicate with the TCP/IP stack on Linux without using a TCP/IP stack on z/VSE**

- **All socket requests are transparently forwarded to a Linux on System z system running in the same z/VM**
- **On Linux on System z, the LFP daemon must run**
  - This daemon fulfills all socket requests by forwarding them to the Linux TCP/IP stack

- **The fast path to Linux on System z provides standard TCP/IP socket APIs for programs running on z/VSE:**
  - **LE/C socket API** via an alternative $EDCTCPV.PHASE (IJBLFPLE)
  - **EZA SOCKET and EZASMI** interface via an alternative EZA interface phase IJBLFPEZ
  - CSI's (Connectivity Systems, Incorporated) assembler socket interface via the **SOCKET macro**
  - Other than the basic socket API, no other tools are provided

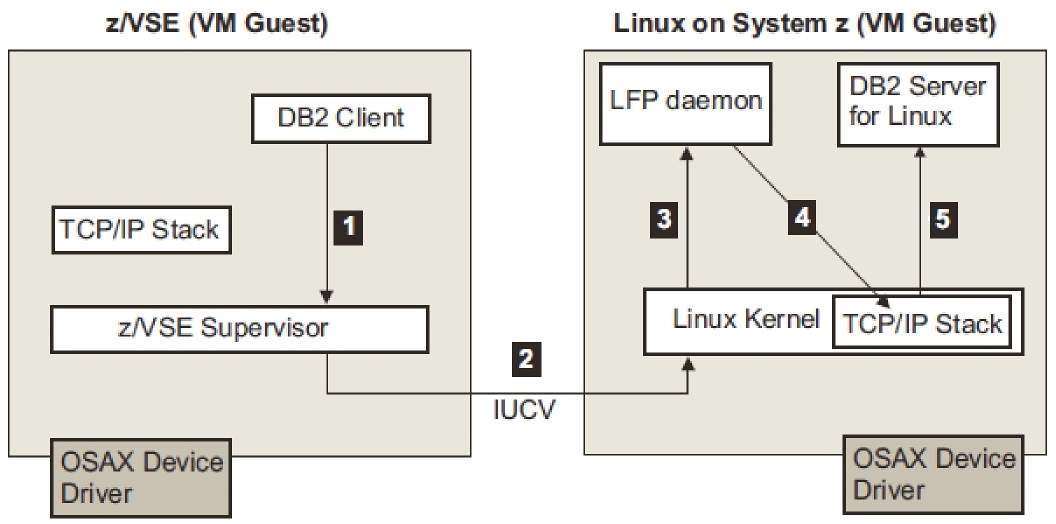
- **Possible performance increase due to:**
  - Less overhead for TCP/IP processing on z/VSE (TCP, sequence numbers and acknowledging, checksums, resends, etc)
  - More reliable communication method (IUCV) compared to HiperSockets, which is a network device, with all its packet drops, resends, etc.

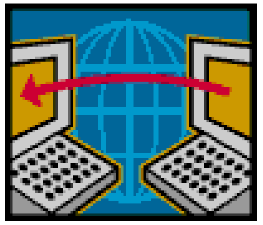
Communication using TCP/IP



1. Data is passed from the application to the TCP/IP stack partition (using cross-partition communication mechanisms, involves dispatching).
2. TCP/IP builds IP packets (including TCP, checksums, sequence numbers, etc) and sends it through the OSAX device driver.
3. The TCP/IP stack passes the packets to the network device driver for use with HiperSockets
4. The HiperSockets network forwards the packets to the Linux image.
5. The Linux HiperSockets device driver receives the packets and passes them to the TCP/IP stack. The TCP/IP stack on Linux checks and unpacks the IP and TCP header. This processing includes handling for retransmissions, sequence numbers and acknowledging, validating checksums and so on.
6. The TCP/IP stack passes the data to the application which processes it.

Communication using Linux Fast Path





1. The data to be sent is passed to the Linux Fast Path (LFP) stack running on z/VSE.
2. The LFP stack builds IUCV packets including the data, and sends the packets via the IUCV channel to the Linux image.
3. The Linux IUCV device driver receives the packets and passes them to the LFP Daemon running on the Linux image. The LFP Daemon then processes the data received from the IUCV channel, and translates it into a socket call.
4. The socket call is processed by the TCP/IP stack. Because the data is to be sent to an application that runs on the same Linux system, the TCP/IP stack simply forwards the data directly to the application (using a Unix pipe, thus no TCP/IP processing required).
5. The application receives the data and processes it.

Performance measurements using Linux Fast Path

Comparison TCP/IP for VSE versus Linux Fast Path:

Workload	TCP/IP for VSE	Linux Fast Path (LFP)	Difference
FTP (BSI FTP server)			
§VSE à Linux (1GB) (NULL file, no I/O)	19 MB/sec 29% CPU (5% App + 24% TCPIP)	72 MB/sec 20% CPU (App)	3.7 times faster 9% less CPU
§Linux à VSE (1GB) (NULL file, no I/O)	21 MB/sec 55% CPU (11% App + 44% TCPIP)	70 MB/sec 20% CPU (App)	3.3 times faster 35% less CPU
Socket Application (running 3 times) §VSE à Linux (100MB) §Linux à VSE (100MB)	4.6 MB/sec (*3 = 13.8 MB/sec) 9.7 MB/sec (*3 = 29.1 MB/sec) 26% CPU (3*1% App + 23% TCP/IP)	14.6 MB/sec (*3 = 43.8 MB/sec) 16.2 MB/sec (*3 = 48.6 MB/sec) 9 % CPU (3*3% App)	3.2 times faster 1,7 times faster 17% less CPU

Environment: IBM System z10 EC (2097-722). TCP/IP connection via shared OSA adapter.

à Significant benefits in transfer rate as well as CPU usage

Prerequisites for using the Linux Fast Path

- **If you use a z/VM-mode LPAR, z/VM 5.4 or later. Otherwise, any z/VM release that is supported by z/VSE**
- **If you use a z/VM-mode LPAR, IBM System z10, z196 or z114. Otherwise, any server supported by z/VSE**
- **z/VSE 4.3 or later**

- **One of these Linux on System z operating systems:**
  - SUSE Linux Enterprise Server 10 Service Pack 3 together with security update kernel 2.6.16.60-0.57.1
  - SUSE Linux Enterprise Server 11 Service Pack 1
  - Red Hat Enterprise Linux 5 Update 5
  - Red Hat Enterprise Linux 6

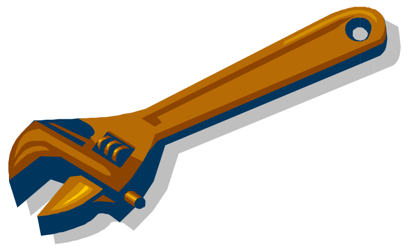
- **z/VSE and Linux on System z are configured as z/VM guests within the same LPAR**

- **The IUCV (“Inter-User Communication Vehicle”) is configured and enabled in both z/VM guests (z/VSE and Linux on System z)**

Preparing to use Linux Fast Path

§ Preparing the LPAR

- For use with LFP, the Linux on System z and z/VSE must run under the same z/VM system
- The use of a z/VM Mode-LPAR is recommended
  - Allows you to mix CPs and IFL in one z/VM Installation
  - Linux runs on IFLs
  - z/VSE runs on CPs
- Change the LPAR Mode to z/VM-Mode and add the IFLs to it



§ Preparing z/VM

- LFP uses IUCV as the underlying communication vehicle. Therefore the z/VSE and the Linux on System z guests on the z/VM system need to be configured for IUCV.
- The following z/VM parameters for the guest systems are relevant:
  - IUCV ALLOW
  - IUCV ANY
  - IUCV MSGLIMIT
  - OPTION MAXCONN *maxno*
- For details about the parameters check the z/VM documentation.

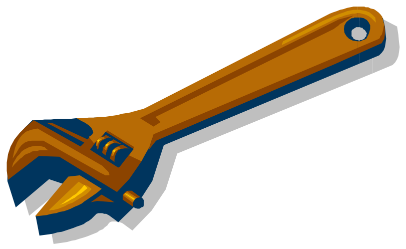
Preparing to use Linux Fast Path

§ Preparing Linux on System z

- **Download and install the LFP Daemon**
  - Part of the “z/VSE Connector Workstation Code“ component 5686-CF8-38 / 02P
  - Member IJBLFPLX.W from PRD2.PROD or download from Internet
  - This ZIP file contains an RPM (RPM Package Manager) that can be used to install the LFPD
- **Configure one or multiple LFPD Instances**
  - Textual configuration files in **`/etc/opt/ibm/vselfpd/confs-available`** and **`/etc/opt/ibm/vselfpd/confs-enabled`**
- It is recommended to use separate (virtual) network adapters or at least separate IP addresses for each LFPD Instance (give each VSE its own IP address)
- Start LFP daemon using lfpd-ctl or automatically at boot via init.d start script

§ Preparing z/VSE

- The LFP code is part of the z/VSE system, **no installation step needed**
- **Start and configure an LFP Instance**
  - Textual configuration statements in LIBR member or SYSIPT of start job
  - LFP Instance operation via IJBLFPOP tool
- **LFP does not require a partition to run**
- Every LFP Instance is identified by a 2 digit number (System ID)
  - Same concept as used by TCP/IP stacks



Sample configuration on z/VSE

```
 * $$ JOB JNM=LFPSTART,CLASS=0,DISP=L
// JOB LFPSTART
// EXEC IJBLFPOP,PARM='START DD:SYSIPT LOGALL'
ID = 01
MTU = 8192
IucvMsgLimit = 1024
InitialBufferSpace = 512K
MaxBufferSpace = 4M
IucvSrcAppName = TESTV
IucvDestAppName = LINR02
IucvDestVMId = LINLFP
WindowSize = 65535
WindowThreshold = 25
/*
/&
* $$ EOJ
```

IJBLFPOP will read input from SYSIPT

IUCV Name of LFP on z/VSE

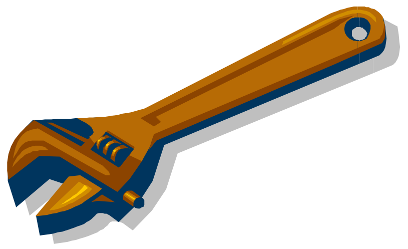
IUCV Name of LFPD on Linux

Guest name where Linux runs

z/VSE Skeletons for use with LFP

§ **The following skeletons are available in ICCF library 59 for use with LFP:**

Skeleton	**Description**
SKLFPSTA	Start an LFP Instance
SKLFPSTO	Stop an LFP Instance
SKLFPLST	List all active LFP Instances
SKLFPINF	Query information about an active LFP Instance
SKLFPACT	Contains control statements to activate LFP you many need to include into the JCL of your applications



Operating an Linux Fast Path on z/VSE

§ **List active LFP Instances**

```
– // EXEC IJBLFPOP,PARM='LIST'
– LFPB025I ACTIVE LFP INSTANCES: 1
                   INSTANCE 01 HAS 3 ACTIVE TASKS
  LFPB026I END OF ACTIVE LFP INSTANCES LIST
```

§ **Display information about an active instance**

```
– // EXEC IJBLFPOP,PARM='INFO <INSTID> [SHOWTASKS] [LOGALL]'
– LFPB023I INFO ABOUT LFP INSTANCE '01':
    *** INSTANCE ***
      STATUS ......................... : UP
      WINDOW SIZE .................... : 65,535
      ...
    *** DEVICE ***
      DEVICE STATUS .................. : ACTIVE
      PACKETS WAITING FOR MSG COMPLETE : 0
      MAXIMUM PACKETS USED ........... : 37
      ...
    *** TASKS ***
      ACTIVE TASK COUNT .............. : 3
     -- TASK #1 –
      TASK ID (PARTITION ID).......... : 2E (Z1)
      SOCKET COUNT ................... : 1
      L2 SOCKET LIST COUNT ........... : 1
      ...
  LFPB024I END OF INFO ABOUT LFP INSTANCE '01'.
```

Operating an Linux Fast Path on z/VSE

§ **Enable or disable socket diagnosis**

–
```
// EXEC IJBLFPOP,PARM='DIAG <INSTID> <ON|OFF>'
```

– If socket diagnosis is ON for an LFP instance, socket statistics are printed to the job listing each time a socket is closed by the application that is using the LE/C socket API:

–
```
LFP Statistics for Socket '0':
  Number of Bytes Sent: 1020364776
  Number of Bytes Received: 943718400
  Number of Packets Sent: 233923
  Number of Packets Received: 183220
  Times in Wait for Send Window: 18158
  Times in Wait for Receive Window: 28002
  Times in Wait for Message Limit: 0
  Times in Wait for Packets: 0
```

Sample configuration on Linux on System z

lfpd-LINR02.conf:

```
 # lfpd configuration file
IUCV_SRC_APPNAME = LINR02
# ensure that only TESTV from VSER05 can connect
PEER_IUCV_VMID = VSER05
PEER_IUCV_APPNAME = TESTV
IUCV_MSGLIMIT = 1024
MTU_SIZE = 8192
MAX_SOCKETS = 1024
INITIAL_IO_BUFS = 128
WINDOW_SIZE = 65535
WINDOW_THRESHOLD = 25
VSE_CODEPAGE = EBCDIC-US
VSE_HOSTID = 10.0.0.1
RESTRICT_TO_HOSTID = yes
LOG_INFO_MSG = no
```

IUCV Name of LFPD on Linux

Guest name where z/VSE runs

IUCV Name of LFP on z/VSE

This is the IP address VSE will appear under

Note: The configuration file must be named “lfpd-XXX”, where XXX is the IUCV_SRC_APPNAME specified in the configuration file !
The XXX characters in the filename must be specified in uppercase !

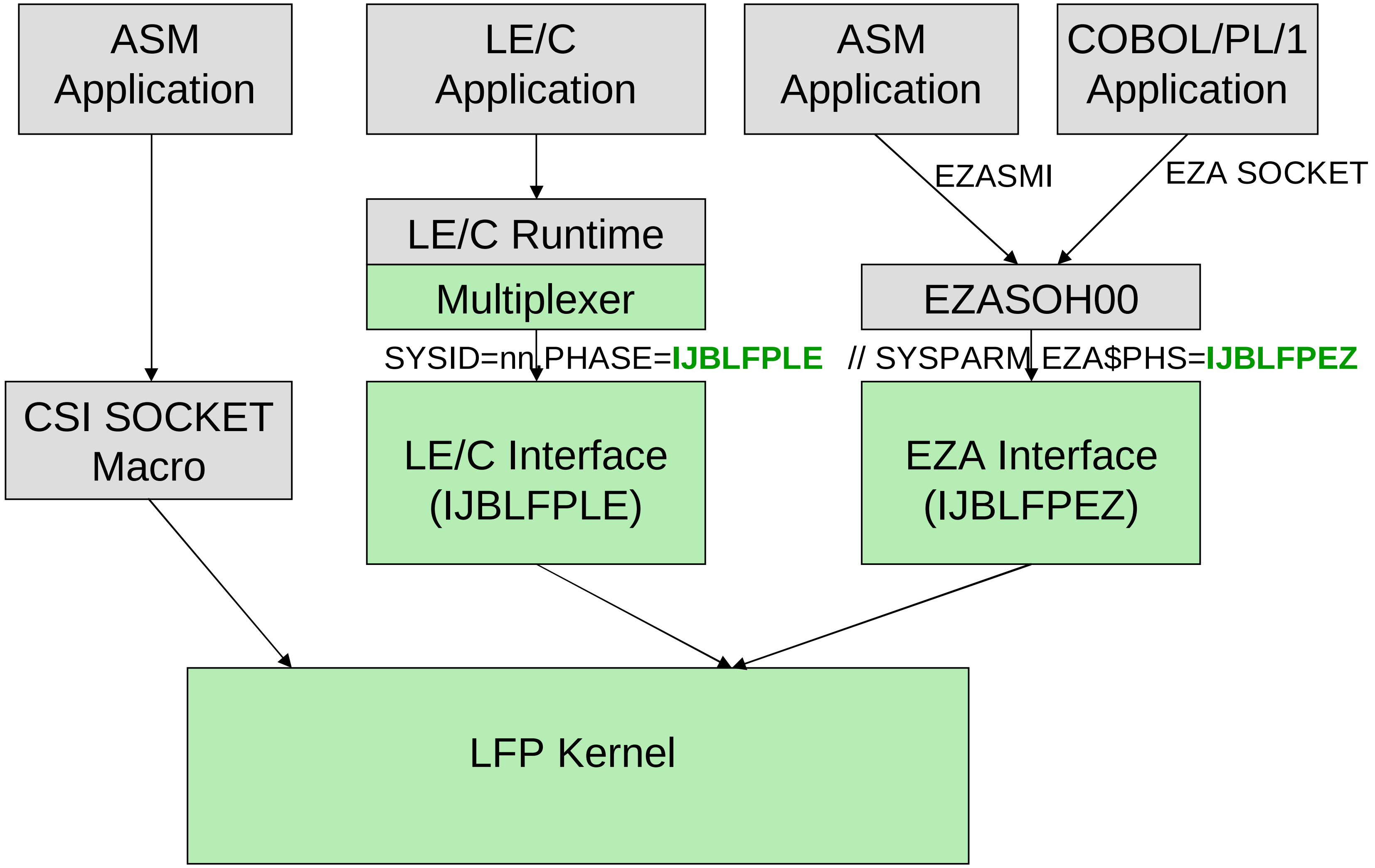
Operating an Linux Fast Path on Linux on System z

§ Display LFP daemon status

– **`lfpd-admin <--iucv_appname|-i appname> <--status|-s>`**

```
Status:
-------
  z/VSE instance is connected.
  Peer VM ID ......... : VSER05
  Peer IUCV Appl. name : TESTV
  Applied host id .... : 10.0.0.1
  Applied host name .. : linlfp
  Allocated I/O buffers ....... : 128
  ...
  Number of active z/VSE tasks : 1
  Number of active sockets : 1
Trace Status:
-------------
  Running in daemon mode
  No trace is running
Configuration:
--------------
  LOCAL_IUCV_APPNAME = LINR02
  PEER_IUCV_VMID = VSER05
  PEER_IUCV_APPNAME = TESTV
  MAX_VSE_TASKS = 512
  MTU_SIZE = 8192
  MAX_SOCKETS = 1024
  INITIAL_IO_BUFS = 128
  WINDOW_SIZE = 65536
  WINDOW_THRESHOLD = 25% (16384 bytes)
  ...
-----------------
```

Socket API Support of Linux Fast Path



Socket API Support of Linux Fast Path

- **Linux Fast Path supports the following Socket APIs**
  - LE/C Socket API
  - EZA SOCKET and EZASMI
  - CSI's SOCKET Macro (limited support)

- **LE/C Socket API considerations**
  - The LE/C interface phase for LFP is shipped as IJBLFPLE.PHASE in IJSYSRS.SYSLIB
  - You must configure the LE/C TCP/IP Socket API Multiplexer to use the LFP LE/C TCP/IP interface phase IJBLFPLE for the IDs of all LFP instances that are running
  - To configure the multiplexer, use skeleton EDCTCPMC in ICCF library 62
  - You can add entries for all your LFP instances with the following statement:
    - EDCTCPME SYSID='01',PHASE='IJBLFPLE'
- **EZA SOCKET and EZASMI considerations**
  - With the EZA SOCKET and EZASMI interfaces you can specify which interface module is to be used
  - For LFP, you must use the EZA interface module IJBLFPEZ
  - You must set the JCL parameter "EZA$PHA" in all your jobs that you want to use LFP
  - To do so use the following statement in your jobs:
    - // SETPARM [SYSTEM] EZA$PHA=IJBLFPEZ
  - If you are using the EZA SOCKET or EZASMI interface under CICS, you need to activate the EZA 'TASK-RELATED-USER-EXIT' (TRUE)

LE/C Socket API Multiplexer

§ **Different Stacks use different Interface routines**

– TCP/IP for VSE (CSI/IBM):	$EDCTCPV
– Linux Fast Path:	IJBLFPLE
– IPv6/VSE (BSI/IBM):	BSTTTCPV

§ **Avoid complicated setup using specific LIBDEFs for different stacks**

§ **Interface phase is selected by System ID**

§ **Use skeleton EDCTCPMC in ICCF library 62   (ß Attention: Typo in documentation)**

```
// EXEC ASMA90,SIZE=(ASMA90,64K),PARM='EXIT(LIBEXIT(EDECKXIT)),SIZE(MAXC
               -200K,ABOVE)'
EDCTCPMC CSECT
EDCTCPMC AMODE ANY
EDCTCPMC RMODE ANY
*
         EDCTCPME SYSID='00',PHASE='$EDCTCPV'
         EDCTCPME SYSID='01',PHASE='IJBLFPLE'
         EDCTCPME SYSID='02',PHASE='BSTTTCPV'
*
         END
/*
```

Specifying the System ID (Instance ID)

- **Using the System ID, you specify which Stack or LFP Instance an application will use**
- **The following table shows how to specify instance IDs and where they can be applied**
  - The settings are checked from top to bottom as listed in the table

	LE/C Socket API	**EZA SOCKET and EZASMI APIs**	**CSI SOCKET Macro**
'LFP$ID' (environment variable)	X		
// SETPARM [SYSTEM] LFP$ID=NN	X	X	
'SYSID' (environment variable)	X		
IDENT.TCPNAME passed to INITAPI call		X	
ID parameter on SOCKET macro			X
// OPTION SYSPARM='NN'	X	X	X
Default '00'	X	X	X

CICS task isolation options

- **LFP isolates CICS tasks from each other**
  - This means that sockets that are allocated by one CICS task, can **not** be used by another CICS task
    - except the socket is passed to the other CICS task via GIVESOCKET/TAKESOCKET calls
  - When a CICS task ends, all sockets allocated by this task will be closed (terminated) automatically
    - Except it has been given to another task prior to task termination

- **Some programs rely on passing sockets from one CICS task to another without the use of GIVESOCKET/TAKESOCKET**
  - For example, DB2 (client or server) application requestor, requires socket sharing, if running under CICS
- **To allow such programs to work with LFP, you need to specify the following JCL statement for the program:**
  - **// SETPARM [SYSTEM] LFP$CIC=SHARE**
  - This setting applies to the LE/C socket interface as well as the EZA interfaces

- **If socket sharing is active, the applications are responsible to close sockets that are no longer needed**
  - No automatic cleanup will be performed at end of CICS task
  - If the applications miss to do proper cleanup, dead sockets may be left over

CSI SOCKET macro considerations

- **For the CSI SOCKET macro, the Linux Fast Path only supports the following connection types:**
  - TCP
  - UDP
  - CONTROL
  - Other connection types (such as CLIENT, TELNET, FTP, RAW, and so on) are not supported and will be rejected if used with the Linux Fast Path.

- **For CONTROL type connections, the only commands supported are:**
  - GETHOSTBYNAME
  - GETHOSTBYADDR
  - GETHOSTNAME
  - GETHOSTID
  - For details, refer to the individual macro descriptions in the "TCP/IP for VSE V1R5F Programmers Guide" manual.

- **For CONTROL type connections, these commands (from Barnard Software, Incorporated) are also supported:**
  - NTOP
  - PTON
  - GETVENDORINFO
  - For details, refer to the “IPv6/VSE Programming Guide” manual

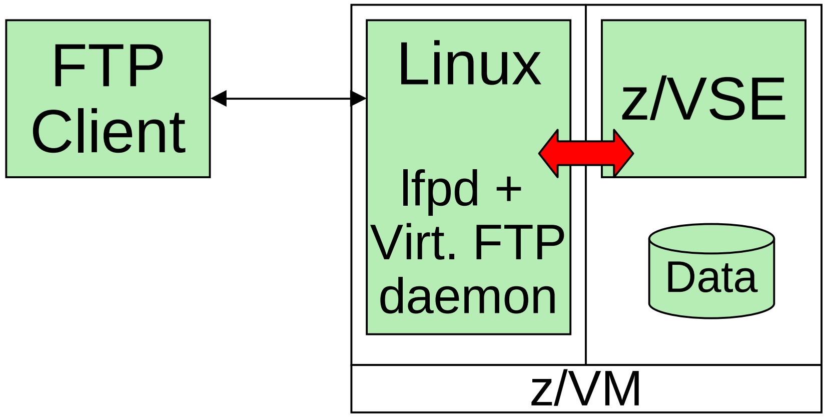
Using existing Applications with Linux Fast Path

- **Most existing applications run unchanged with Linux Fast Path**
  - Provided they use one of the supported Socket API (LE/C, EZA or ASM SOCKET)
    - And they do not use any CSI specific interface, features or functions
- **IBM Applications supporting Linux Fast Path**
  - VSE Connector Server
  - CICS Web Support
  - VSE Web Services (SOAP) support (client and server)
  - CICS Listener
  - DB2/VSE Server and Client
  - WebSphere MQ Server and Client
  - VSAM Redirector
  - VSE VTAPE
  - VSE LDAP Support
  - VSE Script Client
  - POWER PNET
  - TCP/IP-TOOLS included in IPv6/VSE product (e.g. FTP Server/Client)
- **Customer applications should run unchanged:**
  - Provided they use one of the supported Socket API (LE/C, EZA or ASM SOCKET)

New Tool: Virtual z/VSE FTP Daemon

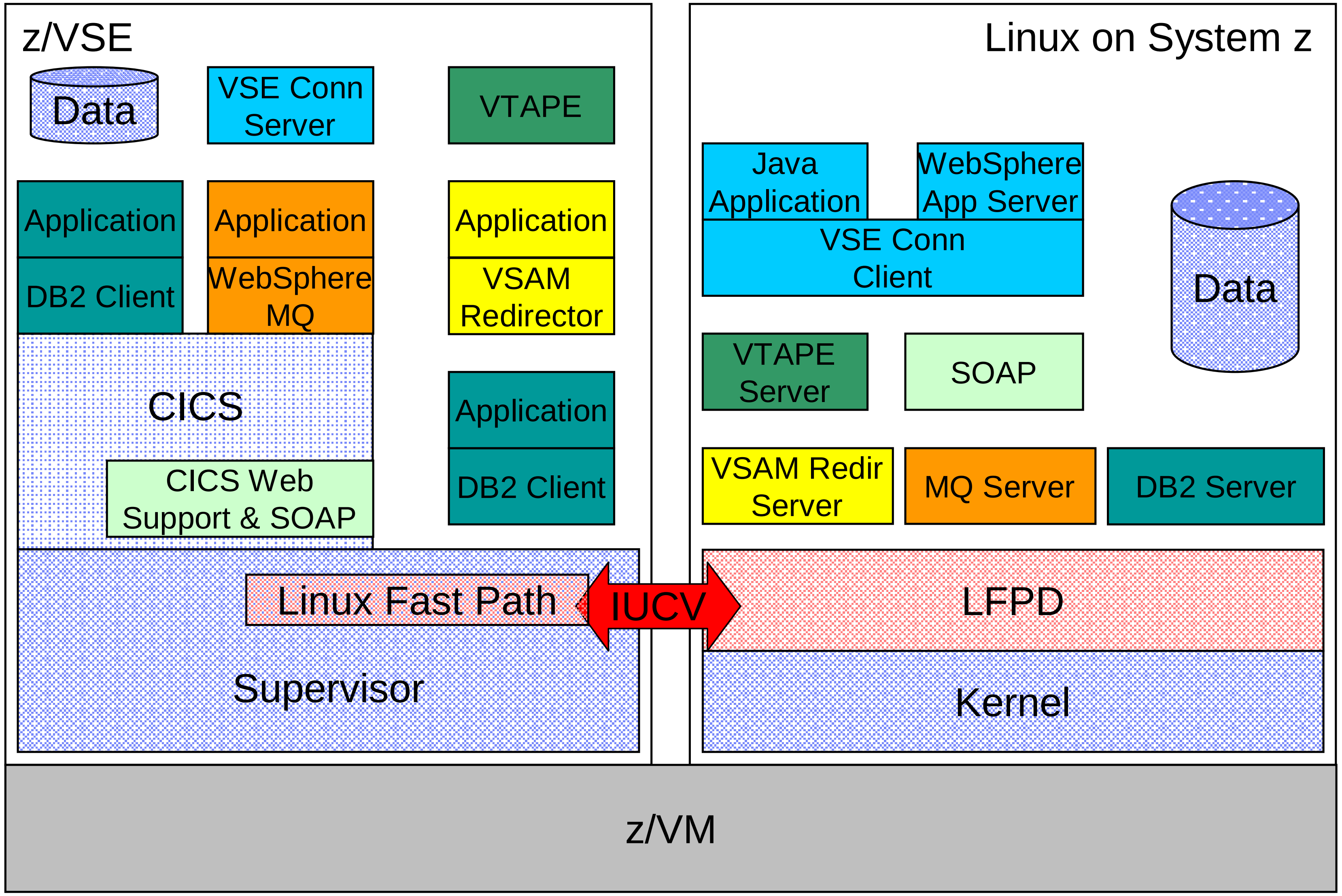
- **The Virtual z/VSE FTP Daemon can be installed on any Java-enabled platform and emulates an FTP server**
  - The actual access to z/VSE resources is done using the VSE Connector Server.
- **Download:** http://ibm.com/zvse/download

à Fits perfectly to Linux Fast Path



- **The Virtual z/VSE FTP Daemon:**
  - Handles all incoming FTP clients.
  - Connects to one or multiple VSE Connector Servers.
  - Is responsible for connection-handling.
  - Is responsible for data translation (ASCII-EBCDIC).
  - Is IPv6 ready
    - You can connect FTP clients using IPv6, the Virtual z/VSE FTP Daemon connects to the VSE Connector Server using IPv4.
  - Supports SSL
    - both for the FTP connection (between FTP client and Virtual z/VSE FTP Daemon, using implicit SSL (FTPS)),
    - and for the connection to the VSE Connector Server (between Virtual z/VSE FTP Daemon and z/VSE host).

z/VSE Applications communicating with Applications on Linux



Questions ?

?