Socket Programming 101 – IPv6 enabling socket applications

Ingo Franzki

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

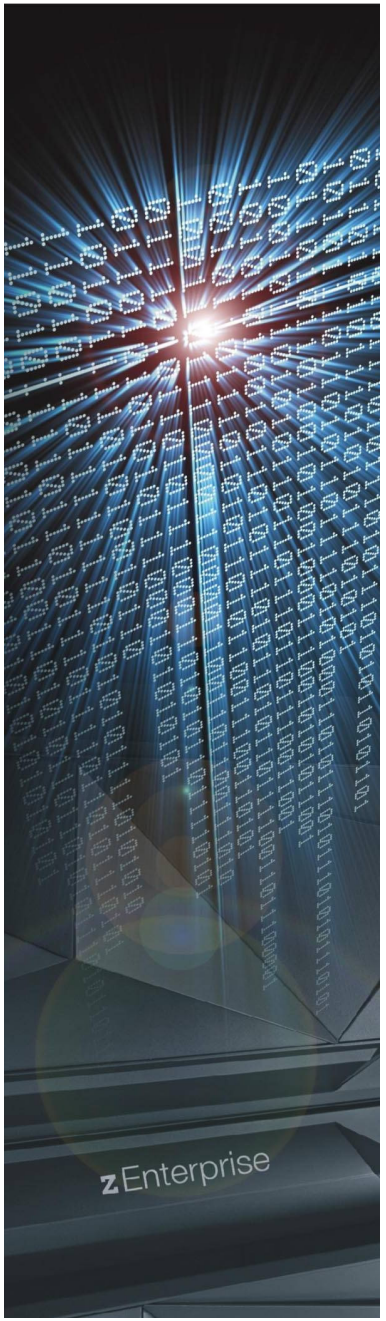
Notes:
Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

- Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at http://www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”).
- No other workload processing is authorized for execution on an SE.
- IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Agenda

- **IPv4 and IPv6 basics, differences and overview**
- **Available TCP/IP Stacks in z/VSE**
- **Available Socket APIs in z/VSE**
  - – Vendor independent Socket APIs
- **BSD style socket programming**
  - – Simple Client & Server example
- **IPv6 related changes in the Socket APIs**
  - – New address families
  - – New control blocks and structures
  - – New socket calls
- **Best practices for adding IPv6 support to applications**
  - – Vendor independent implementation
  - – IPv6 addresses in user interfaces
- **Detecting which TCP/IP Stack you are running with**
- **Do's and Don't Do's**



IPv4 Basics

- IPv4 addresses are 32-Bit (4 Bytes) in length
  - Theoretically up to 4.294.967.296 unique addresses
  - IPv4 addresses are usually written in dot-decimal notation, which consists of the four octets of the address expressed in decimal and separated by periods:
    - Example: 207.142.131.235
    - Each block is 8 bits (1 byte). That is, the value range for each block is 0 to 255.

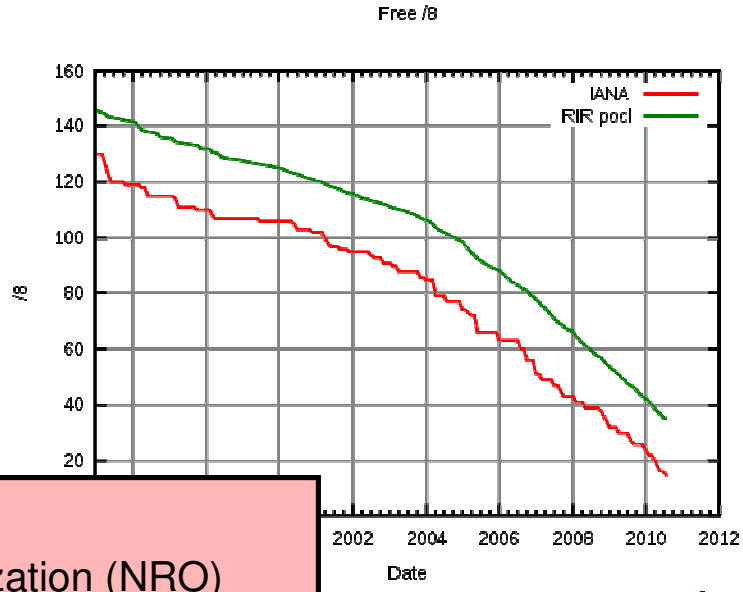
- In order to facilitate routing a data packet across multiple networks, the address is divided into two parts:
  - Network prefix: A contiguous group of high-order bits that are common among all hosts within a network.
  - Host identifier: The remaining low-order bits of the address that are not designated in the network prefix. This part specifies a particular device in the local network.
  - All endpoints which have the same network prefix are in the same IP network.
    - This implied that the endpoints can communicate directly with each others (e.g. through a switch, a hub or a crosslink cable), without the need to use a router.
  - To communicate between different networks, a router is required.

- In IPv4, subnet masks consist of 32 bits, usually a sequence of ones (1) followed by a block of 0s. The last block of zeros (0) designate that part as being the host identifier.
  - Example: 255.255.255.0

	Dot-decimal notation	Binary form
IP address	192.168.5.130	11000000.10101000.00000101.10000010
Subnet Mask	255.255.255.0	11111111.11111111.11111111.00000000
Network Portion	192.168.5.0	11000000.10101000.00000101.00000000
Host Portion	0.0.0.130	00000000.00000000.00000000.10000010

What's the problem with IPv4?

- The depletion of the IPv4 allocation pool has been a concern since the 1980s when the Internet started to experience dramatic growth
  - IPv4 only provided for approximately 4 billion addresses, a limit that is estimated to be reached before 2012

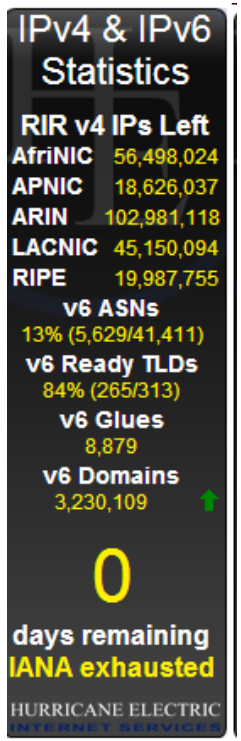


Reasons for t

- Unforeseen (?
  - Example:
- Every mobile
- Always-on cor
addresses sta
- Almost every
- Inefficient add
assigned clas
  - Universities and governmental organizations in the US hold about 74 % of the worldwide assigned IPv4 addresses.
  - Example: Genuity is a IP network provider in the US. They have reserved 3 class A networks. That is about 48 million addresses. However, China has only about 20 million addresses, which is not even half of what Genuity uses.

On 3 February 2011, the Number Resource Organization (NRO) announced that the free **pool of available IPv4 addresses is now fully depleted.** The Internet Assigned Numbers Authority (IANA) allocated the last two blocks of IPv4 address space to a Regional Internet Registry.

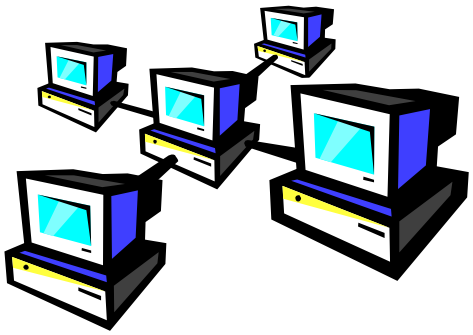
This means that there are **no longer any IPv4 addresses available** for allocation from the IANA to the five Regional Internet Registries.



Other disadvantages of IPv4

- Network Address Translation (NAT)
  - Allows to remap multiple internal IP addresses to one external IP address
    - Every DSL router does that
  - But:
    - Quite complicated and processing time consuming
    - Only a limited number of connections possible at a time (limited due to the size of the NAT table)
    - Only connections from internal networks to external networks (outbound) are possible. For inbound connections a service like DynDNS is required.

- IPv4 contains many redundant and inefficient features:
  - Variable Length IP Header Options
    - Very processing time consuming for routers
  - IP Header Checksum
    - TCP Header also contains a checksum
  - Fragmentation of IP packets
    - Very processing time consuming and inefficient
  - Classification of IP packets
    - Still exists, but is not used anymore



The solution: IPv6

- **Wait a second, what about IPv5 ?**
  - Yes, it did exist as a test protocol only
  - Not used anymore

- **Design goals of IPv6**
  - Avoid errors made when designing IPv4
  - Much larger address range (16 bytes instead of only 4 bytes)
  - Increase scalability
    - ‘Jumbograms’ (up to 4 GB-1 per packet)
  - More efficient processing
    - Fixed Length IPv6 Header
    - No fragmentation
    - No Checksums
    - IPv6 header fields are aligned at 64 bit boundaries
  - Easily extendable
  - Simpler routing
  - ‘True’ multicasting
  - Auto configuration
    - Neighbor Discovery
    - Router solicitation
  - Support for mobile devices

IPv6 Basics

- IPv6 Addresses
  - 128 Bits in length (16 bytes)
    - 4 times larger than a IPv4 address
  - Up to 2^{128} (about 3.4×10^{38}) unique addresses
    - That's approximately 5×10^{28} (roughly 2^{95}) addresses for each of the roughly 6.8 billion (6.8×10^{9}) people alive in 2010.
    - In another perspective, this is the same number of IP addresses per person as the number of atoms in a metric ton of carbon!
  - IPv6 address are usually written as eight groups of four hexadecimal digits (each group representing 16 bits, or two bytes), where each group is separated by a colon (:).
    - Example: 2001:0db8:85a3:08d3:1319:8a2e:0370:7344
  - Leading zeroes in a group may be omitted (but at least one digit per group must be left):
    - 2001:0db8:0000:08d3:0000:8a2e:0070:7344 is the same as 2001:db8:0:8d3:0:8a2e:70:7344
  - A string of consecutive all-zero groups may be replaced by two colons. In order to avoid ambiguity, this simplification may only be applied once:
    - 2001:db8:0:0:0:0:1428:57ab is the same as 2001:db8::1428:57ab

IPv6 Basics - addressing

- IPv6 Addresses gets assigned to interfaces (network adapters)
- One interface (network adapter) can have multiple IPv6 addresses
  - Assigned address
  - Link local address
- Every IPv6 address has a "scope“:
  - Link local
  - Site local
  - Global

- IPv6 addresses are typically composed of two logical parts:
  - Routing prefix
    - The length of the prefix is specified with the address separated by a slash: /64
  - Interface identifier
    - Usually automatically determined from the MAC address of the interface
  - Internet service providers (ISPs) usually get assigned the first 32 bits (or less) as their network from a regional internet registry (RIR)

IPv6 Basics – address types

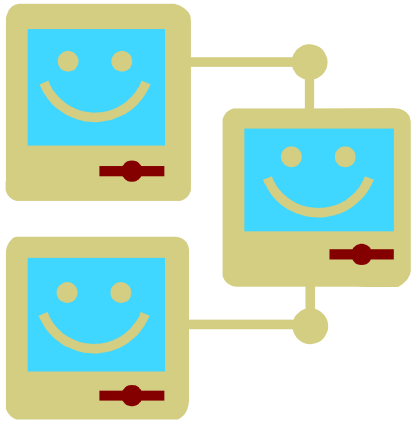
- IPv6 address types:
  - ::1/128 Is the loopback IPv6 address
  - ::/128 Is an unspecified IPv6 address
  - FF00::/8 Is a multicast IPv6 address
  - FE80::/10 Is a link local IPv6 address
  - FEC0::/10 Is a site local IPv6 address
  - FC00::/7 Is a unique local IPv6 address (private address)
  - All others are global unicast IPv6 addresses

- Interfaces (network adapters) have at least 2 IPv6 addresses:
  - Assigned (global) IPv6 address
    - 806::1:2
  - Link local IPv6 address
    - FE80 + Mac Address (020000000008)
    - FE80:0:0:0:0200:0000:0100:0008
    - FE80::200:0:100:8

- Further address types:
  - Site local IPv6 address (not used anymore)
  - Multicast IPv6 address

IPv6 Basics – auto configuration

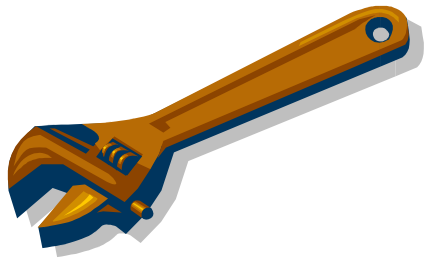
Goal: Plug 'n' Play network



- An IPv6 endpoint needs at least 3 pieces of information to be able to communicate:
  - IPv6 address
  - IPv6 network
  - IPv6 gateway
- Right after the start, an endpoint only knows its link local address
  - E.g. determined from the MAC address of the interface
  - With that, it can only communicate within its local network segment
- The interface then uses Neighbor Discovery Protocols to search for routes in its local network segment
  - It sends requests to the multicast address FF02::2, which all routes are reachable at (Router Solicitation)
  - Available routes then reply with information about the network
- Router also send Router Advertisements in regular intervals to all hosts in the network(s) segment they are responsible for
- ICMPv6 provides essential functions in an IPv6 network
  - Address Resolution Protocol (ARP) is replaced by Neighbor Discovery Protocol (NDP)

Migration from IPv4 to IPv6

- Contrary to popular belief, IPv6 is not backward compatible !
- But: IPv4 and IPv6 networks can be used concurrently over the same cable and with the same endpoint



Transition methods:

- **Dual IP Stacks**
  - That's the easies possibility
  - The IP stack supports both protocols concurrently
    - Examples: Linux since Kernel 2.6, Windows since XP SP1
  - Existing IPv4 applications can continue to run unchanged
    - Applications can be IPv6-enabled over time, one after the other
- **Tunneling**
  - IPv6 packets are sent as payload of other protocols (usually IPv4) to a tunneling broker, which is located in an IPv6 network. The broker extracts the IPv6 packet from the payload and sends it as IPv6 packet through IPv6 routing to the final destination.
    - Example: 6in4 using Tunneling-Broker

Why should a z/VSE customer care about IPv6?

Independent on your concrete benefits

→You will have to care about IPv6, sooner or later!

Why?

- Your internet service provider (ISP) migrates to IPv6
- On 3 February 2011, the Number Resource Organization (NRO) announced that the free pool of available IPv4 addresses is now fully depleted.
- Your customers or partners are only reachable via IPv6 (e.g. China)
- Governmental organizations may only allow manufacturers of IPv6 capable products and applications to participate in advertised biddings
  - Example: The US Department of Defense (DoD) only allows products that are on the "Unified Capabilities Approved Products List" (UC APL) for its advertised biddings.
    - "This list is used by procurement offices in the DoD and the U.S. Federal agencies for ongoing purchases and acquisitions of IT equipment"

Migration from IPv4 to IPv6

Which infrastructure parts needs to be migrated?

- **Layer 1 devices (e.g. hubs)**
  - Those are completely transparent for IPv6
- **Layer 2 devices (switches)**
  - Devices which have been purchased within the last 10 years most likely support IPv6 already
- **Layer 3 devices (routers)**
  - Usually not required for local LANs
  - Today most router manufacturer provide IPv6 capable routers
  - Routers that use Multiprotocol Label Switching (MPLS) are protocol independent
- **Endpoints (PCs, Server, etc.)**
  - Most modern operating systems support IPv6
- **Applications**
  - May have to be adapted (IPv6-enabled) to be able to work with IPv6 addresses

Available TCP/IP Stacks in z/VSE

- **TCP/IP for VSE/ESA 1.5(F)**
  - IPv4 only
  - IPv6 support pending
  - Standard applications (IPv4 capable)

- **IPv6/VSE V1.1**
  - The IPv6/VSE product contains 2 TCP/IP stacks
  - IPv6 Stack
    - Provides support for the IPv6 protocol
    - IPv6 application programming interfaces (APIs)
    - IPv6-enabled applications
    - Supports IPv6 only, no IPv4
  - IPv4 Stack
    - Provides support for the IPv4 protocol
    - IPv4 application programming interfaces (APIs)
    - IPv4-enabled applications
    - Supports IPv4 only, no IPv6
  - Dual stack support through coupling of both stacks

- **Fast Path to Linux on System z (LFP)**
  - No full ‚Stack', provides Socket APIs only
  - IPv4 and IPv6 (since zVSE 5.1) support (dual stack)
  - Available in z/VM and LPAR (since z/VSE 5.1) environment
  - No applications provided

CSI International

BSI BARNARD SOFTWARE, INC.

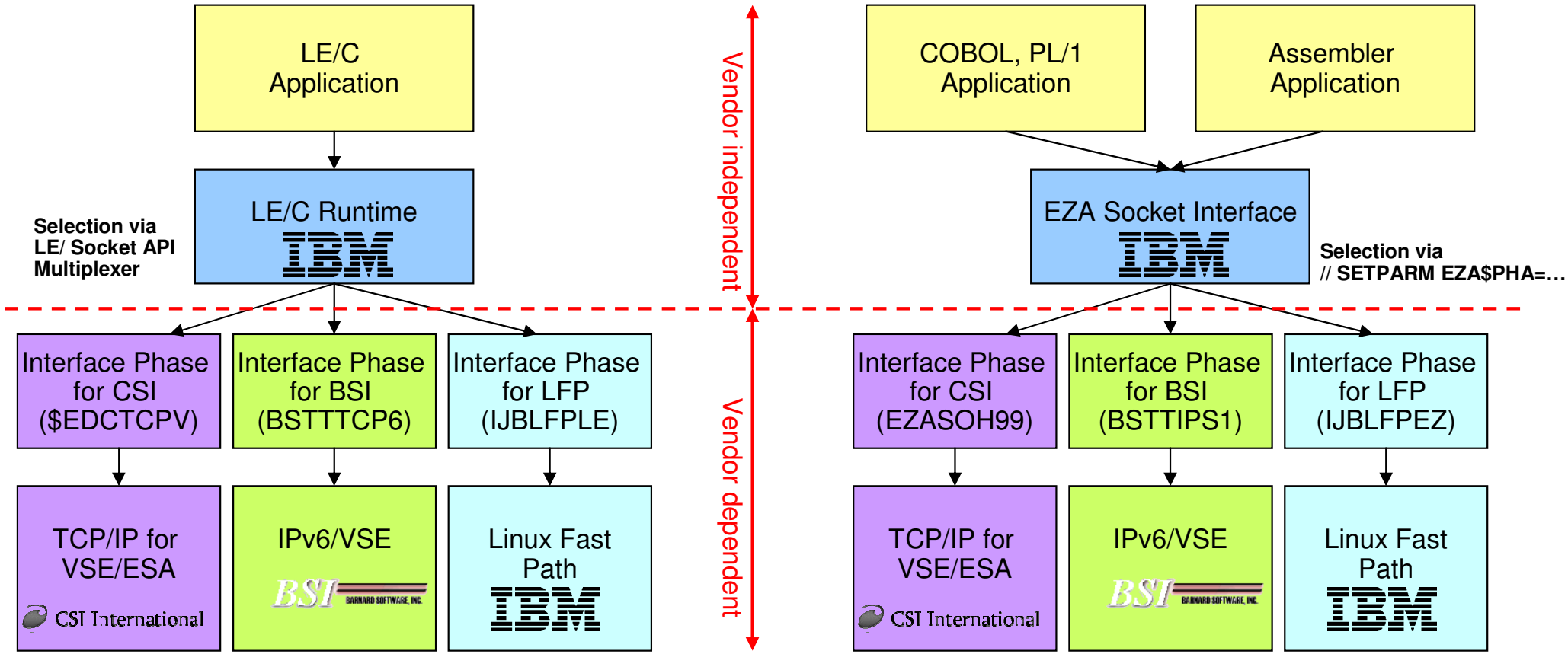
IPv6/VSE is a registered trademark of Barnard Software, Inc

Available Socket APIs

API	Programming languages	Standard / Proprietary	SSL Support	IPv6 Support	Stack support
LE/C Socket API	C	Standard BSD Sockets, mostly compatible with z/OS and other distributed systems (e.g. Unix, Linux, Windows, Mac, …)	Yes	since z/VSE 5.1, IPv6/VSE and LFP only	TCP/IP for VSE/ESA, IPv6/VSE, LFP
EZA APIs - **EZASMI** - **EZASOCKET**	 Assembler COBOL, PL/1, Assembler	Standard BSD Sockets, mostly compatible with z/OS	Yes	since z/VSE 4.2 with IPv6/VSE, since z/VSE 5.1 with LFP, too	TCP/IP for VSE/ESA, IPv6/VSE, LFP
CSI SOCKET Macro	Assembler	CSI proprietary	No	No (IPv6/VSE and LFP support IPv6 extensions)	TCP/IP for VSE/ESA, BSI*, LFP* *) Not all features supported
CSI Preprocessor API (EXEC TCP)	COBOL, PL/1, Assembler	CSI proprietary	No	No	TCP/IP for VSE/ESA (BSI provides conversion tool for EXEC TCP to EZA for COBOL programs)
CSI BSD Socket API	C, Assembler (COBOL, PL/1)	BSD-like socket API with limited standard compatibility CSI proprietary	Yes	No	TCP/IP for VSE/ESA
IBM REXX Sockets	REXX	Standard BSD Sockets	Yes	No	TCP/IP for VSE/ESA, BSI, LFP (internally uses LE/C Sockets)
CSI REXX Sockets	REXX	CSI proprietary	No	No	TCP/IP for VSE/ESA

TCP/IP Stack independent Socket APIs

- **TCP/IP Stack independent Socket APIs typically consists of 2 parts**
  - A generic part (provided by IBM as part of z/VSE)
  - A vendor specific part (provided by the vendor or IBM)



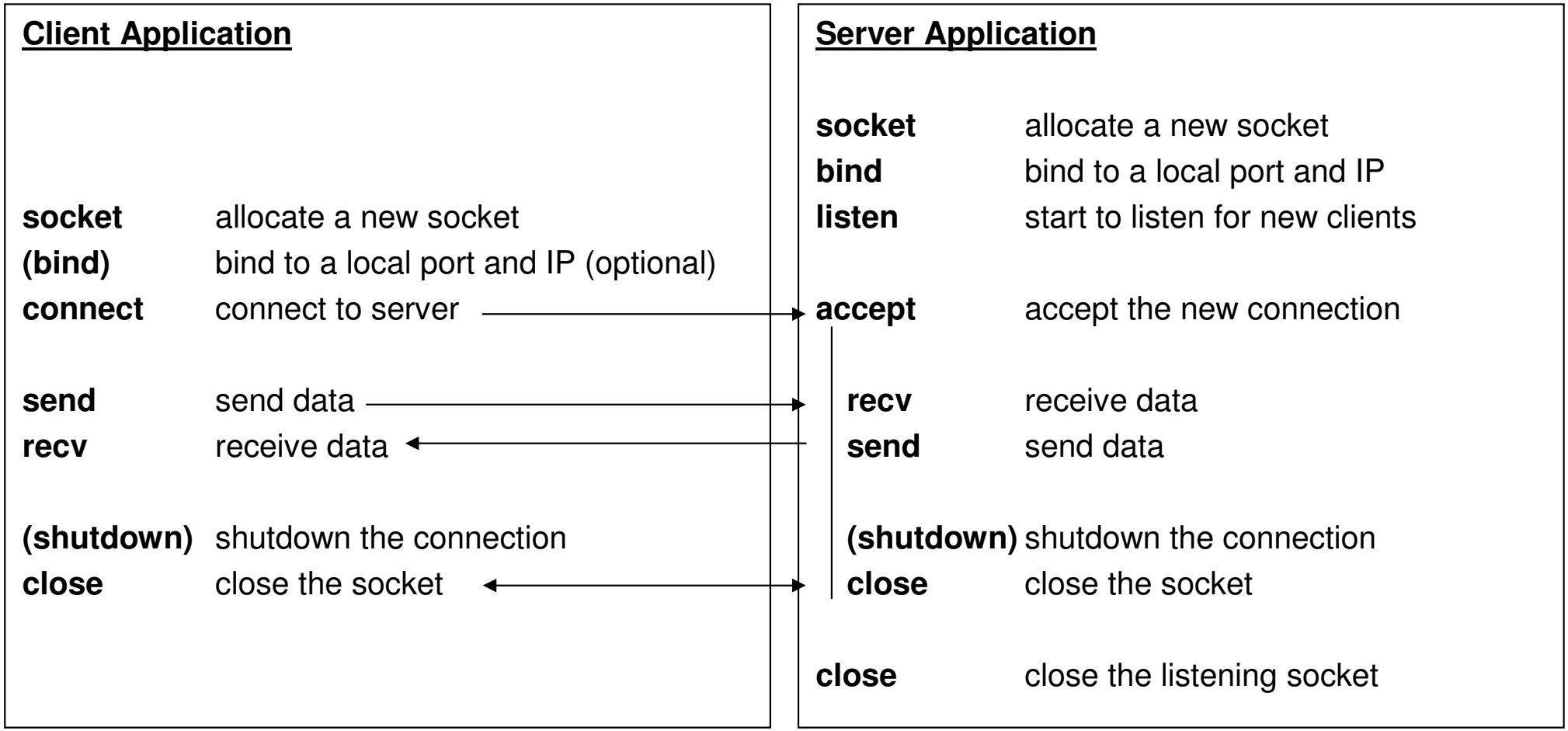
BSD Sockets

- From Wikipedia (http://en.wikipedia.org/wiki/Berkeley_sockets):
  - Berkeley sockets (or BSD sockets) is a computing library with an application programming interface (API) for internet sockets and Unix domain sockets, used for inter-process communication (IPC)
  - As the API has evolved with little modification from a de facto standard into part of the POSIX specification, POSIX sockets are basically Berkeley sockets

- Most operating systems provide BSD Sockets
  - Unix, Linux, (including Android), AIX, HP-UX, …
  - Windows
  - Mac, iOS
  - z/OS
  - z/VM
  - z/VSE
  - z/TPF
  - System I
  - ...

- Programming languages supporting BSD style sockets
  - The BSD sockets API is written in the programming language C
  - Most other programming languages provide similar interfaces, typically written as a wrapper library based on the C API

Simple BSD Sockets Application



- Additional socket calls are available for waiting on sockets (**select**) and ECBs (**selectex**)
- Sockets can be turned into non-blocking mode via **setsockopt** or **ioctl**
- Domain name resolution via **gethostbyname** socket call

Terms: Socket, Address Family, Socket Address

- **Socket:**
  - A network socket is an endpoint of an inter-process communication flow across a computer network.
  - A socket usually contains or consist of a unique combination of a local address and a local port
- **Socket Address Structure (sockaddr):**
  - A socket address is the combination of an IP address and a port number
  - Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread
- **Address family (AF):**
  - The address family is a naming scheme and format of addresses used for computers on the network

Protocol type	Symbolic constant	Socket Address Structure	Note
IPv4	AF_INET	sockaddr_in	Used on Internet with the IPv4 addresses
IPv6	AF_INET6	sockaddr_in6	Used on Internet with the IPv6 addresses
UNIX Sockets	AF_UNIX, AF_LOCAL	sockaddr_un	For local interprocess communication. Not supported in z/VSE

Changes in the LE/C and EZA Socket API to support IPv6

- **The following socket calls now support AF_INET6 (in addition to AF_INET):**
  - socket - allocate an IPv6 socket
  - getclientid - get the callers ID (needed for givesocket/takesocket)
  - givesocket - give a socket to another thread, task or process
  - takesocket - take a socket from another thread, task or process
  - gethostbyaddr - get domain and alias names of an address

- **The following functions now also accept sockaddr_in6 structure:**
  - bind - bind a socket to a local address (IP and port)
  - connect - establish a connection to a foreign address
  - getsockname - get the socket's local address
  - getpeername - get the address of the communication partner
  - accept - accept a new client and get its foreign address
  - sendto - send a message to a foreign address (typically used for UDP)
  - recvfrom - receive a message and get the senders address (typically used for UDP)

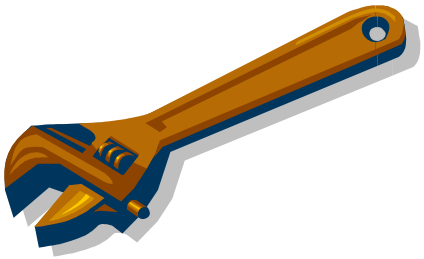
- **The following new functions have been added to support IPv6:**
  - getaddrinfo - translate a domain name and/or service name into a socket address (replaces gethostbyname)
  - getnameinfo - translate a socket address into a domain name and/or service name (replace gethostbyaddr)
  - freeaddrinfo - frees information returned by getaddrinfo
  - inet_ntop - convert a binary address into its textual form (replaces inet_ntoa)
  - inet_pton - convert a textual IP address (IPv4 or IPv6) into a binary address (replaces inet_addr

Adding IPv6 support to socket applications

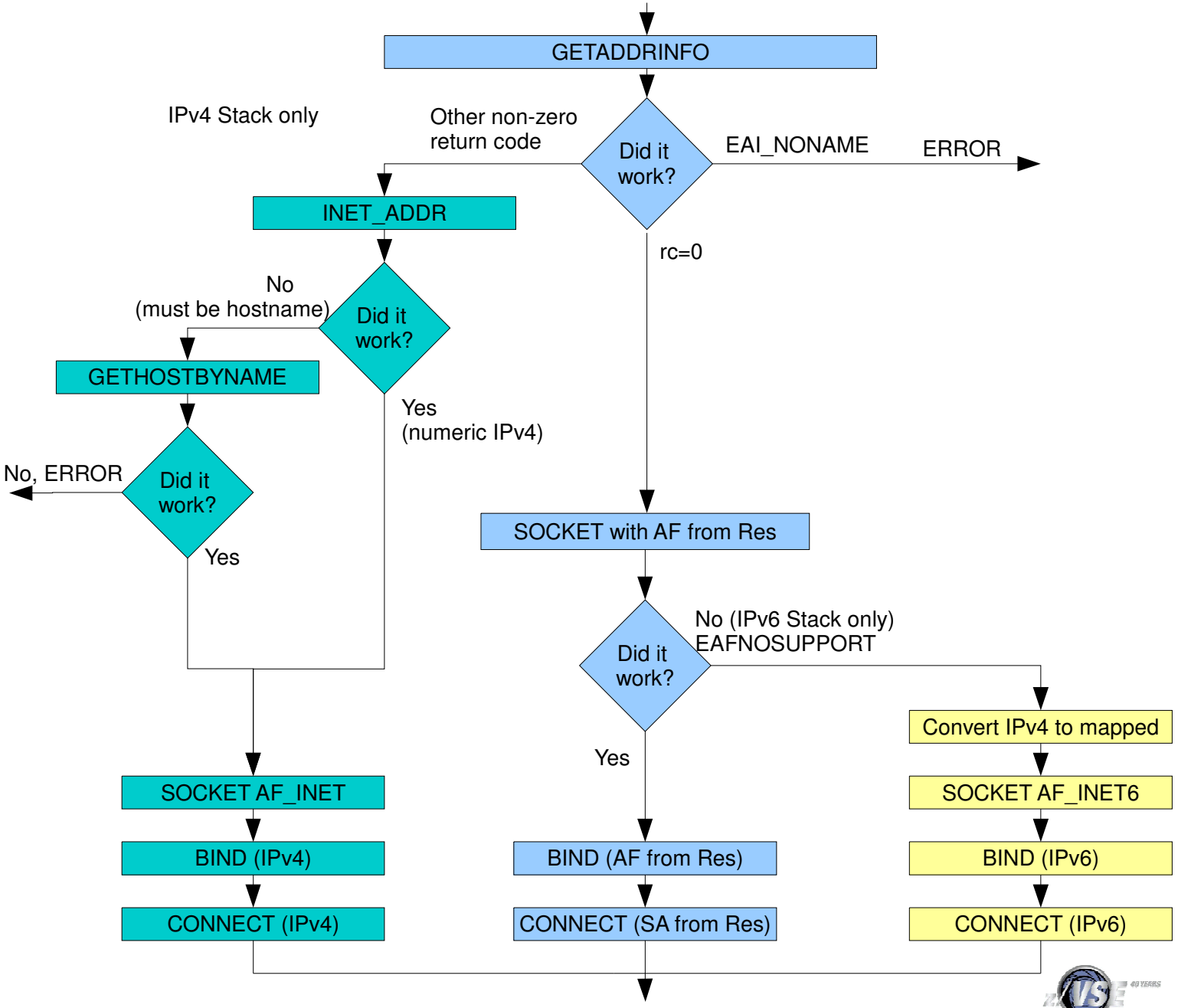
- **Prerequisite:**
  - Your application uses a Socket API that supports IPv6:
    - LE/C or EZA
    - or CSI SOCKET Macro with BSI or LFP

- **How to make you application work with any of the 3 stacks**
  - **Either:** Detect the TCP/IP stack vendor and version you are working with
  - **Or:** Implement it in a way that works with all stacks

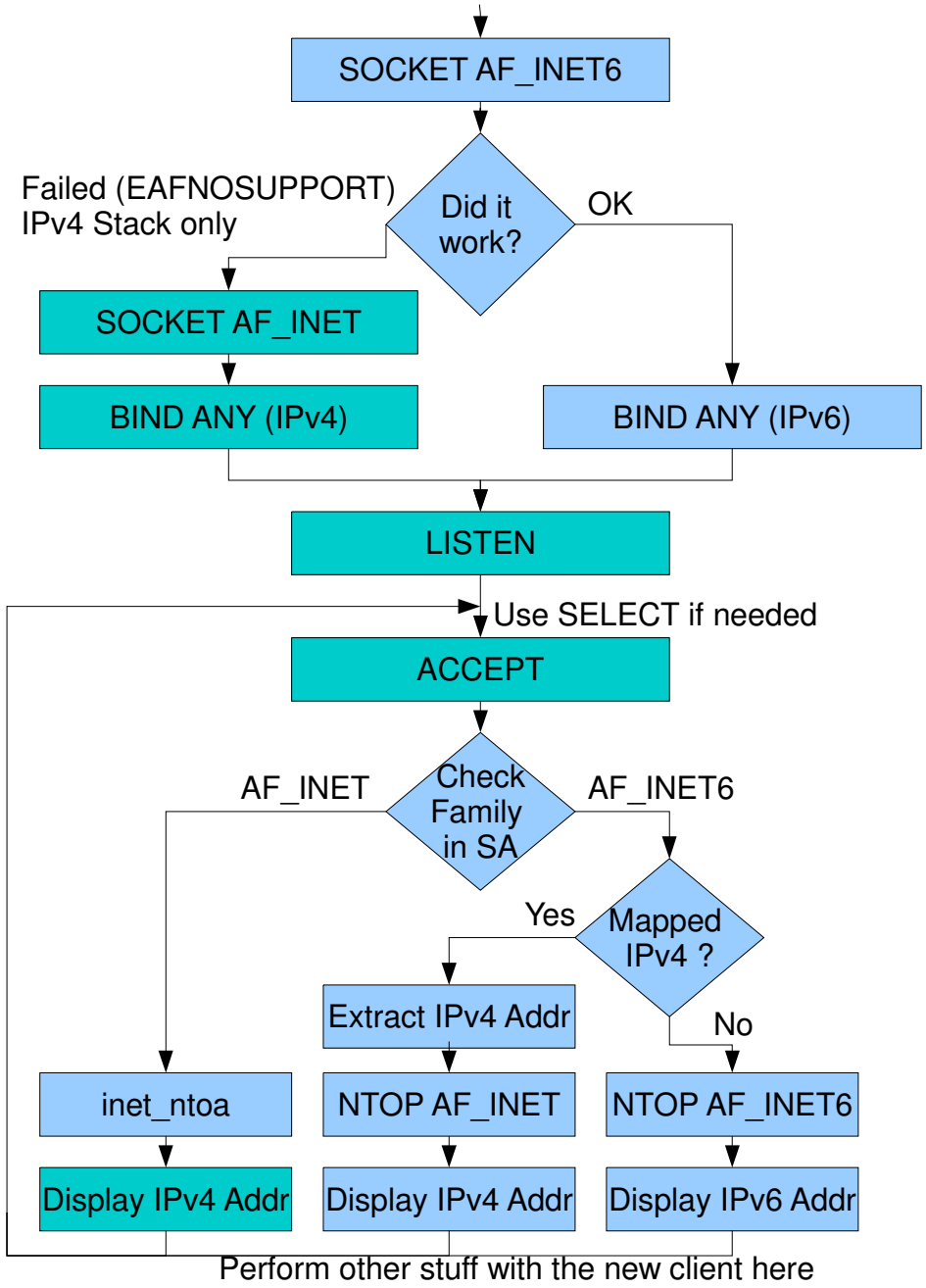
- **Possible situations:**
  - **IPv4-only stack** (e.g. CSI stack, BSI IPv4 stack, LFP on z/VSE 4.3)
    - Supports AF_INET only, no IPv6 addresses accepted
    - New IPv6 related socket calls may not be available
  - **IPv6-only stack** (e.g. BSI IPv6 stack)
    - Supports AF_INET6 only, no IPv4 addresses accepted
    - You can connect to IPv6 endpoints only
    - Supports the new IPv6 related socket calls
  - **Dual stack** (e.g. BSI IPv6 stack coupled with BSI IPv4 stack, LFP since z/VSE 5.1)
    - Supports both AF_INET and AF_INET6
    - Supports the new IPv6 related socket calls



Adding transparent IPv6 support to client applications



Adding transparent IPv6 support to server applications



IPv6 Extensions to CSI SOCKET Macro

- **Although the CSI SOCKET Macro was designed with IPv4 (i.e. 4 byte IPv4 addresses) in mind, it can also be used with IPv6**
- **IPv6 Extensions are currently available with**
  - IPv6/VSE
  - Fast Path to Linux on System z (since z/VSE 5.1)

- **When using the SOCKET macro with IPv4, the 4 byte IPv4 address fits into a single fullword of storage or register**
  - As parameter FOIP= to OPEN, SEND, RECEIVE
  - As result in SRBLOK structure (SRFOIP)
- **When using the SOCKET macro with IPv6 the single fullword or register contains the 31 bit address of a 30 byte IPv6 Socket Address Structure instead**
  - The Socket Address Structure has the same format as the sockaddr_in or sockaddr_in6 structures
- **IPv6-X Flag**
  - The IPv6-X Flag indicates to the SOCKET macro that IPv6 Extensions are to be used
  - The flag is located in the first byte of the SRBLOK
    - The first field in an SRBLOK is a z/VSE ECB field
    - The SRBLOK is specified by the ECB= parameter in the SOCKET macro
  - Setting the first byte of the SRBLOK to “6” (x'F6') during the OPEN call indicates that
    - the IPv6 Extensions are being used
    - AND that IPv6 extensions will continue to be used for this socket.
  - It is not necessary to set the IPv6-X flag on any other type of ASM SOCKET request

Detect which TCP/IP Stack you are running with

- **LE/C and EZA Socket Interfaces**
  - getibmopt socket call may be supported
    - If call fails (bad return code) → assume CSI
    - TCP-IMAGE-NAME is 'SOCKETnn'
    - TCP-IMAGE-STATUS:
      - X'80vv': Active
      - X'40vv': Terminating
      - X'20vv': Down
      - X'10vv': Stopped or stopping

```
03 NUM-IMAGES                PIC 9(8) COMP.
  03 TCP-IMAGE OCCURS 8 TIMES.
    05 TCP-IMAGE-STATUS       PIC 9(4) BINARY.
    05 TCP-IMAGE-VERSION      PIC 9(4) BINARY.
    05 TCP-IMAGE-NAME         PIC X(8)
```

```
vv:   00: CSI
      01: BSI IPv4
      02: BSI IPv6
      03: IBM LFP
```

- **CSI SOCKET Macro**
  - Use CONTROL connection command "GETVENDORINFO"
    - If command fails (bad return code) → assume CSI
    - Response "BSIIPv4" → BSI IPv4
    - Response "BSIIPv6" → BSI IPv6
    - Response "IBMLFP4" or "IBMLFP6" → IBM LFP (IPv4 only, or IPv4 + IPv6)

Best practices for IPv6 addresses in user interfaces

- **A target address can be specified as follows:**
  - Domain name (host name) e.g. "myhost.mydomain.com"
  - Dotted IPv4 address e.g. "10.4.7.23"
  - Numeric IPv6 address e.g. "2001:0db8:85a3:08d3:1319:8a2e:0370:7344"

- **If a port number is required as well:**
  - Use a separate field to specify the port number
  - Allow to append the port number to the address separated by colon:
    - myhost.mydomain.com:4711
    - 10.4.7.23:4711
    - 2001:0db8:85a3:08d3:1319:8a2e:0370:7344:4711
      → Can not distinguish between colon inside IPv6 address and the colon that separates the port number!

- **RFC 2723 recommends to enclose the numeric IPv6 address in square brackets if a port number is specified:**
  - [2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:4711
  - http://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:4711/index.html

- **Attention: square brackets are codepage dependent characters**
  - Consider to use angle brackets (<…>) instead, or a separate field

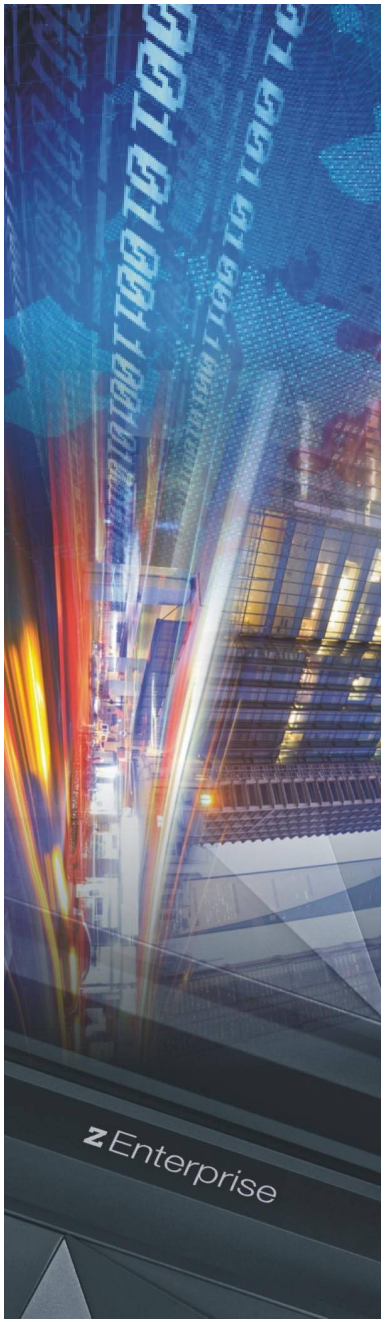
Do's and Don't Do's

- **Do's**
  - Use a socket API that is available on all stacks
    - LE/C Socket API, EZA Socket API, or maybe CSI SOCKET macro
  - Implement transparent support for IPv6
    - Detect the stack or protocol support automatically
    - Add fallback code if one protocol is not available

- **Don't Do's**
  - Do not attempt to parse a textual IPv6 address manually
    - Use getaddrinfo or inet_pton instead
  - Do not use vendor specific features
    - This will make your application fail when running with other TCPIP stacks
  - Do not assume that IPv6 support is always available
    - CSI still supports no IPv6
  - Do not even assume that IPv4 support is always available
    - BSI IPv6-only stack (not coupled)
  - Do not assume that the new IPv6 related functions are always available
    - Older z/VSE versions may not support them
    - The TCP/IP Stack may not support them

Summary

- **IPv6 becomes more and more important in the industry**
  - Make sure your network environment and your applications are IPv6 ready
- **Use a vendor or stack independent Socket API in your applications**
  - EZA Socket APIs (EZASMI or EZASOCKT)
  - LE/C Socket API
- **z/VSE's IPv6 enabled Socket APIs behave the same as the industry wide de factor standard**
  - BSD Sockets
- **Implement 'smart' IPv6 support**
  - Use a transparent approach
  - Automatically detect stack, vendor or IPv6 capabilities
- **Its time to start working on IPv6 support !**



Questions ?

THANK YOU