Software changes for Website and Application IPv6 Readiness

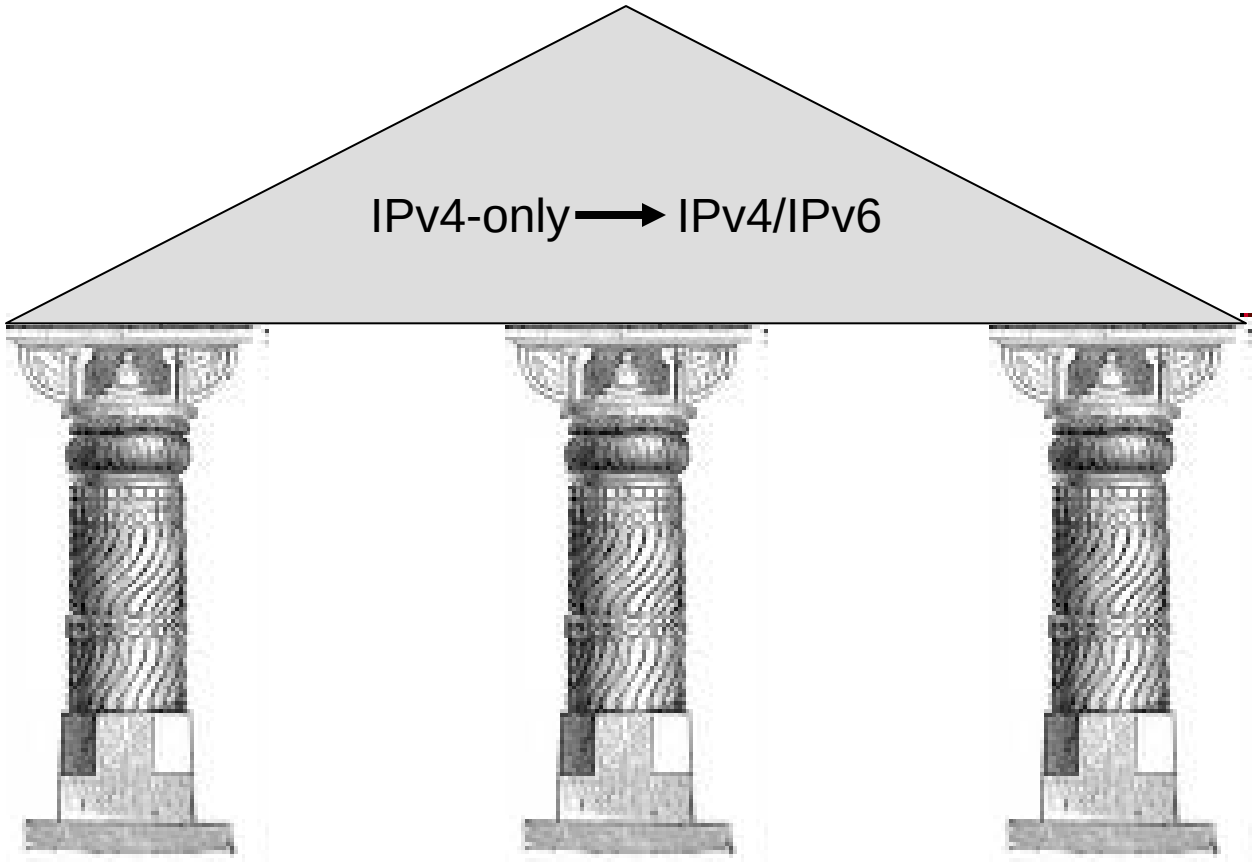
Ahmed Abu-Abed, P.Eng.
Tamkien Systems

ahmed@tamkien.com

Agenda

- Introduction
- Enabling Website IPv6 and Forum Certification
- Intro to Socket Programming for IPv6
- Issues with Legacy IPv4 Code
- Migrating code to IPv6
- IPv6 Compliance Code Checking Tools
- On Java, Perl and PHP IPv6 migration
- References

The 3 Pillars of IPv6 Readiness



Computer
OS + IP Stack

Software
Applications

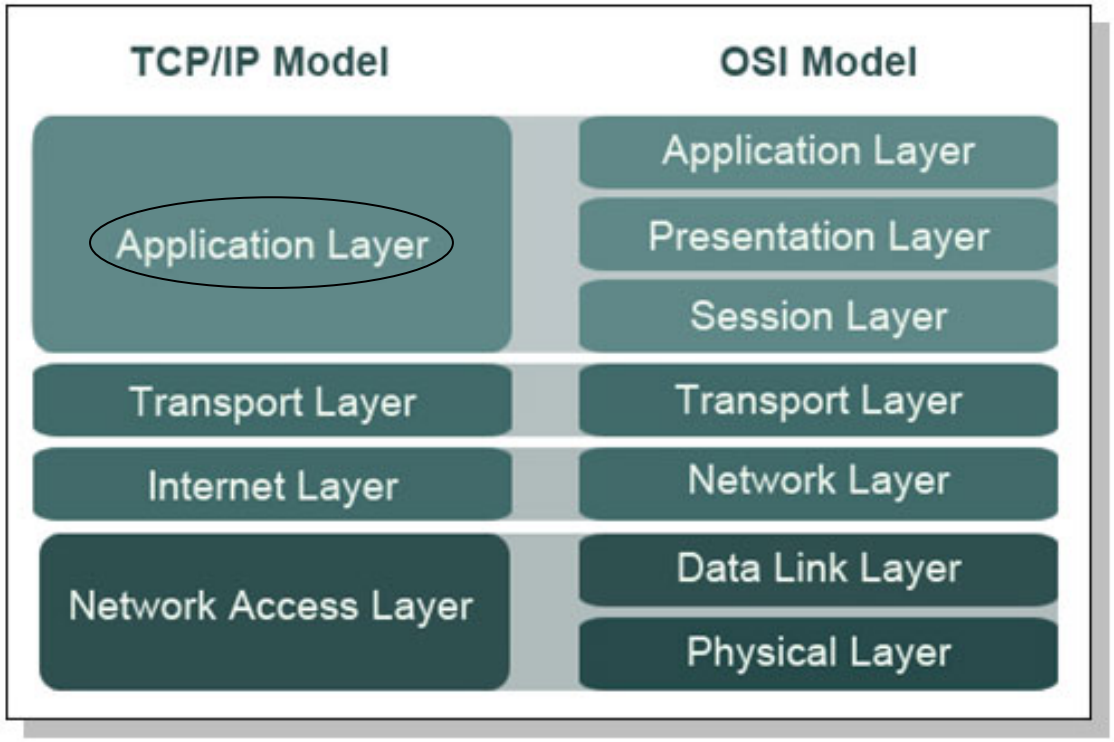
Networks
IX/Core/Edge + Access

Need for Transitioning Applications

- Applications are the reason we use networks
  - Some examples: *VoIP/SIP , websites, OSS/BSS, cloud computing, distributed database, Apache, "ping", Microsoft Exchange, etc.*
- IPv4 has been 'hard coded' into apps, via:
  - Address fields for IPv4, i.e. 32 bits wide
  - 127.0.0.1 loopback address
  - Data-structures and functions that are unusable in a dual-stack environment
  - Broadcast calls, not explicitly supported in IPv6
- Porting applications code to IPv4/IPv6 *now* is necessary for migration.
  - Microsoft Office, Exchange, and Linux apps are dual-stack already
  - Other critical applications like OSS/BSS need checking with the vendor
  - This presentation highlights how to change your code if necessary

Applications in a familiar model

Do not use IPv4 dependant code or addressing



Making websites Ready for IPv6

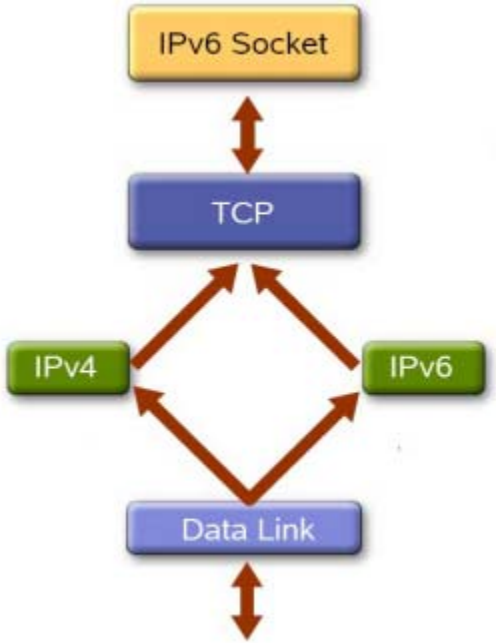
- Upgrades are needed in *reachability* and *webcode*
  - Most of the work needed is on reachability only
- Webcode update needed if
  - IPv4 addresses are hard coded in website, and/or
  - IPv4 functions being used in scripts (PHP, CGI, etc.)
  - More on this later
- Follow the guidelines mentioned on
  - http://www.ipv6forum.org/ipv6_enabled
- Then get listed on IPv6 Forum as an approved site
  - And put the IPv6 Forum WWW logo on your web site
  - IPv6 Forum *only needs reachability* check before approval

IPv6 WWW Certification Steps

- v6eSG (v6 Enabled Logo Steering Group) of the IPv6 Forum tests the website to ensure:
  - It has a global IPv6 address, AND
  - AAAA record in DNS (to translate domain name to IPv6 addr), AND
  - Accessible via HTTP protocol.
- Tests above repeated periodically
- Once the tests pass the website gets registration number & logo
  - Exact test procedures are documented on IPv6 Forum website
- How to do the above ?
  - Host the IPv6 version of website on an IPv6 ready provider (HE.com, etc...)
  - IPv6 Provider will register the AAAA record on a DNS serving the IPv6 global community.
  - Then apply to v6eSG for testing

IP code flow on computers

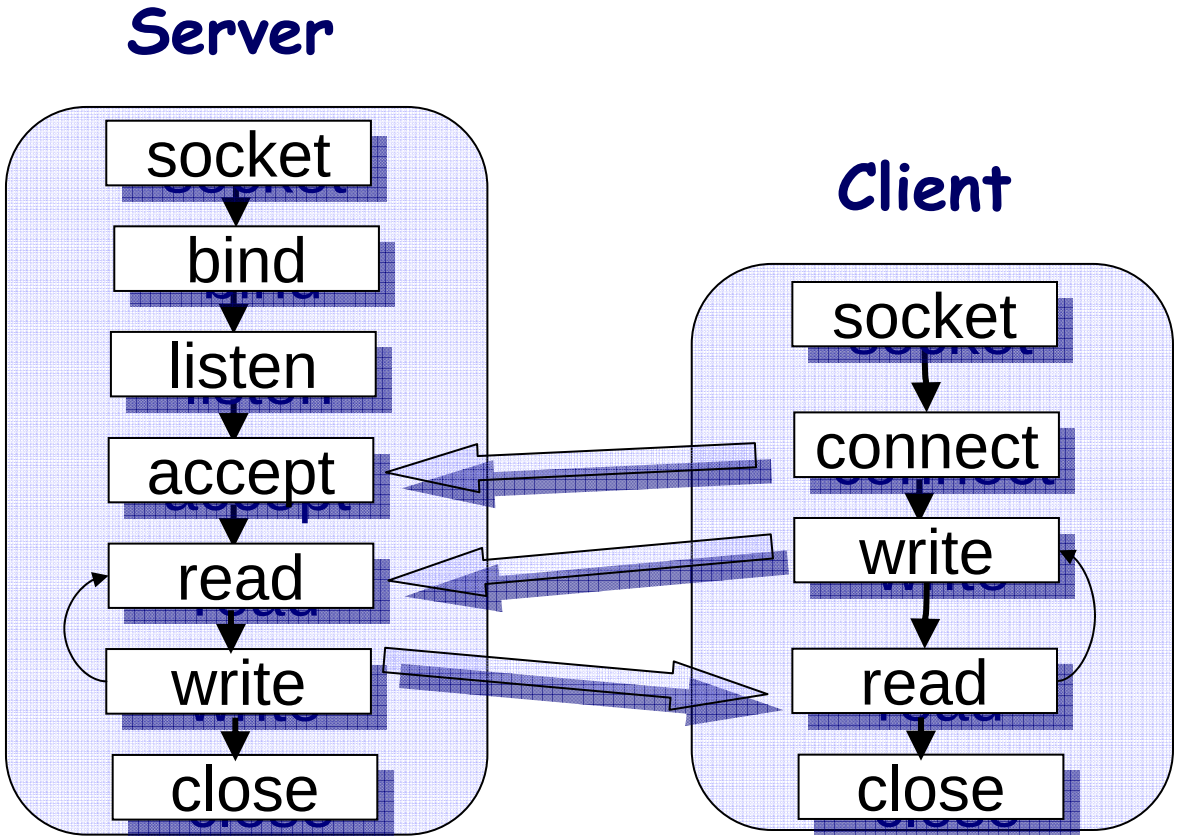
- Assume that system (operating system + IP) has a dual-stack
- Application code should be able to execute on either IPv4 or IPv6
- The OS and IP stack on the computer inspect the header for version field
- Using *dual applications in dual-stack nodes*, application decides to use IPv4 or IPv6 protocol code at runtime



Introduction to Sockets for IPv6

- A IP socket is defined as an address and port number of a network connection plus the transport protocol (UDP or TCP) in use
- Enables programmers to create network-capable applications to transmit application data across the wire (or wireless)
- ALL applications and protocols use the SAME socket programming approach
  - Standardized as Berkeley Sockets (BSD) from UC Berkeley in 1983
  - Uses C/C++ programming languages (Java has different approach)
- For IPv4/IPv6 dual-stack implementation
  - On Windows use Winsock 2 library
  - On Unix use BSD 4.4 library

TCP Socket Calls for IPv4 & v6



Ref: Eva Castro, US IPv6 Summit Dec. 2003

- **Same fundamental calls and procedures are used in v4 and v6. UDP is similar**

Changing Legacy IPv4 Code

- Variables size: IPv6 addresses need 4x the space in octets
  - Use new headers that take care of reserved space
- Lots of references hard coded loop back (127.0.0.1) addresses
  - Change to loop back variable name
  - The variable gets its value depending on v4 or v6 automatically
  - Done through header file variable definitions
- Changed functions: For example, to get IP address from host name
  - Must use *getaddrname()* instead of *gethostbyname()*
- Do not forget user graphical interface, hard coded IPv4 address dialog boxes
  - Adjust the address field to handle both v4 and v6 addresses

Structures and functions to change

IPv4	IPv6	IP version Independence
hostent	addrinfo	yes
sockaddr_in	sockaddr_in6	no
sockaddr_in	sockaddr_storage	yes
in_addr	sockaddr_storage	yes
INADDR_ANY	in6addr_any	no
INADDR_LOOPBACK	in6addr_loopback	no
gethostbyname	getaddrinfo	yes
gethostbyaddr	getnameinfo	yes
inet_addr	getnameinfo	yes
inet_ntoa	getnameinfo	yes
inet_ntoa	inet_ntop	yes
inet_aton	inet_pton	yes
inet_addr	inet_pton	yes

Dual-stacking coded differently in Operating Systems

- WinXP sp1, Win2003
  - Create two sockets, one for IPv4 and one for IPv6
  - Application has code for both options
  - During runtime the appropriate socket is invoked depending on IP version in use
  - More cumbersome
- Vista, Win7, Win2008, Linux 2.4 & 2.6
  - Create one IPv6 socket
  - This can handle both v4 and v6 traffic by setting IPV6_V6ONLY socket option to zero
  - Place IPv4 address in lower 32 bits with 0::FFFF padding
  - Thus *a.b.c.d* becomes 0::FFFF:*a.b.c.d*

Best Practices when Coding

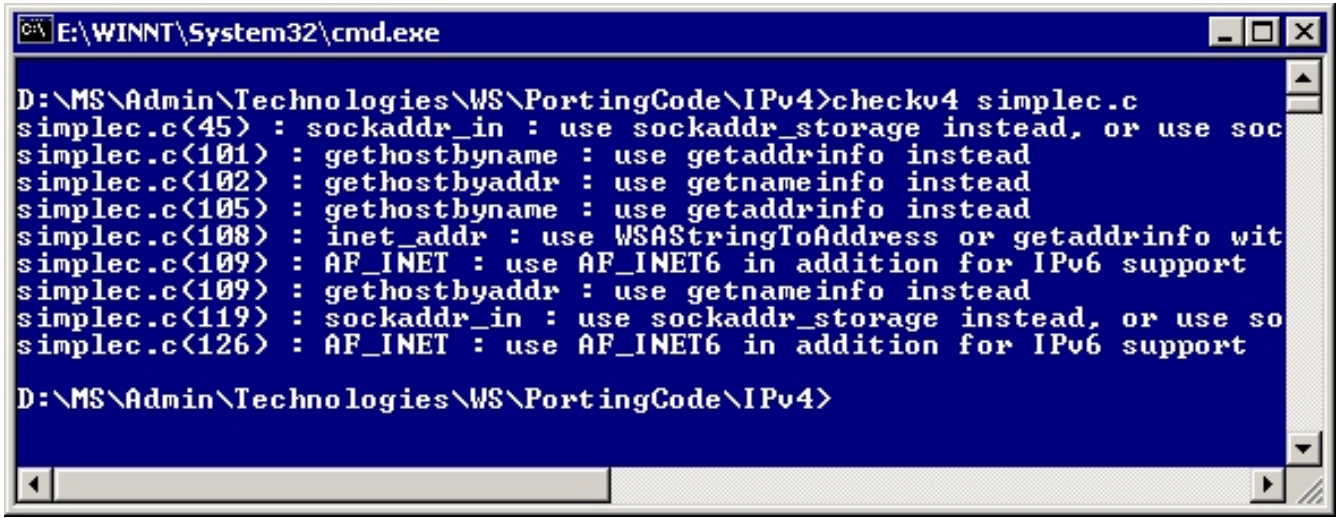
- Make your code *protocol-version-independent*
  - by using IPv4/v6 common functions and structures shown before
- Use *data structures* to hold addresses
  - 32bit fields/variables are good only for IPv4
- Loop through *all IP addresses* when needed
  - Since IPv6 can assign multiple addresses for an interface (link-local, unicast, etc.)

Summary code changes

- The following needs to be revised to have IPv4/IPv6 applications:
  - Data Structures
  - Function Calls
  - Use of Hardcoded IPv4 Addresses
  - User Interface Issues (32 bit dialog boxes)
  - Underlying Protocols like RPC calls
  - Dual-Stack Sockets (see previous slide)
- Plus dual-stack programming sequence has slightly changed
- For detailed list of changes see the references of this presentation
  - This is not a major change from the existing IPv4 Socket Programming
- After changing use code checking tools to verify

Windows dual/stack application checking tool

- *Checkv4* utility, acts on source code (.c) file
  - Already installed with Microsoft SDK for Vista and later
  - Earlier version (since Win2000) available but with limited features
  - Execute on command line with source file as argument
  - Act on messages Checkv4 presents



Linux dual/stack application checking tool

- Linux has an IPv6 "Compliance Automatic Runtime Experiment“, or CARE, tool
  - Similar to Windows tool but diagnoses *running* programs
  - Can check Java, Perl and Python code (as long as they use the C library)
  - Can check any running program or command in real-time
    - Thus one can have it on ALL the time to catch any IPv6 non-compliance
  - See screen shot for sample output

Linux CARE code checking tool

```
etienne@gl:~$
etienne@gl:~$ ipv6_care shell -v

IPv6 compliance checking is now enabled for all programs started in this bash session.
Diagnosis will be reported in '/tmp/ipv6_diagnosis'.
All network-related programs will be logged.

[IPv6 CARE] ~$ telnet www.google.fr 80
IPV6 CARE detected: getaddrinfo() with [ ai_family=AF_UNSPEC ai_socktype=SOCK_STREAM nodename=www.google.fr servname=80 ]
IPV6 CARE detected: getaddrinfo() with [ ai_family=AF_UNSPEC ai_socktype=SOCK_STREAM nodename=www.google.fr servname=80 ]
IPV6 CARE detected: getnameinfo() with [ sa.ip=209.85.129.99 sa.port=80 ]
Trying 209.85.129.99...
IPV6 CARE detected: socket() with [ domain=AF_INET type=SOCK_STREAM protocol=ip ]
IPV6 CARE detected: connect() with [ socket=3 address.ip=209.85.129.99 address.port=80 ]
IPV6 CARE detected: freeaddrinfo()
Connected to www.l.google.com.
Escape character is '^]'.
IPV6 CARE detected: select() with [ readfds[0]=1 readfds[3]=1 errorfds[3]=1 ]
IPV6 CARE detected: select() with [ readfds[0]=1 readfds[3]=1 errorfds[3]=1 ]
^]

telnet> quit
IPV6 CARE detected: close() with [ fd=3 ]
Connection closed.
IPV6 CARE detected: select() with [ writefds[1]=1 ]
IPV6 CARE detected: select() with [ writefds[1]=1 ]
IPv6 diagnosis for 'telnet www.google.fr 80' was generated in: /tmp/ipv6_diagnosis/telnet/by_pid/pid_14103
[IPv6 CARE] ~$
[IPv6 CARE] ~$ exit
exit

End of IPv6 CARE shell.

etienne@gl:~$
```

On Java, Perl and PHP migration

- Java functions (API) have been IPv4/IPv6 compliant since ver 1.5 (Windows) or 1.4 (Linux). IPv6 support in Java is there if
  - Every application does not use hard coded IPv4 address
  - Address and socket information uses Java Socket API
- Perl: Add the "*Socket6"* module for IPv6 in addition to "*Socket"* module for IPv4
  - Otherwise similar concept to C Socket Programming explained above
- PHP: IPv6 Supported in 4.3.4 and 5.2.3 modules
  - A few functions have been added for support of IPv6
- For all above, refer to R. Nucara document in References for details

References

- W.R. Stevens et al, UNIX Network Programming, 3rd Edition, Addison-Wesley
- Owen DeLong, Porting to Dual Stack - - Not That Hard, ARIN XXIV, October 2009
- Introduction to IPv6 Programming, Rino Nucara, 2007, GARR, Italy
- MSDN Library, http://msdn.microsoft.com/en-us/library/ms738649(VS.85).aspx
- Checkv4 Windows Utility, http://msdn.microsoft.com/en-us/library/ms740624(VS.85).aspx
- IPv6 CARE UNIX tool, http://sourceforge.net/projects/ipv6-care/
- Eva M. Castro, IPv6 Porting Applications, Dec. 2003, US IPv6 Summit.
- Marc Blanchet, Migrating to IPv6, 2006, John Wiley.
- IPv6 Forum WWW Enabled Logo procedures, http://www.ipv6forum.org/ipv6_enabled/specification/IPv6_WWW_Enabled_Specification_v2.0.1.pdf