HPCMP UGC'11

Birds of a Feather

IPv6 & Applications

Ipv6-team@hpcmo.hpc.mil

21 June 2011

Distribution A: Approved for Public release; distribution is unlimited.

DREN IPv6 Leader

- DREN is an IPv6 network, supporting legacy IPv4
- "all green" (one of the first to achieve this)
-

	Web	Mail	DNS	NNTP	XMPP (Jabber)
Defense Research and Engineering Network (dren.net)	SUCCESS	SUCCESS	0/0 3/3	Stratum 1	SUCCESS

- Sharing lessons learned with broad community (nationally and internationally)
- Have gained international recognition
- Good example of how DREN should work
  - Stay abreast of trends; see where the technology is going
  - If applicable to DoD, adopt early and gain experience (be a pioneer)
  - Help set priorities for industry
  - Share lessons learned with rest of DoD

Distribution A: Approved for Public release; distribution is unlimited.

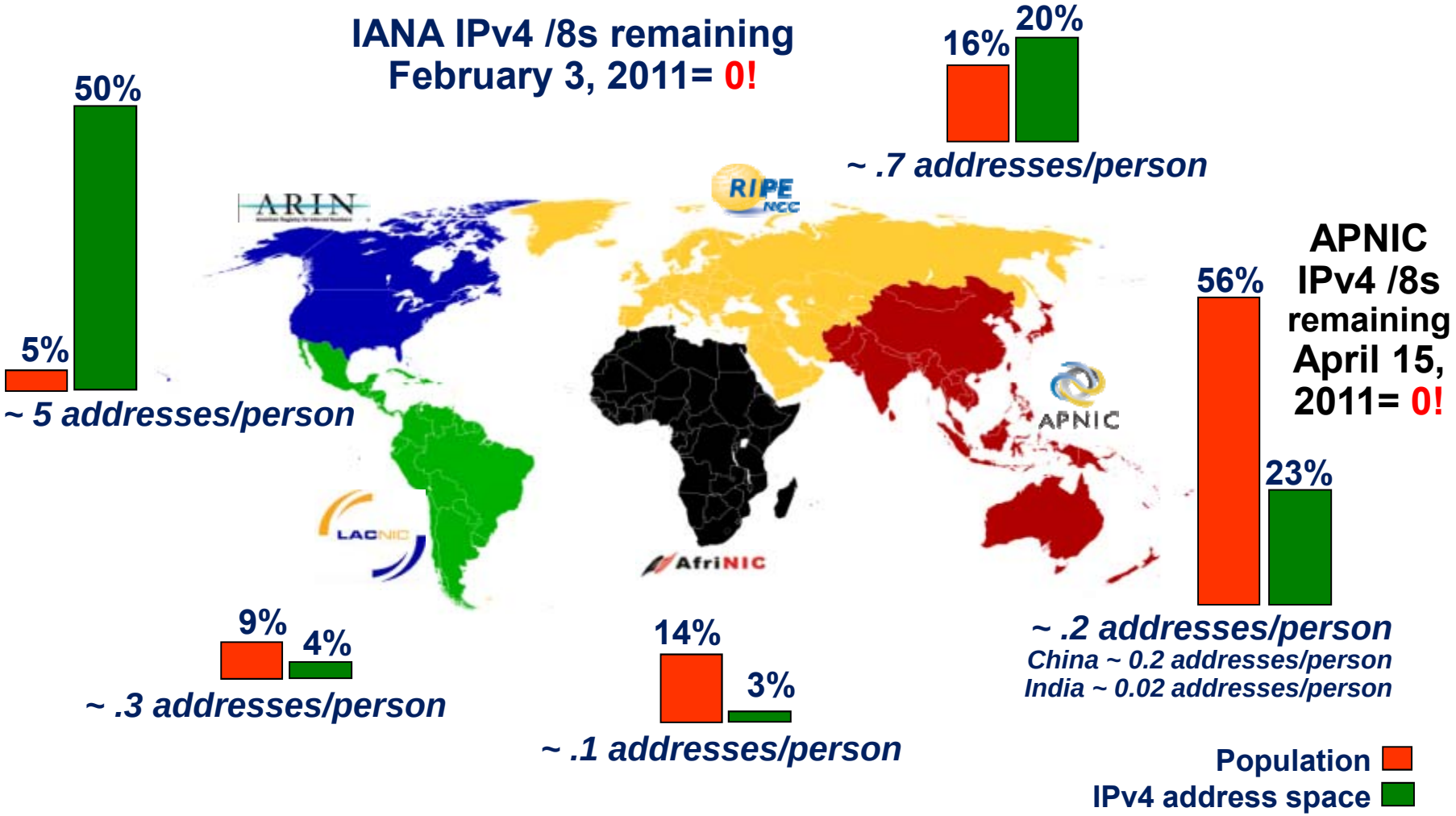
IPv6 Application Conversion

- Why IPv6 at all?
- Why IPv6 in DoD?
- Some examples of programs
- Automated Methods for Converting (sort of)

Distribution A: Approved for Public release; distribution is unlimited.

Why IPv6 at all?
It all depends on where you live

IANA IPv4 /8s remaining February 3, 2011= 0!



APNIC IPv4 /8s remaining April 15, 2011= 0!

Regional IPv4 depletion will occur unevenly
(see www.ipv4depletion.com for details)

Distribution A: Approved for Public release; distribution is unlimited.

Why IPv6 at all?
Churn happens

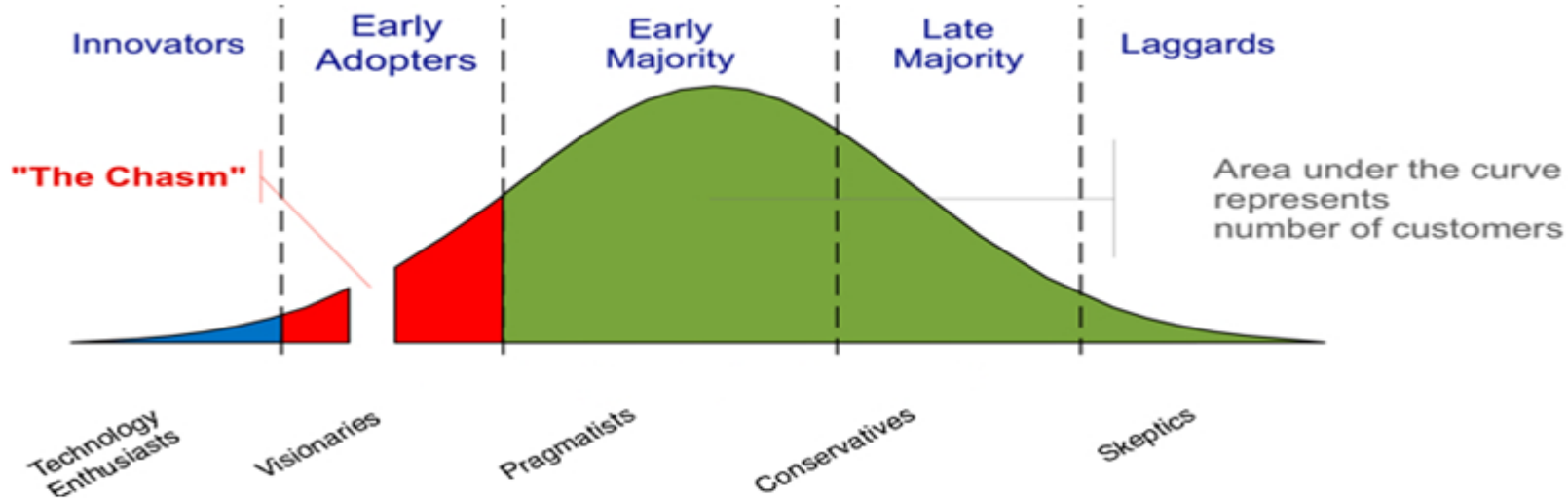
- **Why does your network infrastructure exist:**
  - For internal use? – To enable external use?
- **What happens internally**
  - "We have lots of addresses for our own use." So?
- **What happens externally**
  - Growth: new citizens/customers/clients/partners arrive
  - Churn: old citizens/customers/clients/partners leave
  - When more leave than arrive over time, you shrink

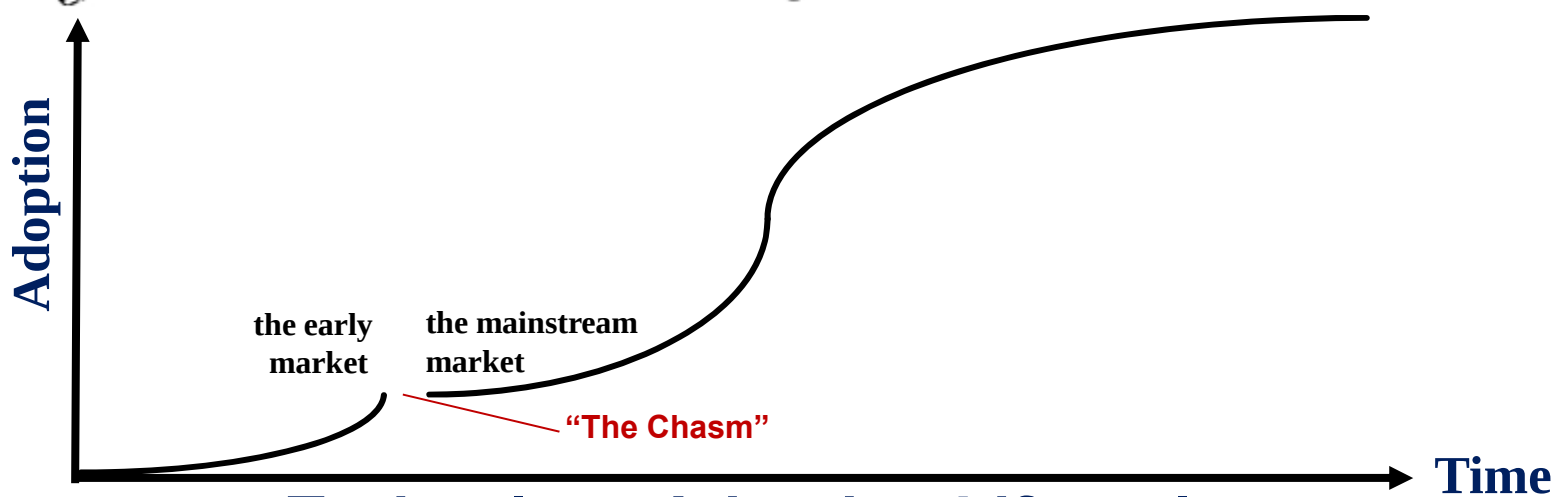
- **No infrastructure change → churn without growth**
  →→ you will shrink

"It's the addresses, 'stupid'!!" – apologies to Pres. Clinton

Distribution A: Approved for Public release; distribution is unlimited.

Why IPv6 at all?
IPv6 has emerged from "The Chasm"





Technology Adoption Lifecycle

Bohlen, Beal & Rogers, "Diffusion Process", 1957, Moore, "The Chasm", 1991

Distribution A: Approved for Public release; distribution is unlimited.

IPv6 Application Conversion

- Why IPv6 at all?

➢ Why IPv6 in DoD?

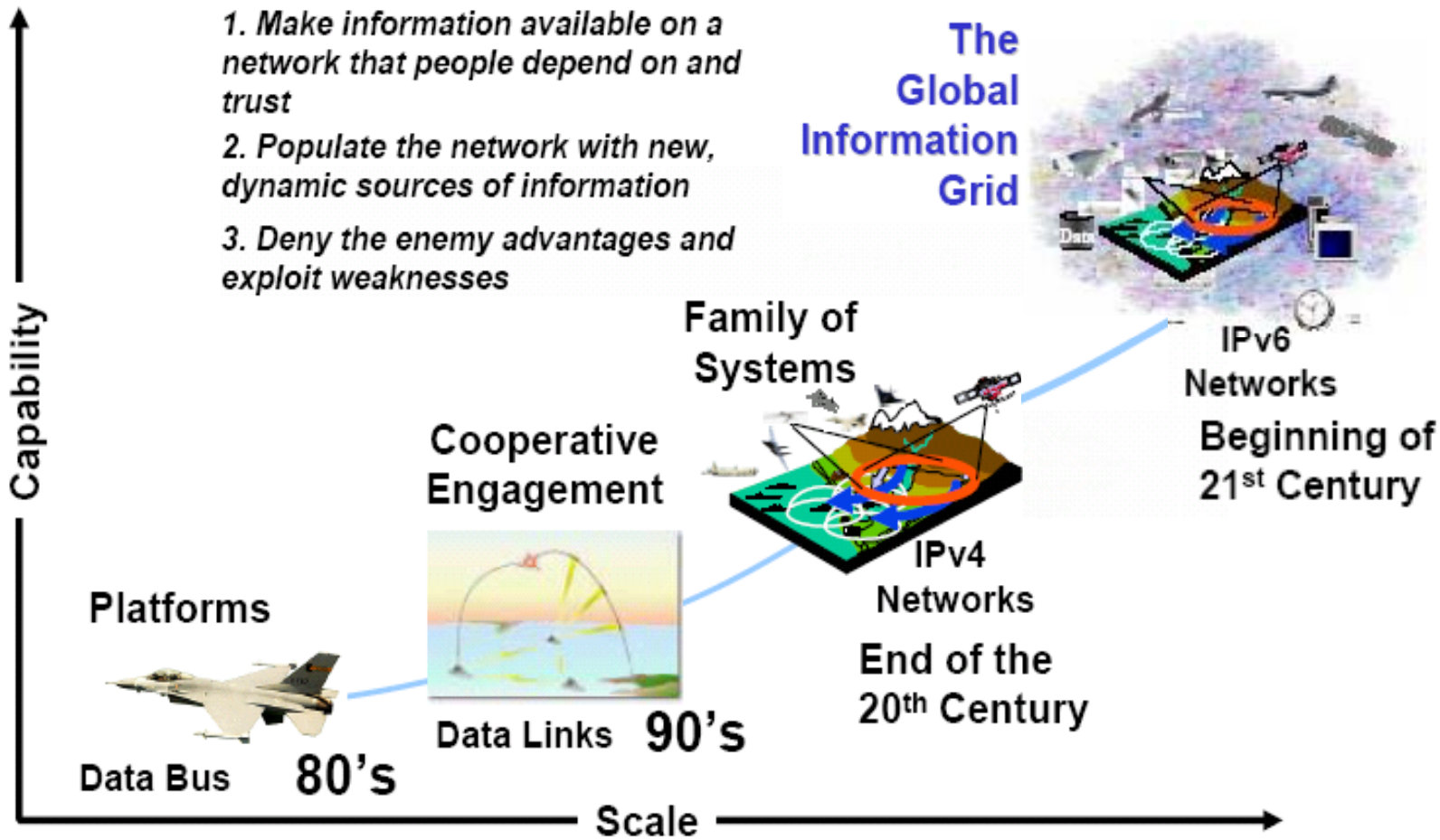
- Some examples of programs

- Automated Methods for Converting (sort of)

Distribution A: Approved for Public release; distribution is unlimited.

Why IPv6 in DoD?

"DoD Net-Centricity"



Distribution A: Approved for Public release; distribution is unlimited.

Why IPv6 in DoD?

- **There are important long-term reasons to transition:**
  - Global pressures for transition to IPv6 (e.g., Asia, Japan, Europe)
  - Major industry S/W and H/W vendors are committed to IPv6 and in the long run will reduce support for IPv4
  - Maintain interoperability with IPv6-based joint, allied, and coalition forces in overseas operational environments
  - Provide support for the increasing mobile, interconnected soldiers, platforms, wired and wireless sensor webs, and facilities that will be exchanging information on an end-to-end basis across DoD's net-centric Global Information Grid (GIG)
  - There are long lead-times across DoD, due to multi-year life cycles of embedded systems, and lengthy system development cycles for major programs being fielded in 2011+ with IPv6 capabilities

Distribution A: Approved for Public release; distribution is unlimited.

Why IPv6 in DoD?

- **DoD and each service have established an IPv6 transition planning organization and have a Transition Plan**
  - DoD tasked DISA to establish the DoD IPv6 Transition Office (DITO)
  - The Army established an IPv6 Transition Plan Working Group (ITPWG)
  - The Navy established a Navy IPv6 Transition Project Office (NITPO)
  - The Air Force established an IPv6 Transition Management Office (TMO)
  - The Marine Corps established an IPv6 Transition Working Group (IPv6TWG)

Distribution A: Approved for Public release; distribution is unlimited.

IPv6 Application Conversion

- Why IPv6 at all?
- Why IPv6 in DoD?
- ➢Some examples of programs
- Automated Methods for Converting (sort of)

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
Anecdotal instances of conversion

- **Most examples are for "modern" languages**
  - C/C++
  - Java is <u>really</u> easy
  - FORTRAN ???
    - Only reference said he had to turn off IPv6 on a Mac
- **Network enabled multiplayer games**
  - DOOM: a few hundred lines of code changed, ~2 weeks elapsed time to learn IPv6, analyse the code, make changes
  - QUAKE II: 32 hours effort to download code for game and server, modify code, set up server

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
Anecdotal instances of conversion

- **Numerous OS network utilities**
  - Vanilla telnet ~20 lines of code
  - Secure (kerberized) telnet harder, because of encryption
- **Other network utilities (ssldump, et cetera)**
  - On the order of a few hours to a couple of days to transition
- **SNORT network analysis tool**
  - highly IP packet intensive software for IPv6 packet analysis
  - 20,000 context differences from one version to the next

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
Anecdotal instances of conversion

- **Typical code sequence from server side**
  - `socket - open a socket`
  - `bind - bind our local address to the socket`
  - `listen - Tell that we are listening to a port`
  - `accept - wait for connections`
  - `read and/or write if TCP`
  - `recvfrom and/or sendto if UDP`
- **From client side**
  - `socket - open a socket`
  - `connect - connect to the server`
  - `read and/or write if TCP`
  - `sendto and/or recvfrom if UDP`

SAME as IPv4

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
JAVA is a very high level language ☺

This sample code is IP version agnostic automatically!

```
/*
 * EchoServer2.java
 * Created on May 26, 2005, 10:18 PM
 */
package EchoServer2;
/**
 *
 *
 * @author  rdeterman
 */
import java.io.*;
import java.net.*;

public class EchoServer2 {
   public static void main(String args[]) {

// declaration section:
// declare a server socket and a client socket for the
      server
// declare an input and an output stream

      ServerSocket echoServer = null;
      String line;
      DataInputStream is;
      PrintStream os;
      Socket clientSocket = null;
```

```
g// Try to open a server socket on port 9999
// Note that we can't choose a port less than 1023 if we are not
// privileged users (root)

      try {
        echoServer = new ServerSocket(9999);
      }
      catch (IOException e) {
        System.out.println(e);
      }

// Create a socket object from the ServerSocket to listen and
      accept
// connections.
// Open input and output streams

try {
        clientSocket = echoServer.accept();
        is = new
       DataInputStream(clientSocket.getInputStream());
        os = new PrintStream(clientSocket.getOutputStream());
```


Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
C Modifications for IPv6 addresses

- **This sample C code illustrates opening a Socket for IPv4**

```
/* Open socket. */
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
      perror("socket");
      return (-1);
sin.sin_family = AF_INET;
    sin.sin_port = htons(port);
    (void) memcpy((void *) &sin.sin_addr, (void *)hp->h_addr, hp->h_length);
```

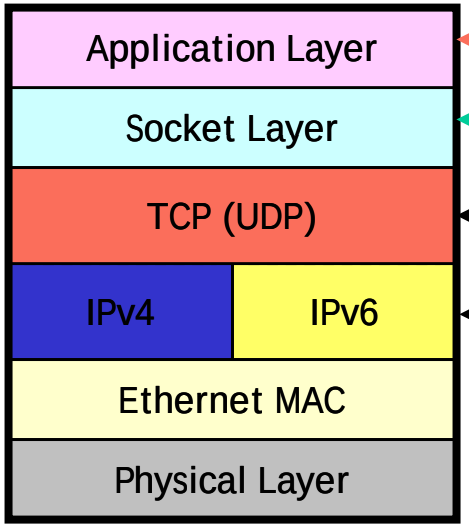
- **This sample C code illustrates opening Sockets for IPv6**

```
SOCKET ServSock[FD_SETSIZE];
    ADDRINFO AI0, AI1;
    ServSock[0] = socket(AF_INET6, SOCK_STREAM, PF_INET6);
    ServSock[1] = socket(AF_INET, SOCK_STREAM, PF_INET);
    ...
    bind(ServSock[0], AI0->ai_addr, AI0->ai_addrlen);
    bind(ServSock[1], AI1->ai_addr, AI1->ai_addrlen);
    ...
    select(2, &SockSet, 0, 0, 0);
    if (FD_ISSET(ServSocket[0], &SockSet)) {
      // IPv6 connection csock = accept(ServSocket[0], (LPSOCKADDR)&From, FromLen);
```


Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
What needs to be changed?

Applications use new socket API (user)



IPv6 Extensions to existing socket application program interface (new API RFC)

New protocol changed to handle larger (128 bit) addresses (OS vendor)

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
New socket API changes

- **Specifications change for API socket calls**
  - RFC3493: Basic Socket Interface Extensions for IPv6 (though Microsoft only supports the deprecated RFC2553)
  - RFC3542: Advanced Sockets Application Program Interface (API) for IPv6
- **Most applications: minimal changes to API**
  - Change the socket, name-service, and data declarations
  - Get an IP agnostic program (will use either IPv4 or IPv6)
- **Network-intensive applications: can be major**
  - IDS, firewall, network/security analysis tools
  - Security tools that use addresses in protocol

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
New socket API changes (continued)

- **Basic `socket()` system call is unchanged**
  - Just a new protocol family for IPv6

    `s = socket(AF_INET, SOCK_DGRAM, 0);` *[IPv4]*

    `s = socket(AF_INET6, SOCK_DGRAM, 0);` *[IPv6]*
  - IPv4-only sockets continue to work as they always have
  - System calls that bind or receive address to/from IPv6 sockets must use IPv6 socket addresses
    - bind(), connect(), sendmsg(), sendto()
    - accept(), recvfrom(), recvmsg(), getpeername(), getsockname()

Distribution A: Approved for Public release; distribution is unlimited.

IPv6 Application Conversion

- Why IPv6 at all?
- Why IPv6 in DoD?
- Some examples of programs
- ➢ Automated Methods for Converting (sort of)

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting

"Build a little, test a little, learn a lot" – RADM *Meyer*

- **Experience is the best teacher**
  - Setting up an IPv6 test lab
    - 2 computers, a router, and some cabling is enough to begin
    - Microsoft describes a virtual lab using only one real computer
  - Need to test your external IPv6-only connectivity?
    - www.v6.dren.net lists many IPv6-only links
  - Need to test your IPv6 infrastructure?
    - www. ipv6-test.com, www. test-ipv6.com verify connectivity
    - www.ipv6tools.org allows you to ping, trace, and query DNS
    - ipv6-speedtest.net allows you to test performance ('-' *not* '.')
  - Want IPv6 connectivity today? (you can try this at home)
    - Use the gogoCLIENT from gogoware.gogo6.com and the authenticated.freenet6.net Freenet6 tunnel broker
    - Others are available: tunnelbroker.net and www.sixxs.net

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Tools you can use

There are some software tools available that can automatically determine if an IPv4 program contains IP-specific calls, and suggest needed changes. A few are:

- porttoipv6.sourceforge.net
  - PortToIPv6 Framework
- ipv6-care.sourceforge.net
  - IPv6 Compliant Automatic Runtime Environment (CARE)
- msdn.microsoft.com. The web address changes, search for:
  - IPv6 Guide for Windows Sockets Applications
  - Checkv4.exe utility program

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Microsoft Checkv4.exe example

- This checker is good at looking for the old IPv4 commands and suggests the newer IP version agnostic commands
- Good for code segment identification; but does not rewrite code
- Be aware that 'getaddrinfo' requires non-trivial amounts of code rewrite.

E:\WINNT\System32\cmd.exe

```
D:\MS\Admin\Technologies\WS\PortingCode\IPv4>checkv4 simplec.c
simplec.c(45) : sockaddr_in : use sockaddr_storage instead, or use sockaddr_in6 in addition for IPv6 support
simplec.c(101) : gethostbyname : use getaddrinfo instead
simplec.c(102) : gethostbyaddr : use getnameinfo instead
simplec.c(105) : gethostbyname : use getaddrinfo instead
simplec.c(108) : inet_addr : use WSAStringToAddress or getaddrinfo with AI_NUMERICHOST instead
simplec.c(109) : AF_INET : use AF_INET6 in addition for IPv6 support
simplec.c(109) : gethostbyaddr : use getnameinfo instead
simplec.c(119) : sockaddr_in : use sockaddr_storage instead, or use sockaddr_in6 in addition for IPv6 support
simplec.c(126) : AF_INET : use AF_INET6 in addition for IPv6 support

D:\MS\Admin\Technologies\WS\PortingCode\IPv4>_
```


Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Sources of useful information

- **WWW resources: DREN IPv6 knowledge base**
  - **www.hpcmo.hpc.mil/cms2/index.php/ipv6-knowledge-base-deployment**
    - **Product Information** article
      - **"2.3 Information about specific IPv6 conversion products"**
    - **IPv6 Training and Learning** article
      - **Books and Programming Techniques**

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting

Things to keep in mind

1. Determine if the underlying protocol (FTP, RPC, etc) is IPv6 aware
2. Use an automated utility to look for needed changes
   a) Change data structures to use new format (if required) (socket_storage for example)
   b) Modify the API calls to use IPv6/IPv4 or agnostic calls
3. Examine source code for hard-coded address (v4 or v6). Ensure no loopback coding is present
4. Look for literal IP data formats
   a) Examine program UI to determine if literal IPs requested/displayed
   b) Examine file and data base access to determine if literal IPs used
5. (Microsoft only) Use the compile flag 'IPV6STRICT'. This allows the compiler to produce IPv6 specific errors during a compile
6. Or, just use JAVA ☺

Distribution A: Approved for Public release; distribution is unlimited.

Backup Slides

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
Anecdotal instances of conversion (continued)

- **Oracle data base server**
  - 9i has supported IPv6 since 2003 (Japan <u>only</u>)
  - 10g r3 in USA in 2008
- **SAP AG R/3 (2004) – internals of ERP**
  - Porting efforts were not expected to be very high
  - Socket C-API extensions allow for smooth migration
  - Use of IP addresses encapsulated in libraries
  - Tough parts were external interfaces
    - IP addresses as payload in protocols
    - IP addresses used by applications you don't even think about
    - IP addresses misused (e.g. as "unique" host identifiers)

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
Anecdotal instances of conversion (continued)

- **OpenH.323 multimedia communications library**
  - 500,000+ lines of code
  - Replacing hard-coded IPv4 addresses with IPv6 addresses, or eliminate them altogether by properly modifying the source code
  - The IP address was indirectly assumed to have a 4-byte length. This included the size of arrays, the number of repetitions for loops and the variables used. This was by far the most time-consuming part of the modification phase
  - A common phenomenon in network applications is the difference between various computer architectures in the order they store 2-byte values. (little endian or big endian). This difference caused some implications in the way IPv6 addresses were stored and manipulated that did not appear with the 4-byte IPv4 addresses, since IPv6 addresses comprised of 8 2-byte fields
  - Inspect a large number of classes and functions. They could not possibly be tested using just a single application, so porting was extended to include a wide range of OpenH323-based applications. Most of the modifications in the applications were relatively small and only had to do with hard-coded IPv4 addresses

Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
C Modifications for IPv6 addresses (continued)

- **This C code illustrates opening a Socket for IPv4 only**

```
SOCKET OpenAndConnect(char *ServerName, int Port, int SocketType) \
{
SOCKET ConnSocket;
LPHOSTENT hp;
SOCKADDR_IN ServerAddr;
ConnSocket = socket(AF_INET, SocketType, 0); /* Open a socket */ if (ConnSocket <0 ) { return
INVALID_SOCKET;
 }
 memset(&ServerAddr, 0, sizeof(ServerAddr));
ServerAddr.sin_family = AF_INET;
ServerAddr.sin_port = htons(Port);
if (isalpha(ServerName[0])) { /* server address is a name */ hp = gethostbyname(ServerName);
if (hp == NULL) { return INVALID_SOCKET;
 }
ServerAddr.sin_addr.s_addr = hp->h_addr; } else { /* Convert nnn.nnn address to a usable one */
ServerAddr.sin_addr.s_addr = inet_addr(ServerName); } if (connect(ConnSocket,
(LPSOCKADDR)&ServerAddr, sizeof(ServerAddr)) == SOCKET_ERROR) { closesocket(ConnSocket);
return INVALID_SOCKET; } return ConnSocket; }
```


Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
C Modifications for IPv6 addresses (continued)

- **This C code illustrates opening a Socket for IPv6 only**

```
SOCKET OpenAndConnect(char *ServerName, int Port, int SocketType) \
{
SOCKET ConnSocket;
LPHOSTENT hp;
SOCKET_STORAGE ServerAddr;
ConnSocket = socket(AF_INET6, SocketType, 0); /* Open a socket */ if (ConnSocket <0 ) { return
INVALID_SOCKET;
 }
 memset(&ServerAddr, 0, sizeof(ServerAddr));
ServerAddr.sin_family = AF_INET6;
ServerAddr.sin_port = htons(Port);
if (isalpha(ServerName[0])) { /* server address is a name */ hp = gethostbyname(ServerName);
if (hp == NULL) { return INVALID_SOCKET;
 }
ServerAddr.sin_addr.s_addr = hp->h_addr; } else { /* Convert nnn.nnn address to a usable one */
ServerAddr.sin_addr.s_addr = inet_addr(ServerName); } if (connect(ConnSocket,
(LPSOCKADDR)&ServerAddr, sizeof(ServerAddr)) == SOCKET_ERROR) { closesocket(ConnSocket);
return INVALID_SOCKET; } return ConnSocket; }
```


Distribution A: Approved for Public release; distribution is unlimited.

Some examples of programs
C Modifications for IPv6 addresses (continued)

- **This C code is IP version agnostic and will accept IPvX**

```
SOCKET OpenAndConnect(char *ServerName, char *PortName, int SocketType)
{
SOCKET ConnSocket;
ADDRINFO *AI;

if (getaddrinfo(ServerName, PortName, NULL, &AI) != 0) {
return INVALID_SOCKET;
}
ConnSocket = socket(AI->ai_family, SocketType, 0);
if (ConnSocket == INVALID_SOCKET) { return INVALID_SOCKET;
}
if (connect(ConnSocket, AI->ai_addr, AI->ai_addrlen) == SOCKET_ERROR) {
closesocket(ConnSocket); return INVALID_SOCKET;
}
 return ConnSocket; }
```


Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Things to keep in mind (continued)

- **Resolve names <u>before</u> opening socket**

```
getaddrinfo(. . .)
s = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);
connect(. . .)
```

- **<u>not</u>**

```
s = socket(AF_INET, SOCK_xxx, 0);
gethostbyname(. . .)
connect(. . .)
```

- **To support both IPv4 and IPv6, iterate when trying to create a connection with the host through all the results returned from the DNS query (use getaddrinfo())**

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Things to keep in mind (continued)

- **Build application-specific address structure in the code.**
  - This would typically be a structure that includes the address type, address data, and optionally address size. This allows a single structure for dealing with multiple address types.
- **Build small set of functions that deal with these address structures**
  - Functions may include: setting, comparing, printing, etc., address structures.
- **When replacing IPv4 addresses in code**
  - Rename variables or structure members so that the compiler can help you find all instances of the address variable that need to be adjusted
- **Use "struct sockaddr_storage" and cast to the appropriate sockaddr_***
- **Hostname lookups**
  - Expect multiple addresses to be returned. This should be obvious for hosts with multiple IPv4 addresses, but consider the potential for several IP addresses (at least 2) per interface. May include multi-cast, anycast, and (rather unlikely) link-local addresses.

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Things to keep in mind (continued)

- **Hostname lookups (continued)**
  - Every interface is assigned multiple addresses:
    1. A link-local address
    2. Zero or more global unicast or anycast addresses (manually or automatically configured).
    3. An “all nodes” multicast address.
       - FF02::1
    4. A “solicited node” multicast address for each of its unicast and anycast addresses.
    5. Multicast addresses for any groups to which it belongs.
  - Routers also have:
    6. Subnet-router anycast address (<subnet prefix>::0)
    7. The “all routers” multicast address
       - FF02::2
  - Example (Microsoft Windows XP)
    - preferred global 2001:480:10:20:20d:56ff:fe3c:9247 (public)
    - preferred link-local fe80::20d:56ff:fe3c:9247, life infinite
    - multicast interface-local ff01::1
    - multicast link-local ff02::1
    - multicast link-local ff02::1:fe3c:9247

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Things to keep in mind (continued)

- **When processing packets, look for:**
  - IPv6 extension headers -- may need to skip for transport layer access
  - Tunneling of IPvX-in-IPvY (how many layers of encapsulation are sufficient to handle?)
  - Libpcap and Berkeley Packet Filters (BPF) issues
- **Write protocol-independent code**
  - Will you be retired before IPv7 or IPv8 is deployed??

Distribution A: Approved for Public release; distribution is unlimited.

Automated Methods for Converting
Things to keep in mind (continued)

- **Libpcap and BPF issues**
  - Used by tcpdump, ethereal, etc.
  - Have supported IPv6 for a few years
    - But must be enabled in the build
  - tcpdump ip6
  - tcpdump net 2001:480:31:10::/64
  - Filter on port X -- verify that v4 & v6 will be processed
    - tcpdump -d port 22
    - tcpdump protochain 6
    - Look for ldh[12] and compare to 0x86dd

Distribution A: Approved for Public release; distribution is unlimited.