DCF255

Lecture 5 | Internet Layer Layer

Agenda

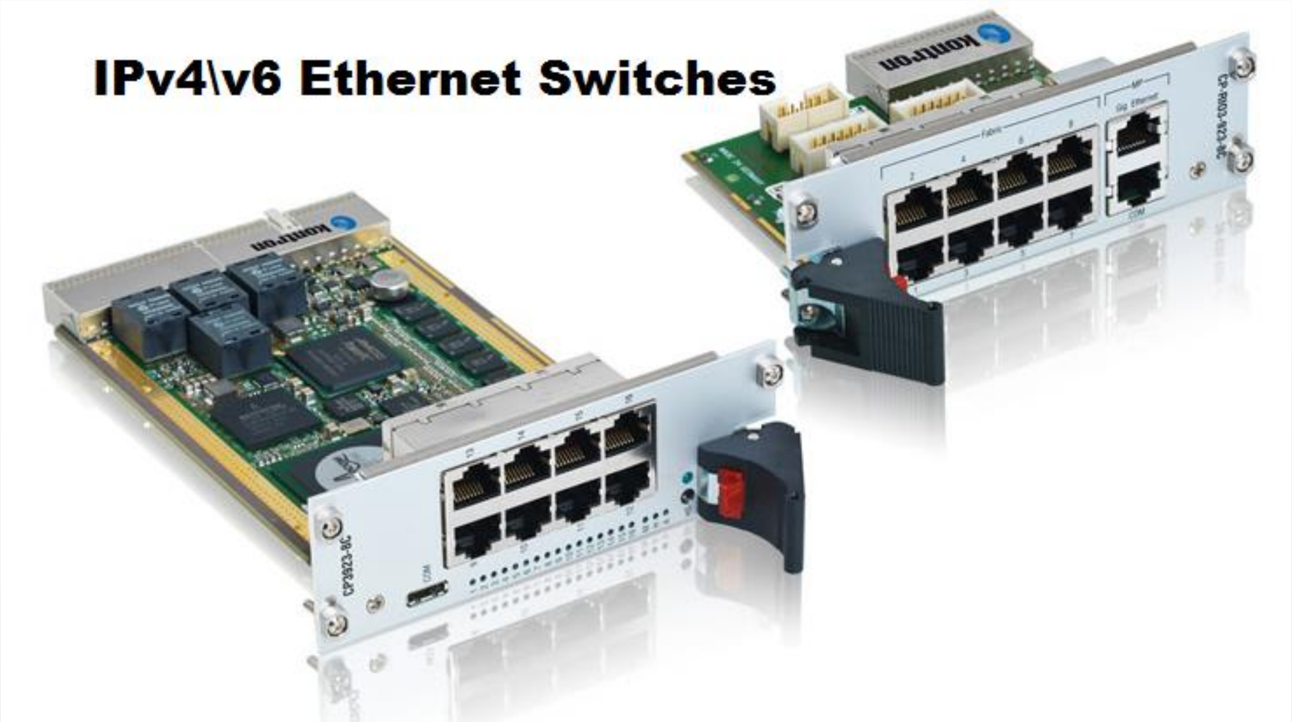
- Internet Layer Functions
  - Dual Stack Environment IPv4/v6
- IPv4 Addressing
- IPv4 Changes to Preserve the Address Space
- IPv6 Addressing
  - Tunneling
  - Scoped Addresses
- IPv6 From a Programming Perspective
  - User Interface Design
  - IP Family Independence
  - Determine IP Family Before Creating Socket

Internet Layer

Functions

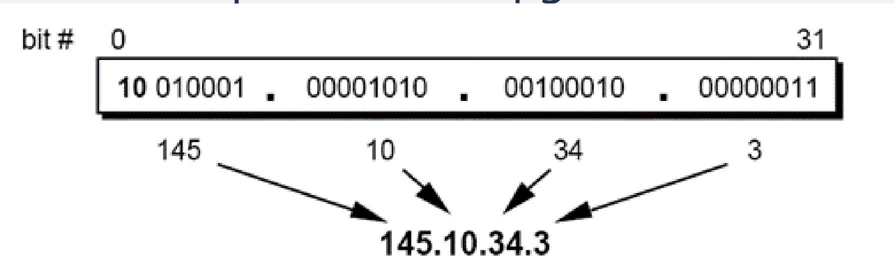
Internet Layer Functions

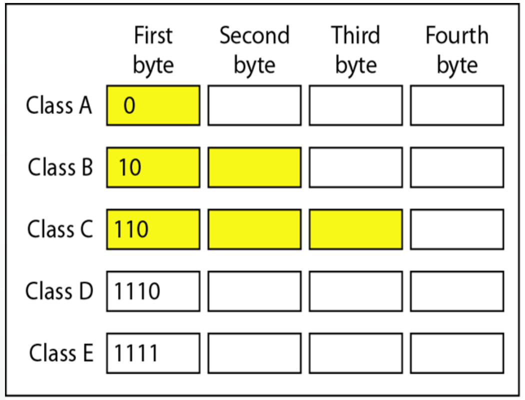
- Internet Layer accepts all content from the Transport Layer as Data
- Responsible for adding header containing the source and destination IP addresses
- Responsible for making Routing Decisions to decide the best route
- For a smooth transition to IPv6 which commenced in 2012, all new hardware most be dual protocol stack enabled called IPv4/v6



IPv4 Addressing

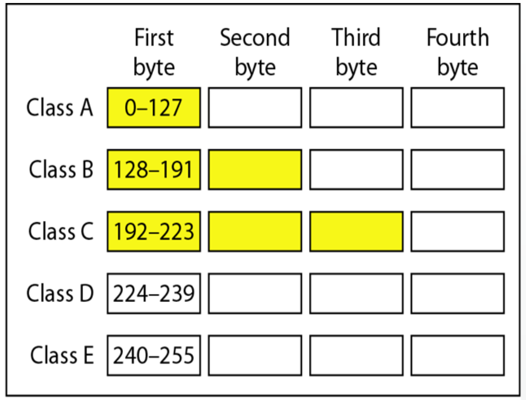
- Classful addressing
- Ipv4 addresses are 32-bit long, separated by a dot and written in dotted decimal notation
- Total address space is 2^{32} = 4.3 billion addresses





	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation



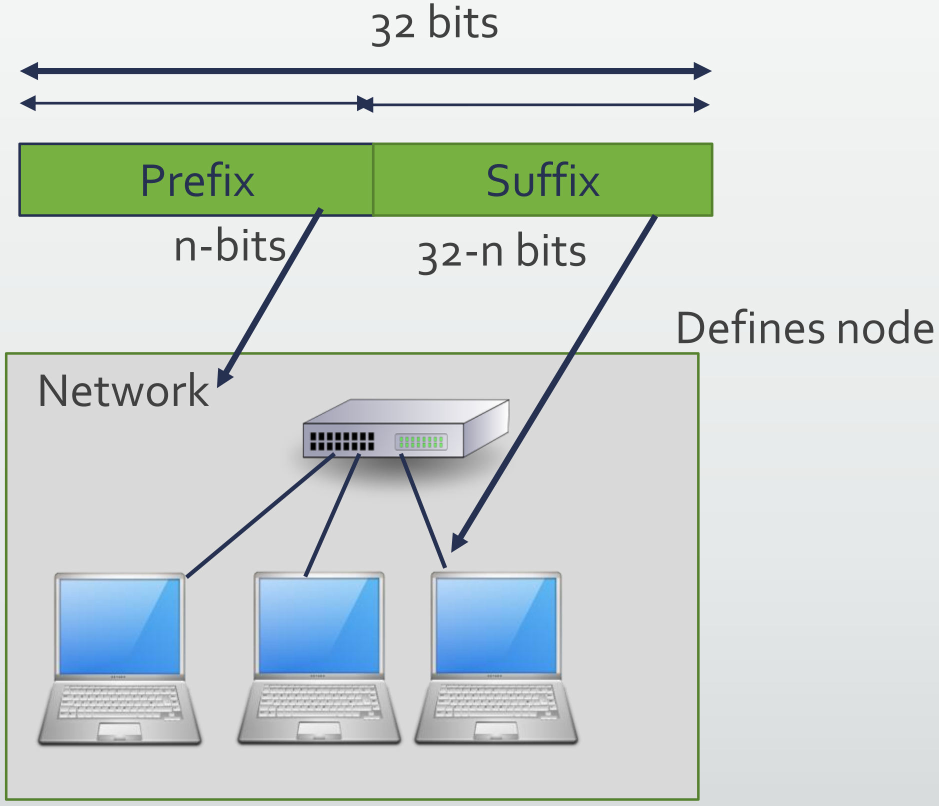
	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

Subnet Mask- Hides, or "masks," the network part of a system's IP address and leaves only the host part as the machine identifier.

Class	Binary	Dotted Decimal	Prefix /CIDR
A	**1111111.00000000.00000000.00000000**	**255.0.0.0**	/8
B	**11111111.11111111.00000000.00000000**	**255.255.0.0**	/16
C	**11111111.11111111.11111111.00000000**	**255.255.255.0**	/24

Default Mask of Classful Addressing



Hierarchy of Ipv4 Addressing

IPv4 and Changes to Preserve the Address Space

- IPv4 32 bit address space, each octet can contain a value from 0-256

- Maximum address size is 256 X 256 X 256 X256 or 4,294,967,296 addresses (4.3 billion)

- With growth of WWW in 1991, and development of smartphones, tablets, gaming systems and IP enabled devices such as phones and VoIP systems

- Address space quickly became depleted, running out in fall 2015

- To preserve the address space as long as possible the IETF instituted changes

IPv4 and Changes to Preserve the Address Space

1. Networks were allowed to create Private address spaces.

2. NAT, Network Address Translator, was created to act as a proxy gateway converting private host addresses to public addresses

3. DHCP, Dynamic Host Configuration Protocols was created to act as a server to allocate addresses from a pool of available LAN addresses.

4. Internet address were assigned using CIDR, Classless Inter-Domain Routing which uses the address space more efficiently

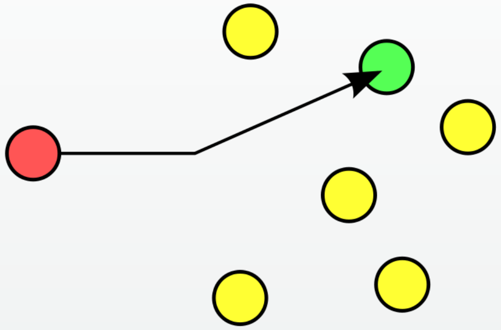
- All of these preservatives have given us the network structure we are familiar with at home and at Seneca.

IPv6

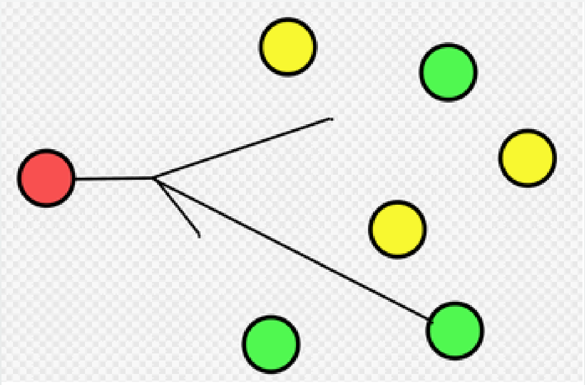
Addressing

IPv6 Addressing - 3 Types

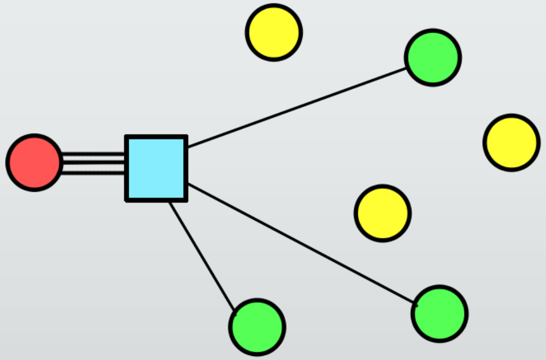
1. Unicast



2. Anycast

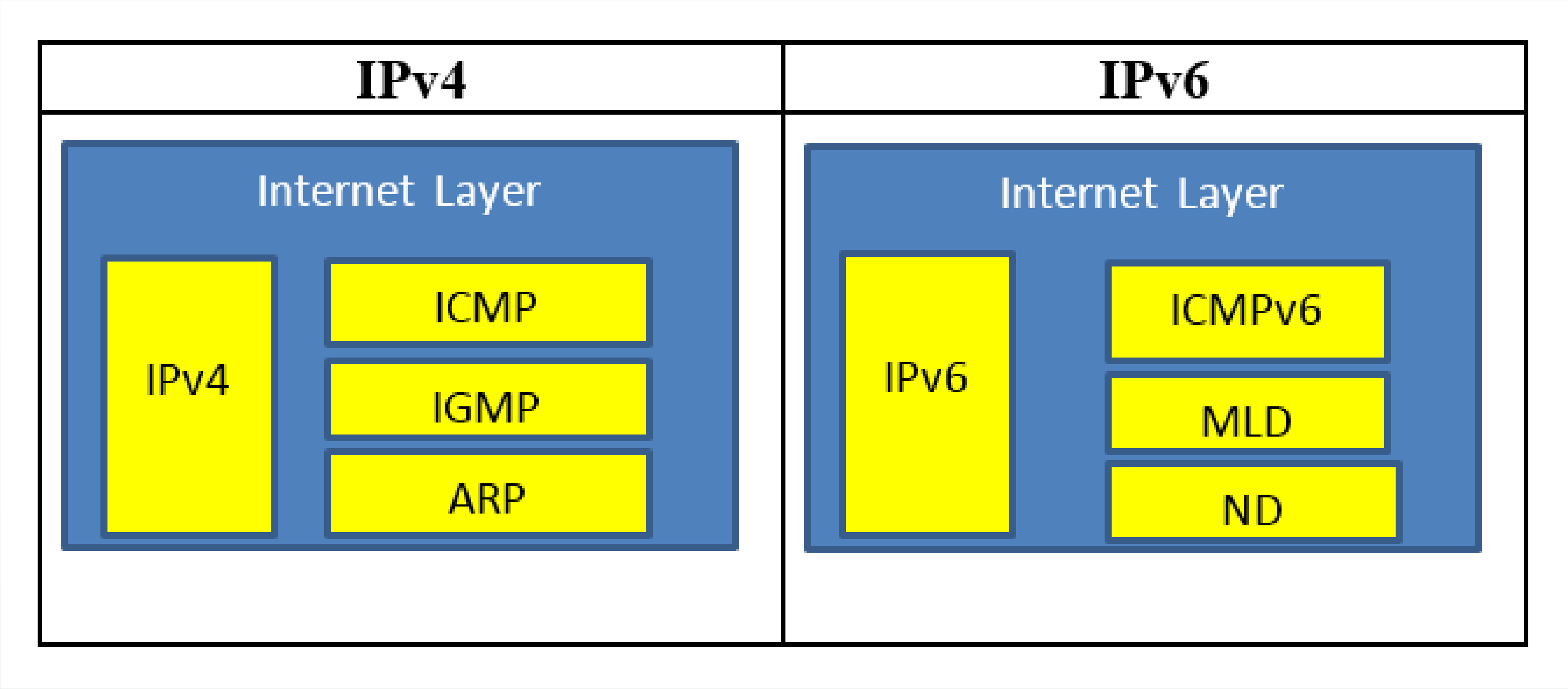


3. Multicast



IPv6

- Both IPv4 and IPv6 are connectionless protocols



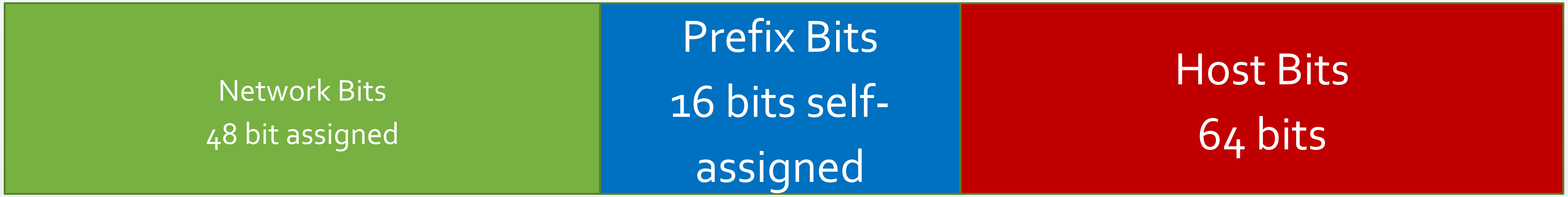
IPv6

1. Inevitability

2. Efficiency

3. Security

IPv6 Address Space



Network Bits
48 bit assigned

Prefix Bits
16 bits self-assigned

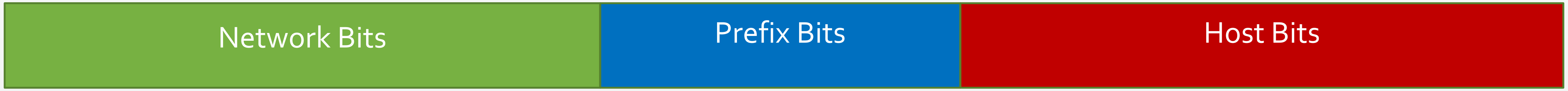
Host Bits
64 bits

- Network address of 48 bits which is assigned by
- Network prefix 16 bits used by businesses to di structure
- Host address 64 bits.
- Total 128 bits – 340 trillion, trillion, trillion addre

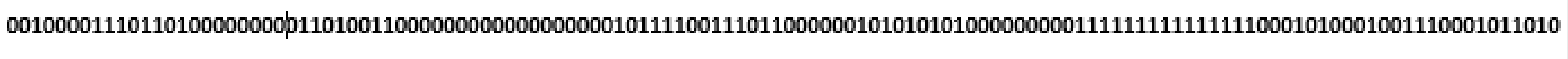
How big is this number?

If you allocated an IPv6 address to every atom on earth, you would still have enough addresses for about 100 more Earths. Before we run out of addresses the human race will be extinct!

IPv6 Address Space

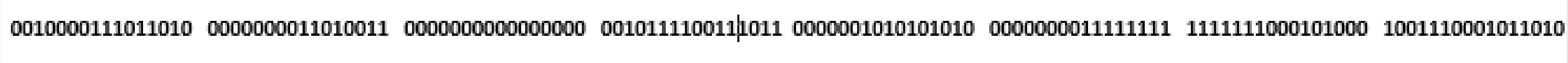


- 128-bit address is divided along 16-bit boundaries. Each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is known as colon-hexadecimal
- IPv6 Address in binary form:



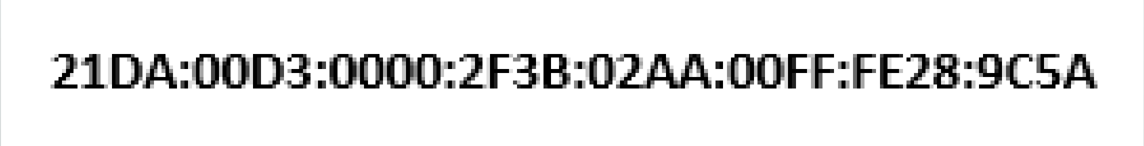
```
00100001110110100000000011010011000000000000000000101111001110110000001010101010000000001111111111111110001010001001110001011010
```

- Divided into 16 bit blocks:



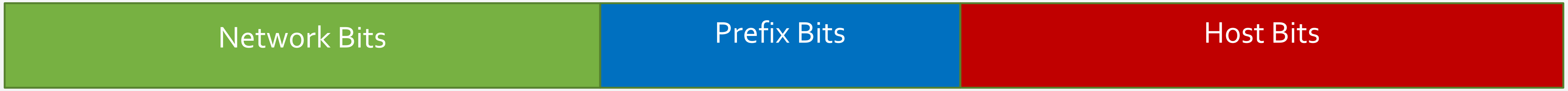
```
0010000111011010  0000000011010011  0000000000000000  0010111100111011 0000001010101010  0000000011111111  1111111000101000  1001110001011010
```

- Each 16 bit block converted to hexadecimal and separated by colons:



21DA:00D3:0000:2F3B:02AA:00FF:FE28:9C5A

IPv6 Address Space - Simplification

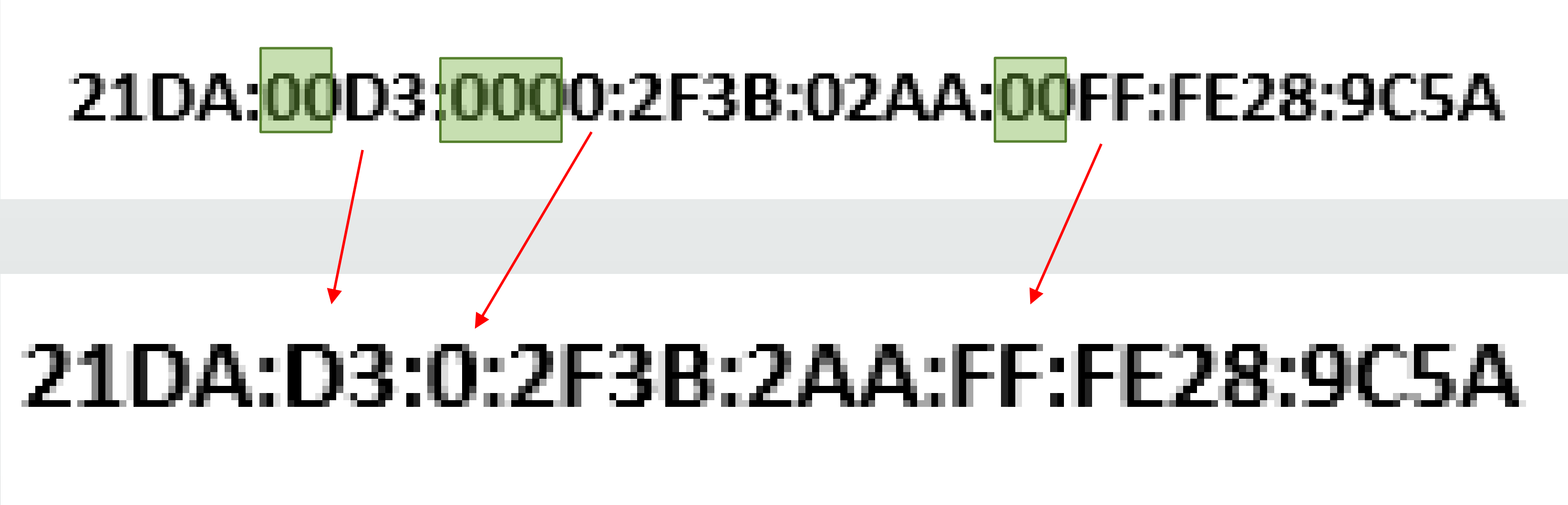


Network Bits

Prefix Bits

Host Bits

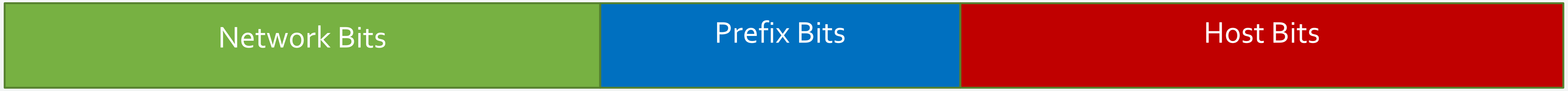
- IPv6 representation removes leading zeros within each 16-bit block
- Each block must have at least a single digit.



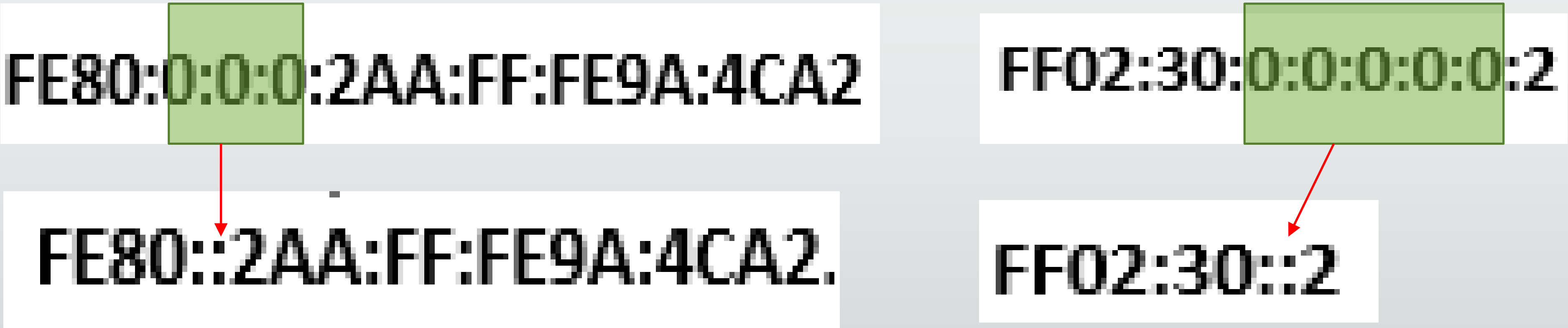
21DA:00D3:0000:2F3B:02AA:00FF:FE28:9C5A

21DA:D3:0:2F3B:2AA:FF:FE28:9C5A

IPv6 Address Space - Simplification



- IPv6 representation also allows for the compression of zeros if the address has long strings of zeros
- Missing zeros represented by a double colon “::”
- Can only be used once in an address



FE80:0:0:0:2AA:FF:FE9A:4CA2

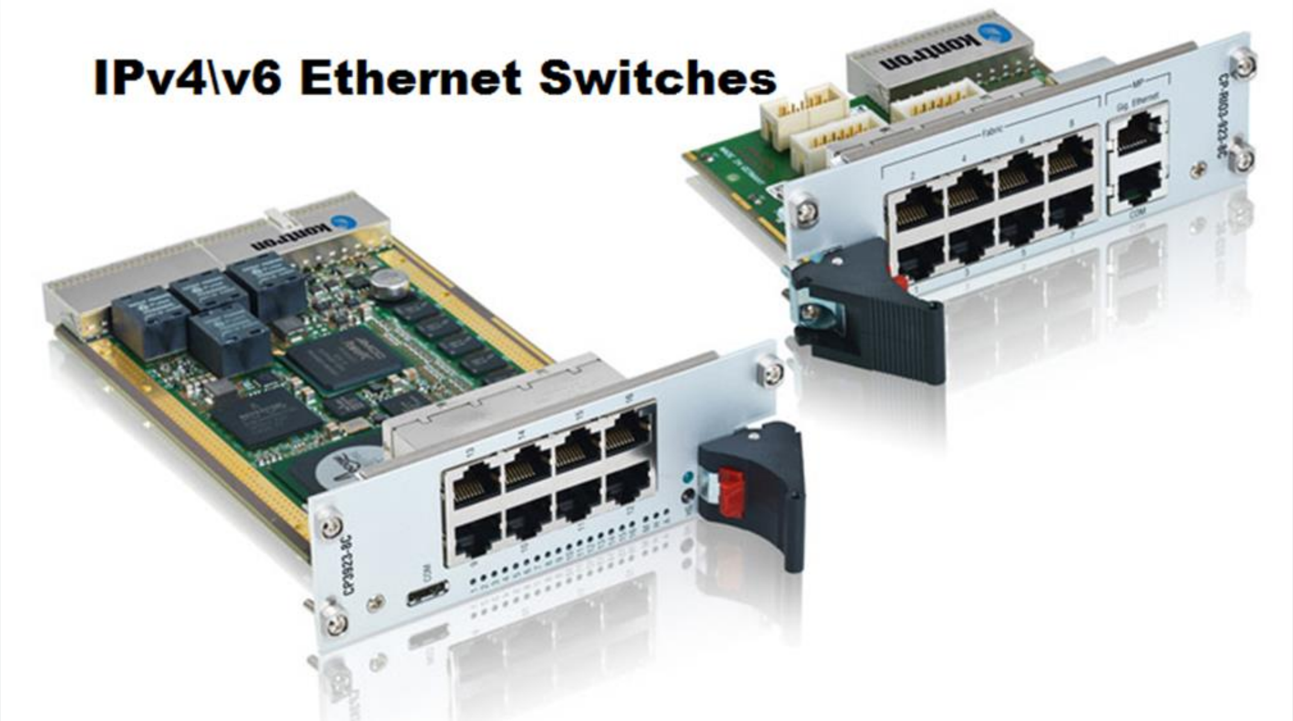
FE80::2AA:FF:FE9A:4CA2.

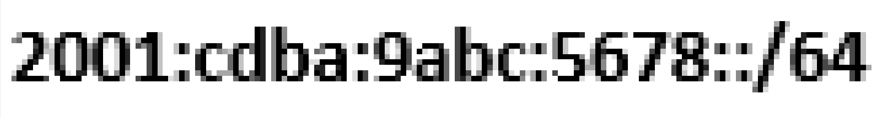
FF02:30:0:0:0:0:0:2

FF02:30::2

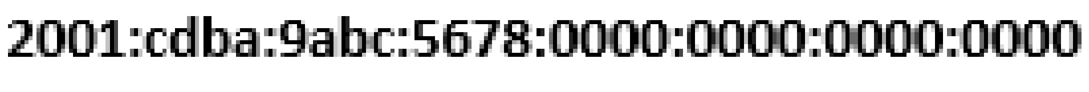
IPv6 Prefix and Dual environments

- 128 bits must show network and host addresses
- Within a network all hosts will have same network prefix
- Prefix indicated by “/” followed by the number of bits used for the network portion of the address

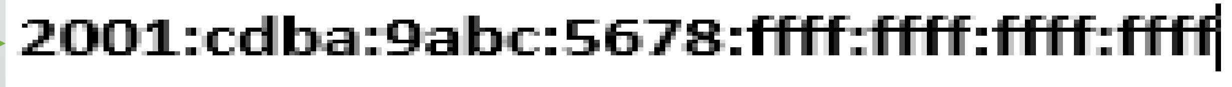



2001:cdba:9abc:5678::/64

- Host portion of address ranges from

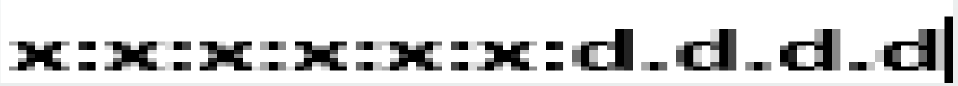

2001:cdba:9abc:5678:0000:0000:0000:0000

→

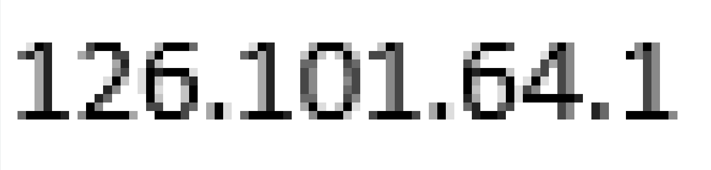

2001:cdba:9abc:5678:ffff:ffff:ffff:ffff

IPv6 Prefix and Dual environments

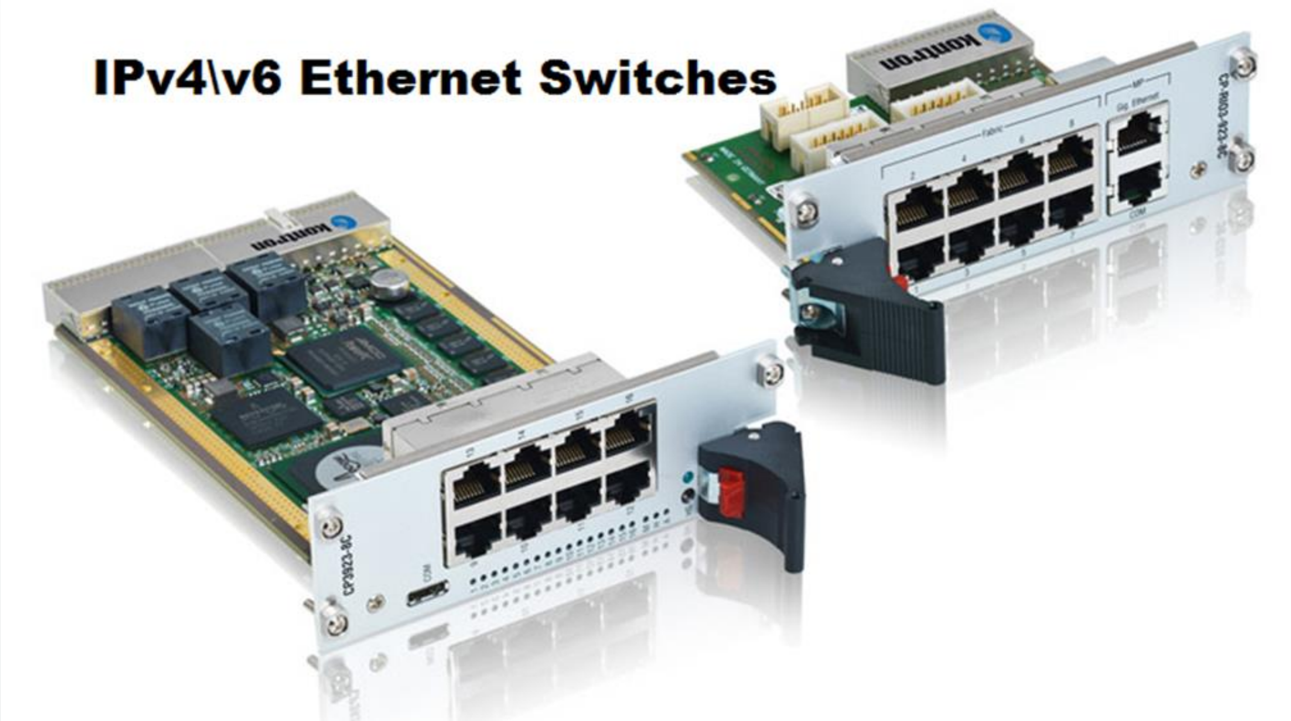
- For dual environments a special syntax is permitted
- “X” represent first 6 high order 16 bit pieces of the address (96 bits)
- “d” represent the decimal values of the four low order 8 bit pieces (32 bits)

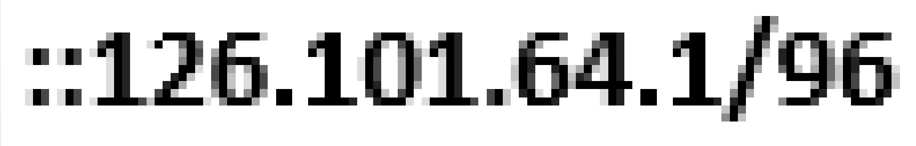


x:x:x:x:x:x:d.d.d.d



126.101.64.1

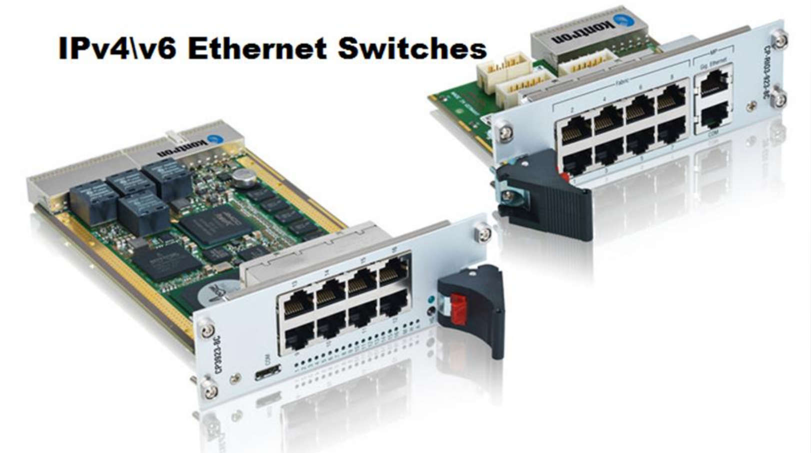


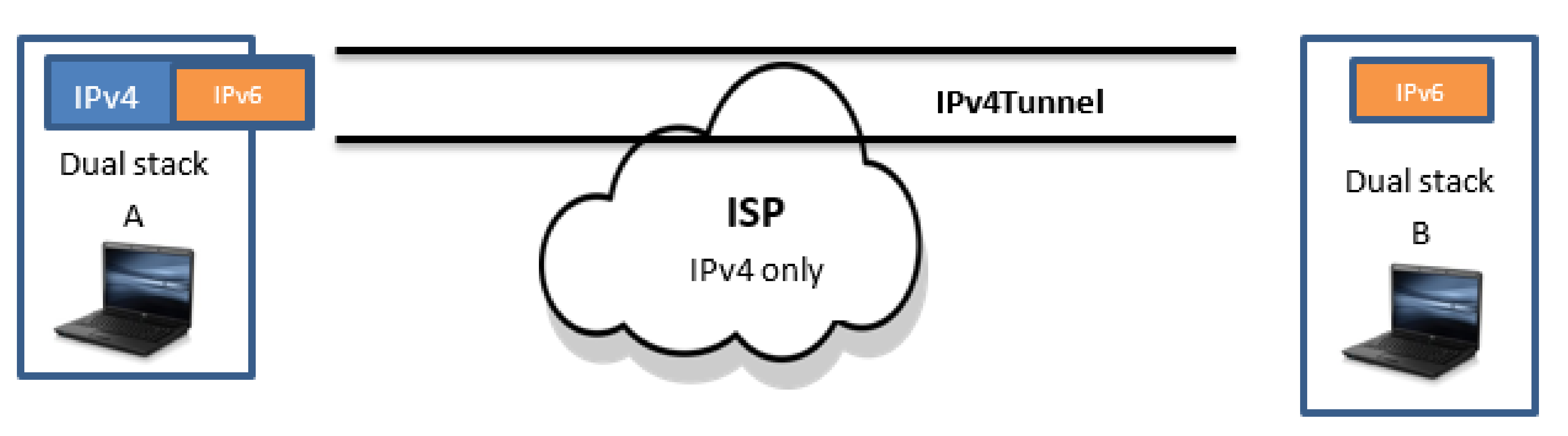


::126.101.64.1/96

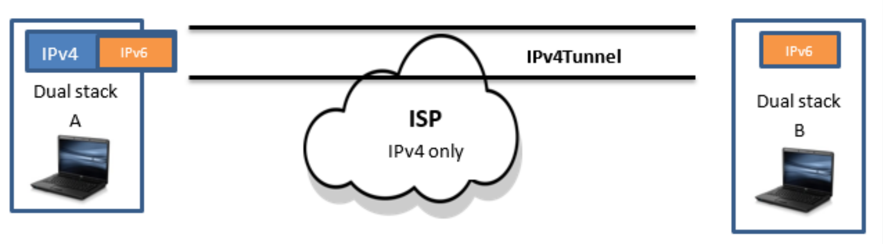
IPv6 Tunnelling: Dual Environments

- Packets travelling on the Internet will meet routers that are not IPv6 capable
- To ensure compatibility IETF created IPv6 over IPv4 Tunneling





IPv6 Tunnelling: Dual Environments



- To determine which protocol to use a router will use DNS records

If IPv4 DNS database record is:
www.senecacollege.ca. IN A 142.204.0.0
If IPv6 address, DNS database record is::
www.senecacollege.ca. IN AAAA FE80:DC28:ffff::1234

If multiple IPv4/v6 addresses, IPv6 addresses will be tried first, and then IPv4 addresses will be tried.

- Connecting to **FE80:DC28:ffff::1234**, then

www.senecacollege.ca IN AAAA FE80:DC28:ffff::1234
www.senecacollege.ca IN AAAA FE80:DC28:ffff::5678
www.senecacollege.ca IN A 142.204.0.0

IPv6

Programming Perceptive

IPv6 – Programming Perspective

1. User Interface Design
2. IP Family Independent
3. Determine IP Family before Creating Socket

IPv6 – Programming Perspective

User Interface Design

- IPv6 address space is much larger and simplification standards require extra programming.

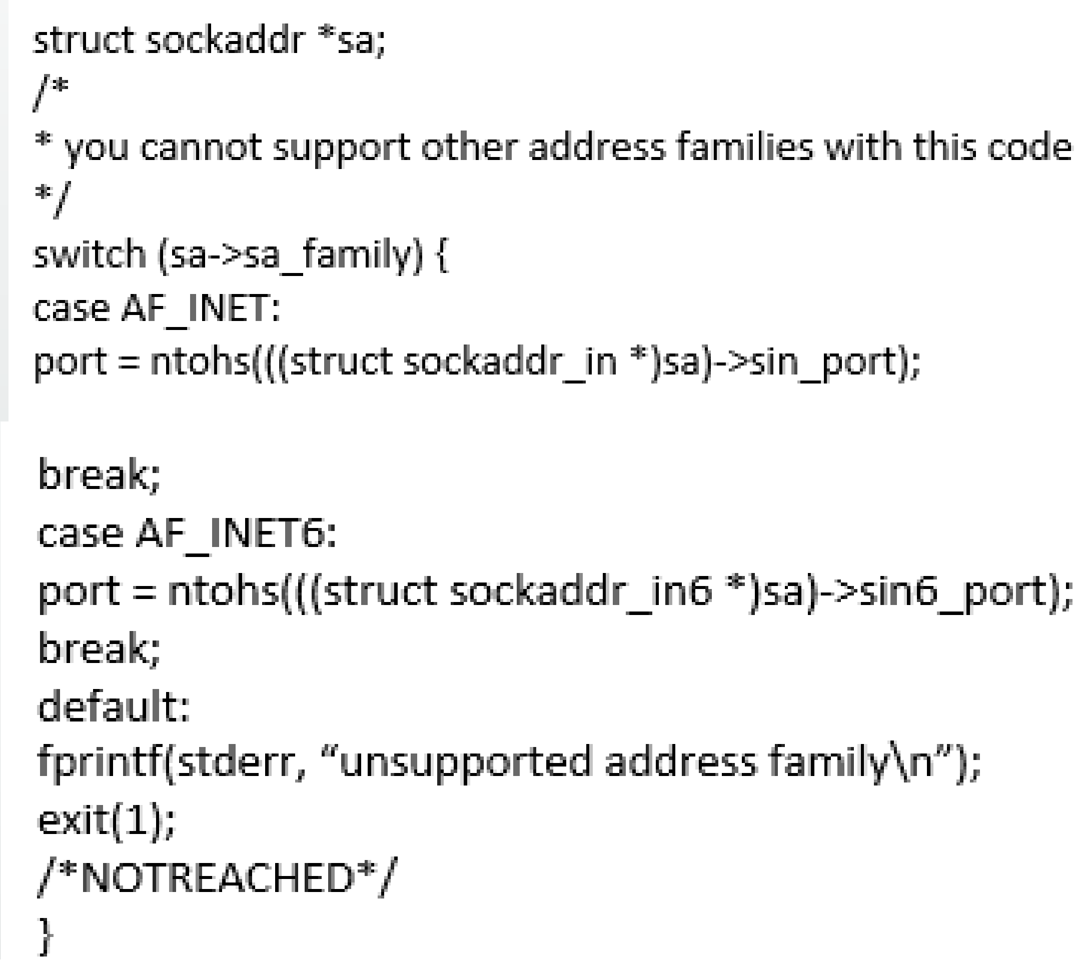
- GUI applications which supply text boxes, for users to change address should be avoided, except in administrative applications

- Programmers need to ask the following questions:

  1. Should number based (IP) or name based (DNS) notation be used?
  2. Should the truncated addresses be used in the interface? The double colon is an optional methods of notation to simplify the address, not a specification.
  3. Does the user need specific parts of the address, such as the subnet prefix, scope identifier or other subfields?
  4. IPv6 specification requires the address to be enclosed in square brackets when part of a URL: For example: **http://[F380:DC28:ffff::1]:80/64**

IPv6 – Programming Perspective

IP Family Independent



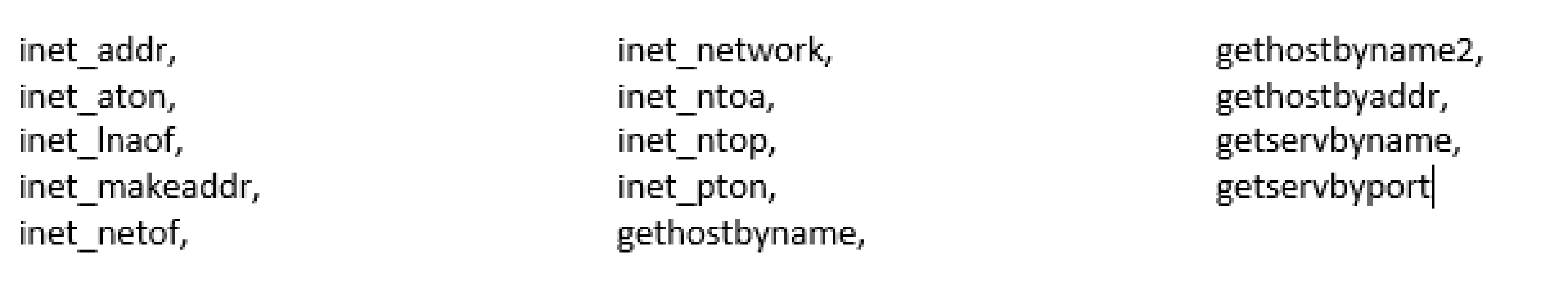
```
struct sockaddr *sa;
/*
* you cannot support other address families with this code
*/
switch (sa->sa_family) {
case AF_INET:
port = ntohs(((struct sockaddr_in *)sa)->sin_port);

break;
case AF_INET6:
port = ntohs(((struct sockaddr_in6 *)sa)->sin6_port);
break;
default:
fprintf(stderr, “unsupported address family\n”);
exit(1);
/*NOTREACHED*/
}
```

Avoid:

1. Hardcoding the address family will make interoperability difficult.
2. Family dependent applications will need to be changed if new protocol developed
3. A device that is IPv4 specific will not know how to tunnel a IPv6 address inside of an IPv4 packet.

IPv6 – Programming Perspective

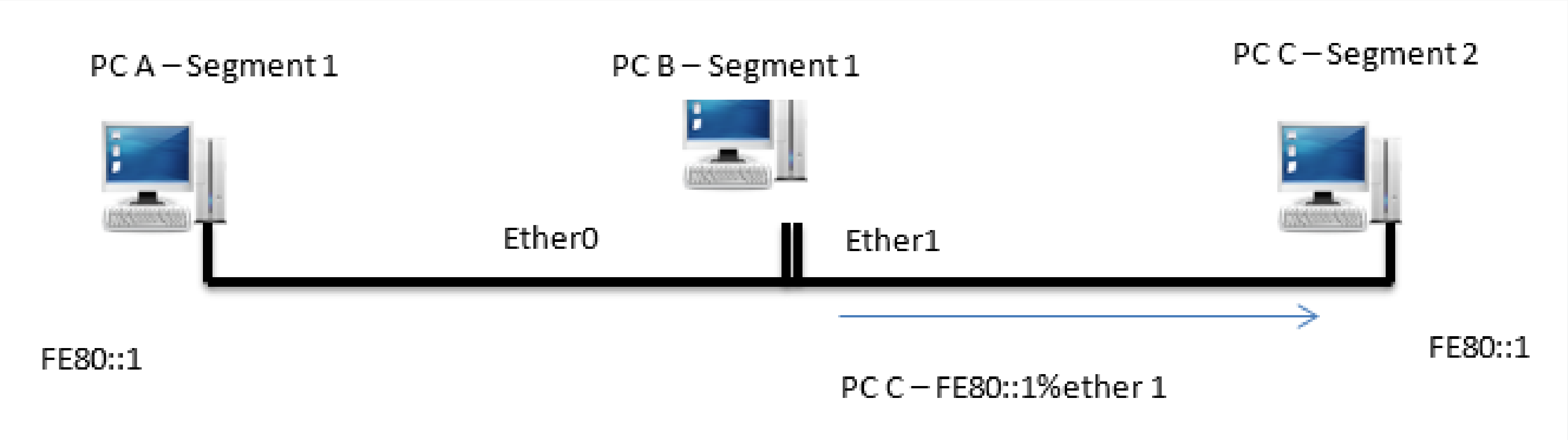
Avoid: Family Specific APIs


inet_addr,
inet_aton,
inet_lnaof,
inet_makeaddr,
inet_netof,
inet_network,
inet_ntoa,
inet_ntop,
inet_pton,
gethostbyname,
gethostbyname2,
gethostbyaddr,
getservbyname,
getservbyport

- Microsoft has a SDK utility called Checkv4.exe which will check for family dependent function calls in your code
- UNIX use grep to search for the above function calls

Scoped IPv6 Addresses

- The IPv6 address is tied to the “interface” not the host
- The 128 bit address alone does not uniquely identify a host

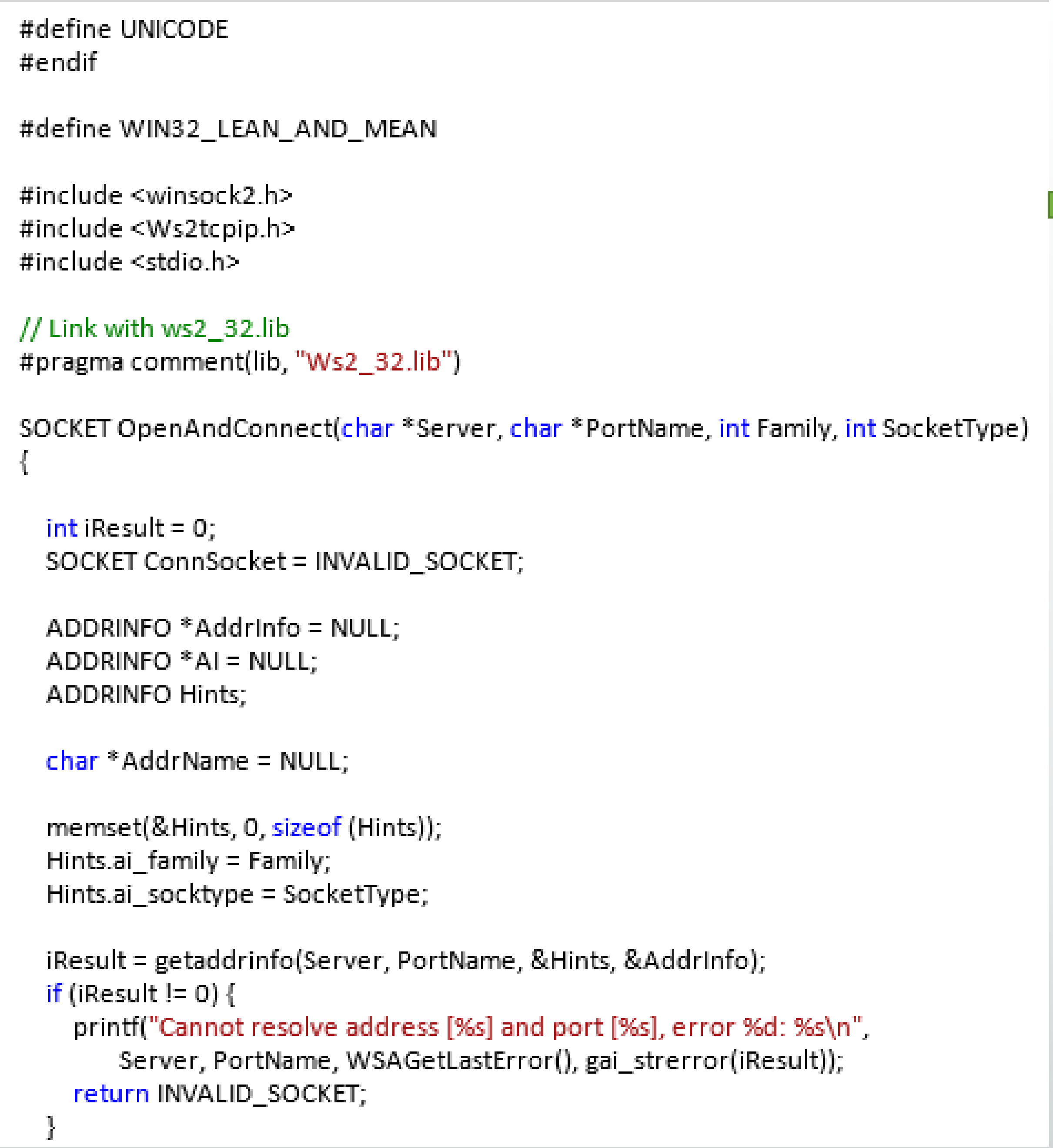


- To communicate with hosts A or C, PC B must use the “link-local” address to specify the NIC to use: **fe80::1%ether1**
- **Sockaddr_in6 has a member sin6_scope_id to add the scope identifier of the interface**

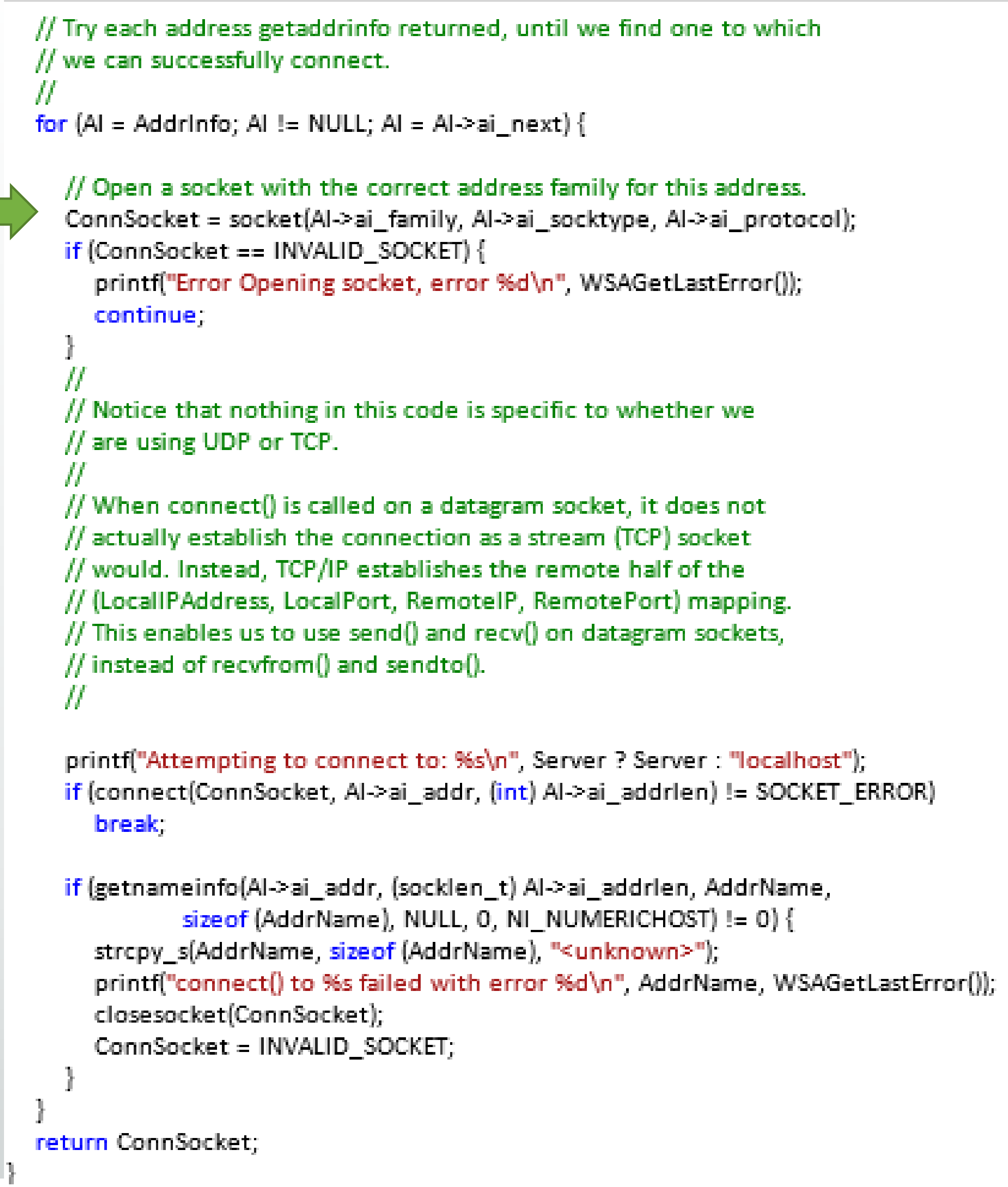
Determine IP Family Before Creating Socket:

1. IPv4 common practice to create socket first using AF_INET address family and then use gettaddrinfo and gethostbyname functions
2. IPv6 the address resolution using getaddrinfo function by be completed FIRST to determine the IP address and address family of the remote host
3. Only then can the socket function be called to open a socket
4. If the name resolution returns bot IPv4 and IPv6 addresses, then separate IPv4 and IPv6 sockets must be used

Windows applcations can use the agnostic **WSAConnectByName**



```
#define UNICODE
#endif

#define WIN32_LEAN_AND_MEAN

#include <winsock2.h>
#include <Ws2tcpip.h>
#include <stdio.h>

// Link with ws2_32.lib
#pragma comment(lib, "Ws2_32.lib")

SOCKET OpenAndConnect(char *Server, char *PortName, int Family, int SocketType)
{

    int iResult = 0;
    SOCKET ConnSocket = INVALID_SOCKET;

    ADDRINFO *AddrInfo = NULL;
    ADDRINFO *AI = NULL;
    ADDRINFO Hints;

    char *AddrName = NULL;

    memset(&Hints, 0, sizeof (Hints));
    Hints.ai_family = Family;
    Hints.ai_socktype = SocketType;

    iResult = getaddrinfo(Server, PortName, &Hints, &AddrInfo);
    if (iResult != 0) {
        printf("Cannot resolve address [%s] and port [%s], error %d: %s\n",
               Server, PortName, WSAGetLastError(), gai_strerror(iResult));
        return INVALID_SOCKET;
    }
```



```
    // Try each address getaddrinfo returned, until we find one to which
    // we can successfully connect.
    //
    for (AI = AddrInfo; AI != NULL; AI = AI->ai_next) {

        // Open a socket with the correct address family for this address.
        ConnSocket = socket(AI->ai_family, AI->ai_socktype, AI->ai_protocol);
        if (ConnSocket == INVALID_SOCKET) {
            printf("Error Opening socket, error %d\n", WSAGetLastError());
            continue;
        }
        //
        // Notice that nothing in this code is specific to whether we
        // are using UDP or TCP.
        //
        // When connect() is called on a datagram socket, it does not
        // actually establish the connection as a stream (TCP) socket
        // would. Instead, TCP/IP establishes the remote half of the
        // (LocalIPAddress, LocalPort, RemoteIP, RemotePort) mapping.
        // This enables us to use send() and recv() on datagram sockets,
        // instead of recvfrom() and sendto().
        //

        printf("Attempting to connect to: %s\n", Server ? Server : "localhost");
        if (connect(ConnSocket, AI->ai_addr, (int) AI->ai_addrlen) != SOCKET_ERROR)
            break;

        if (getnameinfo(AI->ai_addr, (socklen_t) AI->ai_addrlen, AddrName,
                    sizeof (AddrName), NULL, 0, NI_NUMERICHOST) != 0) {
            strcpy_s(AddrName, sizeof (AddrName), "<unknown>");
            printf("connect() to %s failed with error %d\n", AddrName, WSAGetLastError());
            closesocket(ConnSocket);
            ConnSocket = INVALID_SOCKET;
        }
    }
    return ConnSocket;
}
```

Summary

1. The Internet layer is responsible for determining the IP address of the remote host and adding the sending and receiving IP addresses to the header. As the message is forwarded to the receiving host the Internet layer makes routing decisions

2. The depletion of the IPv4 address space led to the creation of private addresses, CIDR, DHCP and NAT technologies which have created the present network landscape. Although IPv6 was launched in 2012, if is expected that IPv4 will be around for a long time in North America. Companies should change to IPv6 for better performance and security

3. For a smooth transition to IPv6 it is mandated that all hardware be able to handle IPv4/v6 dual environments. These devices will be able to Tunnel IPv6 addresses inside IPv4 addresses for IPv4 specific devices. . IPv6 addresses are assigned to an interface and not the host, therefore there is a link-local address which allows the 128 address to be assigned with a scope identify to avoid ambiguity.

4. IPv6 address space uses 3 types of addressing unicast, multicast and anycast. The address space is 128 bits long and is in hexadecimal format. A number of techniques are allowed to simplify the address.

5. The use of IPv6 has made fundamental changes in how programmers design user interfaces, and write socket programs. Given the size and complexities of displaying IPv6 addresses programmers must decide if they want to use a name based system. Also, in writing programs the socket address type must be determined before opening the socket.