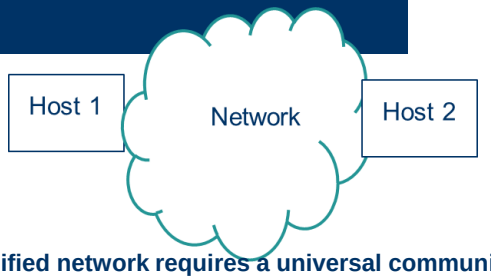
Introducing TCP/IP: Host Identifiers



A unified network requires a universal communication service.

Multiple levels of naming

- Highest level: human readable names
  - Domain names names like www.google.com
- IP level: IP addresss
  - Two types : V4 or V6
- Lowest level: machine addresses
  - Format depends on the lin,/physical layer
  - Many physical layers conform to the Ethernet standard
    - Ethernet frame includes a 6 octet source MAC address and 6 octet dst MAC address in the MAC header

- In TCP/IP terminology, a Host is a computer that communicates with another computer using TCP/IP
- The network between Host 1 and Host 2 can range from
  - directly connected by a physical cable (serial cable or USB cable).
    - This type of network is 'point-to-point'
  - Directly connected to the same Local Area Network (like Ethernet)
    - This type of network is 'broadcast' network- many Hosts can attach to the network.
    - When one Host sends, all Hosts receive.
      - A Host should 'accept' the frame only if the destination MAC address is 1)that of Host 2; 2)A broadcast (dst MAC address of all 1's)
  - Indirectly connected - this means there is >1 network which implies there is a router
  - The Internet - implied indirectly connected. Host 1 generates frames over each link, contained in each frame is an IP datagram.
    - An IP network is able to deliver an IP datagram to the destination network specified in the IP header's destination network field
- A TCP/IP network is a number of autonomous networks that operate in a manner presenting a unified network to end users

Host Identifiers

- Host identifiers:
  - Name: identifies what the object is
  - Address: identifies where it is
  - Route: identifies how to get there
    - Three methods for routing
      - Circuit switching
      - Packet switching
      - Message based (email, pub/sub, future Internet extensions for named data networking)
- Network identifiers:
  - Usually by domain (Clemson's network implies all IPs assigned to clemson)
  - Or by network prefix notation : 130.127/16 where the /xx specifies how many bits define the network prefix
- How does an address relate to a host ?
  - Historically, one address maps to one interface in a Host
  - A Host that has >1 interface is multihomed
    - Routers are mutlihomed (otherwise not very functional !!!)
  - Today, one IP address might map to many machines (we will talk about DNS load balancing)
  - A Host can have multiple interfaces
    - Historically, a Host uses only 1 interface at a time
    - Today….it is possible to use >1 interface at a time (SCTP, Quic, MPTCP0

Original IP V4 Classful Addressing Scheme

- Original scheme was classful:
  Example: 130.127.48.4 This is in dotted quad notation.
  Binary: 10000010.01111111.00110000.00000100
  Hex : 0x827F3004
  Unsigned int: 2,189,373,444
  Class? Binary format….first 3 bits '100' define it as class B

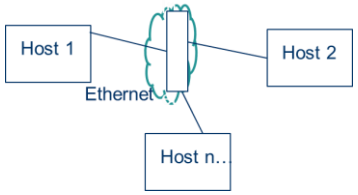
•An address encodes the identification of the network as well as the host (network id, host id)

•Knowing the class, (A,B, C) tells us how many bits are in the network id

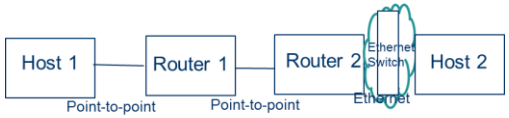
- <130.127> <48.4>
- Clemson's network is identified as 130.127/16 (network prefix format notation)
- The Hostid is the number formed by the host id bits:
  - Host id hex : 0x4804
  - Unsigned int: 18,436

IP Addresses V4

•Directly Connected Hosts….



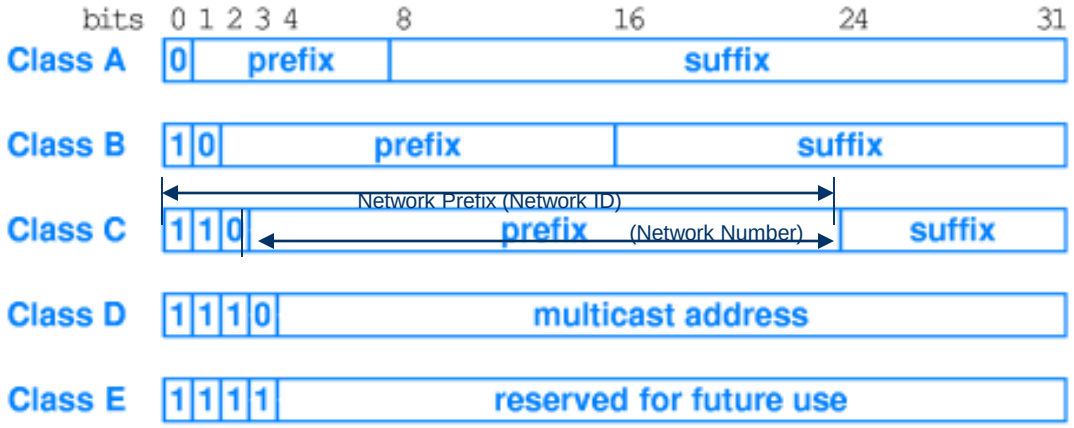
Indirectly Connected



•Three types of addresses?

- •Unicast: one-to-one
  - E.g., ping 8.8.8.8 - our host communicates via a socket with one other host with its unicast IP address of 8.8.8.8
- •Broadcast: one-to-all (where the all is 'all Hosts directly connected to a network')
  - Local: destination address all 1's: 255.255.255.255
  - Network directed : class "C" address, network prefix: 192.168.1/24
    - Network directed broadcast….a host on a different network (e.g., sending host IP : 192.168.1.1 issues ping 192.168.2.255 : all hosts on 192.168.2/24 respond)
- •Multicast: one-to-group - requires special IP protocol to allow Hosts to join multicast groups
  - But uses dedicated class D IP addresses

Original Classful Addressing Scheme



Original address scheme was classful:

- Class A for large networks (>64K hosts)
- Class B for medium networks (>256 hosts)
- Class C for small networks (<256 hosts)
- Class D for multicast
- Class E reserved

Note: classful addressing is considered obsolete- the Internet uses CIDR or classless interdomain routing
Basically, addresses must come with information that indicates the size of the network prefix.

Classful Address Ranges

Class	Lowest Address	Highest Address
A	1.0.0.0	126.0.0.0
B	128.1.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

Dotted Decimal Class Address Ranges

Original Classful Addressing Scheme

- IP V4 address: 32 bits:
  - Total address space: 2exp32 or 2.4 Billion
- Original scheme was classful:
  Example: 130.127.48.4 This is in dotted quad notation.
  Binary: 10000010.01111111.00110000.00000100
  Hex : 0x827F3004
  Unsigned int: 2,189,373,444
  Class? Binary format….first 3 bits '100' define it as class B

- How many networks are there?
  - Class B: 16 bits define the network prefix…but the first 2 bits used
    - 2exp14 minus the all '1s' and all '0s' = 8190
  - Classs A: How many class A networks? 2exp15 or 127
- How many Hosts can operate on a single class B network?
  - 2exp16 minus all '1s' and all '0s' : 65535 -2

Special Addresses

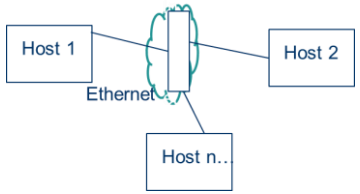
Host Addresses with Special Meaning

- All 0's and all 1's – imply this host and all hosts
- Loop Back : 127.x.x.x (e.g., 127.0.0.1)
  - What address class? Class A!!!!
- Private address space:
  - RFC 1918 defines certain address ranges for private use.
    - 10.0.0.0 - 10.255.255.255 (Class A space)
    - 172.16.0.0 - 172.31.255.255(Class B space)
    - 192.168.0.0 - 192.168.255.255 (Class C space)

Special Addresses

- A Host can never be assigned the IP: 192.168.1.255
- There can never be a class A network of 127.1.1.1 for two reasons
  - First, it's all 1's
  - Second, it implies local host!!
- There can never be a class B network : binary: 10111111.11111111
  - Hex: 0xBFFF
  - Prefix: 191.255/16
- What happens if you png 0.0.0.0
  - An error, or a valid ping to local host

Broadcast Addresses and ARP



- Host 1 and Host 2 communicate directly
- Host 1 needs to know Host 2's IP address AND MAC address
- Address Resolution Protocol (ARP) – allows Host 1 to dynamically learn
  - Try 'arp –a'
- Let's say Host 1 issues ping –c 1 Host 2 (assume all arp caches empty)
- Host 1's IP stack creates the IP datagram that contains the ping message.
  - To build the Ethernet frame it needs Host 2's MAC address
- Host 1 transmits an ARP frame – dst MAC: all 1's (broadcast)
  - Payload: ARP msg called 'who is' .
  - The Msg contains the Host 2's IP address, Host 1's IP address and MAC address.
- All Hosts on that network receive the ARP message and passes the message to the instance of ARP running on that machine.
- Only Host 2 responds with an ARP 'I am" msg that contains it's IP address and its MAC address. This is UNICAST to Host 1
- All Hosts run arp – it maintains a local cache (entries last 20 minutes ??)
  - Host 2 caches the mapping for Host 1
  - Host 1 caches the mapping for Host 2
  - All other Host's on the network that might have received Host 1's 'who is' will NOT cache the mapping
- Question: for that ping –c1 Host 2 (from Host 1). Assuming all arp caches initially empty, how many frames all together are sent:
  - Answer: ARP 'who is', ARP 'I am', ping echo IP datagram, ping reply IP datagram = 4

Broadcast Addresses and ARP

•Let's say Host 1 issues ping –c 1 Host 2 (assume all arp caches empty)
•But….Host 1 and Host 2 are on different networks, 1 router is in between.
•Indirect forwarding at Host 1 – it realizes that the dst IP of Host 2 is a different network and therefore it MUST forward to its configured first hop router (and Host 1 MUST know the IP of Router 1)

Host1 Router 1 Host 2
----→ARP who is ? Broadcast
←-------ARP I am unicast
----ping ICMP Echo IP (dst IP: Host 2, frame dst: router 1's MAC)
Forwards the IP packet to Host 2 but needs to learn Host 2's MAC
-----ARP 'who is' -→
<=-=====ARP 'I am'
---------IP datagram -→
←----------IP datagram(ping reply) (HOST 2 has learned Router 1's MAC)
←-----------IP datagram (ping reply ---

•Total frames sent by all 3 nodes: 8 frames
•Question if Host 1 issues ping –c 2 Host2, how many total frames sent?
- •8 for the first iteration
- •4 for the second iteration since ARP is not required (all MAC addresses are in local caches)
- •Total: 12 frames

Subnet/Supernets

- Limitations of original IP v4 classful addressing
  - Assumes the world is ok with 3 size networks
  - requires a unique network prefix for each physical interface.
  - Does not readily support mobile nodes
  - IP addresses of 32 bits do not provide enough!!!

•Extensions

- •Subnetting:
  - Allow a classful network to be subdivided into any size subnetwork.
- •Supernetting:
  - Allows multiple classful IP addresses to be aggregated together. In general, a network can be a block of contiguous IP addresses (aligned by power of 2)
- •IP V6 – extends IP address from 32 bits to 128 bits

Subnets: break up classful net into smaller subnets

- Allow a classful network to be subdivided into any size subnetwork.
  - Routers and Hosts must know the subnet mask
    - E.g., 192.168.1/24 – let's create 3 subnets
      - Subnet mask 255.255.255.xyz0000 (last octet shown in binary)
      - The xyx form 8 possible subnet ids (but let's assume all 1's and all 0's not valid – so 6 valid)
        - 001, 010, 011,100,101,110
      - Subnet 1: 192.168.1.00100000/27
        - 192.168.1.32/27
      - Subnet 2: 192.168.1.96/27
      - …
      - Subnet 6: 192.168.1.1100 0000/27
        - 192.168.1.192/27

Supernets- aggregate blocks of addresses

- Supernet 4 class c addresses:
  - 192.168.0/16- 192.168.3/16: aggregated address:
    - 192.168.00000011. *
      - 192.168.0/22
  - So the network address 192.168.0/22 represents
  - 192.168.000000xx. *
    - 192.168.0/24…192.168.3/24 4 class C networks (but assume all 0's 192.168.0/24 not valid….so 3 class c's)
- Big advantage- makes it much more efficient for the global routing protocols to identify paths for aggregated networks (rather than having to specify each classful network)
  - Routing protocols call this classless interdomain routing or CIDR
  - An aggregated IP address AKA:
    - Supernetted address
    - CIDR address

IP V6

- Refer to the PDF “IPv6 Basics, a chapter from a Cisco online document”
- 8 sets of 16bit hex values separated by ‘:’ s
  - 2001:0db8:1234:5678:9abc:def0:1234:5678
- Leading zero’s can be omitted, consecutive 0’s represented by ‘::’
  - 2001:0db8:0000:130F:0000:0000:087C:140B can be replaced by
  - 2001:0db8:0:130F::87C:140B
- CIDR applies
  - 2001:db8:12::/64 represents a IP V6 aggregated network prefix
- Usually the network prefix is the first 64 bits, the host id is the last 64 bits
  - The network prefix is usually set by the organization,
  - The host id can be set by
    - Randomly
    - DHCPv6
    - Extended Unique ID (EUI-64) format - extends the MAC interface address to 64 bits by adding FFFE in the middle 16
- Broadcast not used….multicast is used instead!

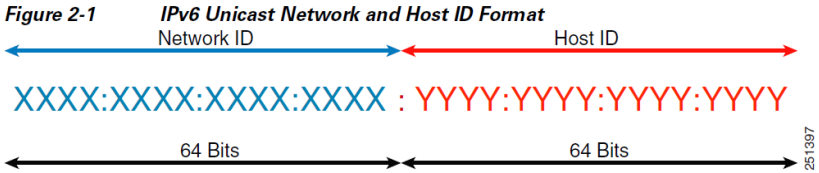


Figure 2-1 IPv6 Unicast Network and Host ID Format

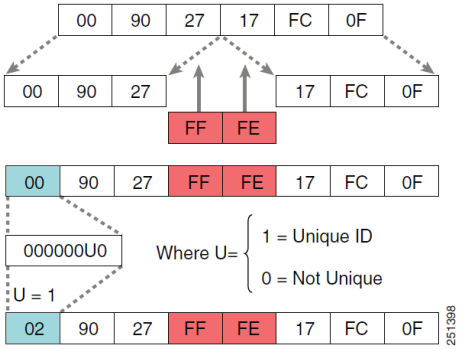


Figure 2-2 Conversion of EUI-64 MAC Address to IPv6 Host Address Format

IP V6 - three scopes

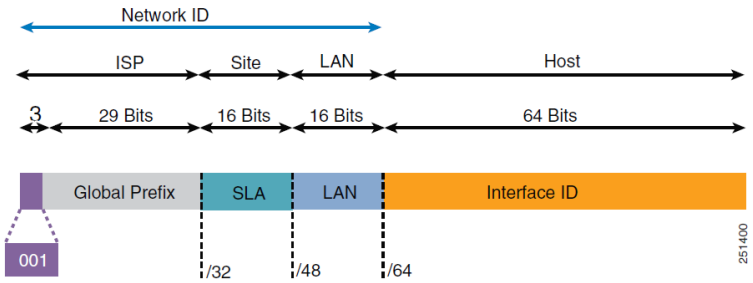
- Gobal Unicast Address
  - Routable over the Internet
  - Id'ed by the three high level bits set to 001 (2000::/3)
  - Example:
  - **2001:0DB8:BBBB:CCCC:0987:65FF:Fe01:2345**
- Unique Local
  - Similar to IP V4 private addresses
  - Used for local communications…
  - Will not route over the Internet
  - Identified: FD00::/7
  - Example:
  - **FD00:aaaa:bbbb:CCCC:0987:65FF:FE01:2345**

- Link Local
  - Mandatory addresses for communicating btween two IPV6 devices on same link
  - Auto assigned by device
  - Not routable
  - Identified: first 10 bits: FE80
  - **FE80:0000:0000:0000:0987:65FF:FE01:2345**
    - Short hand: **FE80::987:65FF:FE01:2345**

Figure 2-3 IPv6 Address Scopes

IP V6 – global scope

- Global Unicast Address
  - Routable over the Internet
  - Id'ed by the three high level bits set to 001 (2000::/3)
- Figure illustrates the portions of a global scope
  - Global routing prefix assigned to a service provider by IANA
  - Site level Aggregate (SLA) or subnet id, assigned to a customer by a service provider



Example….

- Googles IPV6 Name server (ip v4 is 8.8.8.8)
  - **2001:4860:4860::8888**
- Koala5.cs.clemson.edu –ifconfig shows:
  - **eno1 Link encap:Ethernet HWaddr 98:90:96:d7:81:a0**
  - **inet addr:130.127.48.106 Bcast:130.127.49.255 Mask:255.255.254.0**
  - **inet6 addr: 2620:103:a000:401:9a90:96ff:fed7:81a0/64 Scope:Global**
  - **inet6 addr: fe80::9a90:96ff:fed7:81a0/64 Scope:Link**
  - **UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1**
  - **RX packets:32491376 errors:0 dropped:0 overruns:0 frame:0**
  - **TX packets:15173215 errors:0 dropped:0 overruns:0 carrier:0**
  - **collisions:0 txqueuelen:1000**
  - **RX bytes:23183098792 (23.1 GB) TX bytes:11711322477 (11.7 GB)**
  - **Interrupt:20 Memory:f7100000-f7120000**

```
ping6 -c 2 -n google.com
PING google.com(2607:f8b0:4002:804::200e) 56 data bytes
64 bytes from 2607:f8b0:4002:804::200e: icmp_seq=1 ttl=54 time=3.23 ms
64 bytes from 2607:f8b0:4002:804::200e: icmp_seq=2 ttl=54 time=3.16 ms

--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 3.163/3.197/3.232/0.066 ms
```

Abstract Socket Address Structure

Generic Address structure defined in <sys/socket.h>

```
struct sockaddr {
        u_char    sa_len;          /* total length */
        u_char    sa_family;       /* address family */
        char      sa_data[14];     /* actually longer; address value */
};
```

Address Structure IPv4

For sa_family = AF_INET use the following (defined in <netinet/in.h>):

```
struct in_addr {
        u_int32_t s_addr;
};
struct sockaddr_in {
        u_char    sin_len;
        u_char    sin_family;    //This is the address family- not the same as PF
        u_short   sin_port;
        struct    in_addr sin_addr;
        char      sin_zero[8];
};

//Note: the 'in' implies 'Internet'
```

Address Structure IPv6

```
struct sockaddr_in6 {
                                                        6)
        in_port_t sin6_port; // Address port (16 bits)
        uint32_t sin6_flowinfo; // Flow information
        struct in6_addr sin6_addr; // IPv6 address (128 bits)
        uint32_t sin6_scope_id; // Scope identifier
}

struct in6_addr {
        unsigned char  s6_addr[16];
};
```

Note that an IPv6 address consumes more octets than what the generic InetAddr allows….that is ok as long as the code supports this.

Helper Function : to convert between IP addresses and dotted decimal

- Inet_pton and inet_ntop:
  - Replacements for inet_aton and inet_ntoa.
  - [redacted]ic"
  (binary)
  - Supports IPV4 and IPV6
- Example:
  - **int rtnVal = inet_pton(AF_INET, servIP, &servAddr.sin_addr.s_addr);**
  - if (rtnVal == 0)
    - DieWithUserMessage("inet_pton() failed", "invalid address string");
  - else if (rtnVal < 0)
    - DieWithSystemMessage("inet_pton() failed");
  - servAddr.sin_port = htons(servPort); // Server port

- See the helper routine setuptcpclientsocket ( ) in TCP client utility

Helper Functions : writing network programs that work for either V4 or V6 addresses

- Donahoo recommends using a method that supports either V4 or V6
- **int rtnVal = getaddrinfo(addrString, portString, &addrCriteria, &addrList);**
- We send a string, which would be a domain name or a string in dotted quad notation
  - The router Issues the getAddrInfo() passing hints if it was all IP addresses assigne. The results area linked list of adder info.

Helper Functions : to convert between IP addresses and names :

```
char *portString = argv[2];   // Server port/service

// Tell the system what kind(s) of address info we want
struct addrinfo addrCriteria;                   // Criteria for address match
memset(&addrCriteria, 0, sizeof(addrCriteria)); // Zero out structure
addrCriteria.ai_family = AF_UNSPEC;             // Any address family
addrCriteria.ai_socktype = SOCK_STREAM;         // Only stream sockets
addrCriteria.ai_protocol = IPPROTO_TCP;         // Only TCP protocol

// Get address(es) associated with the specified name/service
struct addrinfo *addrList; // Holder for list of addresses returned
// Modify servAddr contents to reference linked list of addresses
int rtnVal = getaddrinfo(addrString, portString, &addrCriteria, &addrList);
if (rtnVal != 0)
  DieWithUserMessage("getaddrinfo() failed", gai_strerror(rtnVal));

// Display returned addresses
for (struct addrinfo *addr = addrList; addr != NULL; addr = addr->ai_next) {
  PrintSocketAddress(addr->ai_addr, stdout);
```

Helper Functions : to convert between IP addresses and names : OLD !!!!

gethostbyname, gethostbyaddr : convert between hostnames and IP addresses

```
struct hostent *  gethostbyname (const char *name)

struct hostent {
  char *h_name;   //official name of host
  char **h_aliases; //alias list
  int h_addrtype;   //host address type
  int h_length;      //length of address
  char **h_addr_list;  //list of addresses from name server
}
```

•getservbyname, getservbyport : converts between services and ports

Network Byte Order

•Network Byte Order: concept to isolate a network from machine architectures.

•Big Endian machines: lowest memory has high-order byte.
•Little Endian machines: lowest memory has low-order byte.
•Functions: htons(), htonl(), ntohs(), ntohl()

Network Byte Order is Big Endian. The MSB of an integer gets sent first.

Protocol Data must be in Network Byte Order but User Data does not have to be.

Appendix - figures