UNIX System Programming
Lecture 17: Networking

- Outline
  - Introduction
  - Ethernet
  - TCP/IP Addresses
  - Name Resolution
- Reference
  - man pages: ip (7), getaddrinfo, htonx, ntohx, inet_aton, inet_ntoa, inet_addr
• Although the standard OSI data communication reference model is a 7 layer model, the TCP/IP protocols don't match its structure exactly. The TCP/IP protocols are usually presented in a 4 layer model:

(4) Application Layer (HTTP, FTP, SSH, etc)
(3) Transport Layer (TCP, UDP)
(2) Internet Layer (Routing protocols - IP)
(1) Network Access Layer (ARP, RFC 984)
Note that a separate *physical layer* is not listed. This layer is included in the Network Access layer. TCP/IP protocols can be used on various hardware technologies: Ethernet, FDDI, ATM, serial lines, Packet Radio, etc.

Each layer has its own independent data structures. Each layer encapsulates information from the layer above and adds its own control information (header) to ensure proper delivery.
Lecture 17: Networking

Introduction

- Header and data from upper layers are encapsulated in new datagrams by lower layers.
Lecture 17: Networking
Introduction

- The Network Access layer is usually ignored by application programmers (but not by network device driver programmers). These protocols allow one system to deliver data to another on a directly attached network.

- At this level IP datagrams are encapsulated into the frames that are transmitted on the network and IP addresses are mapped to the physical addresses.
Ethernet is the most common physical layer protocol. There are several types:

- Thicknet (10Base5) uses RG-5 coax (obsolete).
- Thinnet (10Base3) uses RG-58 coax (very rare).
- Twisted Pair (10BaseT) should use Cat-5 cable.
- Fast Ethernet (100BaseT) uses Cat-5 cable.
- Gigabit Ethernet (1000BaseT) uses Cat-5+ cable.
- Wireless Ethernet (WIFI)
Lecture 17: Networking Ethernet

- Each Ethernet card (network interface card or NIC) has a 48 bit hardware address (MAC, physical, or hardware address).
- An Ethernet address can address a single card (unicast), the entire network (a broadcast address), or a group (multicast).
- The Ethernet frame contains several fields in the header including the destination and source (MAC) addresses.
Lecture 17: Networking
IP Addresses

- At the Internetworking level each machine has a 32 bit (4 byte) IP address, usually given in dotted quad notation: 10.58.240.55. Each number in dotted quad notation represents a byte of the address. The numbers are decimal and lie in the range 0 - 255.

- Each address can be broken down into a network part and a host id part.
Lecture 17: Networking
IP Addresses

- In the original class-based addressing method, the IP address space was divided into a few fixed-length address classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>netid</td>
<td>hostid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>netid</td>
<td>hostid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>netid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lecture 17: Networking

IP Addresses

- There could be 127 Class A networks (1.0.0.0 - 127.0.0.0) with $16 \times 10^6$ hosts ($2^{24}$).
- There could be 16,320 Class B networks (128.0.0.0 - 191.255.0.0) with 65,024 hosts.
- There could be $2.1 \times 10^6$ Class C networks (192.0.0.0 - 223.255.255.0) with 254 hosts.
The number of class B addresses was rapidly depleted. Assigning multiple class C addresses to an organization would make the routing tables too large.

The solution was to move to Classless Internet Domain Routing (CIDR). With **CIDR** a bit mask (the network mask) is applied to an IP address to split it into network and host parts.
Here's an example:

**Host Address:** 192.168.110.130  
**Net Mask:** 255.255.255.128

- The network address is 192.168.110.128. The host address above corresponds to the host number 2 (of 127) on this network.
- A host part of all zeros refers to the network and should not be assigned to any host. A host part of all ones is used for a broadcast.
At the TCP/IP level all addressing is done by IP number.

If the network portion of the destination address does not correspond to the local network the packet is passed to a router.

If the network portion of the destination address matches the local network address, it is passed directly to the destination host.
Lecture 17: Networking
Name Resolution

- All datagram addressing is by IP number. How is a name (google.com) converted to an IP address?

- An IP address is obtained either from the /etc/hosts file or by DNS (dynamic name services). A DNS server translates names into addresses.
Lecture 17: Networking
Name Resolution

• As an application programmer, if our computer is properly configured, we don't need to worry about the details. We can use the `getaddrinfo()` routine to convert a name to a binary network address.

• The `inet_ntoa()` routine will convert a network address to dotted-quad notation. `inet_aton()` will convert from dotted quad notation to a (32-bit) binary address.
The `getaddrinfo()` routine replaces `gethostbyname()` which is now obsolete. `gethostbyname()` is used in (lots) of older software, books or documentation.

`getaddrinfo()` allows us to write code that will work with either IPv4 or IPv6. Similarly `inet_ntop()` will work with either IPv4 or IPv6. `inet_ntoa()` works with only IPv4 addresses.
Note binary network addresses should be in “network byte order” (big endian order). Intel processors are little endian and the following will NOT work when trying to convert “10.5.50.100” to a network address:

```c
unsigned int addr =
(10<<24) + (5<<16) + (50<<8) + 100;
```
The preceding example would work correctly on a big endian architecture. To convert 32-bit addresses between host and network order use the `htonl()` and `ntohl()` routines.

These routines do nothing on big endian machines, but should be used for portability.

Pay careful attention to the documentation to determine if a routine needs (or returns) an address in network or host order.
Lecture 17: Networking
Network Comm. APIs

• There are two major APIs for network programming: sockets and LTI/XTI. XTI is not included by default with most Linux distributions (it is available as an open-source library) and is not widely used. We will only discuss sockets.

• An implementation of the Berkeley sockets API under Windows is known as Winsock.
Lecture 17: Networking

TCP and UDP

• We will discuss only the sockets API to the transport layer. (Raw sockets can be used to directly interface to the IP layer.)

• There are two major TCP/IP transport protocols: TCP (Transport Control Protocol) and UDP (User Datagram Protocol). UDP is a simple, unreliable, datagram protocol, while TCP is a sophisticated, reliable, byte-stream (unlimited length) protocol.
Lecture 17: Networking
TCP and UDP

- There is no guarantee that a UDP datagram will reach its destination. To ensure delivery our application must handle time-outs,acknowledgments,retransmissions,etc. UDP is a connectionless protocol.

- TCP provides a connection between two machines. When data is sent via TCP to the other end, it requires an acknowledgment. If one is not received the data will be resent.
Lecture 17: Networking
TCP and UDP

- TCP provides *sequencing*. When a large amount of data is sent via TCP, the data will be sent in segments. If they arrive out of order, they will automatically be reordered.

- TCP provides *flow control*. A TCP peer advertises how many bytes it can accept. This prevents overflow of the receive buffer.

- A TCP connection is *full-duplex*. Data can be sent and received on the same socket.
Lecture 17: Networking
TCP and UDP

- TCP provides *error detection and correction*. If an error is detected in a packet at the receiver, a request is sent to the send for a re-transmission. This is done until the packet is received without error.

- This is all done automatically. The programmer just needs to setup a TCP connection. The TCP layer software takes care of the rest.
Lecture 17: Networking
In Class Exercise

• Modify the `getip.cpp` program to display the IPv4 address of the specified host in dotted-quad notation.

• Try the following:
  
  ./getip localhost
  ./getip csserver
  ./getip csserver.evansville.edu
  ./getip google.com
  ./getip microsoft.com