Internet

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Computer Networks

- **Network**
  - A network is a hierarchical system of boxes and wires organized by geographical proximity
    - LAN (local area network) spans a building or campus.
      » Ethernet is most prominent example.
    - WAN (wide area network) spans country or world.

- **Internet**
  - An internetwork (internet) is an interconnected set of networks
    - The Global IP Internet is the most famous example of an internet.
## Global IP Internet

- Most famous example of an internet.
- Based on the TCP/IP protocol family.
  - IP (Internet Protocol):
    » Provides basic naming scheme and unreliable delivery capability of packets (datagrams) from host-to-host.
  - UDP (Unreliable Datagram Protocol)
    » Uses IP to provide unreliable datagram delivery from process-to-process.
  - TCP (Transmission Control Protocol)
    » Uses IP to provide reliable byte streams from process-to-process over connections.
- Accessed via a mix of Unix file I/O and functions from the **Sockets interface**.
Network Protocols (1)

The notion of an internet protocol

- How is it possible to send bits across incompatible LANs and WANs?
- Solution: protocol software running on each host and router smooths out the differences between the different networks.
- Implements an internet protocol (i.e. set of rules) that governs how hosts and routers should cooperate when they transfer data from network to network.
  - TCP/IP is the protocol for the global IP Internet.
Network Protocols (2)

- OSI reference model

Diagram:

- Application
- Presentation
- Session
- Transport
- Network
- Data link
- Physical

Node A

Node B
Network Protocols (3)

- **Application:**
  - Supporting network applications. (HTTP, SMTP, FTP, etc.)

- **Transport**
  - Process-to-process data transfer.
    - TCP, UDP

- **Network**
  - Routing of datagrams from source to destination.
    - IP, routing protocols

- **Data link**
  - Data transfer between neighboring network elements.
    - PPP, Ethernet

- **Physical**
  - Bits on the wire.
Network Layer (1)

- **Network layer**
  - Given a packet, get it to the other side of a large collection of networks.
  - Implemented in every host, router.
  - Portability: provides an interface that works across heterogeneous networks.
  - Scalability: provides names and routing that works with billions of end hosts.
Network Layer (2)

- **IP addressing**
  - IP addresses form a 2-level hierarchy.
    - Network part + host part
    - Hosts on same network have same prefix.

![IP Addressing Diagram]

- **Class A**: 0.0.0.0 to 127.255.255.255
- **Class B**: 128.0.0.0 to 191.255.255.255
- **Class C**: 192.0.0.0 to 239.255.255.255
- **Class D**: 240.0.0.0 to 247.255.255.255

32 bits
Network Layer (3)

- **IP**
  - Connectionless (datagram) protocol
    - No call setup at network layer.
    - Packets between same host-dest pair may take different paths.
  - Host-to-host
    - IP gives each host a globally unique IP address
  - Best effort service model
    - IP does its best to deliver it.
    - No attempt is made to recover from lost, reordered, duplicated, or corrupted packets.
    - Synthesize reliability at higher levels. (e.g. TCP)
  - Portable
    - A common packet format that gets used on all networks.
    - Invisibly translating, splitting and reassembly packet as it traverses over different physical networks.
    - A global, network-wide address space.
Network Layer (4)

- **IP (cont’d)**
  - **Advantages**
    - No round-trip delay to setup connection.
    - Each packet forwarded independently of last: if switch or link fails, will be routed around it.
    - Resources allocated dynamically, which let each “flow” achieve peak bandwidth of idle link.
  - **Disadvantages**
    - Busy link = unpredictable, wild service fluctuations.
    - Each packet carries full destination address, which makes per packet overhead higher.
    - Packets can be dropped in intermediate nodes.
**Network Layer (5)**

- **IP routing example:**
  - (A to E)
  - Look up network address of E
  - E on different network
  - Routing table: next hop router to E is 223.1.1.4
  - Link layer sends datagram to router 223.1.14 inside link layer frame
  - Datagram arrives at 223.1.1.4
  - continued...

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>Nhops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

![Network diagram showing IP routing example](network_diagram.png)
Transport Layer (1)

Transport layer

- Data transfer between processes.
  (cf) Network layer: data transfer between end systems
- Use “ports”.
- TCP: reliable, in-order unicast delivery
- UDP: unreliable (“best-effort”) unordered unicast or multicast delivery
- Services not available:
  - Real-time
  - Bandwidth guarantees
  - Reliable multicast
Transport Layer (2)

- **UDP (User Datagram Protocol)**
  - Best-effort, process-to-process
    - may be lost or delivered out of order to application
  - Connectionless
    - No handshaking between UDP sender & receiver
    - Each UDP segment handled independently of others
  - No congestion control: UDP can blast away as fast as desired.
  - Lives on top of IP: adds corruption detection and “ports”
  - Ports allow multiple connections; multiple application protocols between the same machines
    - HTTP: 80, TELNET: 23, SMTP: 25, DNS:53, FTP: 21
Transport Layer (3)

- TCP (Transmission Control Protocol)
  - Point-to-point: One sender, one receiver
  - Reliable, in-order byte stream
    - No message boundaries
  - Pipelined
    - TCP congestion and flow control set window size
  - Full duplex data
    - Bi-directional data flow
  - Connection-oriented
    - Handshaking (exchange of control messages) initializes sender, receiver state before data exchange
  - Flow-controlled
    - Sender will not overwhelm receiver.
### Internet Applications

#### Common applications

<table>
<thead>
<tr>
<th>Applications</th>
<th>Application layer protocol</th>
<th>Underlying transport protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail</td>
<td>smtp [RFC 821]</td>
<td>TCP</td>
</tr>
<tr>
<td>Remote terminal access</td>
<td>telnet [RFC 854]</td>
<td>TCP</td>
</tr>
<tr>
<td>Web</td>
<td>http [RFC 2068]</td>
<td>TCP</td>
</tr>
<tr>
<td>File transfer</td>
<td>ftp [RFC 959]</td>
<td>TCP</td>
</tr>
<tr>
<td>Streaming multimedia</td>
<td>proprietary (e.g., RealNetworks)</td>
<td>TCP or UDP</td>
</tr>
<tr>
<td>Remote file server</td>
<td>NFS</td>
<td>TCP or UDP</td>
</tr>
<tr>
<td>Internet telephony</td>
<td>proprietary (e.g., Skype)</td>
<td>Typically UDP</td>
</tr>
</tbody>
</table>
Network Addressing

- **Data link: MAC address**
  - 48 bits (Ethernet)

- **Network: IP address**
  - 32 bits (IPv4), 128 bits (IPv6)
  - Hierarchical: network + host part

- **Transport: Socket**
  - \(<\text{host address, port number}>\)

- **Application: Domain name**
  - Hierarchical
IP Addresses (1)

- Storing IP addresses
  - IP addresses (and other integer values such as port number) are always stored in memory in network byte order (big endian)

```c
/* Internet address structure */
struct in_addr {
    unsigned int s_addr; /* network byte order (big-endian) */
};
```

- Handy network byte-order conversion functions:
  - **htonl()**: long int from host to network byte order
  - **htons()**: short int from host to network byte order
  - **ntohl()**: long int from network to host byte order
  - **ntohs()**: short int from network to host byte order
IP Addresses (2)

- **Dotted decimal notation**
  - By convention, each byte in a 32-bit IP address is represented by its decimal value and separated by a period.
  
  - IP address 0x739198B5 = 115.145.152.181

- **Converting functions**
  - `inet_aton()`: a dotted decimal string to an IP address in network byte order
  
  - `inet_ntoa()`: an IP address in network byte order to its corresponding dotted decimal string

  - “n” denotes network representation. “a” denotes application representation.
Domain Naming System (1)

- Internet domain names

```
  unnamed root
    /     /     /     /     /     /     /     /     /     /
   mil   edu   com   gov   kr   jp   ....
   |     |     |     |     |     |     |     |     |     |
  cmu   berkeley   amazon   ac   co   re   ne   go
   |     |     |     |     |     |     |     |     |     |
  kaist   skku
   |     |     |     |     |     |     |     |     |     |
  adam   cs   icc
  143.248.136.3  143.248.136.2  115.145.152.181
```

First-level domain names

Second-level domain names

Third-level domain names
Domain Naming System (2)

- DNS
  - The Internet maintains a mapping between IP addresses and domain names in a huge worldwide distributed database.
    - Conceptually, programmers can view the DNS database as a collection of millions of host entry structures.
  - Functions for retrieving host entries from DNS:
    - `gethostbyname()`: query key is a DNS domain name.
    - `gethostbyaddr()`: query key is an IP address

```c
/* DNS host entry structure */
struct hostent {
    char  *h_name;       /* official domain name of host */
    char  **h_aliases;   /* null-terminated array of domain names */
    int   h_addrtype;    /* host address type (AF_INET) */
    int   h_length;      /* length of an address, in bytes */
    char  **h_addr_list; /* null-terminated array of in_addr structs */
};
```
Domain Naming System (3)

- Properties of DNS host entries
  - Each host entry is an equivalence class of domain names and IP addresses.
  - Each host has a locally defined domain name **localhost** which always maps to the loopback address (127.0.0.1)
  - Different kinds of mappings are possible:
    - Simple case: 1-1 mapping
      » icc.skku.ac.kr maps to 115.145.152.181
    - Multiple domain names mapped to the same IP address:
      » www.skku.edu and www.skku.ac.kr both map to 115.145.129.40
    - Single domain name mapped to multiple IP addresses:
      » www.google.com maps to 74.125.71.104, 74.125.71.105, ...
int main(int argc, char **argv) { /* argv[1] is a domain name or dotted decimal IP addr */
    char **pp;
    struct in_addr addr;
    struct hostent *hostp;

    if (inet_aton(argv[1], &addr) != 0)
        hostp = gethostbyaddr((const char *)&addr, sizeof(addr), AF_INET);
    else
        hostp = gethostbyname(argv[1]);
    printf("official hostname: %s\n", hostp->h_name);

    for (pp = hostp->h_aliases; *pp != NULL; pp++)
        printf("alias: %s\n", *pp);

    for (pp = hostp->h_addr_list; *pp != NULL; pp++) {
        addr.s_addr = *((unsigned int *)*pp);
        printf("address: %s\n", inet_ntoa(addr));
    }
}
Domain Naming System (5)

- Querying DNS from the command line
  - Domain Information Groper (dig) provides a scriptable command line interface to DNS.
  - (cf.) nslookup, host

```
linux> dig +short icc.skku.ac.kr
115.145.152.181
linux> dig +short -x 115.145.152.181
icc.skku.ac.kr.
linux> dig +short amazon.com
72.21.194.1
72.21.211.176
72.21.214.128
linux> host icc.skku.ac.kr
icc.skku.ac.kr has address 115.145.152.181
```