Outline

- Networking Overview
- Layering and Protocols
- TCP/IP Protocol Family
- Client-Server Model

Inter-Process Communication

- Intra-node:
  - Files, pipes, FIFOs, semaphores
  - Signals, shared memory, messages
- All these IPC mechanisms are supported by the OS
  - OS provides system calls for IPC
  - OS maintains shared data buffers
  - OS keeps track of IPC data structures
- What if we want IPC across machines?

IPC across machines: Problems

- How to share data structures, data buffers?
  - File descriptors, pipes?
  - Shared memory?
- How do you address/locate the remote process?
  - PID?
- Even the OS on the two machines may be different!
  - Different system calls, IPC mechanisms
Network Communication

- Inter-node communication
- Requires communication at various levels:
  - Processes
  - OS
  - Hardware

Networking: Issues

- Hardware complexity and variety
  - Wide variety of networking media, e.g.: Ethernet, fibre, Wifi, dialup, etc.
  - Large networks have several connections, nodes
- Application diversity
  - Numerous networked applications. E.g.: Web, e-commerce, video streaming, gaming, ...
  - Applications should be portable
  - Developers and users should not have to worry about underlying complexity or variety
- Question: How to support this diversity and hide complexity?

Networking: Key Concepts

- Layering
  - Hides complexity and diversity
- Protocols and interfaces
  - Enable easy communication for networked applications
- Addressing and naming
  - Enable finding other machines, processes

Layering

- Communication takes place at multiple levels
  - Hardware, OS, application
  - Each level has its own implementation details
- Hide complexity of lower levels:
  - OS doesn't need to care about physical connection types, e.g.: Ethernet vs. fibre
  - Application code should be portable across OS’s
  - End-hosts don't need to care about intermediate network characteristics
ISO/OSI 7-Layered Model

- Physical Layer: Basic network hardware
  - Cables, network cards, wireless receivers
- Data Link Layer: Organization of data into frames
  - Ethernet, Wifi, token ring
- Network Layer: Addressing and forwarding of network packets
- Transport Layer: Reliable data transfer
- Session Layer: Connection establishment
- Presentation Layer: Data representation
- Application Layer: Application-specific

Protocols

- Set of rules and conventions between communicating participants
- Well-defined sequence of messages exchanged
- Different messages lead to different actions
- Each layer has its own protocols
  - Application: HTTP, SMTP, FTP
  - Transport: TCP
  - Network: IP
**Interfaces**

- Each layer provides an interface to the layer above it.
- Hides the complexity of underlying layer.
- Can replace the protocol underneath.
- Can have multiple protocols above interacting through the same interface.

**Connectionless vs. Connection-Oriented Communication**

- Connectionless
  - Data sent out in small chunks (packets).
  - Packet-switching: Moving data from source to destination.
  - E.g.: Postal service.
- Connection-oriented
  - Dedicated end-to-end channel.
  - E.g.: Telephone network.
- Computer networks are primarily connectionless.
- Connection-oriented protocols constructed on top.

**TCP/IP**

- The de-facto network protocol family.
- Used almost exclusively over the Internet.
- Most familiar applications run on top of it:
  - Web, email, ftp, video streaming, ...
- All OS's come equipped to support TCP/IP.
**TCP/IP Layered Architecture**

- **Hardware**
  - Ethernet, Wifi,...

- **Data Link/Physical**

- **Network**

- **Transport**
  - TCP, UDP

- **Application**

  - User

  - Operating System

**Network Layer: Internet Protocol (IP)**

- Packet addressing and delivery
- Connectionless:
  - Each packet is independent
  - Packet-switching
- Unreliable:
  - No guarantee of packet delivery
  - Packets can be dropped, duplicated, delivered out-of-order

**Transport Layer: User Datagram Protocol (UDP)**

- Built on top of IP
- Uses datagrams
  - Similar to IP packets
- Connectionless and unreliable
- Provides:
  - Port numbers for process identification
  - Checksum to verify datagram content

**Transport Layer: Transport Control Protocol (TCP)**

- Built on top of IP
- Connection-oriented:
  - Establishes end-to-end connection
  - Data is delivered as a stream of bytes
- Reliable:
  - Data is delivered in-order
  - No duplicates
**Application Layer Protocols**

- HyperText Transfer Protocol (HTTP)
  - Used in the Web
- Simple Mail Transfer Protocol (SMTP)
  - Used in email
- File Transfer Protocol (FTP)
- Trivial File Transfer Protocol (TFTP)

**Addressing and Naming**

- IP Addresses
- Hostnames
- Ports

**IP Addresses**

- Used to identify machines (network interfaces)
- Each IP address is 32-bit
  - IPv6 addresses are 128-bit
- Represented as x1.x2.x3.x4
  - Each xi corresponds to a byte
  - E.g.: 192.168.200.10
- Each IP packet contains a destination IP address
- How does the packet get to its destination?

**Internet Routing**

- The Internet consists of a number of routers
- Each router forwards packets onto the next hop
- Goal is to move the packet closer to its destination
  - Each router has a table
  - Matches packet address to determine next hop
Hostnames
- It would be difficult for us to remember IP addresses of all our machines
- Use names instead
  - E.g.: caesar.cs.umn.edu, www.google.com
- The name consists of multiple parts:
  - First part is a machine name (or special identifier like www)
  - Each successive part is a domain name which contains the previous domain

Domain Name Service (DNS)
- IP routing uses IP addresses
- Need a way to convert hostnames to IP addresses
- DNS is a distributed mapping service
  - Maintains "table" of name-to-address mapping
  - Used by most applications. E.g.: Web, email, etc.
- Advantages
  - Easier for programmers and users
  - Can change mapping if needed

Identifying Remote Processes
- IP addresses and hostnames allow you to identify machines
- But what about processes on these machines?
- Can we use PIDs?

Ports
- Identifiers for remote processes
- Each application communicates using a port
- Communication is addressed to a port on a machine
  - Delivers the packets to the process using the port
- Both TCP and UDP have their own port numbers
- Many applications use well-known port numbers
  - HTTP: 80, FTP: 21
Encapsulation and Multiplexing

- How does data travel from a process on the sender to another process at the receiver?
- **Encapsulation:**
  - Each layer adds its header to the data from upper layers at the sender
- **Multiplexing:**
  - Each layer combines data coming from multiple protocols/processes from upper layers at the sender
  - Data is routed to the destination IP address
  - Headers are peeled out and data is demultiplexed at the receiver

---

**Encapsulation**

```
<table>
<thead>
<tr>
<th>Layer</th>
<th>Headers</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>TCP hdr</td>
<td>Data</td>
</tr>
<tr>
<td>Network</td>
<td>IP hdr, TCP hdr</td>
<td>Data</td>
</tr>
<tr>
<td>Data Link/Physical</td>
<td>Ethernet hdr</td>
<td>IP hdr, TCP hdr, Data</td>
</tr>
</tbody>
</table>
```

---

**Multiplexing**

```
<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols/Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>TCP, FTP, TFTP</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP</td>
</tr>
<tr>
<td>Network</td>
<td>IP</td>
</tr>
<tr>
<td>Data Link/Physical</td>
<td>Ethernet</td>
</tr>
</tbody>
</table>
```

---

**Client-Server Model**

- A common way to communicate and construct applications
- One end (client) requests for a service
- The other end (server) receives requests, performs service and returns results
- E.g.: Web
  - Web browser (client) asks for a URL
  - Web server returns the corresponding file
- Most applications modeled as client-servers
  - Email, ftp, ssh, ...
**Server**
- Daemon that waits for requests
- When a request arrives
  - Handles the request
  - Performs some service
  - Returns result to requesting client
- Goes back to waiting for more requests
- Waiting for new requests and handling existing requests might happen concurrently
- How?

**Client**
- Sends a request to a server
- Waits for response
- Receives response (or error)
- Done or sends more requests

**Client-Server Example: Web**

**Summary**
- Networking Basics
- Layering and Protocols
- ISO/OSI Model and TCP/IP
- IP Addressing, Hostnames and Ports
- Encapsulation and Multiplexing
- Client-Server Model