CMSC 498M: Chapter 10a  
Networking and Multiplayer Games

Reading:  
- Network and Multiplayer, by Chuck Walters (Chapt 5.6 in "Introduction to Game Development" by S. Rabin).

Overview:  
- Multiplayer Games  
- Networking: Protocols and packets  
- OSI Network Layers: Physical, Data Link, Network, Transport, ...

Multiplier Games

Multiplier Games: A number of players communicating through a network.

Persistent Games: Such as "World of Warcraft", where state is maintained, regardless if anyone is playing.

Transient Games: Only exist while people are playing, and reset each time the server-side is reset.

Performance Issues:  
Latency: How long does it take for global state to be updated?  
Reliability: How often is data lost or corrupted?  
Bandwidth: How much data can be transmitted in a given time?  
Security: How is the game-play protected from tampering/cheating?

Tradeoffs:  
All of these considerations interact, and trade-offs must be made.
Overview

- Multiplayer Game Basics
- Packets and Protocol Basics
- OSI Network Structure

Multiplayer Factors: Event Timing

Factors in Multiplayer Games:

- **Event Timing**: How do players interact with the game?
- **Shared Display**: Single platform and multiple players.
- **Connectivity**: How are players connected?

**Event Timing**:

- **Turn-based**:
  - Player's move in **turns** (round robin). Other players wait.

- **Real-time**:
  - Players move **simultaneously**.
  - May need to handle **race-conditions** (e.g., two players attempt to acquire the same resource at the same time).
  - "**Twitch Games**" typically require very low latencies, less than 150 ms.
Multiplayer Factors: Shared Display

**Shared Display:** for multiplayer games on a single platform.

**Full Screen:**
- **Complete Player Visibility:** Such as board games and sports games. Everyone sees everything at all times.
- **Player Funneling:** Restricts players to a small region of game space. As players move this region moves with them. (e.g. Gauntlet)
- **Turn-Based Screen Control:** Active player controls the viewpoint.

**Split Screen:** Each player has a separate portion of display through the use of separate viewports.
- **Components:** System maintains the following information for each player.
  - **Camera:** Viewpoint.
  - **Cull Data:** What portion of the environment is visible.
  - **Heads-up Display:** Game stats displayed on top of scene.
  - **Map Data:** Centered about the current player.
  - **Audio Effects:** Need to be split among players as well.

**Issues:** Maintaining consistency between views. Practical only for a small number of players (e.g. 2-4).

---

Multiplayer Factors: Connectivity

**Connectivity:** How do players connect?

**Direct Link:**
- **What:** Connected directly typically through short connections.
- **Examples:** Serial and USB cable, wireless (infrared and Bluetooth).
- **Pro:** Fast, reliable. **Con:** Few players, only small distances.

**Circuit-Switched Network:**
- **What:** Unshared direct connection between endpoints.
- **Example:** Traditional public telephone system.
- **Pro:** Reliable, low-latency. **Con:** Low bandwidth, few players.

**Packet-Switched Network:**
- **What:** Communication broken into small packets that are routed over a shared network.
- **Example:** Internet.
- **Pro:** Many players, large areas. **Con:** Variations in latency/bandwidth.
Networking Basics

**Network:**
- A group of two or more computers connected together.
- Henceforth we consider packet-switched networks.

**Characteristics:**

**Scale:** The area spanned by the network.
- **LAN:** (Local-area network) E.g., connecting one business or school.
- **WAN:** (Wide-area network) Connecting computers distributed over an entire city, state, country or the world.

**Topology:** How the computers are connected together. Examples: Ring, star, bus, tree.

**Protocol:** Agreed upon rules for communicating and routing information. Examples: TCP, IP, FTP, HTTP.

**Architecture:** General communication structure. Examples: Peer-to-peer, client server.

Overview

- Multiplayer Game Basics
- Packets and Protocol Basics
- OSI Network Structure
Protocol

Protocol:
- A convention for routing and transferring data over packet-switched networks.
- Communication networks may be unreliable and may connect machines having widely varying manufacturers, operating systems, speed, data formats.

Issues:
Packet sizes: Fixed or variable sized packets?
Handshaking: Communication exchange to ascertain how data will be transmitted (format, speed, etc.)
Acknowledgements: For receipt of data.
Error checking/correction: Handling errors in data transmission.
Compression: Reducing data size due to limited bandwidth.
Encryption: To protect private data.

Packets

Packet:
- The logical transmission unit of a protocol.
- Two parts: Header (information) and payload (data).
- Example: Quake network protocol packet structure:
  http://www.gamers.org/dEngine/quake/QDP/qnp.html

Simple Example:

```c
struct Packet {
    // Header
    short packetLength;  // length of the packet (in bytes)
    short packetType;    // E.g. data, control, acknowledgement
    int checkSum;        // checksum used for error checking
    // Payload
    char data[256];      // ...the data
};
```
Packets: Issues

**Issues:**

**Serialization:**
- Pointers and references cannot be reliably transmitted since they refer to local memory. Convert them to names or indices to an array.
- Abstract data types: Are often based on references for inheritance.

**Endianness:** Transmitting multibyte numbers: 0123
- Little Endian: Low-order bytes first: 3, 2, 1, 0
- Big Endian: High-order bytes first: 0, 1, 2, 3

**Intrinsic Types:** Use \_\_int32 rather than int to force compiler to use 32-bit integers.

**Unicode:** Better than ASCII for string data.

---

**Overview**

- Multiplayer Game Basics
- Packets and Protocol Basics
- OSI Network Structure
The OSI Model

Open System Interconnect (OSI) Model: Formalizes the multilayered structure of networks.

Application: End-user processes (e.g., mail (smtp), ftp, telnet).
Presentation: Packetization, byte order, encryption, compression.
Session: Connection and data exchange (e.g., logging in/out, socket).
Transport: Flow control (e.g., TCP/UDP).
Network: Basic routing (e.g., IP).
Data Link: Packet/frame structure.
Physical: Physical medium (wire).

Physical Layer: Latency and Bandwidth

Physical Layer:
- The medium over which data is carried.
- Examples: twisted-pair wire, coaxial cable, wireless.

Latency and Bandwidth:
Time of flight: Time to send a single bit of data.
Includes delays at switches from source to destination.
Bandwidth: Maximum transfer rate from source to dest in bits per second (bps). Includes header.
Transmission time: messageSize / bandwidth.
Transport latency: timeOfFlight + transTime
Sender (receiver) overhead: Time to process packet for sending (receiving).
Total Latency: overhead_{send} + timeOfFlight + transTime + overhead_{rec}
Effective Bandwidth: messageSize / totalLatency
Example of Computing Effective Bandwidth:

Raw bandwidth: 10Mbps (mega-bits per second).
Sender overhead: 250 μsec (0.00025 sec).
Receiver overhead: 300 μsec (0.00030 sec).
Message size: 1000 bytes (8000 bits).
Distance: 1000km (Transmission speed = 150,000km/s).

What is the effective bandwidth?

\[
\text{TransTime} = \frac{8000\text{b}}{10\text{Mb/s}} = \frac{8000\text{b}}{10^6\text{b/s}} = 800\mu\text{s}
\]

\[
\text{TimeOfFlight} = \frac{1000\text{km}}{150,000\text{km/s}} = 6667\mu\text{s}
\]

\[
\text{TotalLatency} = 250 + 6667 + 800 + 300 = 8017\mu\text{s} = 0.008\text{s}
\]

\[
\text{EffBandwidth} = \frac{8000\text{b}}{0.008\text{s}} = 1\text{Mb/s}
\]

Note that this is only 1/10th of the raw bandwidth.

---

Maximum bandwidths of common media connection types:

<table>
<thead>
<tr>
<th>Media Connection Type</th>
<th>Maximum Bandwidth (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial cable</td>
<td>20K</td>
</tr>
<tr>
<td>USB 1&amp;2</td>
<td>12M, 480M</td>
</tr>
<tr>
<td>ISDN</td>
<td>128K</td>
</tr>
<tr>
<td>DSL</td>
<td>1.5M down, 896K up</td>
</tr>
<tr>
<td>Cable</td>
<td>3M down, 256K up</td>
</tr>
<tr>
<td>LAN 10/100/1G BaseT</td>
<td>10M, 100M, 1G</td>
</tr>
<tr>
<td>Wireless 802.11 a/b/g</td>
<td>54M, 11M, 54M</td>
</tr>
<tr>
<td>Power line</td>
<td>14M</td>
</tr>
<tr>
<td>T1</td>
<td>1.5M</td>
</tr>
</tbody>
</table>

Note: Rates vary with direction.

Source: Chapt 5.6 of Rabin

Actual delivery is around 70% of maximum.
Data Link Layer

Data Link Layer:
- Puts data in frames and ensures error-free transmission.
- Controls the timing of the network transmission. Adds frame type, address, and error control information.
- Examples of data link protocols are Ethernet for local area networks and PPP, HDLC and ADCCP for point-to-point connections.
- Network Interface Card (NIC): Performs these operations. Each NIC is associated with a MAC (media access control) address.
- All devices within a given subnetwork must have unique MAC addresses.

Network Layer

Network Layer:
- Performs end-to-end (source to dest) packet delivery, (whereas the data link layer is for node-to-node).
- Performs network routing, flow control, data segmentation/desegmentation, and error control.
- Famous example: The internet protocol (IP).

IP Addresses:

IP version 4: (IPv4)
- Address is 4 bytes, typically displayed in decimal: 255.8.128.16
- Only $2^{32} = 4.3$ billion possible addresses.

IP version 6: (IPv6)
- Address is 16 bytes, displayed as 8 16-bit segments in hex: [2001:0db8:85a3:08d3:1319:8a2e:0370:7344]
- Supports $2^{128} = 3.4 \times 10^{38}$ addresses. Roughly 1 address for every atom in everyone’s body on the planet.
Network Layer: Special IP Addresses

**Unicast Address**: An individual’s IP address. Sources:
- **Static**: Fixed address reserved for servers that require a well-known address. (Examples: DNS server, gateway router.)
- **Dynamic**: Assigned dynamically by a DHCP server (dynamic host configuration protocol) to a specific MAC address. (Typical case.)

**Special Addresses**: Some IP addresses have special meaning:
- **Multicast**: Packets sent to this address are routed to all members of a multicast group.
- **Local Broadcast**: Packets sent to this address are routed to all members of the local network.
- **Loop-back**: Packets sent to this address loop back to the current machine without entering the physical network (e.g., for testing).

**Domain Name**: Human readable address (e.g., www.gamedev.net rather than 16.15.32.1). Stored in domain name server (DNS).

Transport Layer

**Transport Layer**:
- Provides transparent transfer of data between hosts.
- Responsible for end-to-end error recovery and flow control, and ensuring complete data transfer.
- Makes the network layer’s services more reliable.
- Provides notion of ports, as extension of IP addresses.

**Ports**:
- Ports are essentially ways to address multiple entities on the same node. Identified by an integer in the range 0..1023.
- **Net Address = IP Address + Port Number**.
- Each application "listens" for information on a single port.
- Can run multiple network applications at the same time.
Transport Layer: TCP

Transmission Control Protocol: (TCP) Features:
- **Ordered-data transfer**: Assign each packet a unique sequence number and reorder incoming packets in proper sequence order.
- **Error-free data transfer**: Uses checksum to identify corrupted packets and sends negative acknowledgement.
- **Retransmission of lost packets**: If receipt is not acknowledged.
- **Discarding duplicate packets**: By detecting duplicate sequence nos.
- **Congestion avoidance** (flow control): When receipts are not received in a timely manner, the transmission rate is reduced. (Like having a traffic light controlling traffic onto a freeway.)

**Miracle of TCP/IP**:
- These protocols were developed at a time when the Internet was much smaller, but they scaled up remarkably well to today's internet.

Transport Layer: TCP

TCP packet structure:

```
 source port number  destination port number
                      sequence number
                      acknowledgement number
 head len  unused                  window size
 Internet checksum  ptr to urgent data
                    options
                    data
```

32 bits
**Transport Layer: UDP**

User Datagram Protocol (UDP):
- Connectionless protocol with no guarantees of delivery.
- "Send and forget" individual packets.
- Faster than TCP: Smaller packets, lower overhead, lower latency.
- Popular for games, since much state information is nonessential and quickly goes out of date.

<table>
<thead>
<tr>
<th>source port number</th>
<th>destination port number</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>checksum</td>
</tr>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

**UDP packet structure**

**Session Layer and Socket Programming**

**Session Layer:**
- Manages connections between applications.
- Responsible for establishing/terminating connections and coordinating data exchange.

**Sockets API:**
- Library for low-level network programming and inter-process communication.
- Berkeley Sockets API: released with 4.2 BSD release of the Unix operating system. Quickly became the de facto standard.
- WinSock: Windows version of Berkeley sockets, with additional features for the Windows environment.
Socket Programming

Things you can do with Sockets:

- **Create:**
  - You can specify whether the socket is TCP or UDP.

- **Connect:**
  - Connect to a remote listening host by giving the net address (e.g., IP address and port).
  - Support for converting domain names to IP addresses.

- **Stream Transmission:** (for TCP)
  - Send and receive data through a socket.

- **Datagram Transmission:** (for UDP)
  - Analogous operation for UDP sockets.

What to send?

- **Game state/events:** This will be discussed in the next lecture.

Presentation Layer

**Presentation Layer:**

- Responsible for delivery and formatting of information to application layer for further processing or display.
- Relieves application layer of concern of syntactical differences in low-lever data representations.

**Examples:**

- **Format Conversion:** For example strings:
  - Length+Text: "13, thisisastring"
  - Null terminated: "thisisastring\0"

- **Packing:** E.g. packing small enumerations into bit fields.

- **Float to fixed:** Convert floating point numbers to fixed point.

- **Compress data structures:** E.g. encode computational forms (rotation matrices) to more concise representations (quaternions).

- **Encryption:** protecting private data like passwords.

- **Serialization:** removing pointers and references.
Resources

Further information on WinSock:
"WinSock2 for Games": Tutorial on WinSock from gamedev.net:
   http://www.gamedev.net/reference/articles/article1059.asp
"Game Programming with Asynchronous Sockets": Another one.
   http://www.gamedev.net/reference/programming/features/asynsock/

Berkeley Sockets:
Tutorial from RPI:
   http://www.cs.rpi.edu/courses/sysprog/sockets/sock.html

General Network Programming:
"The Internet Sucks:
   Or, What I Learned Coding X-Wing vs. TIE Fighter."
   http://www.gamasutra.com/features/19991003/lincroft_01.htm

Summary

Summary:
- Multiplayer Games
- Networking: Protocols and packets
- OSI Network Layers: Physical, Data Link, Network, Transport, ...