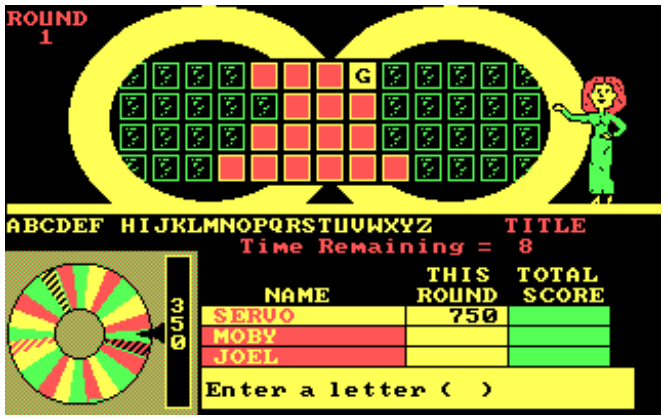
Multiplayer Games and Networking

Overview

- Multiplayer Modes
- Networking Fundamentals
- Networking for Games
- Networking for Unity

Early forms of Multiplayer: **Turn Based**

- Easier to implement
- Puzzle / board game
- Non-real time connection
  - Floppy Disks, Email
  - Database (Door Games)



Early forms of Multiplayer: **Real Time**

- Shared I/O
  - Input Devices
    - Shared Keyboard layout
    - Multiple Device Mapping
  - Display
    - Full Screen vs Split Screen

Multiplayer Modes: **Connectivity**

- Non Real-Time
  - (turn based)
- Direct Link
  - Serial, USB, IrD, … (no hops)
- Circuit Switched (phones)
  - Dedicated line with consistent latency
- Packet Switched
  - Internet
  - Shared Pipe

Multiplayer Modes: **now with Networking!**

- Difficulty based on Event Timing
  - Turn-Based
    - Any connection type
  - Real-Time
    - More data to sync
    - Latency sensitive

Networking: **When do we need it?**

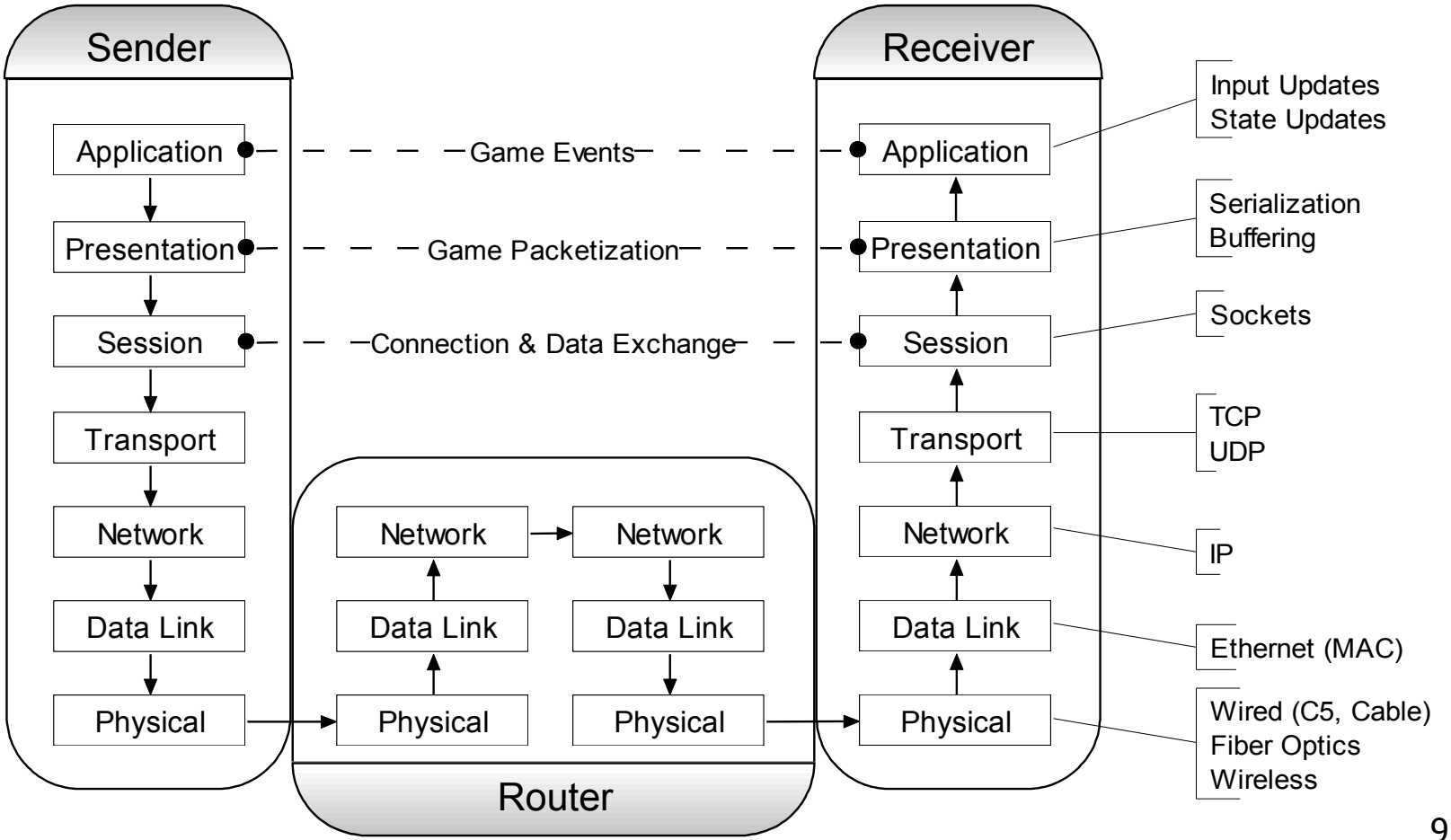
- Single Player Games?
  - Leaderboards and trophies
  - Online data stores
    - (PS+, Steam Cloud)
  - Downloadable content
  - DRM
- Multiplayer
  - Most AAA titles moving toward multiplayer
  - * or at least, single player +

"Portal 2 will probably be Valve's last game with an isolated single-player experience" *

Networking
At a glance

- Connection between multiple computers
- Transmission of data
- How do we design a system that can do….
  - Packet Length Conveyance
  - Acknowledgement Methodology
  - Error Checking / Correcting
  - Compression
  - Encryption
  - Packet Control

Protocol Stack:
Open System Interconnect



Physical Layer

- Bandwidth
  - Width of data pipe
  - Measured in bps = bits per second
- Latency
  - Travel time from point A to B
  - Measured in Milliseconds
- The Medium
  - Fiber, FireWire, IrD , CDMA & other cell

Table: Max Bandwidth Specifications

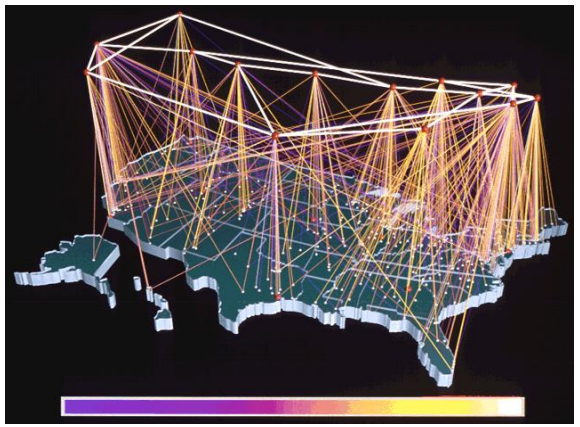
	Serial	USB 1&2	ISDN	DSL	Cable	LAN 10/100/1G BaseT	Wireless 802.11 a/b/g	Power Line	T1
Speed (bps)	20K	12M 480M	128k	1.5M down 896K up	3M down 256K up	10M 100M 1G	b=11M a,g=54M	14M	1.5M

Data Link Layer

- Serializes data to/from physical layer
- Network Interface Card
  - Ethernet
  - MAC Address

Network Layer

- Packet Routing
  - Hops
    - No connection
    - Guarantees sending
    - Doesn’t guarantee receiving
    - Non-deterministic path
  - Routers, Hubs, Switches
- Internet Protocol (IP)
  - Contains Source & Destination IP Address
  - IPv4 vs IPv6
- Unicast, Broadcast, Loop back



Network Layer: **Domain Name Service**

- Domain Name Service
  - Converts text name to IP address
  - Must contact one or more DNS servers to resolve
  - Local cache resolution possible
- Game Tips
  - Store local game cache to use when DNS out of order.
  - DNS resolution often slow, use cache for same day resolution.

Transport Layer

- Manage data deliver between endpoints
  - Error recovery
  - Data flow
- TCP and UDP used with IP
  - Contains Source and Destination Port
- Port + IP = Net Address
  - Port Range = 0-64k
  - Well known Ports 0-1k
    - http, ftp, ssh, …

Transport Layer:
Transmission Control Protocol

- Connection based
  - Keep Alive
  - Handles breaking up data into correct size
  - Packet window
  - Packet Coalescense
- Guaranteed, in order delivery
  - ack, nack, resend
- Flow Control
- Easy to use
  - Reading and writing, just like a file
- Requires more header data

Transport Layer: **User Datagram Protocol**

- No connection
- No guarantees
  - May not arrive
    - TTL (time to live) – hop count limit
  - May not arrive in order
  - May arrive multiple times
  - Source not verified
- Datagram
  - Sent in packets exactly as user sends them
- Capable of broadcasting

Transport Layer: **TCP vs UDP**

- Which to use?
  - Depends on the game!
- TCP
  - Turn based games, leader boards
- UDP
  - More common, especially for time sensitive games
  - Add TCP features as needed
  - Unity uses UDP, with features for reliable, in order transmission

Session Layer

- Manages Connections between Apps
  - Connect
  - Terminate
  - Data Exchange
- Socket API live at this layer
  - Cross platform
  - Cross language

Session Layer: **Sockets**

- Based on File I/O
  - File Descriptors
  - Open/Close
  - Read/Write
- Modes
  - Blocking
    - Use in separate thread
  - Non-blocking
    - Poll the socket periodically

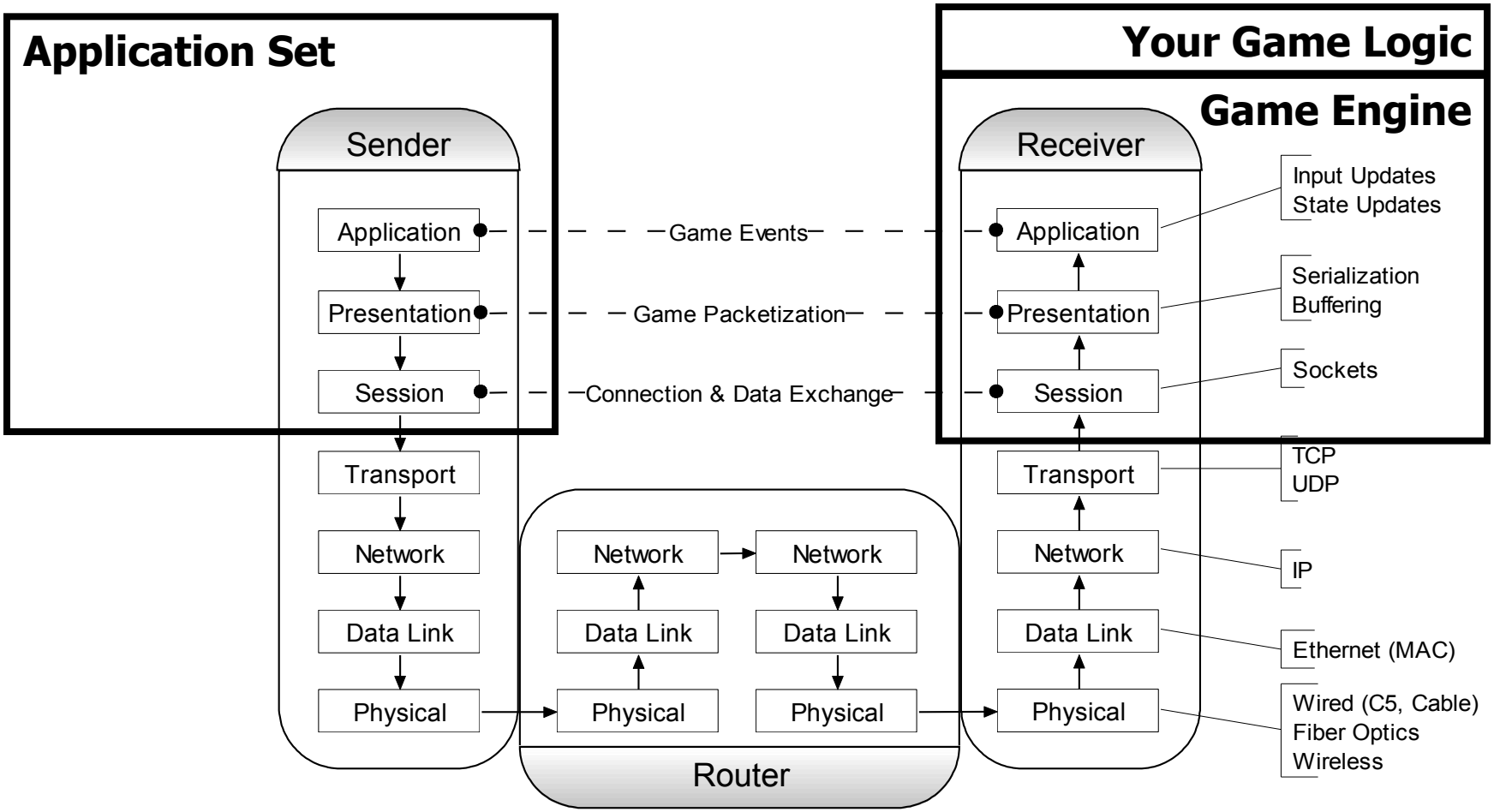
Presentation Layer

- Prepares App Data for Transmission
  - Compression
  - Encryption
  - Endian Order
    - 0b1000 vs 0b0001
  - Serialize
  - Buffering
    - Packet Coalescense
    - Increased Latency
    - Store local data and wait

Application Layer

- Interfaces with user
- Handles game logic
- Transmits the right data
- … at the right time…
- …to the right person

Protocol Stack:
Open System Interconnect



Networking for Games

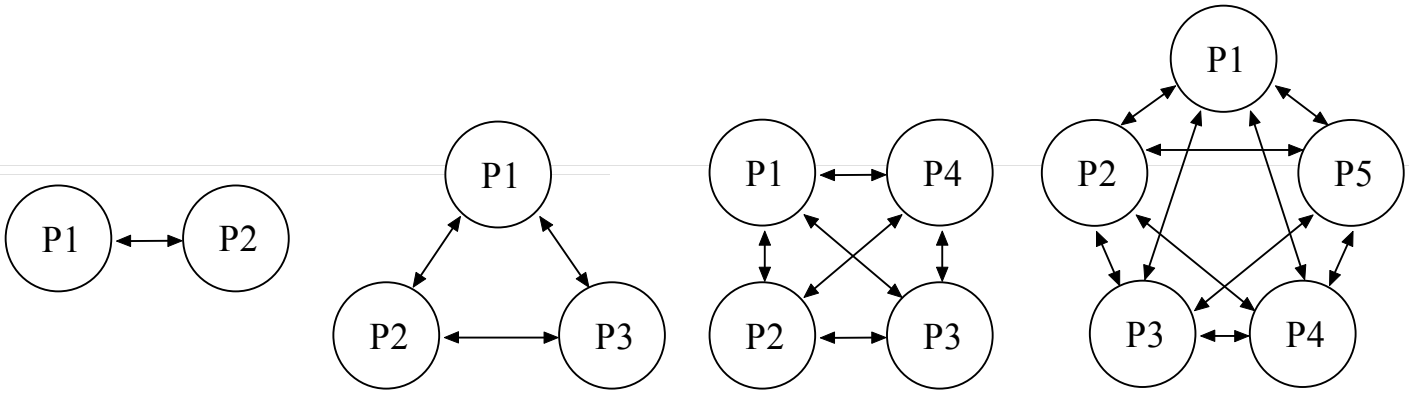
- Who are we communicating with?
- What data needs to be sent?
- How often do we need to send it?
- How do we protect that data?
- How do we handle conflicts?

- (Looking at non-trivial real time applications)

Connection Models

- Broadcast
  - Good for player discovery on LANs
- Peer to Peer
  - Good for 2 player games
- Client / Server
  - Good for 2+ player games
  - Dedicated lobby server great for player discovery

Peer to Peer vs. Client/Server



2 players
1 connection

3 players
3 connections

4 players
6 connections

5 players
10 connections

N = Number of players

	Broadcast	Peer/Peer	Client/Server
Connections	0	$\sum_{x=1}^{N-1} x$	Client = 1 Server = N

	Broadcast	Peer/Peer	Client/Server
Send	1	N-1	Client = 1 Server = N
Receive	N-1	N-1	Client = 1 Server = N

Client / Server Architecture

- Clients connect to Server
  - Server handles all communication between clients
  - “UDP Connection”
- Small high frequency packets (20-30 /sec)
- Packet based comm results in new challenges
  - Packet loss
    - Especially if client asks for higher rate then their connection can handle
  - Inherent latency
    - Bandwidth + Latency => Lag => Player frustration
    - Varies from client to client

Client / Server:
Authoritative vs. Non-Authoritative

- Authoritative
  - Clients send inputs to server
  - Server does all input processing, world simulation, application of data rules
  - Server tells client what happened
  - Client only collects data and renders results!
- Non-Authoritative
  - Clients process user data, applies logic, updates the server
  - Clients have control of their specific objects
  - Server acts as a relay
- Can you trust clients?

Client / Server:
Communication Methods

- Remote Procedure Calls
  - Invoke functions on another machine
    - Client to server
    - Server to a client
    - Server to a set (possibly all) clients
  - Used for infrequent actions and events
    - Loading levels
    - State of infrequently changed object

Client / Server:
Communication Methods

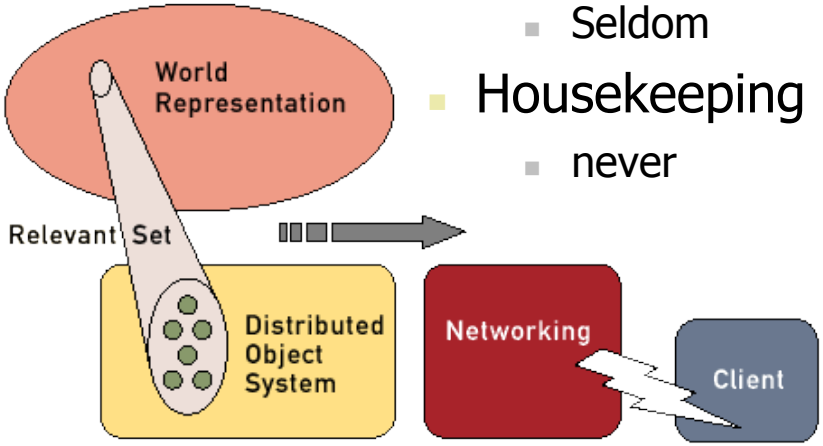
- Update Models
  - Input Reflection
    - Authoritative server mode
    - Slightly process input data
    - People notice delay of 0.1s
    - Synchronous (wait for data from everyone)
    - Asynchronous (predict input)
      - Not appropriate for input reflection
    - Low and consistent latency
    - Seed rand() with same seed on all computers
    - Don't use system time for calculations

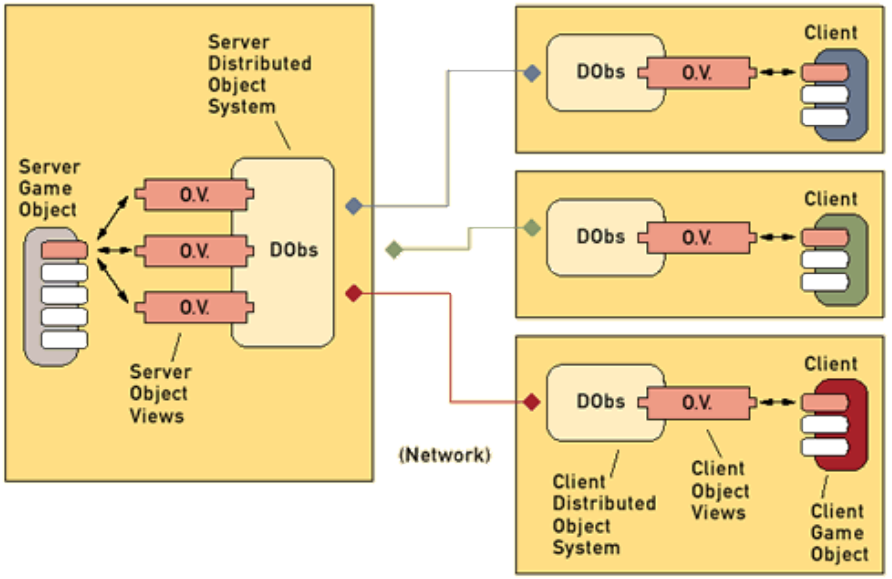
Client / Server:
Communication Methods

- Update Models
  - State Reflection
    - Both server modes
    - Update position, rotation, velocity….
    - Larger packets
      - Prioritize
    - Server Distributed Object System

Client / Server:
Server Distributed Object System

- Relevance Sets
- Object Views
  - Objects consist of three major groups of data
    - Visual & Display
      - always
    - Game logic & AI
      - Seldom
    - Housekeeping
      - never





Client / Server:
Server Distributed Object System

- Synchronization
  - The “art” of network programming
  - Dead Reckoning
    - Works fine until drastic change
  - AI Assist
    - Help transition between waypoints
    - Might cause slight synch problems
  - Arbitration
    - Weighted decision to correct outcome
    - Server is dictator
    - Client might delay critical event while waiting

Client / Server:
Sync Optimizations Techniques

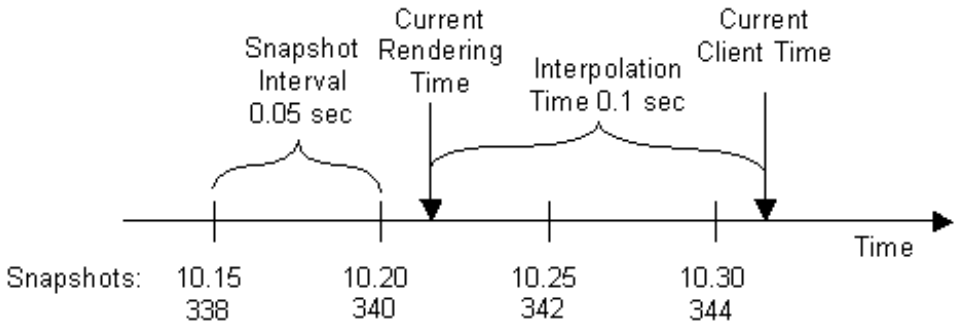
- Solutions (Valve's Source Engine)
  - Delta compression
  - Interpolation
  - Prediction
  - Lag compensation

Client / Server:
Sync Optimizations Techniques

- Delta compression
  - Only send newly updated information
  - Approach used for other types of streaming data
  - Acknowledgement numbers used to keep track of flow
  - Client can request full snapshot when problems occur

Client / Server:
Sync Optimizations Techniques

- Interpolation
  - Snapshot updating results in jerky jittery graphics
  - Interpolate between current snapshot and previous
    - Client runs 100 ms behind
    - Will work with one lost packet
    - Two lost packets will cause errors



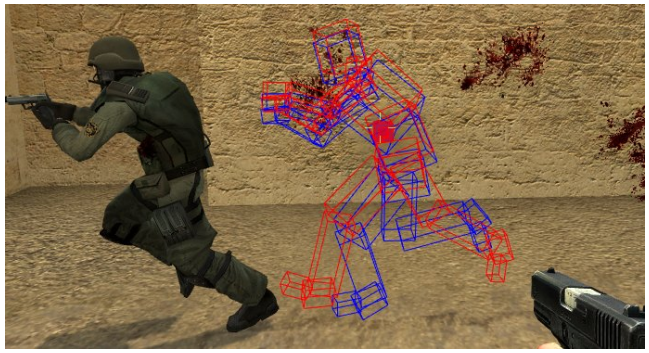
Client / Server:
Sync Optimizations Techniques

- Prediction
  - Player will notice 100 ms delay in own input
  - Client side input prediction
  - Client predicts where player should be using same logic as server
  - When snapshot comes they are compared
    - May be different since server has more information than client
  - Client must correct
    - Smoothing used to make correction less noticeable

Client / Server:

Sync Optimizations Techniques

- Lag compensation
  - When my shot information arrives at server, enemy has moved
  - Server compensates
  - Maintains history of all player positions
  - Looks back in time for player position at time of shot



Cheating

- Why not client do hit detection?
  - Client can't be trusted
  - Cheat proxy
    - "man in the middle"
  - Valve's-Anti-Cheat
  - Blizzard's Warden

Cheating

- Material hacks (wallhacking)
- Aim and trigger bots
  - Color based. Old versions (Quake etc.) replace models with colored ones, used algorithm to scan screen.
    - Can end up aiming at other stuff in the scene
  - Client hook versions use information on the player positions
  - Graphics Driver versions. Get 3D values from renderer and convert to mouse coordinates

Security

- Console network stacks
  - provide additional security functions
- Intel Fair Online Gaming
  - Hardware, firmware, and game software on client

Security

- Encryption
  - Preserve integrity of network traffic
  - Efficiency vs Security
- Execution Cryptopgraphy
  - Prevent reverse engineering to edit game data
- Copy Protection
  - DRM
  - Code sheets
  - Active internet connection

Networking for Unity

- ***This is not a substitute for reading Unity's documentation!***

- UDP based
- Client / Server
  - No dedicated server software
  - Authoritative vs. Non-Authoritative
- Game Lobby

Networking for Unity

- Network Views
  - Required component for transmitting data
  - Not same as an “Object View”, but required to create them in code
  - RPC
  - State Synchronization
    - Reliable Delta Compressed
    - Unreliable
- Tutorials for prediction, lag compensation, interpolation, etc.

Networking for Unity: **3rd Party MMEs**

- Massively Multiplayer Engines
  - Photo, SmartFox, Electroserver, …
- Higher scalability
- API for Unity
- Re-implementing Object View structures

Networking in your game

- Read Unity's documentation first!
  - Overview
  - API for networking classes
- Check out the tutorials
  - Unity's networking tutorials
  - Other's available online ($$$?)
- Get *something* working
  - Then test the different options

References: **Networking Overview**

- Source Engine Overview
  - http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking
  - Overview, Delta Compression, Interpolation, etc.
- Relevance Sets / Object Views
  - http://www.gamasutra.com/resource_guide/20020916/lambright_01.htm
- Glenn Fiedler Overview
  - http://gafferongames.com/networking-for-game-programmers/
  - Includes articles on cross-platforms low level implementation (stuff that Unity already does for you)

References: **Unity**

- Documentation
  - http://unity3d.com/support/documentation/Manual/Networked%20Multiplayer.html
  - http://forum.unity3d.com/threads/29015-UniKnowledge-entry-Unity-Networking-the-Zero-to-Hero-guide
- Example / Tutorials
  - http://www.palladiumgames.net/tutorials/unity-networking-tutorial/
  - http://answers.unity3d.com/storage/temp/13488-networking.zip

References: **NAT Punch-through**

- Overview
  - http://www.mindcontrol.org/~hplus/nat-punch.html
- Unity Master Server
  - http://unity3d.com/support/documentation/Components/net-MasterServer.html