Who am I?

- Jeff Kesselman, CTO Nphos
  - 18years in games and multi-media
    - CTO Rebel Monkey, Blue Fang Games
    - American Interactive Media (Phillips)
    - Crystal Dynamics
    - Total Entertainment Network (TEN)
  - 9 years at Sun
    - Win32 Java 1.3 Performance Tuning
    - Initial leader of the JInput project
    - 2 yrs in Sun "Game Technologies Group"
    - 2.5 years at Sun Labs (Project Darkstar)

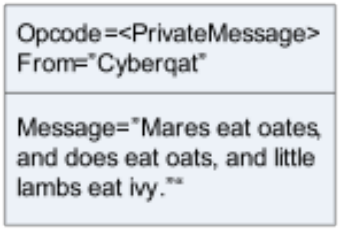
Goals for the Week

- The fundamental structure and technologies of the internet
- The history of multi-player games and the challenges they faced when moving to the internet.
- The influx of web technologies and the rise of so called “casual games.”

The Fundamental Technologies of the Internet

A Packet

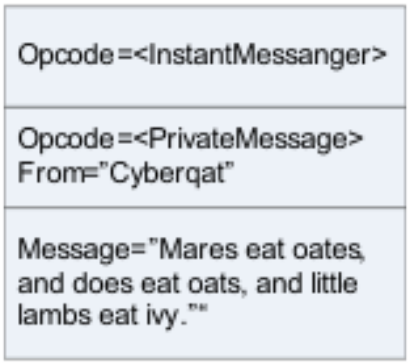
- Fundamental IO Unit
  - Header
    - Generally an op-code
    - Might have other info
      - Sender
      - Packet size
      - etc
  - Payload
    - The data being sent
    - 0 or more fields
      - 0 if no data needed



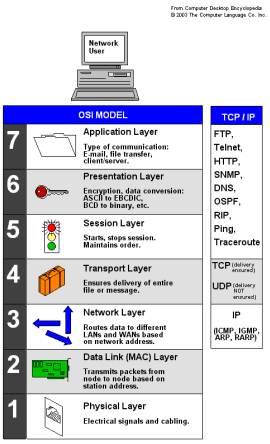
A set of related packets and their meanings together define a Protocol

Nesting Packets

- Entire packet can be the "payload" of a bigger packet
  - Outer packet header prepended to inner packet
- Protocols "stack' in this manner
  - Can be many layers deep

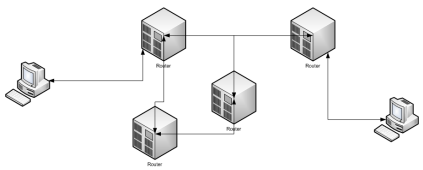


The TCP/IP Stack



Simplified Image of the Internet

- Internet is fundamentally made of two things
  - Client computers
  - Routers
- Data "hops" from one router to the next until it reaches its destination
  - Multiple potential routes
  - Routed by IP address
  - Can watch
    - Traceroute in Unix
    - Tracert in Windows



Packet Loss

- Internet is inherently unreliable
  - Packets can be lost in transmission

Why might a packet be lost?

Packet Loss

- Internet is inherently unreliable
  - Packets can be lost in transmission
    - Router failure
    - Line failure
    - Line partial failure (garbled data)
    - Router over-loaded (dropped from queue)

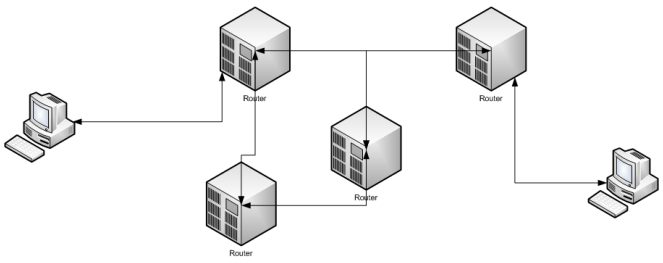
Packet Order

- Internet is inherently unordered
  - Packets can arrive at destination in a different order then they were sent

Why might they arrive out of order?

Packet Order

- Recall Internet has redundant paths
  - Each packet traces its own path
    - Each router makes a packet by packet selection of where to forward to based on current congestion
    - Decision is based on local knowledge only
    - Older packet might get "stuck" on a router queue



TCP and UDP

- TCP and UDP are the fundamental data "carriers" for applications on the Internet
- UDP is
  - A datagram protocol
    - Connectionless, Packet Oriented
  - Unordered and Unreliable
    - Built more or less right on top of IP
- TCP
  - A stream protocol
    - Connections, stream oriented
  - Ordered and Reliable
    - Complex additional protocol layer

TCP Reliability

- Internet is inherently unreliable
  - Routers drop packets when garbled or overloaded
  - Packets can arrive in any order
- Where does TCP get its guarantees?
  - Packets are sequence ordered on send.
  - If a later packet arrives before an earlier one, a resend is requested
  - Delivery of later packets held until earlier packets arrive
  - This is an over-simplification
    - 30 yrs worth of tuning and refining behind TCP

Disadvantages of TCP

- TCP is easy to use
  - Reliable and ordered
  - Easier to secure

What might be some disadvantages of TCP?

Disadvantages of TCP

- Can “stall”
  - Must wait for lost packet to continue
  - Creates latency spike
- Small additional overhead per packet
  - About 28 bytes
- For applications that are more sensitive to latency then loss, UDP can be a better choice.

Application Level Protocols

- All built on top of TCP or UDP
  - HTTP (the web)
    - Built on top of……. ?
  - RTP (streaming audio and video)
    - Built on top of… ?
  - SSH
    - Built on top of… ?
  - Guild Wars
    - Built on top of… ?
  - Unreal Networking
    - Built on top of… ?

Application Level Protocols

- All built on top of TCP or UDP
  - HTTP (the web)
    - Built on top of TCP
  - RTP (streaming audio and video)
    - Built on top of UDP
  - SSH
    - Built on top of TCP
  - Guild Wars
    - Built on top of TCP
  - Unreal Networking
    - Built on top of UDP

Hybrids rare but possible

- TEN's BULLET Protocol
  - TEN was fundamentally a TCP/IP service
  - BULLET traded bandwidth for latency spike reduction
    - Main stream of game traffic TCP/IP
    - Sliding window of packets duplicated in UDP side-channel
    - UDP packets used to "fill in" during TCP stalls if available

HTTP

- Built on top of TCP/IP
  - Every Put/Get involves…
    - Make TCP/IP connection
    - Send request (character coded)
    - Get repsonse (character coded)
    - Close connection
  - Very Inefficient
    - Connection establishment expensive
    - Textual translation costs
  - Asynchronous by nature

HTTP Synchronous Sessions

- Comet
  - AJAX technique to fake session
  - Polling based
    - “Long poll” to reduce costs
  - Really quite absurd
    - Even more inefficeint then HTML
    - Lots of problems
      - Faking connectivity that HTML threw away

HTTP Synchronous Sessions

- HTML 5 Web Sockets
  - Real session
    - Multiple interactions on a single connection
  - Still Textual

RPC (Remote Procedure Call)

- Another layer (usually) over TCP/IP
- Computer A sends a packet to Computer B saying "call this function".
  - Synchronous RPC, computer A waits for return value in a packet from computer B, which it returns from the initiating call onA
  - Asynchronous RPC, computer A sends the request and goes on. B can initiate its own RPC call if it wishes to tlak back to A.
- Structural procedure call
  - ONC RPC a standard (Unix, Windows, etc)
- Object Oriented RPC
  - Java RMI
  - Corba
  - SOAP XML-RPC
  - others.... (python, ruby, etc...)

Network Names

- IP uses numerical addresses
  - IP4
    - 4 bytes per address
    - 128.132.45.1
  - IP6
    - 8 octets per address
    - 2001:0db8:85a3:08d3:1319:8a2e:0370:7334

How does www.google.com become 66.249.91.104?

Domain Name System (DNS)

- Every destination on the internet is served by a DNS registration server
  - Keeps a map of names to IP addresses
- DNS servers tell other DNS servers about the names registered with them
  - Loose, redundant network
    - Every DNS server has at least two other servers that it trades information with
    - Very reliable
    - Takes time to propagate

Recall: Every computer has an IP Address

- IP address is like a street address
  - Routes packet through the internet
  - Packet eventually reaches router to which computer is connected
  - Ip address is bound to that router, like your street name is bound to your street

How does mobile internet work?

Dynamic Host Configuration Protocol

- DHCP is a “conversation” between router and computer when computer first connects
- IP from a free pool is assigned to computer
- Computer generally keeps that Ip until disconnected
  - Might keep it longer on a “lease” arrangement
- Not just mobile computers
  - Often used by ISPs to remotely configure IP of clients

DHCP and Security

Why might DHCP make game security harder?

DHCP and Security

- IP is your "return address"
  - Every packet from you contains your IP so the other computer ("host") can return information to you
- When net was new and hardwired, IP blocking was a common solution to bad behavior
- DHCP makes it very easy to "move" and thus avoid recognition
  - Makes "IP Blocking" very difficult on modern net
    - Have to block entire sections of an ISPs address space
    - Lots of innocents are caught in such a block

Questions?

Tomorrow…. The history of multi-player gaming

The Evolutionary History of the Architecture of Online Games

Where game architecture comes from

- Game software has DNA
  - > It carries the history of the industry within it
  - > In order to understand current games, you need to understand the history

Where game architecture comes from

- Game software has DNA
  - > It carries the history of the industry within it
  - > In order to understand current games, you need to understand the history
- Game software usually evolves incrementally
  - > Game development is generally risk adverse
  - > Game development is on tight schedules
  - > Games general vary only in minor way from what came before

Copyright 2008 Sun Microsystems, Inc. All Rights Reserved. Revision A

Where game architecture comes from

- Game software has DNA
  - > It carries the history of the industry within it
  - > In order to understand current games, you need to understand the history
- Game software usually evolves incrementally
  - > Game development is generally risk adverse
  - > Game development is on tight schedules
  - > Games general vary only in minor way from what came before
- Leaps happen rarely but occasionally
  - > Usually by 'cross-breeding' unrelated software

Multi-player games

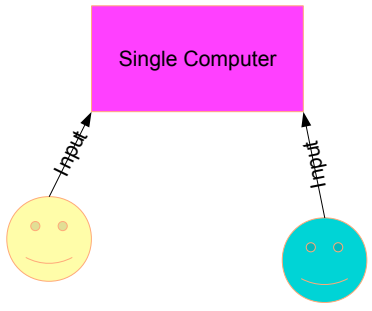
An evolutionary line

The Game Loop

- "Near real time" programming
- Game runs in a tight loop
  - input
  - update
  - calculate
  - display
  - do it all over again.....
- True for almost all games
  - Turn based - wait on input
  - "Real time" - poll input and continue

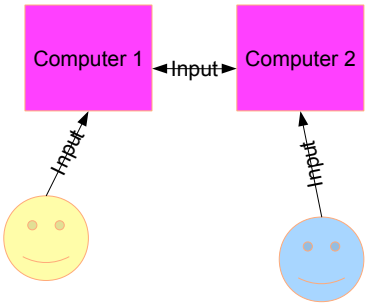
Multi-Player, the first step

- Multiple Players on one computer
- Turn Based
  - > Players each enter their own move sequentially in Update
- Real Time
  - > Each player has their own set of keys or input device
  - > All players are polled in Update



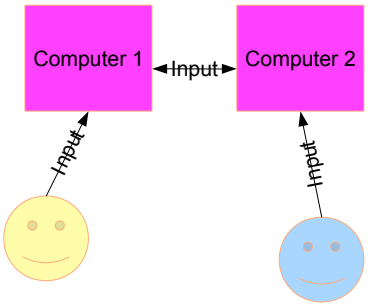
Multi-Station, the first networked games

- Played on LANs
- Non-local players are on virtual devices
  - > Other players input happens on foreign machines
  - > Is communicated over network
  - > Is processed in Update at every machine as if all input was local



Multi-Station, the first networked games

- The "lock-step" model
  - > Every station is running the same game/simulation (sim)
  - > Works because on a LAN, latency is infinitesimal



Stepping out into Cyberspace

- Bandwidth no longer infinite
  - Originally vary constrained
    - 4800 bits per sec
  - Better today
    - So called “broadband”, still can be overload if not careful
- Latency no longer infinitesimal
  - Originally spikey up to abt 500ms.
  - Now locale dependent, but more like 200ms.

Action game needs

- Requires tight synchronization
  - So called “twitch” gaming
  - High pressure on latency
- Requires very frequent communication
  - Min 7 to 10 packets per second per user
  - Puts pressure on Bandwidth
    - Games generally limited to 16 players to limit n-squared
- Can require large amounts of server processing per player
  - Depending on technique used..

N-squared
Its not just a good idea, its the law

- If N players are all sending packets at P packets per second to all other players then:

Total Packets per Second = P*(N^2)

- 4 players @ 10 pps = 160 packets per second
- 12 players @ 10 pps = 1440 packets per second
- 16 players @ 10 pps = 2560 packets per second

...

Internet Play:
Lock Step Pros and Cons

- Pros
  - > Cheat proof
  - > Exact synchronization assured
- Cons
  - > Every player's experience limited by worst case
  - > Handles latency spikes poorly
  - > Handles dropped players poorly
    - > Needs to wait for timeout to determine drop v. spike

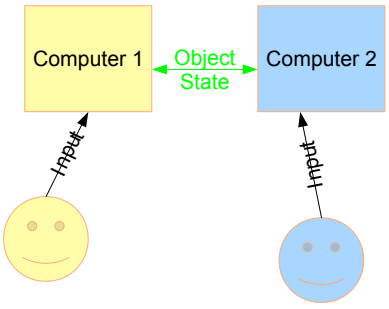
Internet Techniques

Technique 1: Latency Buffering for Lock Step games

- Observation
  - Humans can handle large amounts of predictable latency
  - Humans cannot handle even small amounts of unpredictable latency
  - Mental Model: Steering a battleship
- Technique:
  - Delay ALL rendering by maximum expected latency
  - Render frame when all players data has arrived
- Pros:
  - Exact synchronization across all games
  - All players at same advantage/disadvantage
  - No server intelligence needed (can handle many game sessions at once)
- Cons
  - 'laggy' feeling controls
  - Play is always a worst case
  - Spikes over expected worst case latency stall game

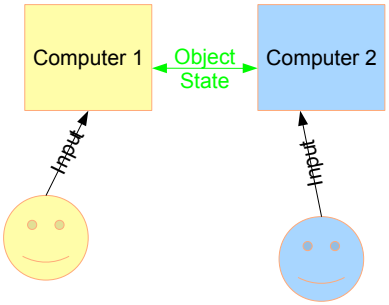
Flight Sims: Open Loop/Asynchronous (Asynch)

- Based on work for SimNet (DIS)
  - > Each system has its own variant world state
  - > Each vehicle is simulated on one machine
    - > Periodic time-stamped state updates sent to others
    - > Lower freq then controller input



Flight Sims: Open Loop/Asynch

- Dead Reckoning
  - > Each sim makes "best guess" at non-local positions
    - Use vehicle model to assist
      - "Tanks don't fly"
  - > Corrects as updates are received
    - > Note: Updates always in past.
  - > Requires conflict resolution mechanism
    - > "shooter decides"



Internet Play: Open Loop/Asynch Pros and Cons

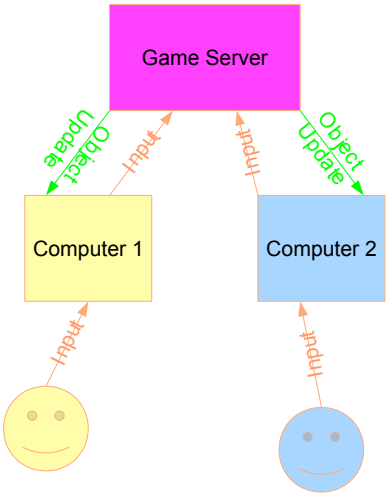
- Pros
  - > Good at hiding latency
    - > Smooth predict/correct over many frames
  - > Better bandwidth control
    - > Can communicate less often
      - 'shape' by distance
      - Out of sight, out of mind
- Cons
  - > Prone to cheating
    - > Need to trust sender as to position
    - > Need to trust shooter as to hit/miss
  - > Occasional 'warping' or other artifacts
- In general, technique used by all vehicle sims

Action Games
Technique 2: Authoritative Server

- Authoritative Server
  - Server = a special client
    - Only server's idea of the world is "real"
    - Prevents cheating if operated by game provider
  - All clients display approximations
  - Server uses "latency compensation" to determine player position at critical moments
    - Basically a "look-back" latency buffer
    - Checks state of game at time on player action packet to determine results

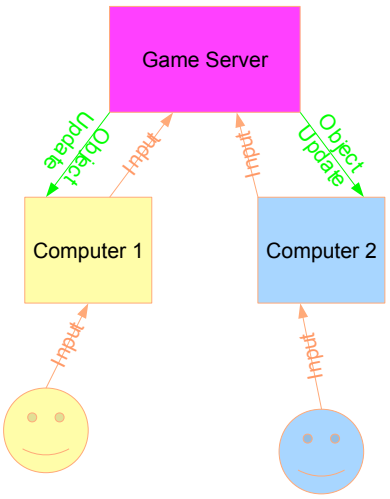
Quake: The first client/server game

- Server runs authoritative simulation
- Clients run open loop/asynch views
  - Really rich "controllers" for server.



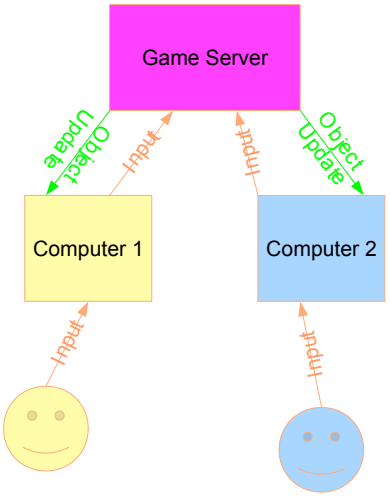
Quake: The first client/server game

- Pros ?



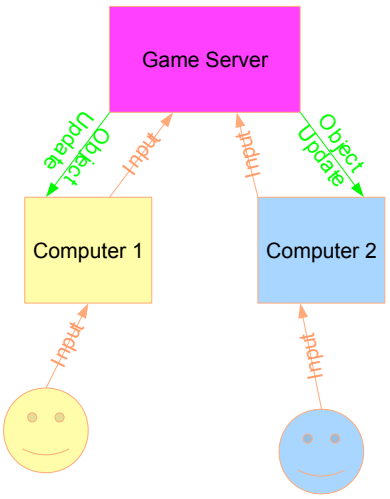
Quake: The first client/server game

- Pros
  - > Cheating is much more difficult
  - > Still not totally impossible
    - > Aimbot
- Cons ?



Quake: The first client/server game

- Pros
  - > Cheating is much more difficult
  - > Still not totally impossible
    - > Aimbot
- Cons
  - > What looks like hit to shooter can miss
  - > "Low Ping Bastard" (LPB) effect



Discovery

- How do you find a game people to play with?

Game Discovery: LANs

- On LAN, players communicated with broadcast
  - First, broadcast play
    - Only one game session per LAN
  - Later, broadcast discovery, unicast play
    - Multiple sessions per LAN

Game Discover: WANs

- In Cyberspace, no one can hear you broadcast
  - On Internet, players need each others IPs
  - Initially, player entered manually
    - Found each other through IRC
  - GameSpy offers discovery service
    - Programmatic, but still over IRC
    - Simple directory server plus chat
    - Funded by advertising on client
  - TEN and MPath offer complete services
    - Net APIs and star architecture comm servers

Game Discovery Today

- TEN and MPath are gone
- Gamespy
  - Industry standard
  - has expanded data services
  - Now has comm API
    - Thin wrapper over peer to peer TCP/IP and UDP
    - Does UDP socket introduction through IRC
  - Licensed per game, advertising in Gamespy client
    - Most games don't use the Gamespy client
- Xbox Live/ PC Live
  - Microsoft's attempt to get into the TEN/MPath space
  - Yearly fee, electronic retailing

Unity Networking

Notes and observations based on what we have learned so far.

Unity Mechanisms and Peer to peer Examples

- Unity provides TCP/IP and RPC calls.
- Unity peer to peer example game mixes latency buffering and psuedo dead reckoning.
  - All other players are latency buffered
  - In order to try to avoid control lag, local player is NOT buffered, but actions are displayed immediately

How can this fail?

Example of Unity peer to peer networking failure:

- Two soccer players trying to kick the ball.
  - A sees himself ahead of B because his display of B is back-time but his display of himself is current.
    - A kicks the ball and sends that information out to the world as a position and velocity of the ball
  - B sees herself ahead of A because her display of A is back-time but her display of herself is current.
    - B kicks the ball and sends that information out to the world as a position and velocity of the ball
  - A receives a ball motion packet from B later then his kick and changes the state of the ball
    - Sudden “warp” effect
  - B receives a packet ball motion from A later then her kick and changes the state of the ball
    - Sudden “warp” effect
- A and B show a warp and are still both out of sync.

Why is this solution wrong?

- Unity docs suggest giving each non-player object a single player controller on creation

How can this fail?

Why is this solution wrong?

- Unity docs suggest giving each non-player object a single player controller on creation
- Latency is doubled for all non local objects
  - A kicks a ball belonging to B. A cannot update it but must send a message to B saying “I kicked this”.
  - B buffers that message for latency L in its latency buffer. When A actually reaches the ball on B’s screen, B calculates the physics and sends the result back to A.
  - A similarly buffers that action for latency L until that time is displayed, when A *finally* sees the result of the action.
  - Result: Major physics lag on any object not locally controlled.

Canonical Mistake

- Mixing Time Frames
  - The further apart those frames, the more obvious the errors will be and the harder they will be to correct.
  - No authoritative server means not having any ‘fair’ mechanism to determine who is right.

Unity with authoritative server

- Better because at least there is a “right” answer
- To do properly would require dead-reckoning
  - Players all get posts about the past, predict the present
- Problem: Unity provides no direct access to the physics engine.

Why is this a problem?

Unity with authoritative server

- Better because at least there is a "right" answer
- To do properly would require dead-reckoning
  - Players all get posts about the past, predict the present.
- Problem: Unity provides no direct access to the physics engine.
  - Dead reckoning requires prediction
    - remember: data is in the past, present is always predicted
  - Physics is always applying forces (drag etc)
    - This is deterministic
    - BUT too hard to calculate if the physics engine is not available
  - Unity example attempts to use simple newtonian prediction (no forces applied)

Result?

- Very poor prediction over any significant period of time
  - Only works if you keep time short
    - Means flooding system with update packets
    - Adds to bandwidth issues and processing costs
- What would work better?
  - Send position, and vector of motion only when a force is applied.
  - Use the physics engine to predict current state from that
  - Unfortunately Unity makes this impossible
    - Hides physics engine
    - Hides application of forces.
      - Arbitrarily sealed classes make this impossible to intercept

MUDs and MMOs or..

"The British are Coming!"

What this lecture is about

The Evolution of MUDs and MMOs

Lecture Overview, Day Two

- The evolution of the MMO
  - > From MUD to WOW in 30 minutes
- The Difficulties facing today's MMO developers
  - > The motivations for Project Darkstar

MUD's and MMOs

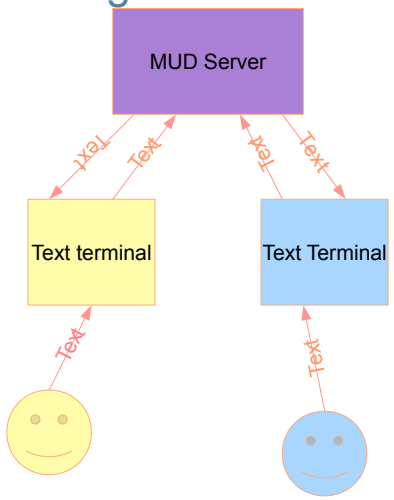
Foreign DNA

Day 3:
MMORPGs and Web Games

Born to Network

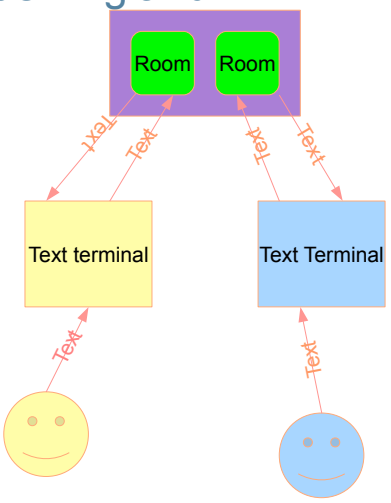
Meanwhile, in merrie olde England

- The Birth of the MUD
  - Multi-user text adventures
  - Event driven servers
  - Textual command based world simulation
    - User submits text, eg "take sword"
    - Server updates world state and sends textual reply
      - Others also see text for world state change



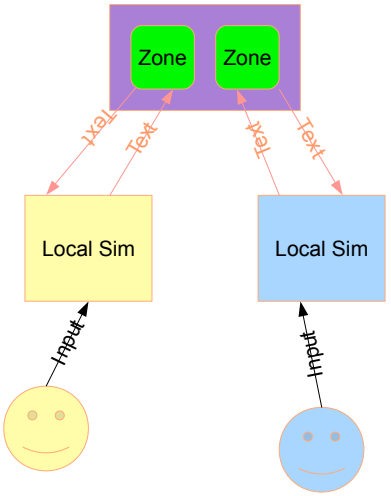
Meanwhile, in merrie olde England

- Used concept of "room" to break down n-squared communication problem
  - Only those in room 'see' changes to room state
  - Only those in room can act on others in room
  - What if you run out of rooms?
    - Virtual 'instanced' rooms



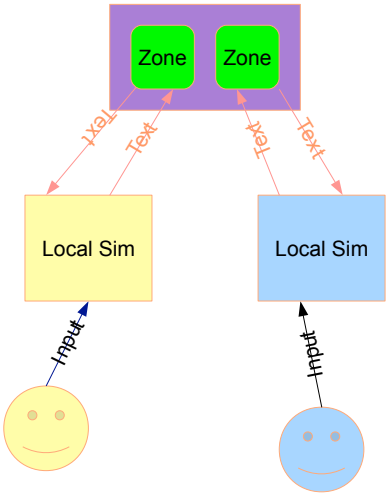
Ultima Online: The Visual MUD

- 2D game for client
  - Levels or "maps" as in previous 2D games
  - Each player on map has a position
- MUD for server
  - Map becomes feature of room (Zone is born)
  - Position on map becomes feature of player object



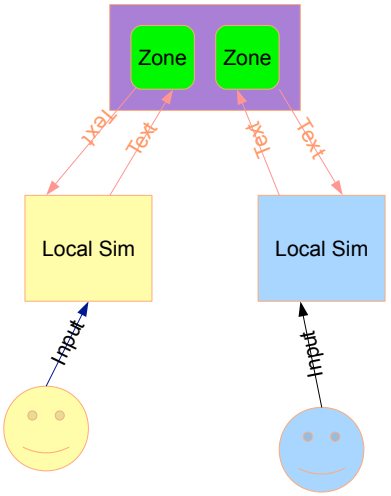
Ultima Online: The Visual MUD

- Issues?



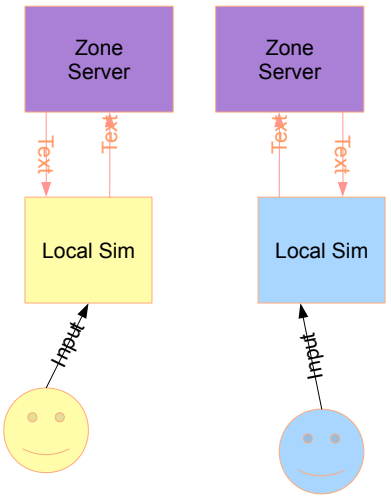
Ultima Online: The Visual MUD

- Issues?
  - > Over-crowding of "popular rooms"
    - > "fire marshal limit"
  - > Scalability limited by power of server
    - > Replicate server
  - > Server crash loses state of whole world
    - > Static worlds
    - > Persistence of users
      - Inventory
      - Experience
      - Quest flags



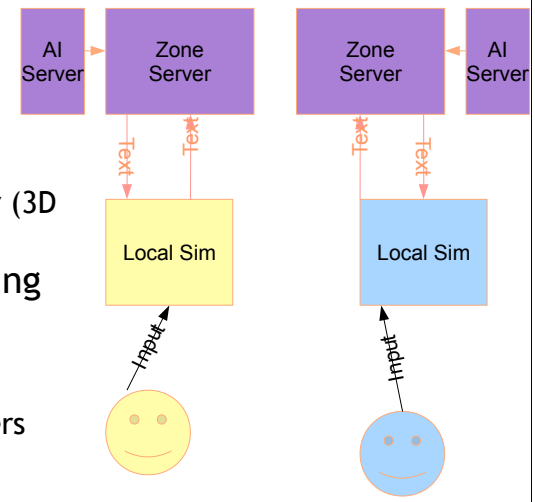
Everquest (EQ): The birth of the Shard

- EQ needed more power
  - > More users
  - > More work per user (3D world)
- Solved by clustering
  - > Server per Zone
  - > One cluster is called a 'shard'
    - > Shard is represented to user as one 'server'
    - > Terminology left over from UOL



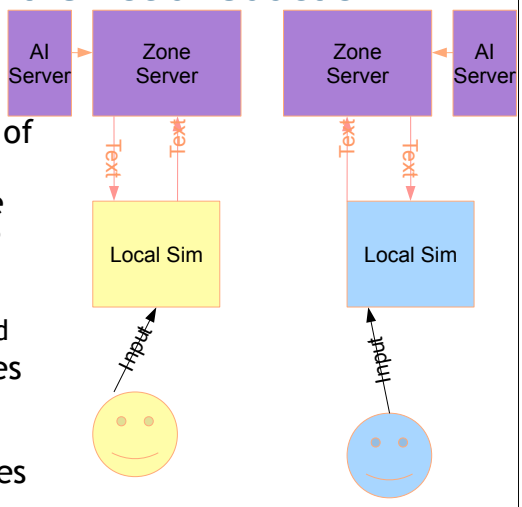
Everquest (EQ): Further load reduction

- EQ needed more power
  - > More users
  - > More work per user (3D world)
- Solved by clustering
  - > Moved MOB AI to separate server
    - > A system "player"
  - > Other special servers
    - > Commerce
    - > Chat
    - > Physics (CoX)



Everquest (EQ): Further load reduction

- Issues?
  - > Many single points of partial failure
  - > Zone server failure means loss of zone state
    - > Like UO but only partial loss of world
  - > Over crowded zones
    - > Return of the fire marshall
  - > Under utilized zones
    - > Wasted CPU resources



Phantasy Star Online: The rebirth of the Virtual Room

- Question: Can we do better scaling then shards?
- PSO Answer: Mission Instancing
  - > One standard zone as a “hub”
    - > Chat
    - > Create parties
    - > Get a 'mission'
  - > Mission is a virtual zone
    - > Created when party enters
    - > Destroyed when party leaves
    - > Limits n-squared to max party size
    - > Only has state while occupied
      - Can be run on a random machine from a pool

That's the state of the art today

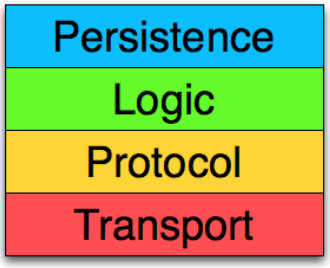
- Various minor tweaks
  - > Incremental improvements
  - > Different mixes of techniques
- Things to remember
  - > Game development is a me-too business
    - > Technical evolution happens slowly due to risk
    - > Mostly focused on client experience
  - > Architectural innovation happens elsewhere
    - > Biggest leaps are usually the adoption of techniques already proven elsewhere

Part 2: Enter the Web

With the web, came web based games

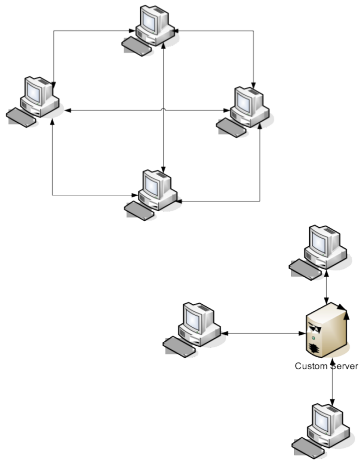
- New Problem: RequiresMassive and Cheap scalability
  - Not 10s or 100s of users per server but thousands
  - Practical reasons, huge audience
    - Farmville as 13 million distinct daily users
  - Economic reasons
    - Only 3% - 5% of users ever pay anything
- Built on enterprise web technologies
  - Thats what the first developers knew
  - Have been handling scale for a long time
  - Problem: State is much more an issue in games

Layers of the Enterprise Stack



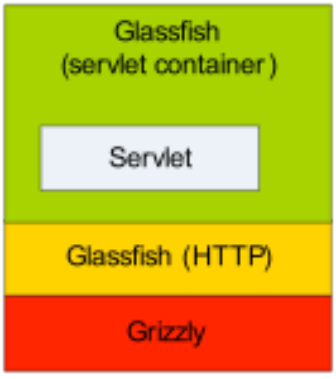
Transports

- Socket Libraries
  - Symmetrical
    - Peer to Peer
    - 'Peer' could be a custom server
  - "Tame" TCP/UDP
    - `May or may not scale
  - Could be custom protocol
    - Eg. Sliding window UDP
  - Could provide discovery
- Providers
  - Gamespy
  - Various Open Source
    - Sun Grizzly Library
    - JGN
    - others



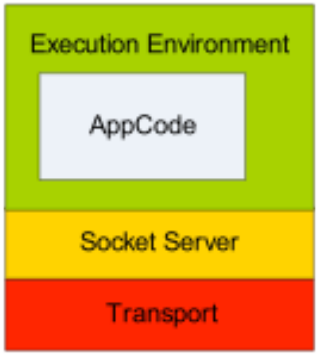
Protocol: HTTP

- Code that responds to HTTP requests
- Glassfish 3 or Tomcat
  - Built ontop of Sun Grizzly TCP/IP library
  - Servlets add programmability (Logic)
  - Supports Standard Java persistence models
    - JDBC, JDO, Hibernate, etc



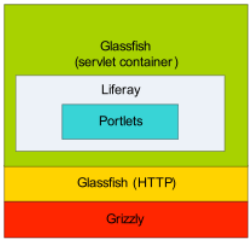
Execution Environments

- Container Systems
  - Built on top socket servers
  - Highly scalable execution
  - Provide an application model
    - EJBs/ Servlets
    - Portlets
    - Darkstar/Reddwarf Applications



Portlets

- Visual widget, part of a portal
  - Column oriented
  - Database backed
  - Has portal DB backing it
- Liferay
  - Is actually a servlet
  - Supports Standard Java persistence models
  - JDBC, JDO, Hibernate, etc
- Others
  - PHP, etc



Standard Java Persistence Models

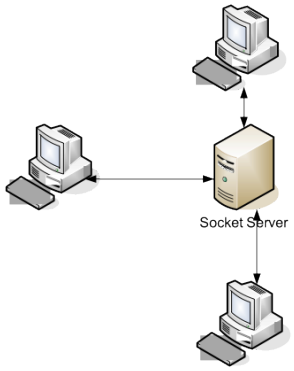
- JDBC
  - SQL interface,
  - Supported by most RDBMs
    - ODBC bridge available
- JDO
  - Object database interface
  - Supported by many databases
- Custom ORMs
  - Hibernate
    - Supports many popular RDBMs
  - Per Database vendor

Game Oriented Technologies

- Standard web stack has some serious limitations
  - Not designed for multi-player
    - Web page viewers are all independent users
  - All state has to be checked back to the database
    - slow and costly

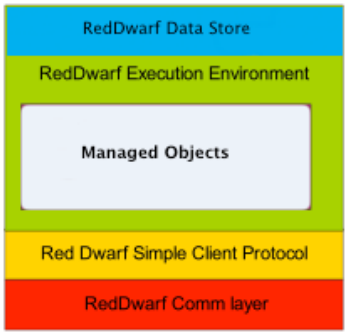
Game Oriented Technologies: Socket Servers

- Star-network hub
  - Built on top transport library
  - Highly scalable
    - Thousands of connections
  - Higher level net concepts
    - "Rooms" or "Channels"
  - Generally provides discovery
  - Could provide limited persistence
  - Limited to no execution support
- Providers
  - Electro-Tank
  - Smart Fox
  - Red Dwarf
    - Subset of Red Dwarf functionality
  - Others



Game Oriented Technologies: Darkstar/RedDwarf

- Open source game server
  - Designed for low-latency response
- Runs "ManagedObjects'
  - Ala Project Darkstar
  - Almost POJO
  - Event driven
- Supports connected sessions
- Transparent Persistence
  - Non-relational
- Transparent Multi-tasking



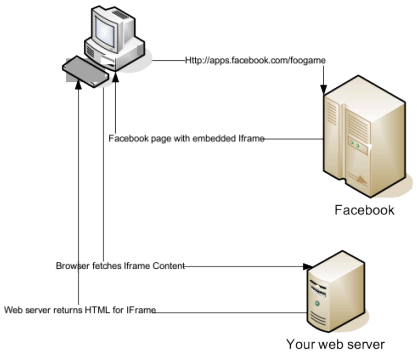
Facebook Integration

Or 50 Ways to Screw Your App

The Facebook Model

- Like the Web… only much worse
- Two Ways to be hosted on Facebook
  - The Old Way
    - FBML
    - Just Don't.
  - The New Way
    - IFrame

How IFrame Works



Facebook Issues

- Facebook is HIGHLY unreliable at doing anything but serving its own pages
  - Its overloaded
  - They break the API weekly in new ways
- Plan your game to rely on as little facebook functionality as you can get away with
  - IFrame
  - Avoid using their UI calls
  - Be aware that they put ‘security’ limits on Javascript
- Plan for facebook to fail
  - Have good fallbacks for any place you call them