Multiplayer Online Games

Dietmar Schreiner

Institut für Computersprachen
Abteilung für Programmiersprachen und Übersetzer

schreiner(at)complang.tuwien.ac.at
Overview

- Introduction
  - Business Facts
  - History
  - Classification

- Communication
  - Requirements
  - Latency
  - Peer-2-Peer models
  - Publisher-Subscriber models
  - Client-Server models

- Load Balancing
  - Interest Management
  - Zoning
Introduction
Online Games – Business Facts (1)

- Game-Industry has grown up to an important social and economic factor
  - Annual turn-over (OECD)
    - 2003: 21 Bil. US$
    - 2007: 35 Bil. US$
  - Massive Multiplayer Games turn-over
    - 2005: 2 Bil. US$
    - 2007: 3,2 Bil. US$
    - 2011: 6,8 Bil. US$ aprox. (DFC International)
  - World of Warcraft
    - 2008: 8+ Mio. Active accounts
      - Monthly fee €12,99 - €10,99
Online Games – Business Facts (2)
Online Games – Business Facts (3)
Online Games – A brief history

- **1958** Tennis for Two; William Higginbotham
  - Oscilloscope and two paddles

- **1962** Two Player Space War; Rick Blomme
  - First networked multiplayer game

- **1972** Pong; Atari

- **1978** Multi User Dungeon (MUD); Trubshaw
  - **1984** MUD1; Compuserve
    - first commercial MUD

- **1984** Island of Kesmai
  - First MMORPG, Pay by use

- **1993** Doom; id

- **1997** Ultima Online; Origin Systems

- **2004** World of Warcraft; Blizzard
Classification required to focus on specific aspects of games, e.g.

- **Economic issues**
  - Business model, ROI, Distribution

- **Game-play and game-mechanic**
  - Real-Time, Turn-Based
  - Linear, Open
  - Simulated, Scripted

- **Resource consumption**

- Etc.
Classifications relevant for this lecture

- **By type/genre**
  - **Role Playing Games (RPG)**
    - Mixture of Soft-RT and Turn-Based
    - Complex World-Simulation
  - **Sports, Simulation**
    - "Hard" Soft-RT
    - Complex physical calculations
  - **Strategy**
    - Turn-Based
    - Large amount of entities/units
  - **Action, FPS**
    - "Hard" Soft-RT
    - Simple world simulation and physics
Computer Games – Class Characteristics (1)

- Role Playing Games
  - Incorporate two discriminative phases of game play
    - Exploration phase
    - Combat phase
  - Exploration phase
    - similar to action games or first-person shooters
      - less critical than FPS
  - Combat phase
    - turn based game play
    - fixed turns, typically hard-deadlined, but long time period
Sports/Racing Games
- Simulations are mainly based on time-discrete physics simulation
  - Timing has major impact
    - “Hard” Soft-RT
- Simulation code allows prediction
Real-Time Strategy Games

- Construction phase
  - research/tech-trees, base, defensive structures
  - „OffLine“

- Exploration phase
  - areal recon
  - long term tasks

- Combat phase
  - micro management
  - fast paced, time critical (?)
Action and FPS Games

- Simplified physics
- Fast-paced
- Large amount of high-speed particles
  - Timing is extremely critical

- Prediction possible
- MMOG?
Computer Games – Classification (3)

- **By number of players**
  - **Single-Player**
    - One player
    - No real player-2-player interaction (e.g. High-Scores)
  - **Multi-Player** *
    - 2-64 players
    - Max. One server or p2p-architecture
    - One dedicated play (session)
  - **Massive-Multi-Player** *
    - “Unlimited” players
    - Multiple (transparent) servers
    - Typically continuous play
Multiplayer Online Games

Communication in MMOGs
MMOGs - Requirements

- MMOGs are typically commercial games
- Requirements on commercial games are
  - High availability
    - Min. (zero) down-time
  - Security
    - Identity theft
    - Fraud / Cheating
    - Data theft
  - Low latency and round trip time
    - Sensitivity depends on classification
    - Prediction and compensation not always possible
  - Low bandwidth demand
MMOGs – Latency (1)

- Time difference between reaction and initiation of an event
  \[ t_{\text{Latency}} = \Delta t_{\text{Action}} = t_{\text{Reaction}} - t_{\text{Initiation}} \]

- Round Trip Time is time between request and response
  - For asymmetric (transport) latency:
    \[ RTT \neq 2t_{\text{Latency}} \]

- Reasons for latency
  - Finite propagation delays
    - Physical limitations of medium
  - Serialization
    - fragmentation, bit-stuffing and framing
  - Queuing delays
    - Queues at routers, gateways, etc.
Player Actions and Latency

from Claypool and Claypool, Latency and Player Actions in Online Games, 2006
Latency Compensation

- Prediction
  - Extrapolation
    - Render current position
    - Calculation of future position
  - Intrapolation
    - Render last position
    - Calculate current position (that becomes the last one for the next turn)

- Time manipulation
  - Clients issue commands with time-stamps
    - Clock synchronization
  - Server calculates world models for specified time
    - Server may rollback world state
MMOGs – Latency (4a)

- **Sports/Racing Games**
  - fast paced time discrete simulation
  - Real-Time

from Yasui et al., Influences of Network Latency and Packet Loss on Consistency in Networked Racing Games
Dead Reconing

- Calculation of future position via integration of objects kinematic properties.
  - calculation of position
    \[
    p' = p + \dot{p}t + \frac{\ddot{p}t^2}{2}
    \]
  - calculation of speed
    - dampening is applied to overcome numeric effects
    \[
    \dot{p}' = \ddot{p}d + \ddot{p}t
    \]
    \[
    \dot{p}' = \ddot{p}dt + \ddot{p}t
    \]
Sports/Racing Games

Graph showing the average difference in positional error between two cars versus average additional delay [ms] for different conditions (NC, DR, Causality, DR+Causality) with standard deviations of 10 ms and 30 ms. The graph includes a 95% confidence interval.
MMOGs – Latency (5)

- Action/FPS

Graphs showing the effect of packet loss on hit fraction and induced latency on hit fraction.

From Beigbeder et al., The Effect of Loss and Latency on User Performance in Unreal Tournament 2003.
Role Playing Games
- Time frames of player actions determine data bursts
- Play actions depend on state updates
  - actions are deadline but have a wide time frame
MMOGs – Latency (7a)

- Real-Time Strategy
  - Three phases (Build/Construct)

from Claypool, The effect of latency on user performance in Real-Time Strategy games
Real-Time Strategy

Three phases (Explore)

from Claypool, The effect of latency on user performance in Real-Time Strategy games
Real-Time Strategy

Three phases (Combat)

from Claypool, The effect of latency on user performance in Real-Time Strategy games
MMOGs – Latency (8)

Summary

from Claypool and Claypool, Latency and Player Actions in Online Games, 2006
Summary

The graph illustrates the performance of different types of MMOGs as a function of latency. The graph shows three types of avatars:

- **Omnipresent (ex: RTS)**: This type of avatar maintains high performance with low latency, showing a steep decline in performance as latency increases.
- **Third-Person Avatar (ex: Sports, RPG)**: This type of avatar has a moderate performance decline as latency increases, with a noticeable drop-off as latency goes up.
- **First Person Avatar (ex: FPS, Racing)**: This type of avatar experiences a significant drop in performance even at low latencies, indicating high sensitivity to latency variations.

The x-axis represents latency (in msec), while the y-axis represents performance.
Multiplayer Online Games

System Architectures at Network Level
Decentralized Architecture

Pros

- No single world representation
  - No single point-of-failure
  - No server-sided bottle-neck
- High scalability
- Low latency (direct connectivity of peers)
- Low Bandwidth requirements

Cons

- Cheating hard to prevent
- Persistence of game world is hard to achieve
- Trustworthy authentication is nearly impossible
MMOGs – Peer-2-Peer (2)

- Bandwidth requirements
  - Smaller than in Client-Server Systems
  - Upstream = Downstream

\[ B_C = \frac{2(N-1)L_U}{T_U} \]

- \( N \) … Number of peers
- \( L_U \) … Size of update message
- \( T_u \) … Update interval

- Bandwidth consumption is linear to number of peers
MMOGs – Peer-2-Peer (3)

- Peer-2-Peer systems are well suited for
  - Multi-player games
  - LAN
  - Small groups of players that know and trust each other
  - Sports/Simulation
  - Action/FPS

- Peer-2-Peer systems are not suited for
  - MMOGs
  - RPGS
    - No persistent world model
    - No trust
**MMOGs – Peer-2-Peer with CA (1)**

- PP-CA is a full-fledge Peer-2-Peer architecture
- A central arbiter is incorporated into the P2P network
- CA receives all state changes
  - Plausibility checks
  - Veto and notification on violations only
  - Peers have to roll-back on veto
- Improved fraud/cheat detection
Bandwidth requirements

- Similar to standard peer-2-peer model

\[
B_{Down} = \frac{(N-1)L_U}{T_U}
\]

\[
B_{Up} = \frac{NL_U}{T_U}
\]

\[
B_C = \frac{(N-1)L_U + NL_U}{T_U} \approx \frac{2NL_U}{T_U}
\]

Bandwidth consumption typically linear to number of peers

- Worst-case if each peer creates a violation per update period

- Increase bandwidth consumption in quadric order
Augmented Peer-2-Peer architecture

- Set of peers is clustered to represent a subset of the game world (states)
  - Peer is client or zone owner
  - DHT provides functionality for
    - Data placement / Data replication (backup)
    - Lookup
MMOGs – Zoned Federation Model (2)
Zoned Federation Models are well suited for

- MMOG
  - Scaleability through Zoning and DHT
- RPG
  - Operator has to keep control of Zone Owner Nodes
- Strategy Games

Zoned Federation Models are not suited for

- Sports/Simulation
- FPS
  - Due to Zone-owner bottle-neck
Communication is based on content
- Creation/Production
- Reception

Typical relations between creator and recipient are 1:n or even n:m
- In client-server systems only 1:1 relations exist

Publisher-Subscriber Model
- Recipients subscribe to “channels” of interest
- Publisher inform their subscribers only
- Selection of channels is crucial for over-all system
Publisher propagates all relevant state information via channels

- Each channel represents one area of interest
- Clients subscribe to channels of interest
  - Neighbouring
  - Filtering
- Clients sent state changes to publishers of their channels
  - Publishers process state change
  - Publishers replicate states

Reduces bandwidth consumption but provides point-to-point connections
- Often used within server-clusters
MMOGs – Publisher Subscriber Model (3)
MMOGs – Client Server Model (1)

- Most used model in today’s MMOG
  - Missing alternatives
  - Harsh constraints due to restrictions
    - Payed service

- Clients are directly connected to one central server (cluster)
  - World state and calculation is located at server
  - Client is viewer only
  - Server in charge of
    - World management
    - World simulation
    - Game execution (rule based game state changes)
    - AAA (Accounting, Authentication, Authorization)
MMOGs – Client Server Model (2)

- **Bandwidth requirements**
  - **Client**
    
    $$ B_C = \frac{L_u + NL_U}{T_u} = \frac{(N+1)L_U}{T_U} $$
    
    - Bandwidth usage of client is linear to number of clients
  - **Server**
    
    $$ B_S = \frac{NL_U + N(NL_U)}{T_u} = \frac{N(N+1)L_U}{T_U} $$
    
    - Bandwidth usage of client is quadric to number of clients
MMOGs – Client Server Model (3)

- **Pros**
  - Cheating/Fraud is nearly impossible
  - Participation only with valid (paid) account
  - Critical updates are typically at the server, not at the clients
    - Game balance/mechanic
    - Quests
  - Persistent world model
    - User data / account state

- **Cons**
  - Single Point of Failure (SPoF)
  - Bandwidth usage
    - Bad scalability
  - Costs (cluster, farming e.g.)
Multiplayer Online Games

Load Balancing
Interest Management - Motivation

- MMOGs host large number of simultaneous players (10k+)
  - Player want to participate in ONE persistent world
  - Current game servers host ~1000 players / server
    - Quests are instantiated
      - Players may not meet even if they are at the same location
    - Global areas may become over-populated
      - Latency becomes too long
      - Games become unplayable

- IM is used to create one persistent, scalable world
Interest Management - Basics

- IM is based on
  - Partitioning of game world
    - Space based (zoning)
    - Object based
  - Publisher-Subscriber architecture

- Aura-Nimbus principle
  - Aura = region surrounding each object in game world
    - E.g. visibility A, B
    - Calculation costly
  - Nimbus = action
    - E.g. sword hit
  - Interaction of nimbus only possible if aurae intersect
    - Supports publisher-subscriber model
      - “If aurae intersect, subscribe to channel of nimbus”
Interest Management – Region Based

- Partitioning of Gameworld into regions of interest
- Regions of subscription are all regions of interest of one client
  - Calculated by the IM system

- Interest Management Algorithms
  - Euclidean Distance Algorithm
    - Aura is circle centered at object
  - Square Tile Algorithm
    - World is divided into squares
    - Object is assigned to center of one square and 8 surrounding ones
  - Hexagonal Tile Algorithm
    - World is divided into hexagons
    - Equidistant to all neighbors
  - Ray Visibility
    - Raycasting of viewing-frustrum
    - Objects are of interest if intersected by a ray
Zoning - Basics

- Persistent world are typically partitioned (cells/zones) and distributed over multiple servers
  - Reduction of load for each single server
  - Reduction of costs
  - Improvement of latency

- Cell size is critical for system behavior
  - Too many cells lead to overloaded cells
  - Too less cells lead to too many channels

- Partitioning may be
  - Static
    - May still lead to overloaded servers
  - Dynamic
    - Problem of cell/zone shifts and channel subscriptions
Zoning – Microcells (1)

- Game world is partitioned into zones
  - Each zone is located on one server
- Zones are partitioned into microcells
  - Microcells may be transferred across servers
    - Zones may change their size dynamically
    - Load between zones may be balanced at run-time
Zoning – Microcells (2)

- Management of microcells
  - Load sharing: avoid load before it occurs
  - Load balancing: balance load, if it occurs
    - Balanced deployment
      - Monitor number of players and actions
      - microcells with highest load are assigned to server with lowest balance
      - Con: now geographic neighborhood considered
    - Clustering deployment
      - Like balanced deployment
      - Considers neighborhood: deploys clusters
    - Simulated annealing deployment
      - Used to optimize running servers
      - Move microcell to other server, measure load, create permutations of microcells
    - Optimal deployment
      - Use ILP to calculate optimal microcell deployment
      - Con: Costly calculation
Zoning - Shards

- Partitioning of real world
  - Instantiation of one server cluster (shard) per partition
  - Shards share account data
    - Users may be routed to different shards, depending on shard load
    - Zoning within shards possible
- Implemented by World of Warcraft
- Pros
  - no SPoF
  - Balancing by routing of login
- Cons
  - Shards are independent
  - limited instances
Multiplayer Online Games

Questions ?
Comments ?
Ideas ?