Matrix: Adaptive Middleware for Distributed Multiplayer Games

Rajesh Balan (Carnegie Mellon University)
Maria Ebling (IBM Watson)
Paul Castro (IBM Watson)
Archan Misra (IBM Watson)

Motivation: Massively Multiplayer Online Games

- > 3 billion dollar industry annually
- Many players
  - Blizzard, Electronic Arts, Microsoft, Sony
- Incredibly hard to get right
  - Game companies know games
  - Infrastructure requires systems people

Scale of Problem

- ~ 1,000,000 concurrent players
  - ~10,000,000 subscribers
- ~ 10,000 servers
- Mix of shooting & adventure games
  - Different latency/consistency constraints

Goal: Clean Separation

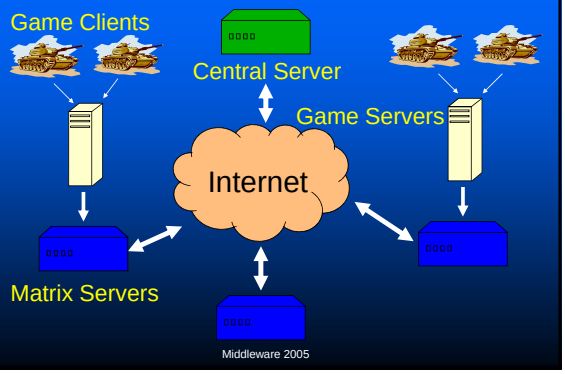
Game Developers
- Concentrate about Game issues
  - Design
  - Graphics
  - Fun

Infrastructure (Matrix)
- Handle low-level issues
  - Latency
  - Filtering
  - Consistency
  - Reliability

Design Criteria

- Provide good game performance
  - Automatically scale to observed load
    - Handle hotspots and time-based variations
  - Adequate consistency
  - Fast efficient packet routing
  - No game player perceived artifacts
- Attractive to Game developers
  - Easy to use
  - Does not add any security holes

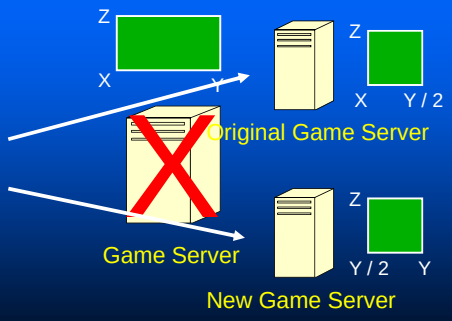
Matrix Architecture



Matrix Benefits

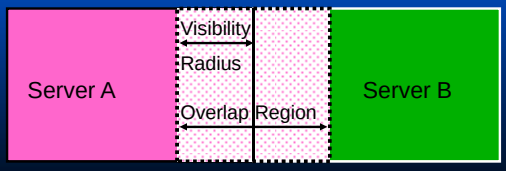
- Client – Server architecture
  - Transparent to game clients
  - Requires no client – server protocol changes
- Fast efficient routing
  - Game packets are spatially tagged
    - Overlap regions used to determine consistency set
    - O(1) route lookups using central server
- Automatic Load Balancing

On-Demand Load Balancing



Maintaining Adequate Consistency

- Insight ➔ Players have limited visibility
  - Use that to create overlap regions between servers
- Only route player packets inside overlap regions
  - Use Central Server to do O(1) routing



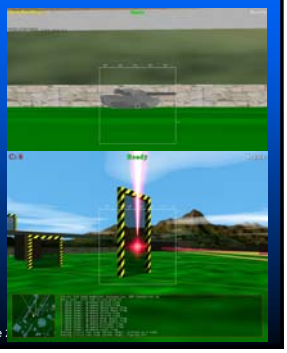
Easy to Use

- Game Clients (Minimal changes needed)
  - Support for dynamic switching of game servers
- Game Servers (Requires more work)
  - Must spatially tag packets
  - Send packets to / from Matrix server
  - Convey load information to Matrix server
  - Accept new map ranges / clients

Status and Evaluation

- Matrix implemented in C
- 3 real multiplayer games modified
  - Bzflag
  - Quake 2
  - Daimonin (RPG)

8 - 16 hours per game



Evaluation Highlights (1)

- Automatic load balancing works
  - Superior to static partionings
    - when hotspots occur
    - Extra players join the game
- Matrix has low overhead
  - Central server is not a bottleneck
  - Overlap regions reduce network traffic

Evaluation Highlights (2)

- Scalability analysis performed
  - Size of overlap region is crucial
  - System I/O limits scalability
- Matrix is fully playable
  - User study with Bzflag and real players
  - Effects of load are greatly minimized
  - Matrix operation mostly invisible

Conclusion

- MMOGs are a huge emerging market
  - large number of technical challenges
- Matrix designed to simplify game development
  - Simple usage model
  - Handles routing, consistency and automatic load balancing

Introduction

- Infrastructure
  - Massively multiplayer games
  - Quake, Everquest, Final fantasy etc.
- Focus is on backend technology
  - Not the game itself

Solution

- Build a generic infrastructure
  - Supports spatial applications (games)
  - Handles all QoS issues related to games
    - Latency of packet routing
    - Consistency
    - Reliability

On-Demand Nature

- Similar to Clash
  - But no hashing is done
- Game server runs on 1 matrix
  - When load exceeds threshold →spawn another matrix/game server combo
  - When load decreases, remove server
- Only local decisions are made

On-Demand (details)

- Server provides load info to matrix
- Matrix then decides how to split game map
- Game server is informed of new map
  - Checks if clients need to be switched
  - Consults matrix if so

Filtering

- Performed locally by each matrix
  - By checking spatial tag of each packet
  - Helps to decrease network load
  - May also help with cheating

Consistency

- Games have weak consistency
  - Consistent only within a small range
- Provided using notion of overlap regions
  - Calculated by central server
  - Packets in overlap region forwarded to other matrixes that care

Reliability

- No special technique
- Reuse state of the art
  - Multiply replicated matrix/game server sets behind a network direction switch
- Matrix servers can also heartbeat neighbours
  - Perform local repair
  - Global repair requires central server

Evaluation

- Ongoing
- Show problem of single server
  - Overload when clients increase
- Show benefits of on-demand
  - Less servers required
  - Better hotspot handling

Questions?

- Contact Info
  - rajesh@cs.cmu.edu

Role of Central Coordinator

- > 3 billion dollar industry annually
  - “Sims” sold > 7,000,000 copies
- Many players
  - Electronic Arts, Microsoft, Sony
- Incredibly hard to get right
  - Game companies know games
  - Infrastructure requires systems people

Matrix Benefits

- Client – Server architecture
  - Transparent to game clients
  - Requires no client – server protocol changes
- Fast efficient routing
  - Game packets are spatially tagged
    - Overlap regions used to determine consistency set
    - O(1) route lookups using central server
- Automatic Load Balancing

Latency

- #1 metric for most games
- Latency increased with each hop through infrastructure
- Achieve O(1) direct routing
  - via central server
  - Fallback → run OSPF
- Filtering and 1-hop routing ensure best possible latency

Matrix Benefits

- MMOGs are a huge emerging market
  - large number of technical challenges
- Matrix designed to simplify game development
  - Simple usage model
  - Handles routing, consistency and automatic load balancing
  - Has good runtime performance