Scaling Peer-to-Peer Games in Low Bandwidth Environments

Jeff Pang
Carnegie Mellon

Frank Uyeda
U.C. San Diego

Jacob R. Lorch
Microsoft Research

P2P Games

Total Bandwidth (Mbps)

Internet

P2P Games

P2P Quake on a LAN
P2P Quake on a Cable Modem

Not Enough Bandwidth

128 kbps

Not enough bandwidth to send to all peers at required rate choppy and unsatisfying game play

0 100 200 300 400 500

Players

0 50 100 150 200

Total Bandwidth (Mbps)

Players

Players

Players
Our Solution: **Donnybrook**

- **Contributions**
  - Provide insights that enable scaling
  - Design techniques that codify these insights
  - A prototype implementing these techniques

- **User study demonstrates effectiveness**
  - Preserves fun in low bandwidth environments
  - Increases scalability by an order of magnitude

---

**Design Principles**

- **Player attention is bounded by a constant**
  - Example: I focus on my current target
    ⇒ Total attention scales linearly with # players

- **Realism should not be sacrificed for accuracy**
  - Example: "Teleporting" to correct locations is jarring
    ⇒ Don’t always minimize error in because of updates

- **Interaction must be timely and consistent**
  - Example: If I shoot you, you should get hit immediately
    ⇒ Prioritize interaction (i.e., inter-object writes)

---

**Focus Set**

- **Who goes in my Focus Set?**
  - Attention(i,j) = how much a player i is focused on player j
  - Attention(i,j) sent from peer i to peer j periodically
  - Peers with the highest Attention(i,j) go in peer j’s Focus Set

  \[
  \text{Attention}(i,j) = f_{\text{proximity}}(d_{ij}) + f_{\text{aim}}(\theta_{ij}) + f_{\text{interaction-recency}}(t_{ij})
  \]

---

**Guidable AI**

- **Problem**: Best effort peers receive infrequent updates

- **Solution**: Smooth state changes with AI
  - Example: Use existing pathfinding code to make replica move

---
Donnybrook in Action

Setup

Evaluation

Questions
- How much does Donnybrook improve playability in low bandwidth environments?
- How close is Donnybrook in a low bandwidth environment to a LAN ("optimal")?

Methodology
- Modified Quake III to run on Donnybrook
- User study to evaluate user satisfaction
- Simulation to evaluate fairness (See paper)

Total Stay Time

Projected Scalability

Three additional metrics in the paper support these conclusions.
Summary

- Donnybrook enables large-scale P2P games in low bandwidth environments
- Three key design principles:
  - Player attention is bounded by a constant
  - Interaction must be timely and consistent
  - Realism should not be sacrificed for accuracy
- Future work:
  - Constant-size Focus Set doesn’t work if everyone is focused on one player (e.g., the flag carrier in CTF)
  - Use a multicast tree to deliver frequent updates in this scenario

Meta-Conclusions

- We should do more games research
  - Has the market-share and user base
  - Numerous large scale distributed systems problems
- Bounded player attention applicable to other systems
  - Prioritizing data collection in P2P sensor aggregation
  - Varying fidelity in P2P video conferencing
- “End-to-end” evaluation should consider user experience
  - Donnybrook’s replica consistency is not great
  - Users don’t notice or don’t care

Why P2P Games?

- Massively multiplayer games are popular
  - Recent research:
    • Colyseus [NSDI ’06]
    • SimMUD [INFOCOM ’04]
- Centralized First Person Shooter (FPS) games scale poorly
  - Quadratic Bandwidth
  - CPU Limited

Other P2P Games Challenges

- Cheating
  - Migration, Witnesses
    • Bharambe [CMU thesis work]
  - Trusted Computing / Hardware Sensors
    • Johnson et al. [Intel]
- Fault Tolerance
  - DHT self-healing / replica placement:
    • Colyseus [NSDI 2006]
    • SimMUD [INFOCOM 2004]
- Persistence
  - General P2P storage replacements for DB

Game Execution Model

- Game State:
  - Collection of distinct objects (players, missiles, items, etc.)
- Game Execution:
  - Each object has a Think function:
    Think() { ReadPlayerInput(); DoActions(); ... }
  - Execute each object once per frame:
    Each 50ms do {
      foreach object do {
        object->Think();
      }
    }
**Area of Interest Filtering**

- Object Location
- Replica Management
- Object Placement

**Design Comparison**

- Existing P2P game architectures
  - Replicas can be as stale as the update interval
  - Update rate per peer is fixed

- Donnybrook’s solution
  - QoS: Vary update rate per peer, some get priority
  - Guidable AI: Secondary replicas think for themselves

**Pairwise Rapid Agreement**

- **Interaction**: when player A modifies player B (i.e., A performs a write on B)
- **Goal**: modification is consistent and applied quickly
- **Insight**: # interactions scales slowly
  - Occur at human time scales ⇒ infrequent
  - Involve only 2 players ⇒ unicast
- **Solution**: prioritize all inter-object writes
  - Player A sends mod to Player B
  - Player B checks preconditions locally
  - Player B applies mod, sends result to A
- **PRAs required in Quake III**
  - Damage, Death, Item Pickup, Door Opening

**Prediction**

- **Motivation**: state snapshots get stale fast
  - Example: players can traverse the entire diameter of a map in several seconds in Quake III
  - Goal: send prediction of state at time of next expected update
  - Example: predict where a player will be the next update
- **Predicted Properties**
  - Predict position: use in-game simulation to figure how where physics brings player in next second
  - Predict viewing angle: use view angles to estimate the target a player is aiming at
  - Predict Events: use number-of-shots-fired to estimate when a player is “shooty,” etc.

**Guidable AI: Convergence**

- **Problem**: Best Effort peers receive very infrequent updates
- **Solution**: Smooth state changes with AI
  - **Position**: use existing path finding code to make replica move smoothly
  - **Angle**: have AI turn smoothly toward predicted targets
- **Convergence**
  - **Motivation**: Players in focus should be represented more accurately, but should not “warp” to actual position
  - **Solution**: Converge to actual state when receiving frequent updates
    - Focus on player B
    - In player B’s Focus Set, get frequent updates
    - Error(replica, actual) decreases with each update
    - When Error() > ε, use player B’s update snapshots instead of AI
    - Error(a,b) = distance(a.pos, b.pos)

**Setup**

- **User Study Stats**
  - LoBW-Donny vs. LoBW: 12 trials
  - LoBW-Donny vs. HiBW: 32 trials
  - 98 total participants

**Prediction**

- **Motivation**: state snapshots get stale fast
  - Example: players can traverse the entire diameter of a map in several seconds in Quake III
  - Goal: send prediction of state at time of next expected update
  - Example: predict where a player will be the next update
- **Predicted Properties**
  - Predict position: use in-game simulation to figure how where physics brings player in next second
  - Predict viewing angle: use view angles to estimate the target a player is aiming at
  - Predict Events: use number-of-shots-fired to estimate when a player is “shooty,” etc.
**Departure Time**

*Time until first vote*  
![Graph](image1.png)

*Time until second vote*  
![Graph](image2.png)

How long before a player wants to switch?

---

**Fun Score**

*LoBW vs. LoBW-Donny*  
![Graph](image3.png)

*LoBW-Donny vs. HiBW*  
![Graph](image4.png)

Survey: How fun was server A? Server B?

---

**Preference**

![Preference Chart](image5.png)

Survey: Was A or B more Fun?

---

**Fairness**

*Random bots*  
![Graph](image6.png)

*All bots level 5*  
![Graph](image7.png)

---

**P2P Games**

Donnybrook: The Elder Slides  
Beta Test

![Diagram](image8.png)

- Not enough bandwidth to send to all peers every frame
- ⇒ choppy and unsatisfying game play
Outbound Capacity Problem

- Residential broadband is asymmetric
- In many games, players can see many others
- Insufficient upload capacity to send updates at required frequency
  
  ⇒

- Must send updates at lower frequency
- Replicas can be as stale as the update interval

Cost per Peer in Quake III:
Update rate = 20 updates/sec
Update size = ~100 bytes

⇒

Bwidth per peer ≥ 16kbps
⇒

Supportable peers at 128kbps ≤ 8

Focus Set

- Who goes in my Focus Set?
  - Attention(i,j) = how much a player i is focused on player j
  - Attention(i,j) sent from peer i to peer j periodically
  - Peers with the highest Attention(i,j) go in peer j's Focus Set

Departure Time

How long before a player wants to switch?