SWIN BUR NE

CENTRE FOR ADVANCED INTERNET ARCHITECTURES

Achieving Fairness in Multiplayer Network Games through Automated Latency Balancing

S. Zander, I. Leeder, G. Armitage, J. But

<szander,garmitage,jbut@swin.edu.au>

<i_leeder@hotmail.com>

Centre for Advanced Internet Architectures

Swinburne University of Technology

SIGCHI ACE 2005, June 15th-17th

Overview

- Motivation
- Fairness Approach
- Implementation
- Client-site Bots
- Evaluation Results
- Conclusions and Future Work

Motivation

- Multiplayer network games have become very popular and have evolved into some kind of sports
  - Competitions and leagues are very popular and comparable to sporting competitions
  - Professional gamers earn their living just by gaming; they have fans and TV shows
  - Many people playing on amature level take it seriously
- Games requiring fast player reactions are very sensitive to the Quality of Service (QoS) of the underlying computer network(s)
- Fairness is very important
  - Game design (we **do not** talk about this)
  - **Network QoS differences**

SWIN BUR *NE* CENTRE FOR ADVANCED INTERNET ARCHITECTURES

Motivation cont'd

- Focus on fast-paced games e.g. first person shooters where fast player reactions are crucial
- Focus on latency/delay (also called lag)
  - Influence of jitter has not been sufficiently studied
  - Influence of packet loss is much smaller
- Previous work has shown that
  - Efficiency of players decreases with increasing latency
  - Latency differences cause unfairness
- Latency differences are caused by
  - Network access technology
  - Distance between client and server (propagation delay)
  - Congestion in the network

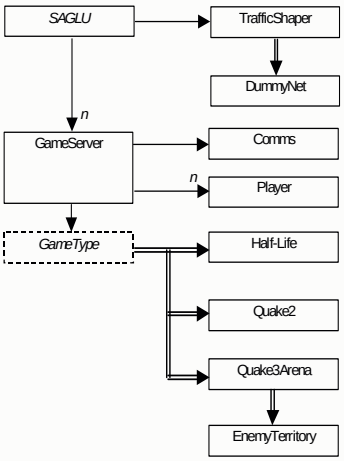
SWIN BUR *NE* CENTRE FOR ADVANCED INTERNET ARCHITECTURES

Fairness Approach

- Implement tool that automatically equalizes players latency by adding artificial lag
- Evaluate effectiveness of approach using human or computer players
  - Compare 'objective' performance metrics (e.g. kill rate) for players (player groups) with different latencies
  - Use hypothesis testing to determine if differences are significant
  - If differences are significant there is unfairness
  - Eliminate factors other than delay

Implementation

- Self-Adjusting Game Lagging Utility (SAGLU)
- Game independent proxy-application between game clients and server
- Extensible multithreaded C++ implementation
- Retrieves player information from the server (IP address, port and latency)
- Equalizes player latencies by adding fake delay



Implementation cont'd

- Delay adjustment algorithm
  - How to determine amount of additional artificial delay?
  - How to add the delay?
  - How frequently to measure player's network delay and adapt the additional delay?
- Implemented simple algorithm

```
for (i in 1:#Players)
    P[i].NetDelay  = getNetDelay()
for (i in 1:#Players)
    P[i].AddDelay = min(max(P[1:#Players].NetDelay), MaxTolerableDelay) –
                    P[i].NetDelay
    if (P[i].AddDelay > 0) setAddDelay(P[i].IPAddress, P[i].Port, P[i].AddDelay)
sleep(AdaptationIntervalTime)
```


Client-side Bots

- Usability trials with human players
  - Necessary for conclusive evaluation
  - Human responses are highly unpredictable (very difficult to eliminate all unwanted factors)
  - Resource and cost intensive (time, equipment, money)
- Client-side computer players (bots)
  - Easy to eliminate unwanted factors e.g. bots behave identical, do not get tired, do not change playing style etc.
  - Far less resources needed
  - Bots are different from humans
    - Incapable of complex navigation (only line of sight)
    - Very effective delay compensation (movement prediction)
    - **But** send real network traffic and therefore should be affected by network delay

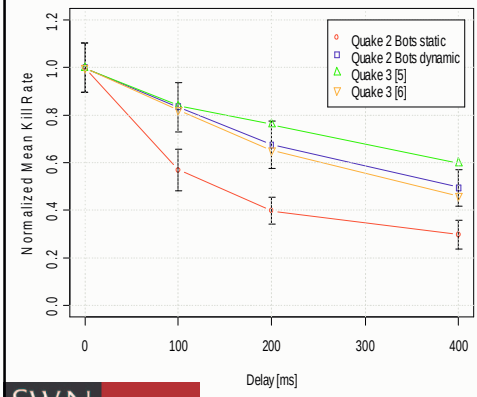
Evaluation

- FreeBSD PC with 2.4GHz and 1.25GB RAM
- Emulate network delay using *dummynet*
  - Static
  - Dynamic (changing every second with exponential distribution)
- Small simple map without obstacles (e.g. lava pits, elevators) and powerful explosive weapons
- 4 bot players (same configuration)
- SAGLU adaptation interval of 5 seconds
- Experiments
  - How do bots react to delay?
  - Do bots experience unfairness?
  - Can SAGLU balance unfair games?
- Average results over 15 games (15 minutes duration)

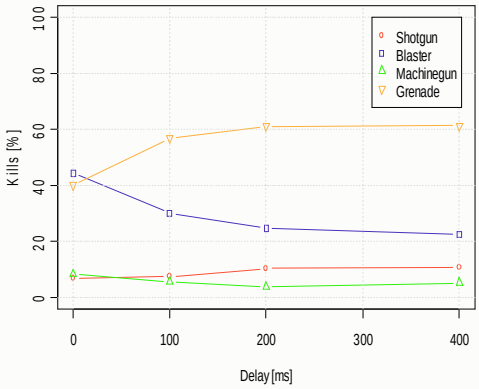
Evaluation Results

- How do bots do react to delay?

Kill rate decrease (bots & humans)



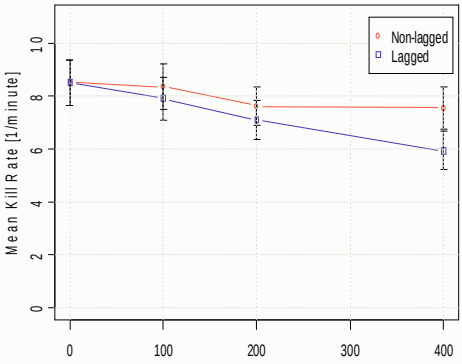
Weapons used for kills (bots)



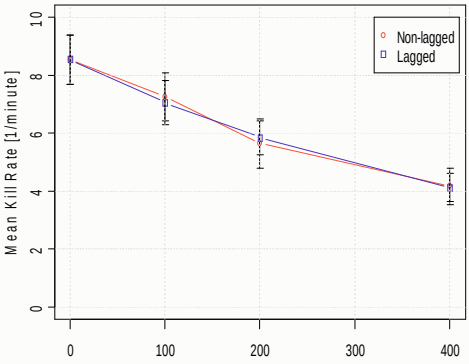
Evaluation Results cont'd

- Do bots experience unfairness and can SAGLU balance the games?

Dynamic delays **without** SAGLU



Dynamic delays **with** SAGLU



Conclusions and Future Work

- Client-side bots behave similar to humans
  - Kill rate decreases and weapons with area effects become more effective with increasing delay
  - Experience unfairness because of delay differences
  - But performance (kill rates) cannot be directly compared
- SAGLU effectively balances the game (http://caia.swin.edu.au/genius/tools/saglu-0.1.tar.gz)
- Usability trials with human players in real networks
  - Refine delay adjustment algorithm
  - Optimize parameters (e.g. adapt. interval, tolerable delay)
- Measure performance and overhead

Thanks for your attention!

SWIN BUR *NE* CENTRE FOR ADVANCED INTERNET ARCHITECTURES