Design and Simulation of a Migration-based Architecture for Massively Populated Internet Games

Gabriele D'Angelo

joint work with

Sandro Pifferi
Ludovico Gardenghi
Luciano Bononi

University of Bologna
Department of Computer Science

Dallas (TX), 29/11/2004

Presentation outline

- Gaming Architecture of MMORPGs
- Problem definition and migration based proposed solution
- Simulation model definition
- Migration mechanism and heuristics description
- Experimental results
- Conclusions and work in progress

Introduction

- In recent years many popular interactive computer games have gained online remote multiplayer functionalities
- **Massive Multiplayer Online Role-Playing Games** (MMORPGs) are the computer-based counterpart of the well-known Role Playing Games (RPGs)
- Big multiplayer games systems are usually based on a pool of servers, geographically distributed across one or more countries
- MMORPG servers usually have to manage *hundreds*, or even *thousands* of avatars co-existing in the *same virtual world*

Gaming Architecture of MMORPGs

- The typical MMORPGs gaming structure is a simple client-server
- The gaming protocol is usually a request/reply scheme: clients send commands to the servers, servers synchronize each other send replies to the clients
- The choice of the server is usually made in the following ways:
  - a round-robin DNS entry, or
  - the “nearest server” selection

    The concept of “**near**” server is usually intended as the “**geographically nearest**” to the client. Another solution is to connect to the “**topologically nearest”** server, i.e. the server reachable by traversing the minimum number of hops

Problem definition

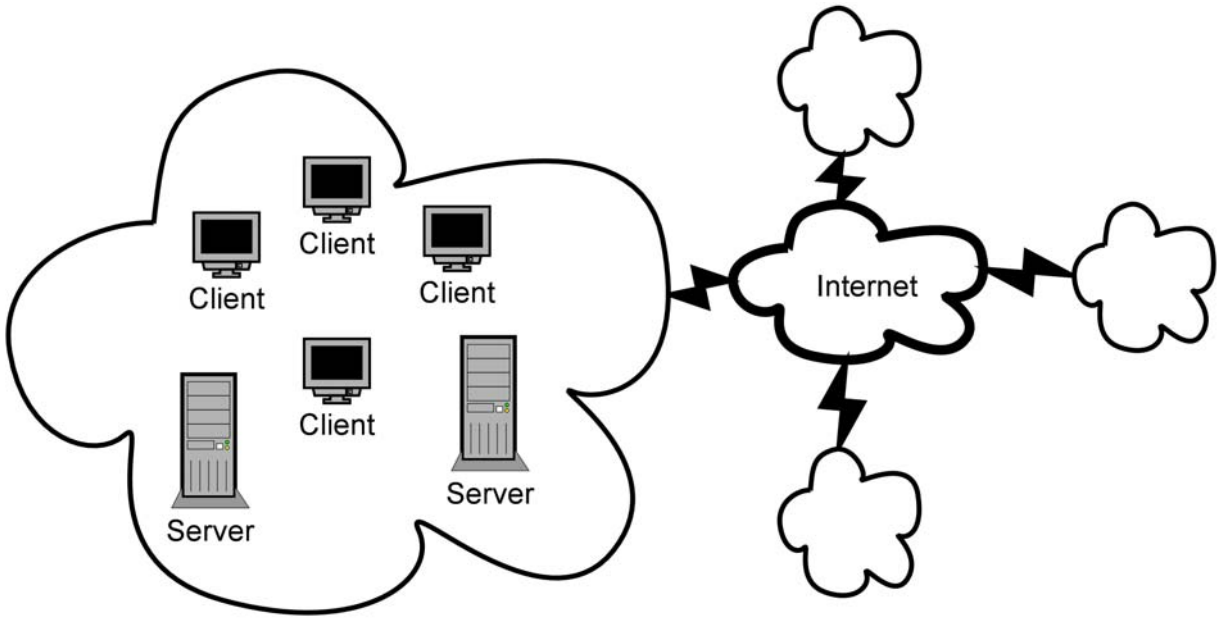
- Due to the heterogeneous communication infrastructure and network asymmetries, some users may suffer slow, congested and unreliable Internet connections, while others may have access to fast and reliable links
- Failing to send a command, or sending it too late, may cause damages to the avatar evolution (death, injury, loss of resources) and may result in unjustified penalties for the player
- Users with high-latency or low-bandwidth Internet connections may be more **disadvantaged** than their opponents supported by fast connections

The proposed solution

- In the classical gaming implementations, each client chooses its server at the *beginning* of its session, and keeps using it *until the end* of the playing
- The **fairness** may be improved with the introduction of the **dynamic migration** of clients among servers
- The target of our modeling and simulation investigation will be a qualitative comparison between a “classic” gaming network infrastructure scenario and the same scenario with the effects of the migration mechanism

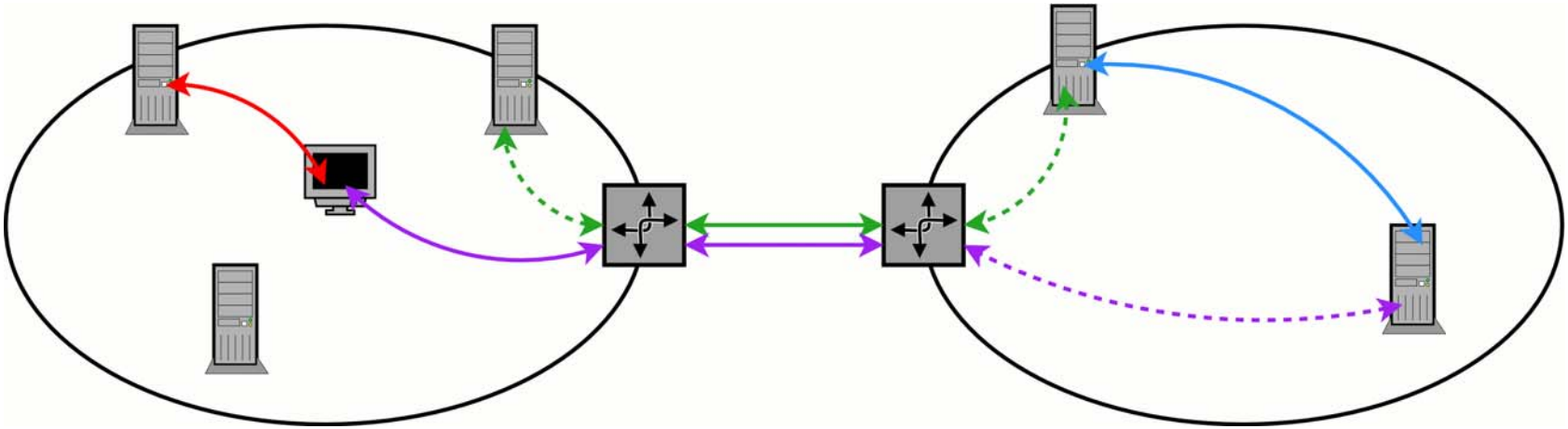
The simulated network model: a set of areas

We assume the network is organized in a numbered set of Areas, roughly corresponding to big Autonomous Systems, Internet Service Provided Networks or Internet Carriers



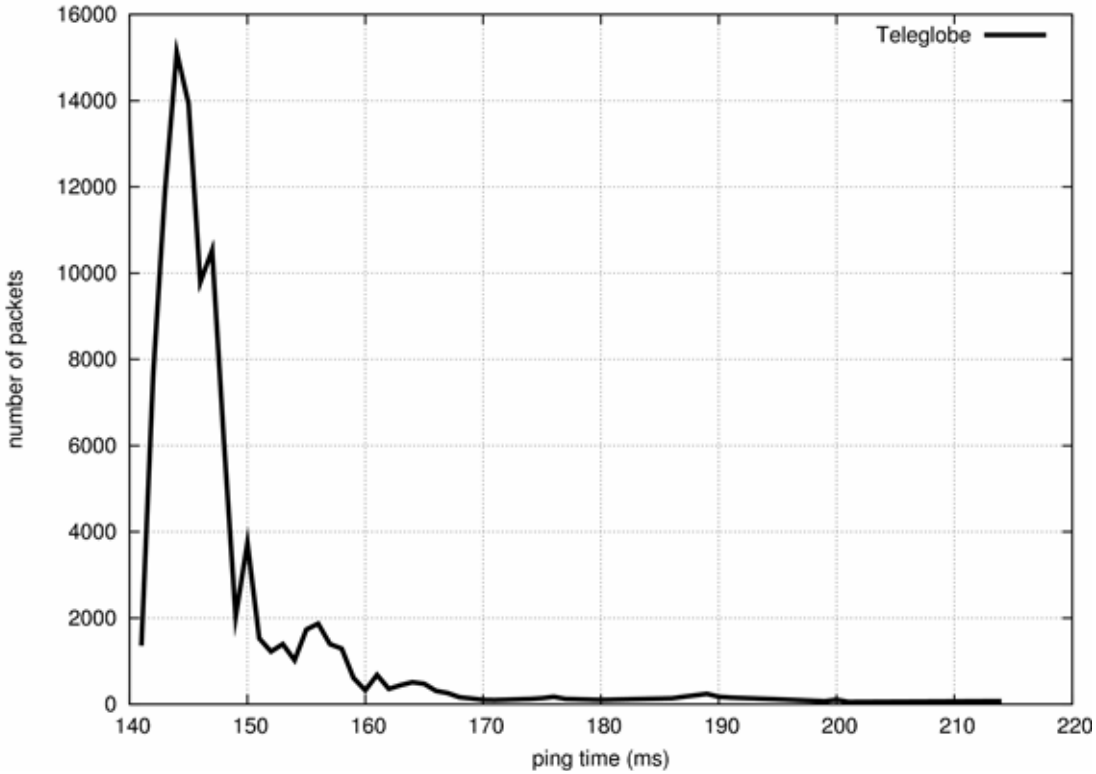
The simulated network model: latency and transfer time

- We assume that transmission times are mainly determined by link bandwidth and link latency network parameters
- To each client and server host is assigned a connection class that identifies the connection type the host adopts (i.e. modem, xDSL..)
- The end-to-end latency is calculated as summation of “intra-area” and “extra-area” latencies and transfer time



The simulated network model: latency distribution

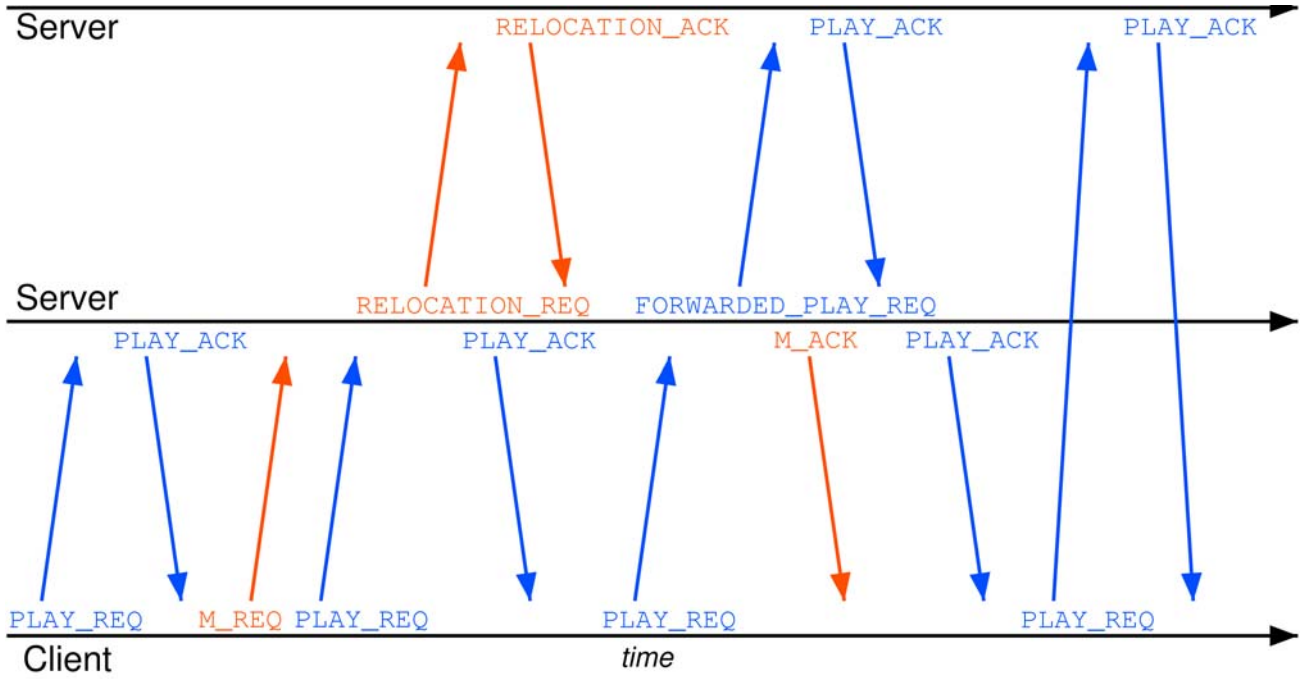
Experimental latency distribution for ping times, across two different areas (GARR <-> TeleGlobe)



It could be approximated by a right-shifted lognormal distribution

The simulated server model

To characterize the CPU computation time distributions of the servers (usually commercial and closed source) external observations has been made. The time passed from the *arrival* of a request to the server and the *departure* of the reply was measured



The migration mechanism

- There is no information to assist the client in making a "*good server choice*" at the beginning of the server session
- The migration mechanism would allow a client to disconnect from a server, and to connect to another server, while continuing to play in way transparent to the user
- One critical point is the design and implementation of the migration heuristic to assist clients in the decision to migrate
- Intuitively, the aim of the migration is to make the latency curve distribution of the clients *as much homogeneous as possible*

The migration heuristics

- The migration trigger: to migrate or not to migrate?
- Two approaches experimented:
  - simple migration

    once every a **fixed number of seconds** the client sends a ping probe to the servers in the list. If a different server appears to be **faster** than the current one, then a migration process will be activated
  - early migration

    in *addition* to the fixed time intervals evaluation, clients collects **runtime statistics** on the network round trip time by exploiting command/reply from/to the server. The client could try to *anticipate* the migration attempt.

Simulation tool and setup

- A dedicated simulator has been designed and implemented. The runtime is opportunely optimized for the RAM management in order to overcome the memory bottleneck in a massive simulation
- Monolithic, sequential, discrete event-based simulation
- The simulated scenario is composed by a network with 20 gaming servers and 2000 concurrent clients. It has been tuned with real world and literature –based parameters
- A number ranging from 10 to 20 independent runs has been executed for each test in order to achieve good confidence level

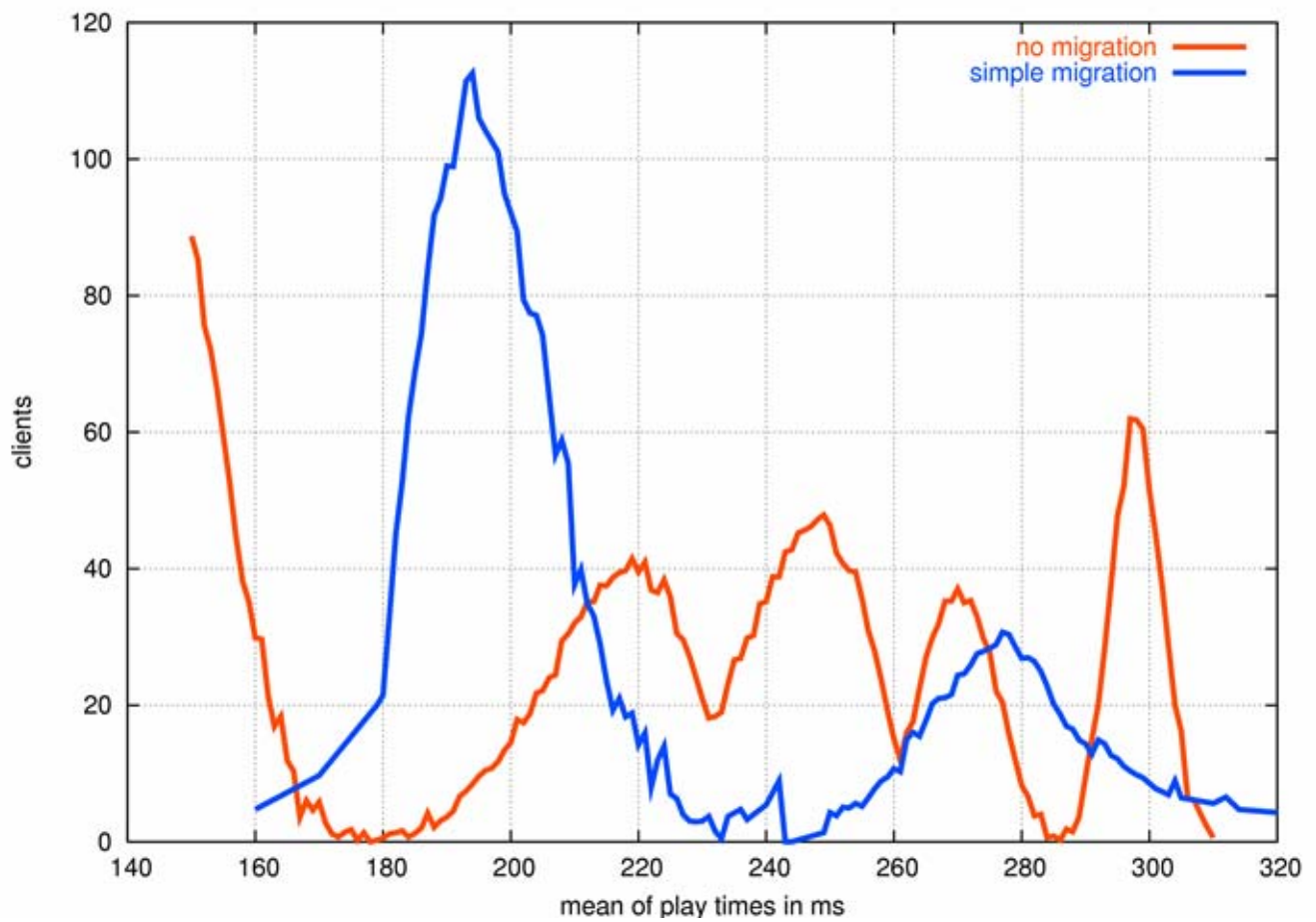
Performance results

- The main **performance index** is the "**play time**":

  how much a client has to wait for the reply after it has sent a request to the server

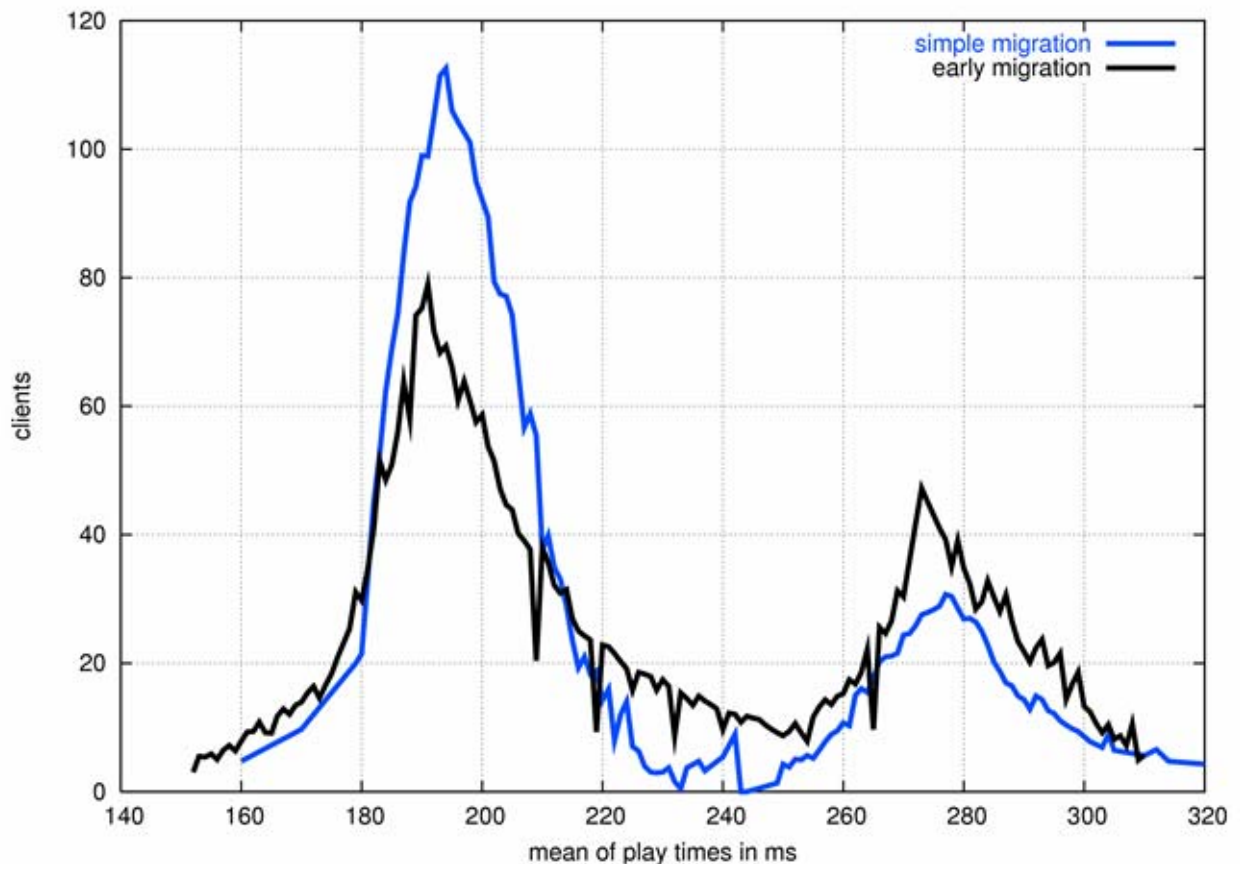
- The play time is composed by the time necessary for the client-to-server communication and the time consumed by the server to compute the next state
- It is expressed in milliseconds (ms) and it is used to compare the **fairness** of a gaming infrastructure under many different scenarios
- We are interested in the distribution, average and variance of play times

Experimental results: no migration vs. simple migration



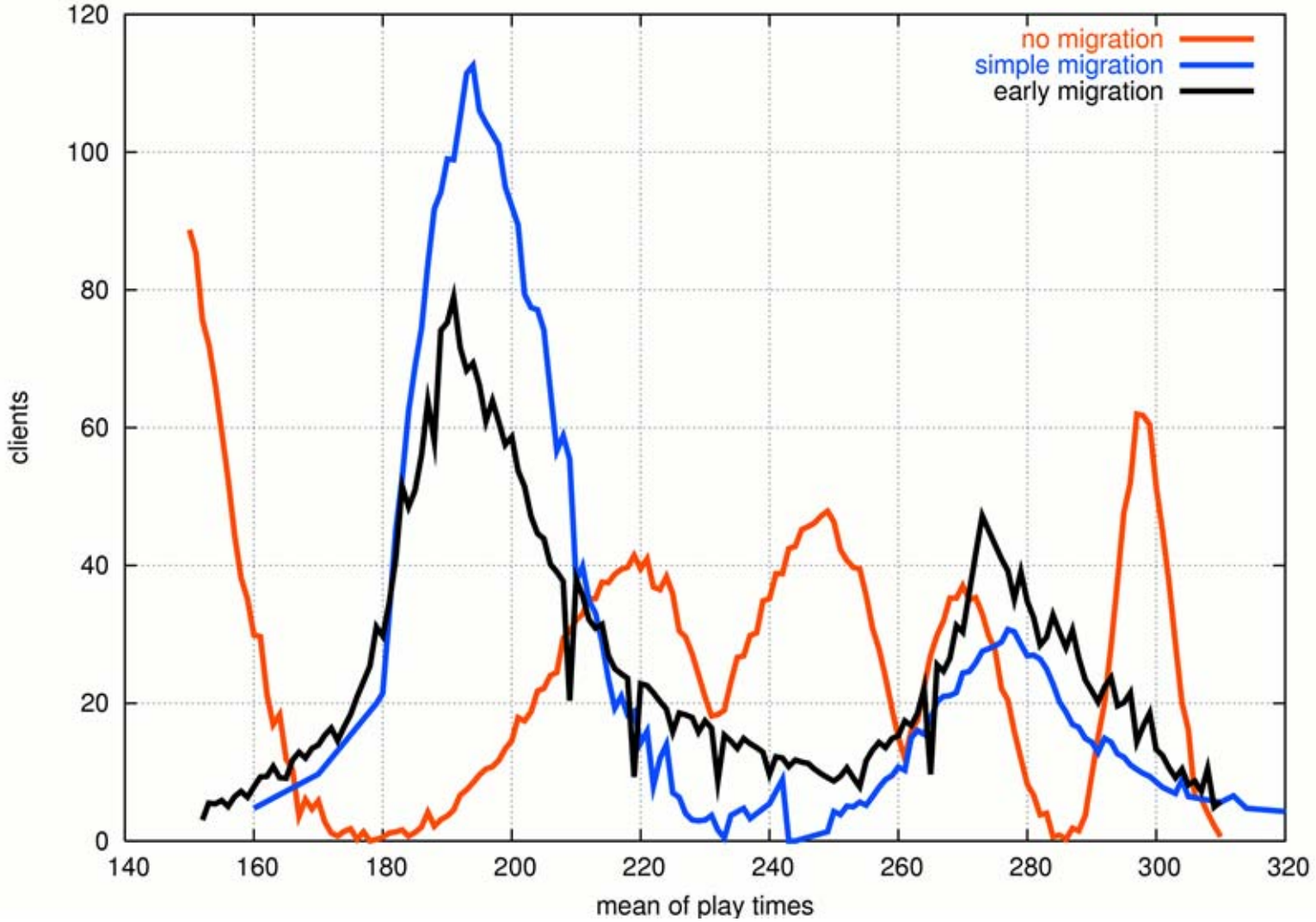
The play times appear to be much more **concentrated** with *migration on*, instead of **distributed** around at lest 5 sets of distant local aggregation values

Experimental results: simple vs. early migration

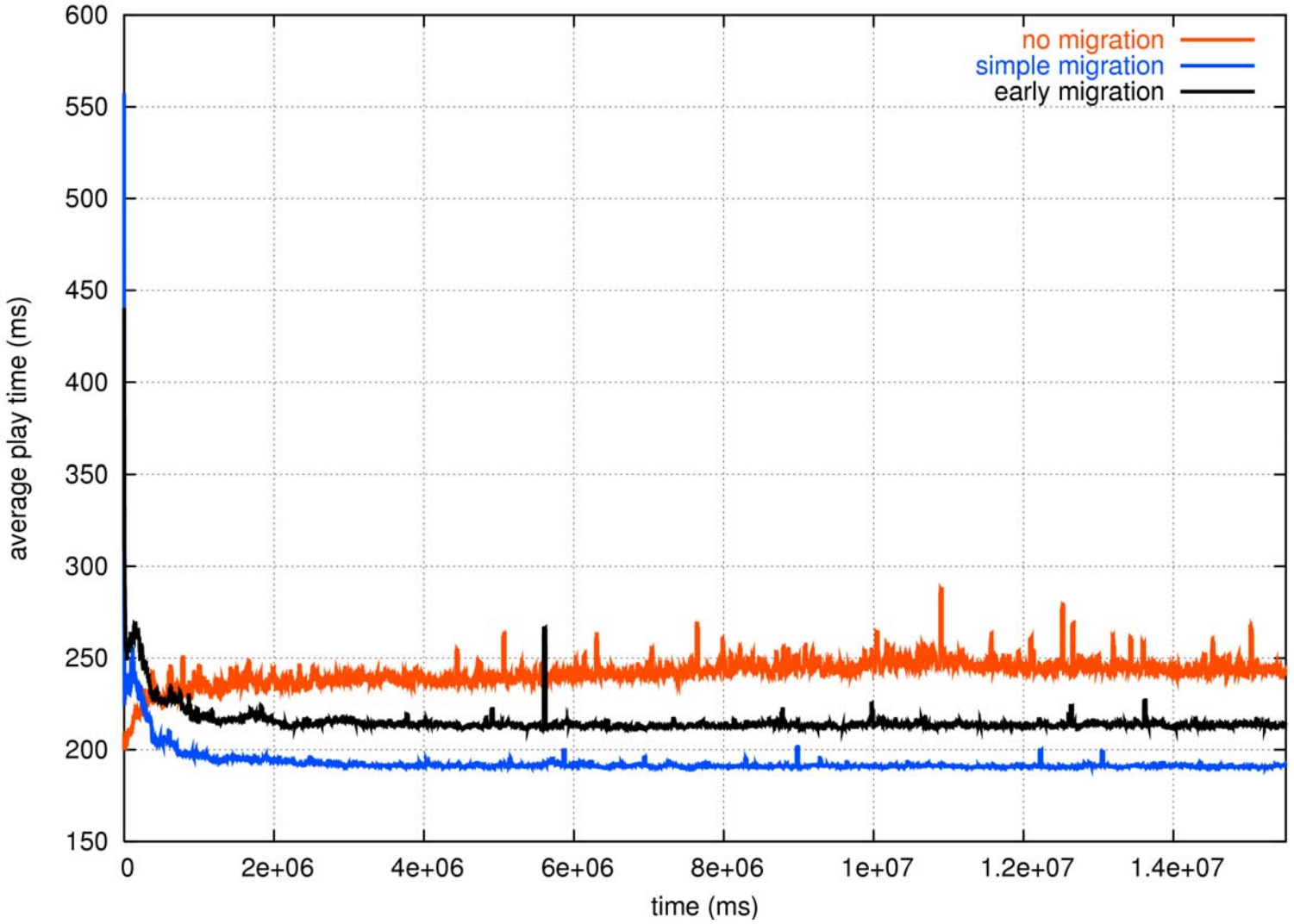


The "**simple migration**" mechanism gives the **better** results, with respect to the "**early migration**" scheme

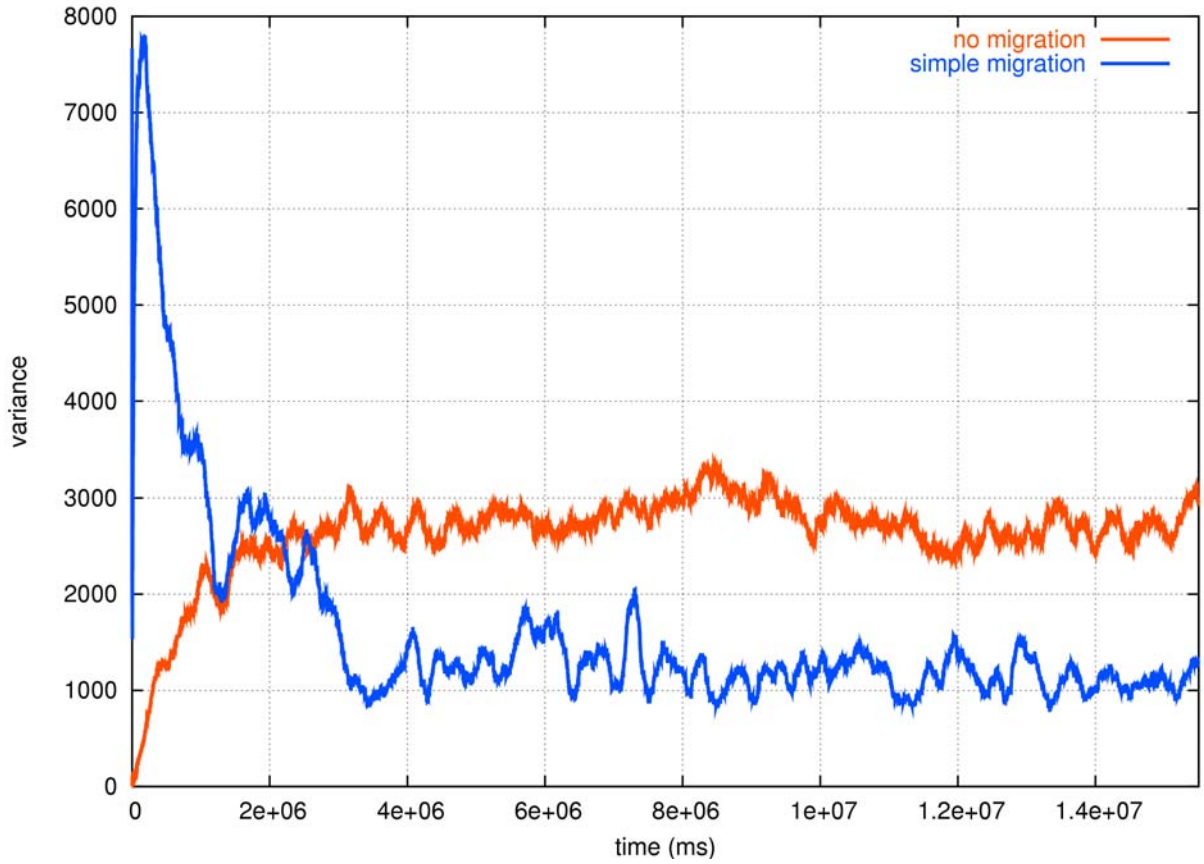
Experimental results: all together



Experimental results: average play time

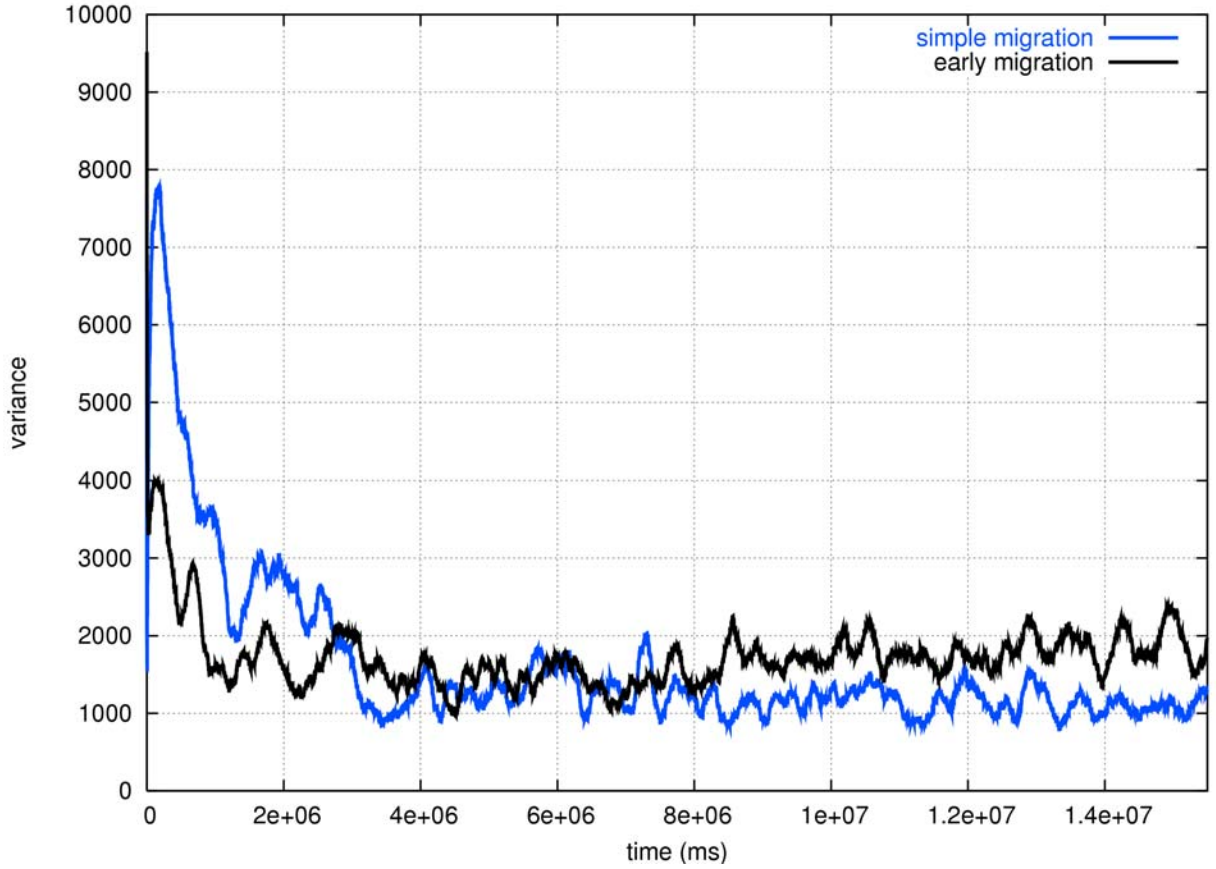


Experimental results: play time variance



After the initial re-allocation phase, the original architecture (*orange line*) leads to **higher variance value** than the same architecture with the “simple migration” (*blue line*)

Experimental results: play time variance



The “early migration” scheme has some difficulties when there are two or more servers with a very *similar latency*. This makes a subset of clients to **swing between servers**, introducing **overheads** and **unbalanced loads** on the servers

Conclusions and future work

- We propose and simulated an improved, migration based architecture for online multiplayer games, focused on MMORPGs with multiple replicated servers

- Work in progress
  - more efficient *migration* and *load balancing* algorithms
  - refinement of server-to-server communication protocols
  - a new architecture integrated by a pool of "**proxies**" accepting connections from the clients and forwarding the requests to the right area server
  - server side *computation migration* and *replicated execution*

Design and Simulation of a Migration-based Architecture for Massively Populated Internet Games

Gabriele D'Angelo
<gdangelo@cs.unibo.it>

joint work with

Sandro Pifferi
Ludovico Gardenghi
Luciano Bononi

University of Bologna
Department of Computer Science

Dallas (TX), 29/11/2004