

Distributed Virtual Environments: From Client Server to Cloud and P2P Architectures

Laura Ricci

**Dipartimento di Informatica
Università degli Studi di Pisa**

Lezione n.19

10 maggio 2013

DVE: A KILLER APPLICATION

- A killer application
 - Massively Multiplayer Online Games
- A continuously growing market
- An interesting (and complex application) including
 - Graphics
 - Networking
 - Artificial Intelligence
- A high number of challenging design requirements
- An application which may greatly benefit from novel architectures
 - Cloud
 - P2P

AGENDA

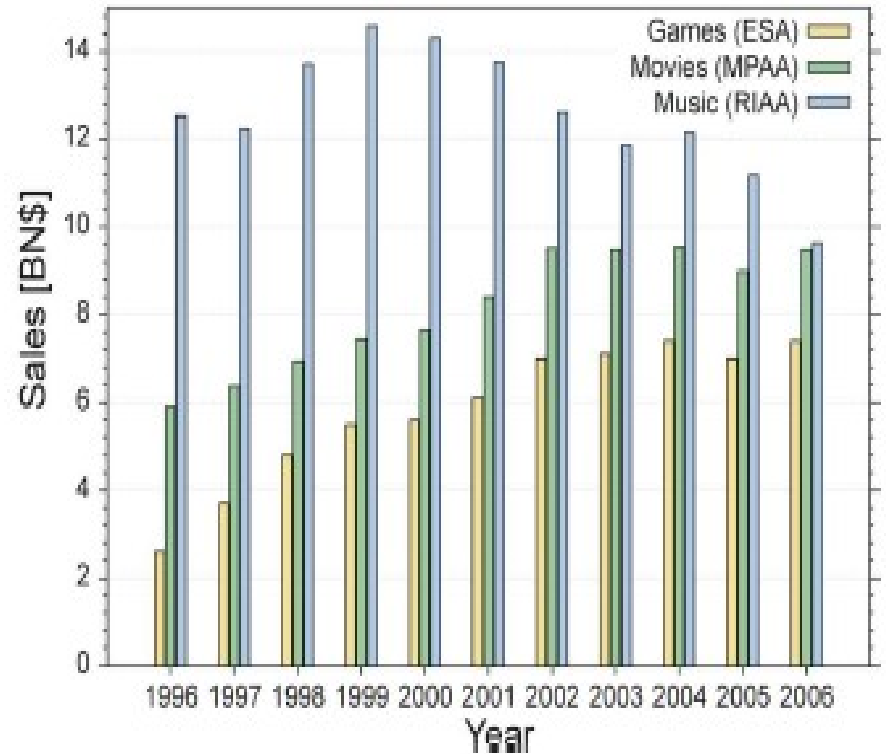
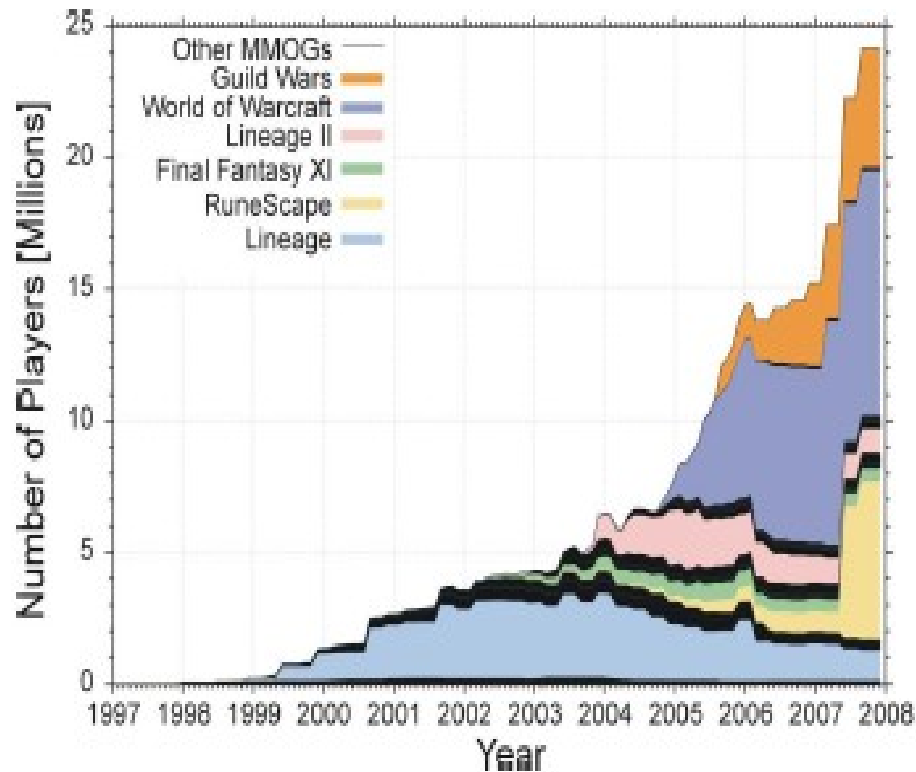
- Distributed Virtual Environments: definition, history, examples
- Requirements
- Scalability issues:
 - Interest Management
 - Dead Reckoning
- Architectures
 - Client Server
 - Parallelizing Techniques: Zoning, Mirroring, Sharding, Instancing
 - P2P
 - DHT based
 - Voronoi based
 - Clouds infrastructure for gaming
- Conclusions

- Note: some material of this tutorial (mainly figures) comes from the course Networked and Mobile Games taught by Prof. Wei Tsang Ooi.

DISTRIBUTED VIRTUAL ENVIRONMENT

- synthetic world where multiple **geographically distributed** participants **interact** via a **virtual character** (avatar)
 - involves a set of hosts communicating over the network
- evolution:
 - first introduced for military applications back to the 80s
 - rapidly evolved to social and entertainment environments attracting millions of users from all over the world
- examples:
 - massively multiplayer online games (MMOG)
 - networked virtual environments
 - military training simulations
- **the main focus is on MMOG because**
 - it is the **killer application**
 - it is the **worst case scenario**

MMOG: A GROWING MARKET



WAR OF WARCRAFT: THE KILLER MMOG



data centres and network monitoring by AT&T

10.2 million of users

10 data centres around the world


13250 server blades

75000 CPU cores

1.3 PTbyte storage

“World of Warcraft made more money overall
and more money per person than Avatar the movie”

World of Warcraft subscriber numbers remain at 10.2 million

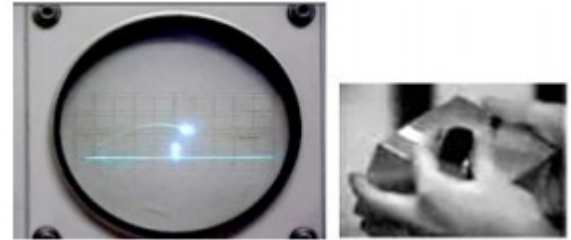
by Adam Holisky  May 9th 2012 at 5:00PM



Blizzard announced today in the investors call that the *World of Warcraft* population remains stable at 10.2 million subscribers. This is after the announced dip to 10.2 million in February 2012.

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
- 1984 Island of Kesmai
 - First MMORPG, Pay by use
- 1993 Doom;
- 1997 Ultima Online; Origin Systems
- 2004 World of Warcraft; Blizzard
- Currently, different kind of game genres
 - Real time strategy (RTS)
 - First Person Shooters(FPS)
 - Massively Multiplayer Online Role Playing Game (MMORPG)



REAL TIME STRATEGY MMOG

- Interaction Omnipresent model:
 - Players view and simultaneously influence the entire set of resources under their control
- Game Perspective:
 - Bird-eye view
- Player coordinates armies
- Goals:
 - resource gathering
 - base building
 - destroy enemy army or take control of a region
- Ex: Age of Empire



FIRST PERSON SHOOTER MMOG

- Interaction model:
 - avatar based: player interacts with the game through a single character
 - player actions are defined in terms of commanding the avatar
- Game perspective:
 - first person prospective
- Weapon-based combat
- Highly frenetic behaviour
- Ex: Doom, Quake, Call of Duty, Halo



MMORPG: MASSIVELY MULTIPLAYER ON LINE ROLE PLAYING GAMES

- Interaction model avatar based
- Game perspective:
 - First or third person perspective
- actions:
 - fight monsters for experience
 - interact with other characters
 - acquire items
- million of subscribers
 - ex: War of Warcraft, Warhammer on line
- we refer mainly to MMORPG

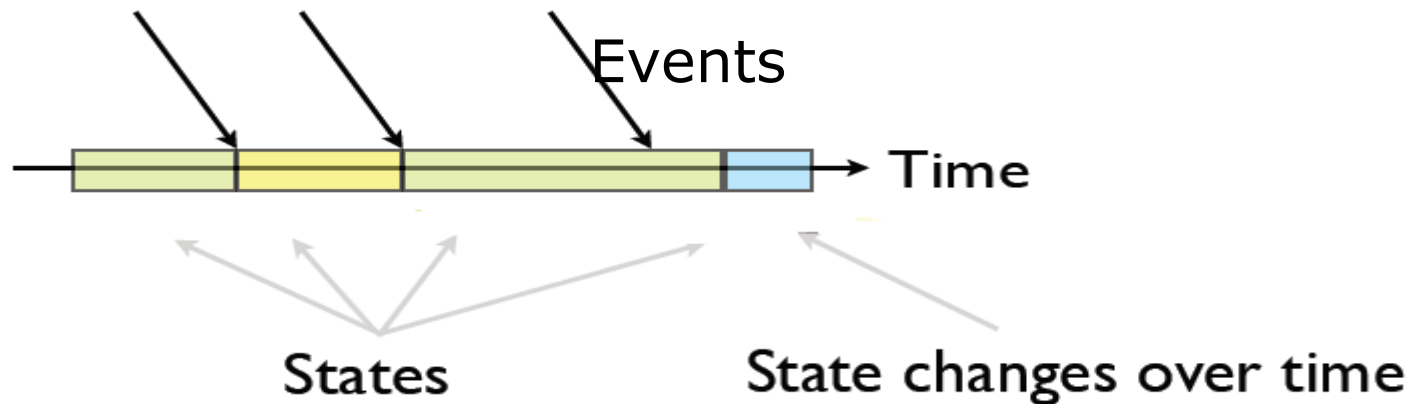


SOCIAL ENVIRONMENTS: SECOND LIFE



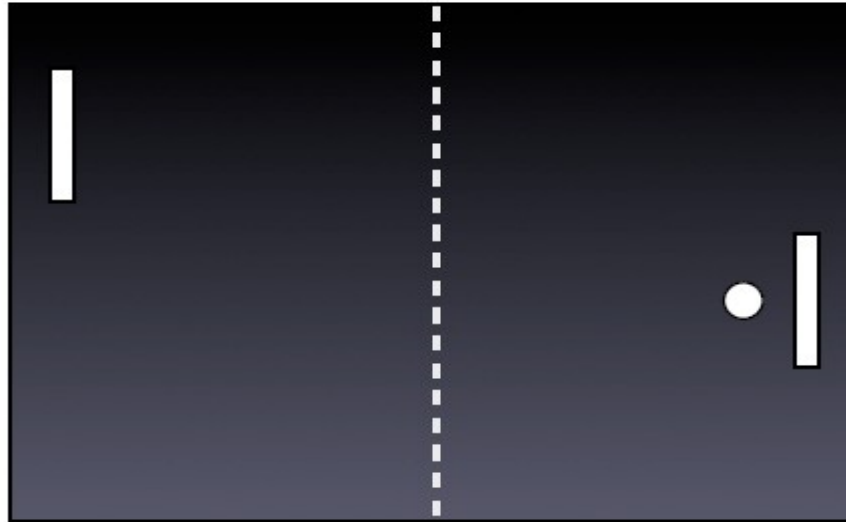
AN ABSTRACT MODEL FOR MMOG

- The game simulation on a distributed node of the network can be modelled as a sequence of **states changes** over time, in reaction to **events**



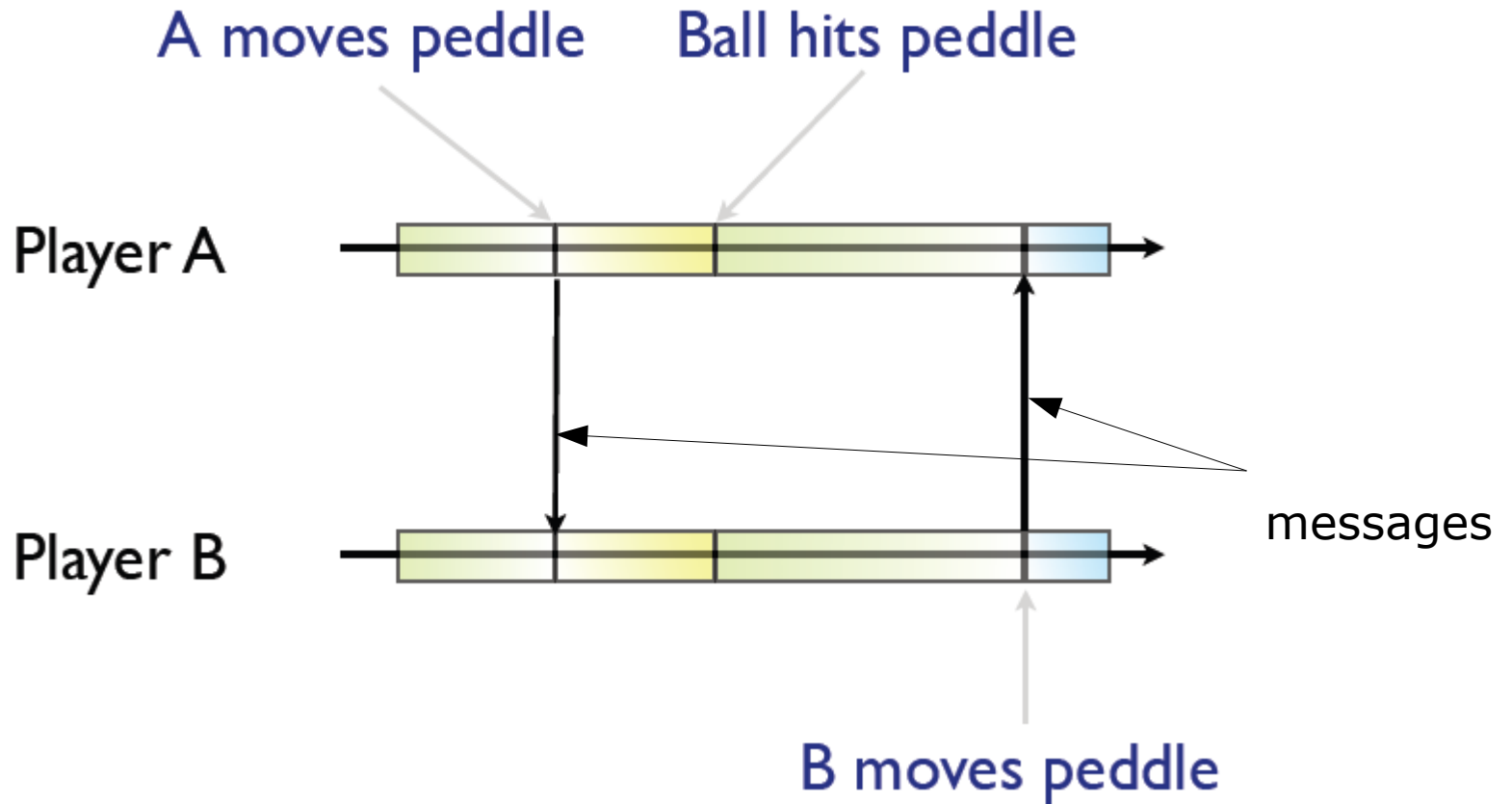
- A node may send messages to other nodes to notify its state change
- Events:
 - user input
 - timers
 - events generated from other nodes

A NAIVE EXAMPLE: TWO PLAYERS PONG



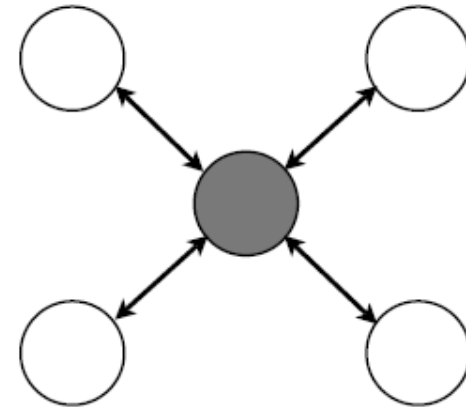
- **state:** position of
 - the paddles
 - the ball (derived from its velocity)
- **events:** movement of the paddle
 - must be notified to the other player

A NAIVE EXAMPLE: TWO PLAYERS PONG



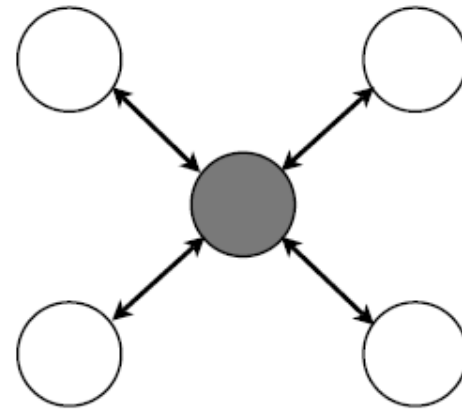
CLIENT/SERVER CENTRALIZED ARCHITECTURE

- Clients send events to server
- Server computes the new state and notifies the updates to all the clients
- no direct interaction between players



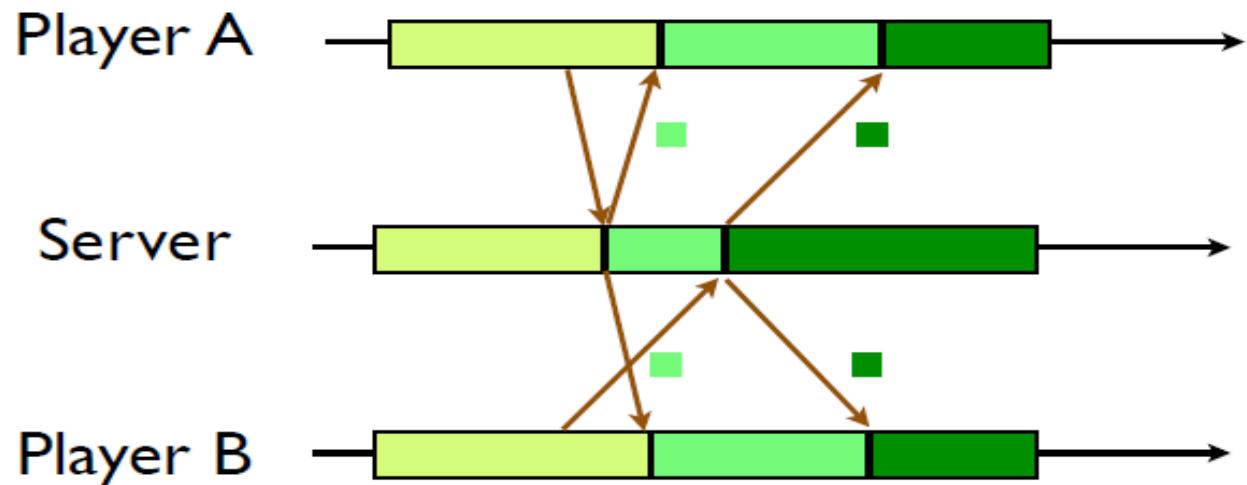
SMART SERVER DUMB CLIENT

- the server
 - maintains the state
 - notifies the clients
 - resolve conflicts
 - simulate game
- the client
 - sends events to the server



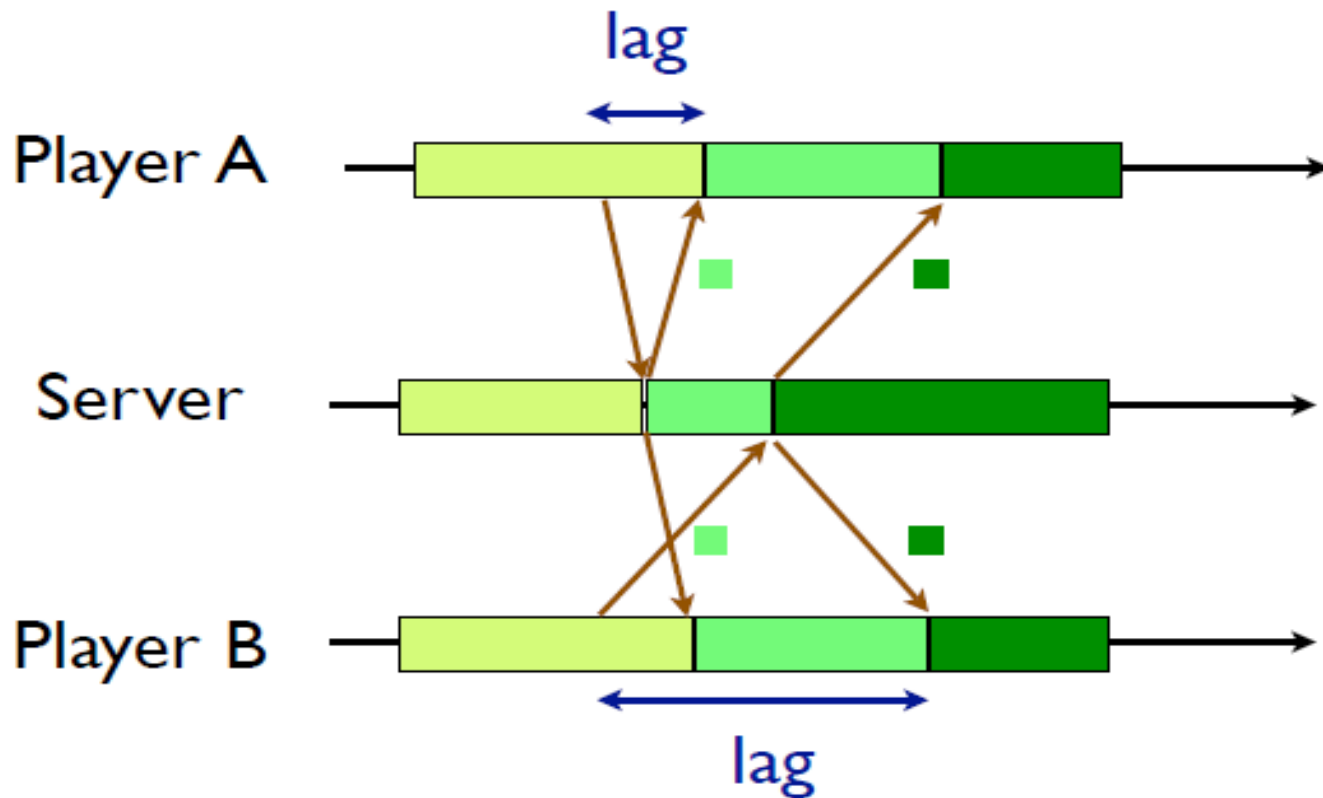
In the picture: state updates happen at the same time on different clients

This is a simplifying assumption



SMART SERVER DUMB CLIENT

- Lag: interval of time between a player action and its rendering
- Larger lag for player with higher latency



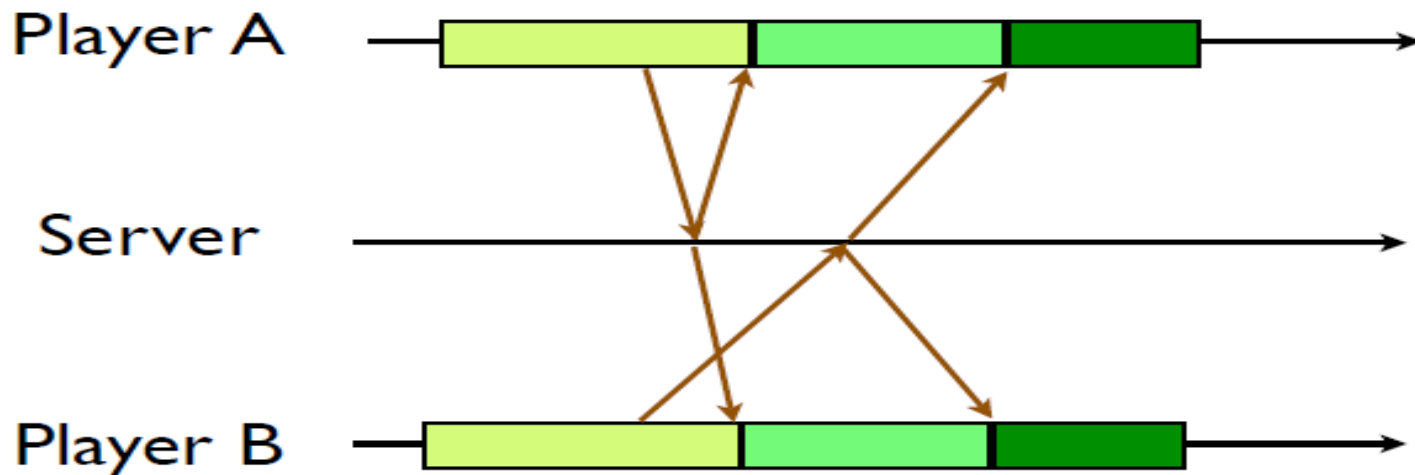
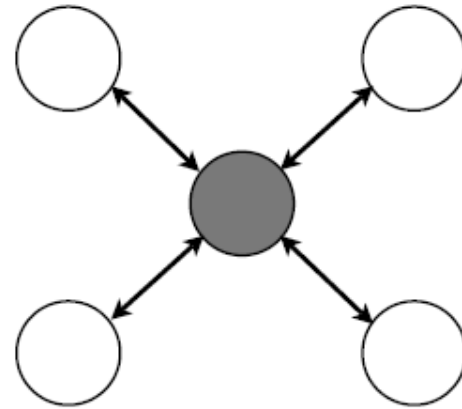
DUMB SERVER SMART CLIENTS

the server

- notifies clients
- resolve conflicts

the client

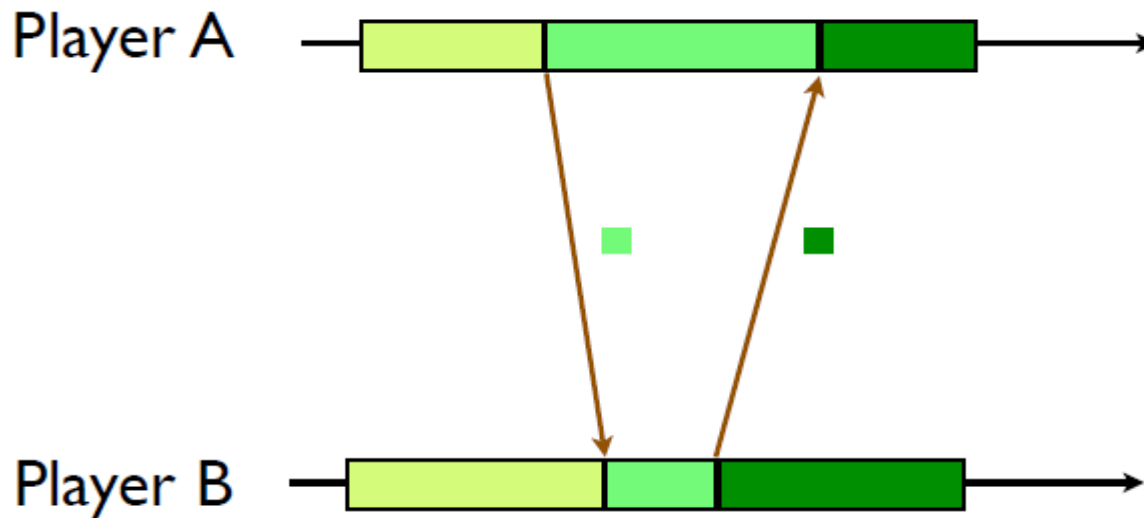
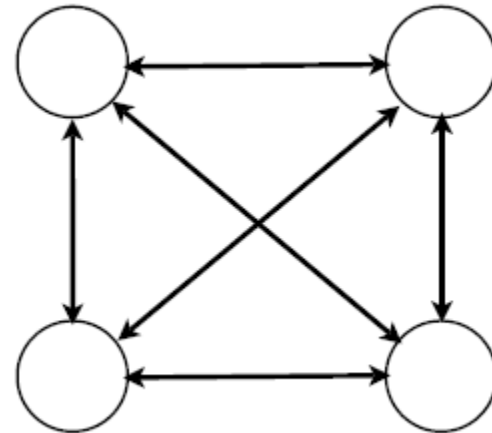
- maintain state
- simulate game



PEER TO PEER

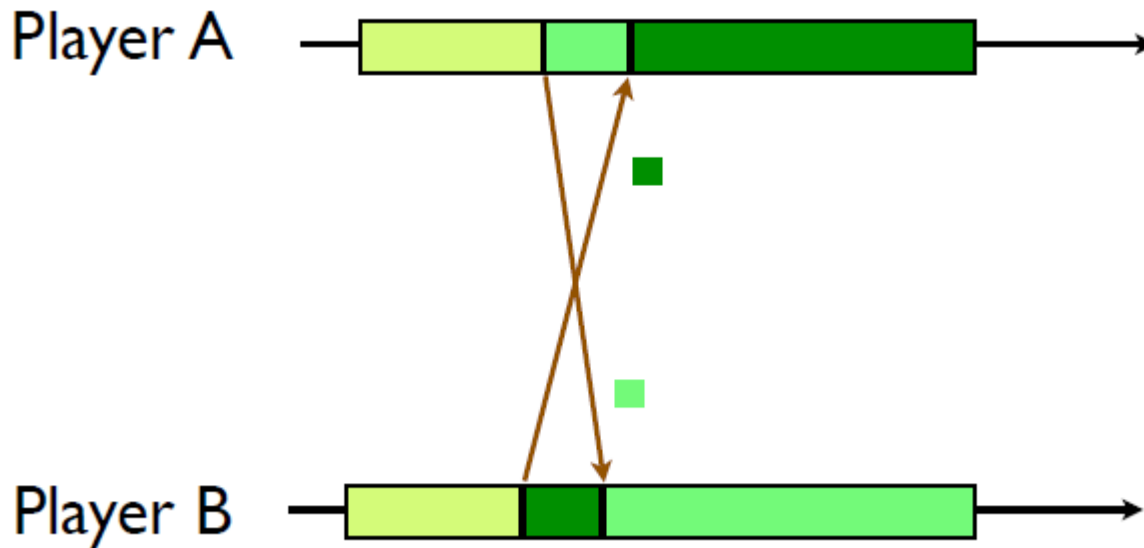
The peer/client

- state management
- simulation engine
- event notification to the other peers
- distributed conflict resolution



PEER TO PEER

Challenge: Maintaining consistency



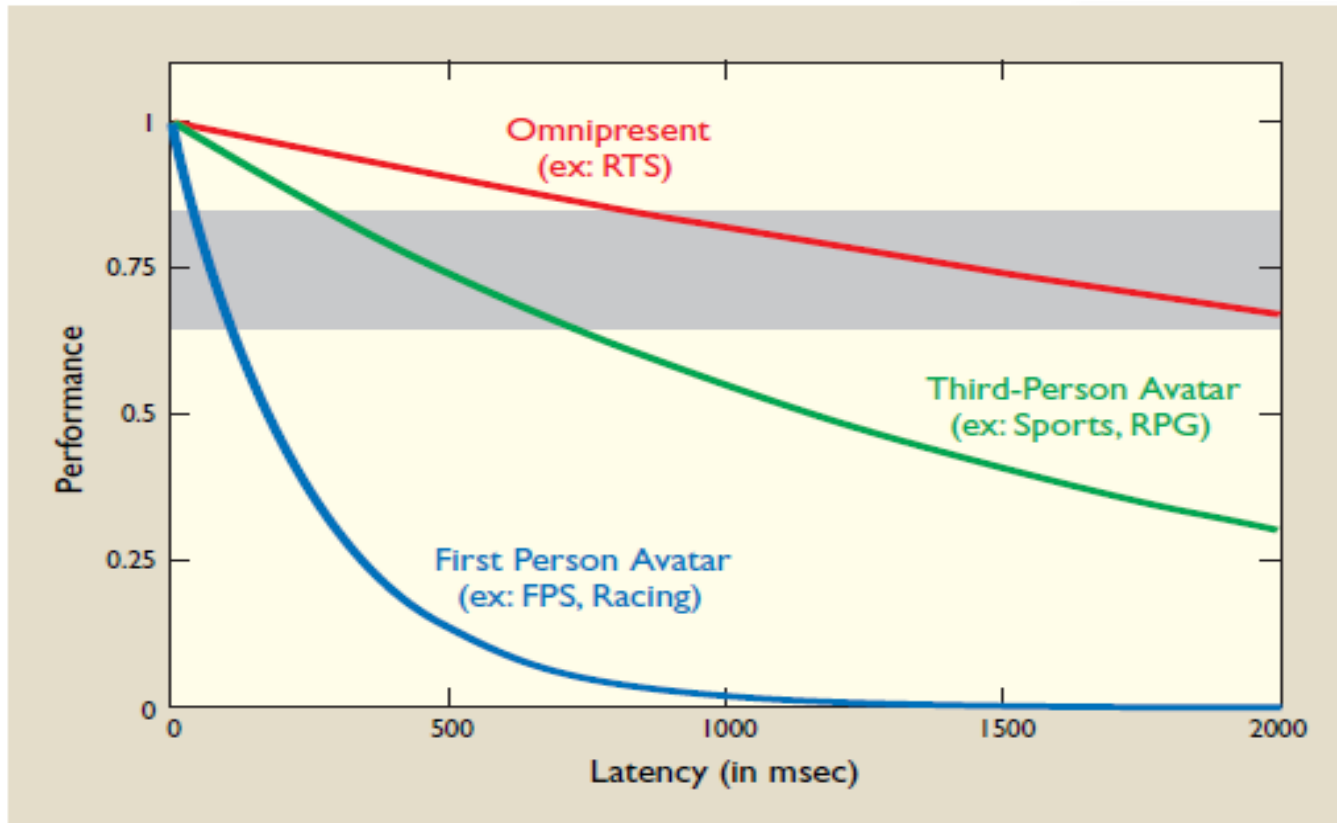
DESIGN REQUIREMENTS

- Responsiveness
- Consistency
- Security
- Fairness
- Resource Efficiency
 - Bandwidth is limited and may not be sufficient for a huge number of players
 - Latency
- Scalability

RESPONSIVENESS

- **Response time:** interval of time between player action and its rendering on the screens of the players
 - acceptable values depending on **game genre** and **players actions**
- **Latency sensitive**
 - The action has to be notified to the server (in C/S architectures) that computes the state update and sends the new state to all the players
- Latency depends on the underlying network:
 - LAN: 10ms
 - Internet: hundreds of msec up to more than one second
 - often depending on 'last mile' access network
- A challenge for best effort networks:
 - packet delay, packet loss

RESPONSIVENESS VS. LATENCY



The horizontal grey area in the figure is a visual indicator of player tolerance for latency

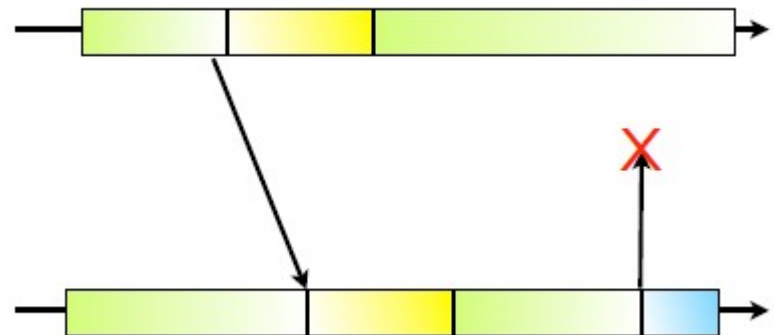
CONSISTENCY

Goal: all the players have the same view of the game state at the same time

Ideally, updates should reach other players immediately, so that states are consistent at all time



Delay and loss lead to inconsistent states



CHEATING RESISTANCE

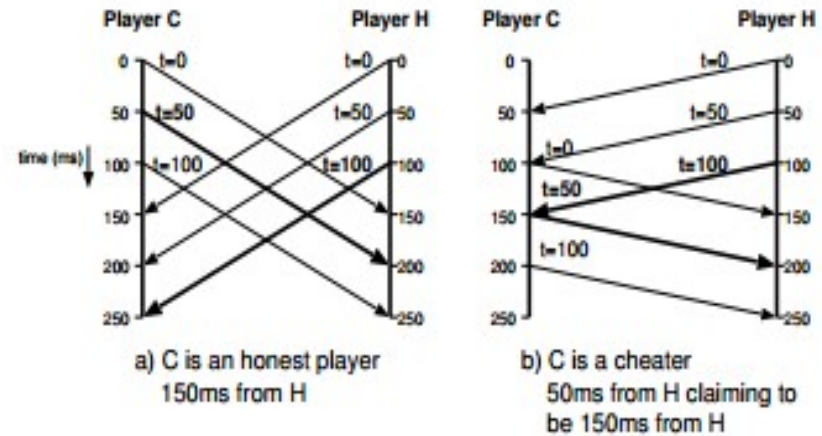
- cheating: an unauthorized interaction with the system aimed at offering an advantage to the cheater
- 15 different types of cheats have been detected

Some examples:

suppress correct cheat: we will see this after introducing dead reckoning

look-ahead cheating:

- the cheating player delays its actions to see what other players do before announcing its own action
- indistinguishable from a player suffering from high latency
- requires modification of the event timestamp



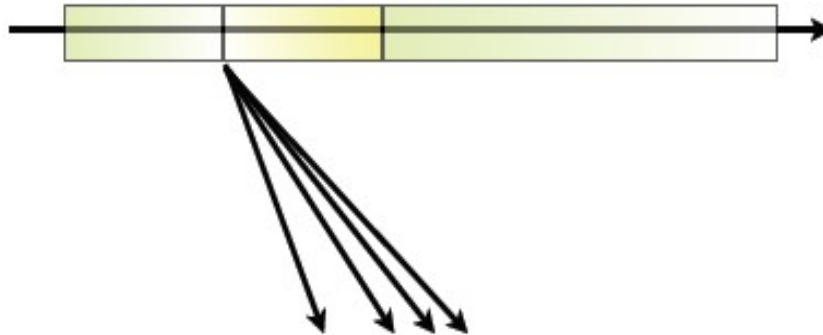
STATE PERSISTENCE

- access to the game entities must be available to users at any time
- simulation evolves even when some users are not actively involved in the game
- permanent data between login sessions
- ephemeral data need not be stored in a permanent repository
- straightforward solution for client/server architectures
 - a single server stores all permanent data
 - scalability problems
- the problem is more complex for P2P architectures
 - what does it happen if all the players/peers logout?

SCALABILITY

Management of a huge number of players is mandatory

- huge cost in network/CPU resources
- resource are limited, in particular **bandwidth**



- Several techniques introduced to cope with resource limitations
 - Area of Interest
 - Delta Encoding

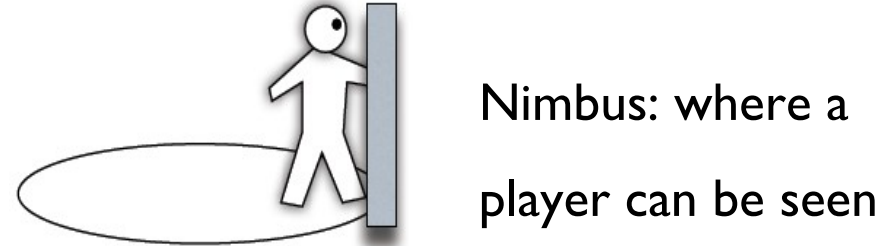
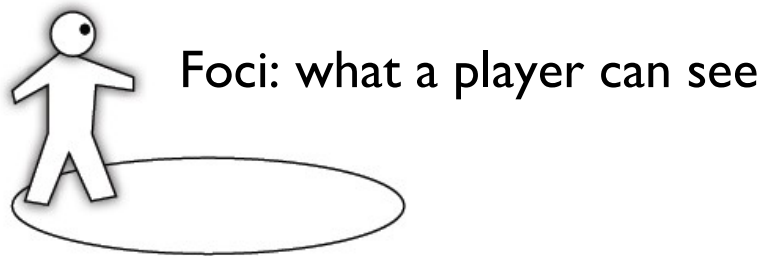
SCALABILITY ISSUES IN MMOG

- Huge amount of bandwidth require at the server: several Gbps
- Hundreds of terabytes of data transferred in a month
- Need to reduce bandwidth requirements
- Several techniques
 - Interest Management
 - Dead Reckoning
 - Relevance Filtering

INTEREST MANAGEMENT

Main idea:

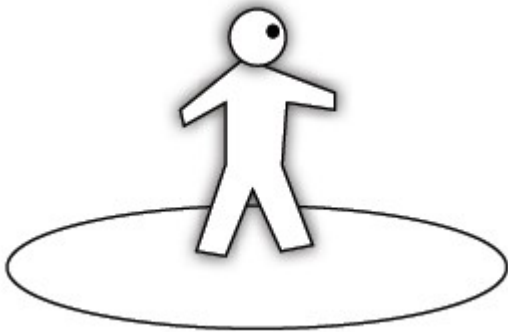
- Reduce traffic for notifying positional and object updates
- Basic idea: an update u should be sent to a player p only if u matters to p



Update of p matters to q if the foci of p intersects nimbi of q

INTEREST MANAGEMENT: AURA BASED APPROACHES

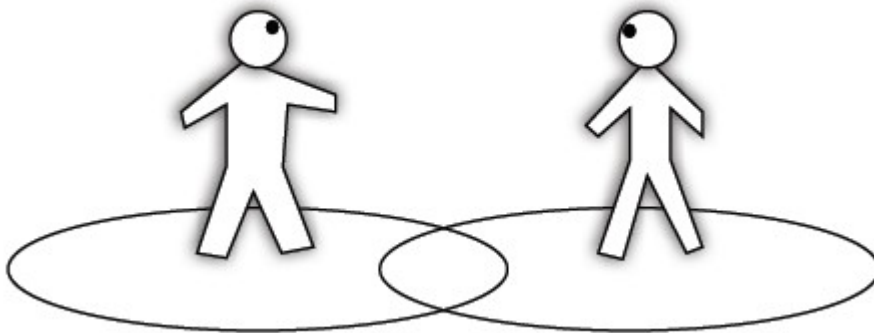
- Computation of exact foci/nimbi may be costly
- Aura based approaches



Area of Interest (AOI) (aura):

region of the virtual world surrounding the entity

dynamic definition of the AOI

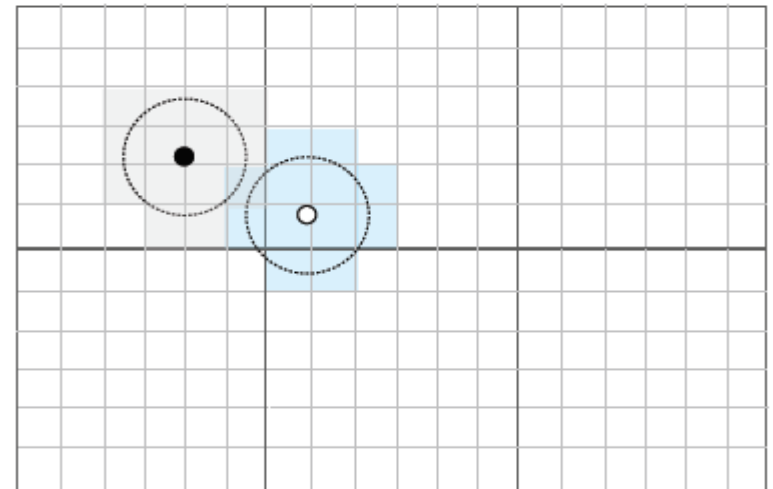


Update of p matters to q if the
their AOI intersect

Entity mutual visibility require complex
computations

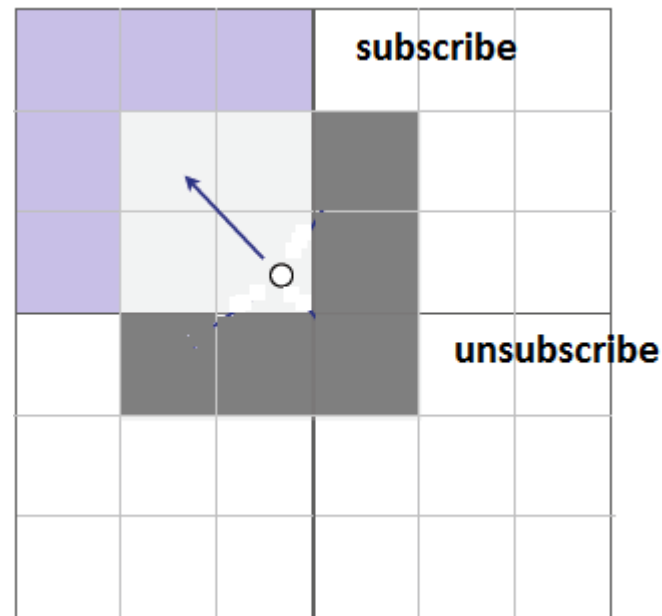
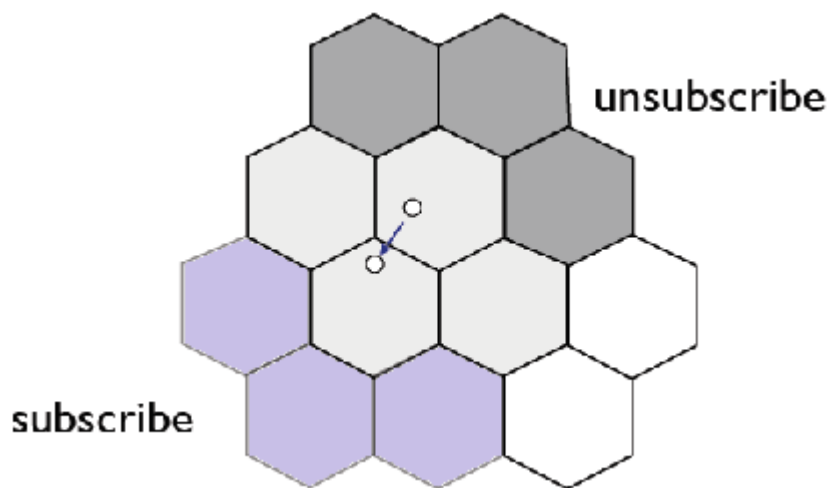
INTEREST MANAGEMENT: CELL BASED APPROACHES

- define a static partitioning of the virtual world into **cells**
 - shapes: rectangular, hexagonal,...
- aura approximated by a set of cells defining bounding box of AOI
- need not be binary: can generalize to multilevel of interest depending on the distance
- multicast group based implementation



INTEREST MANAGEMENT: CELL BASED APPROACHES

Hexagonal cells approximate a circular AOI better than rectangular ones

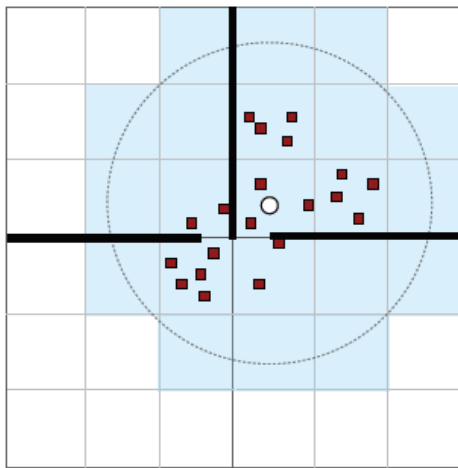


Every move requires 3 new subscriptions
and 3 unsubscriptions

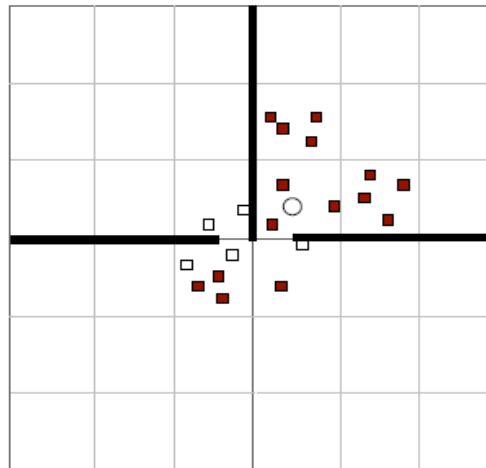
Moving diagonally requires 5
new subscriptions and 5 unsubscriptions

INTEREST MANAGEMENT: VISIBILITY

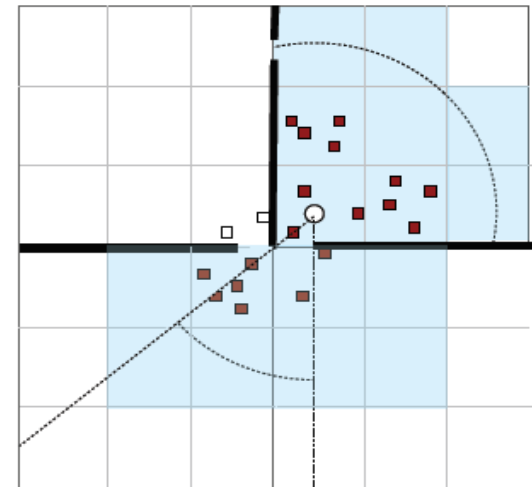
- More complex approaches considers **mutual visibility**
- A player P is interested in an entity Q if P can see Q and Q is close to P
 - consider visual occlusions
 - different level of accuracy: object-to-object visibility, object-to-cell, cell-to-cell visibility



Naive Approach



Object-to-Object visibility



Cell-to-Cell-visibility

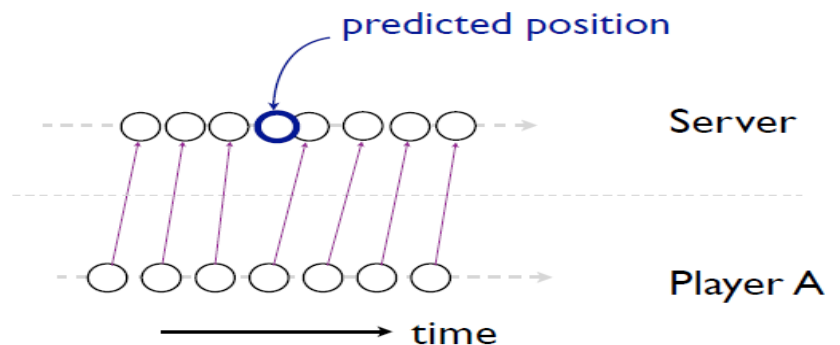
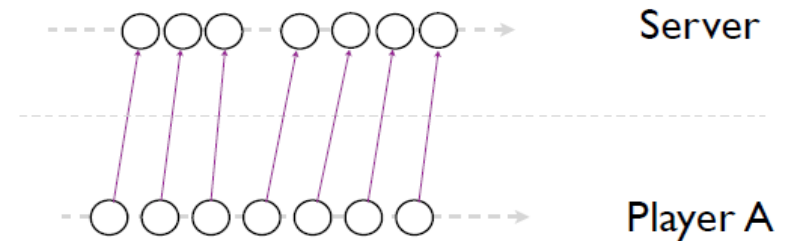
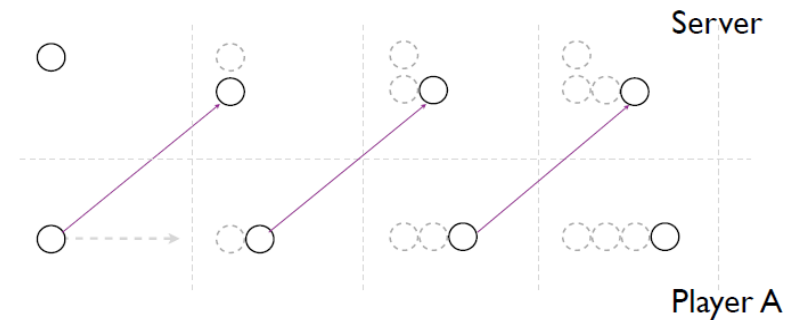
DEAD RECKONING

- technique to predict the position of an entity by exploiting its last known (or the history of past known) **kinematic state** which may include
 - position
 - velocity
 - acceleration
 - orientation
 - angular velocity
- originally exploited in air and marine navigation, GPS, etc.....
- in MMOG
 - bandwidth requirement reduction: not all positional updates (heartbeats) are sent to the server
 - jitter tolerance

$$X_1 = X_0 + v_0\Delta t + \frac{1}{2}a\Delta t^2$$

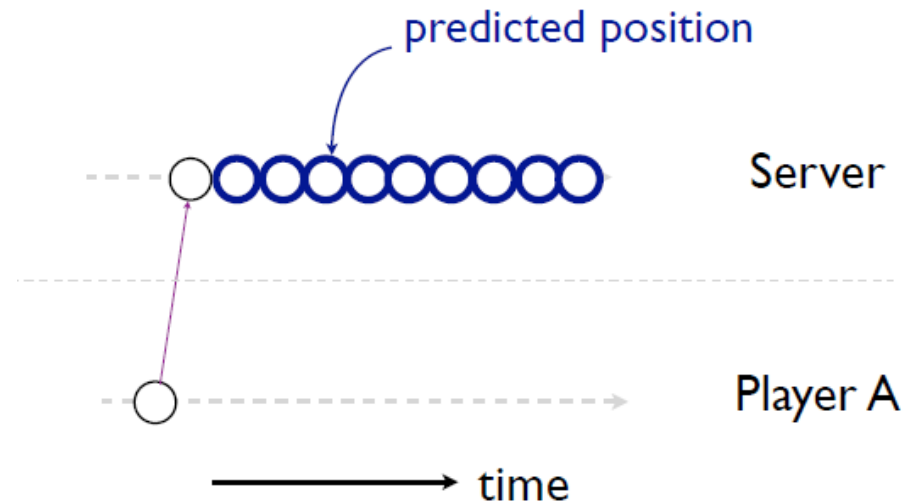
JIITTER TOLERANCE BY DEAD RECKONING

- Basic Solution: server keeps track of the position of the avatar through the updates sent from the players
- Delay jitter causes player's movement to appear erratic
- But, if an update arrives late, dead reckoning predict the position....

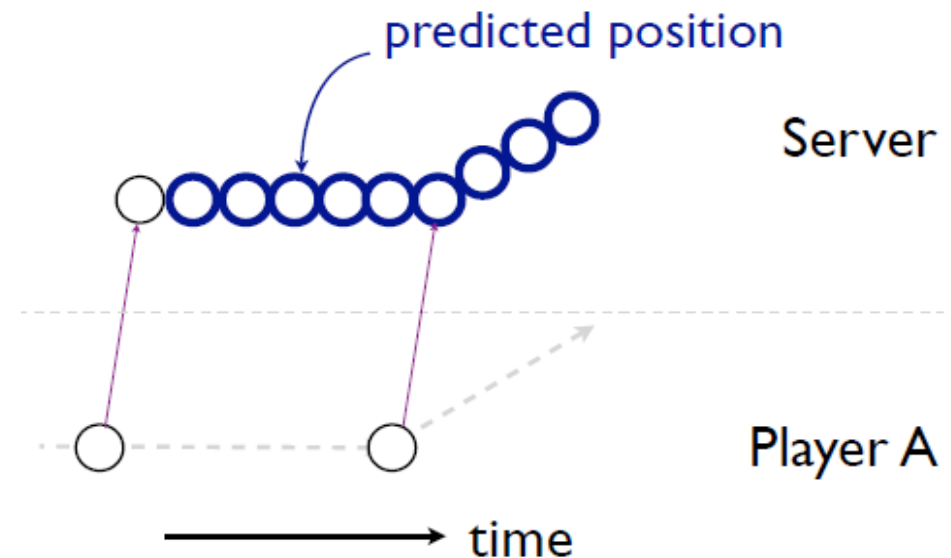


BANDWIDTH REDUCTION BY DEAD RECKONING

- if the velocity remains constant, predict every next position

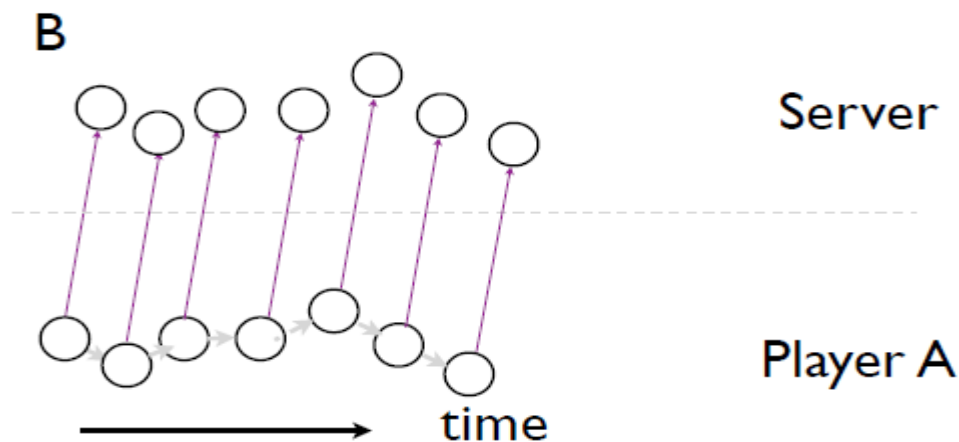


- if the velocity/direction changes client needs to send the update to the server



DEAD RECKONING

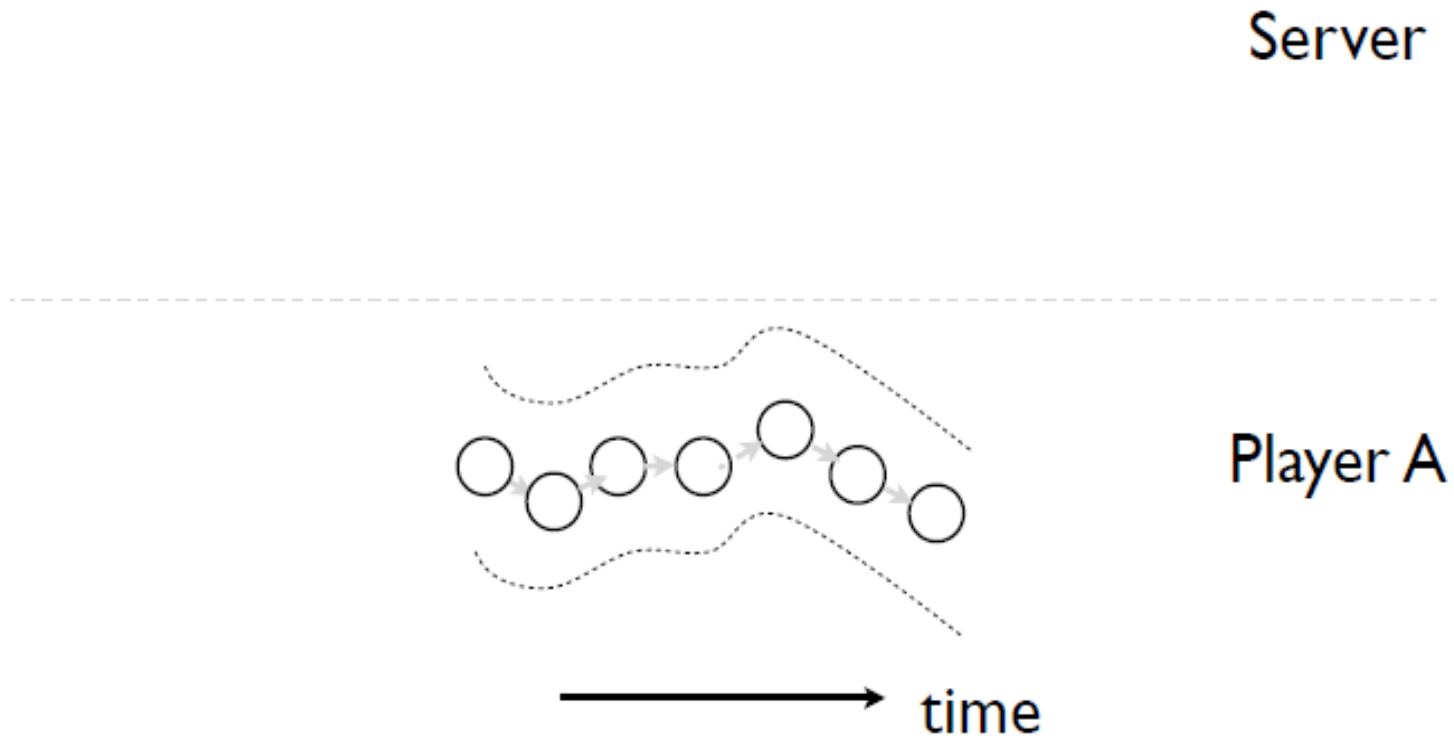
Still needs large number of updates if the velocity/direction changes frequently and each update is notified to the server



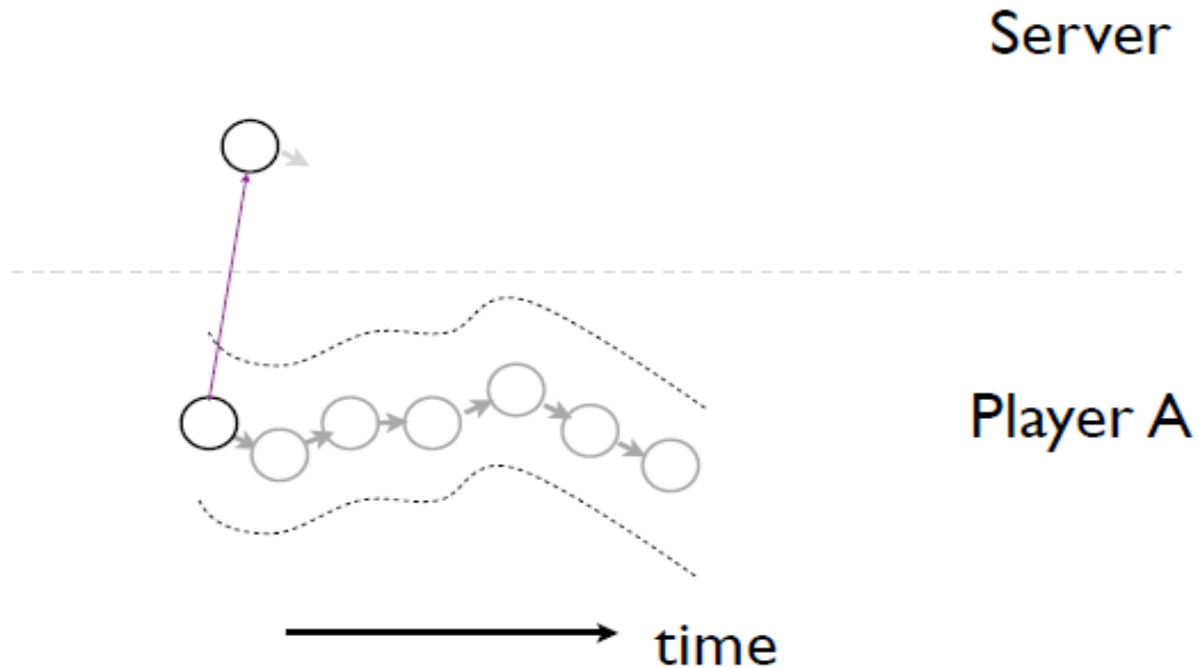
A trade-off solution:

- the same dead reckoning algorithm executed both by the client and the server
- the client sends the update iff the difference between the real position and the predicted one exceeds a predefined threshold

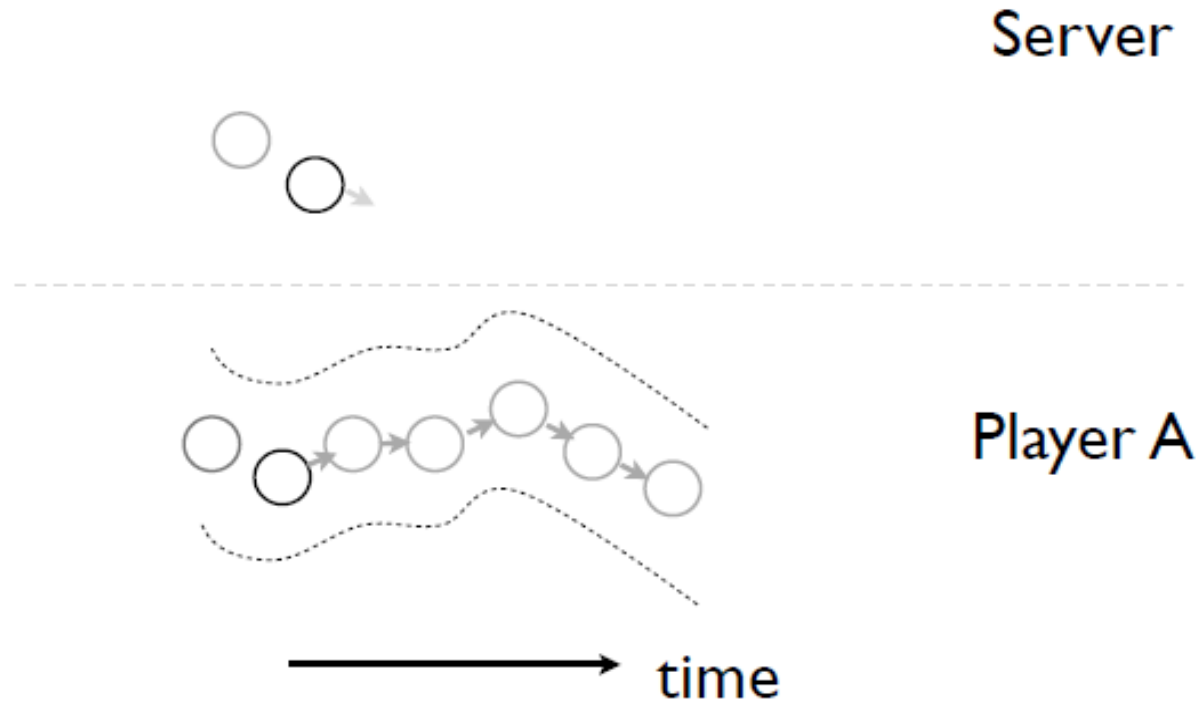
DEAD RECKONING: AN EXAMPLE



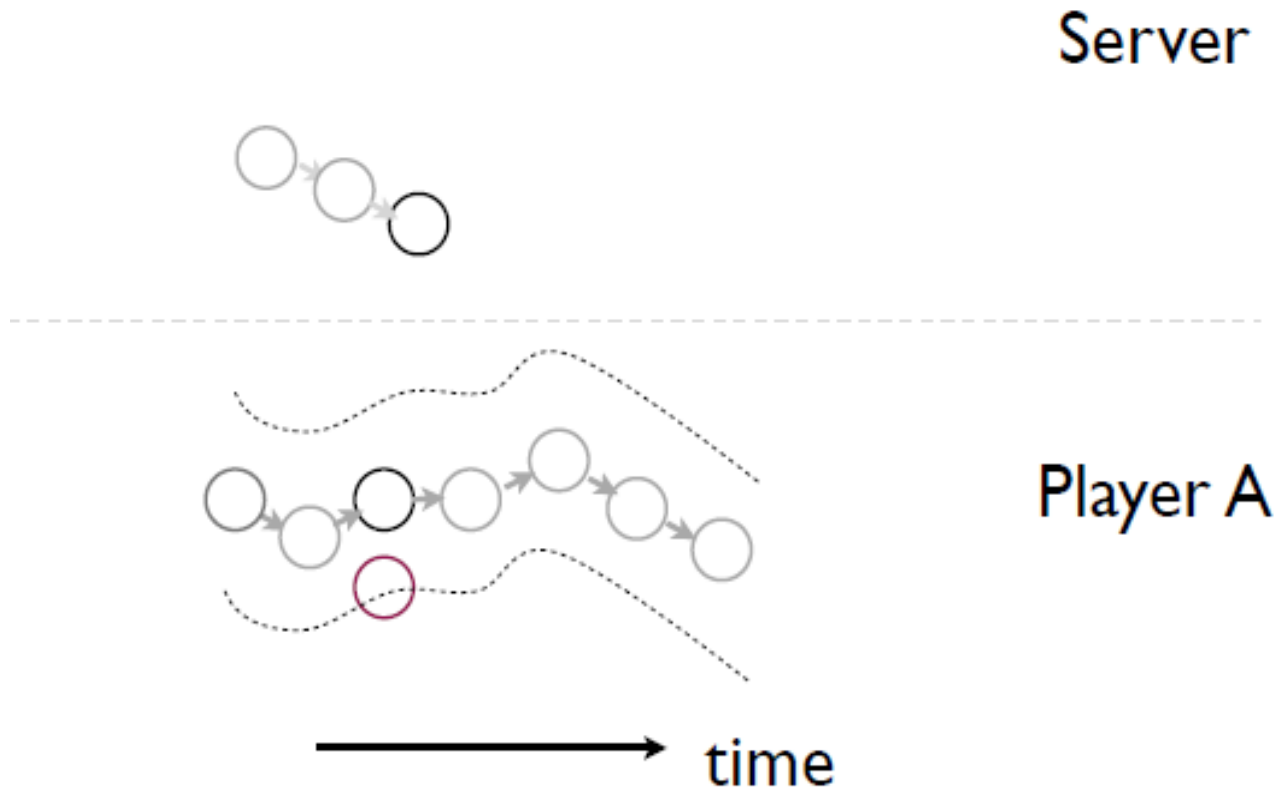
DEAD RECKONING: AN EXAMPLE



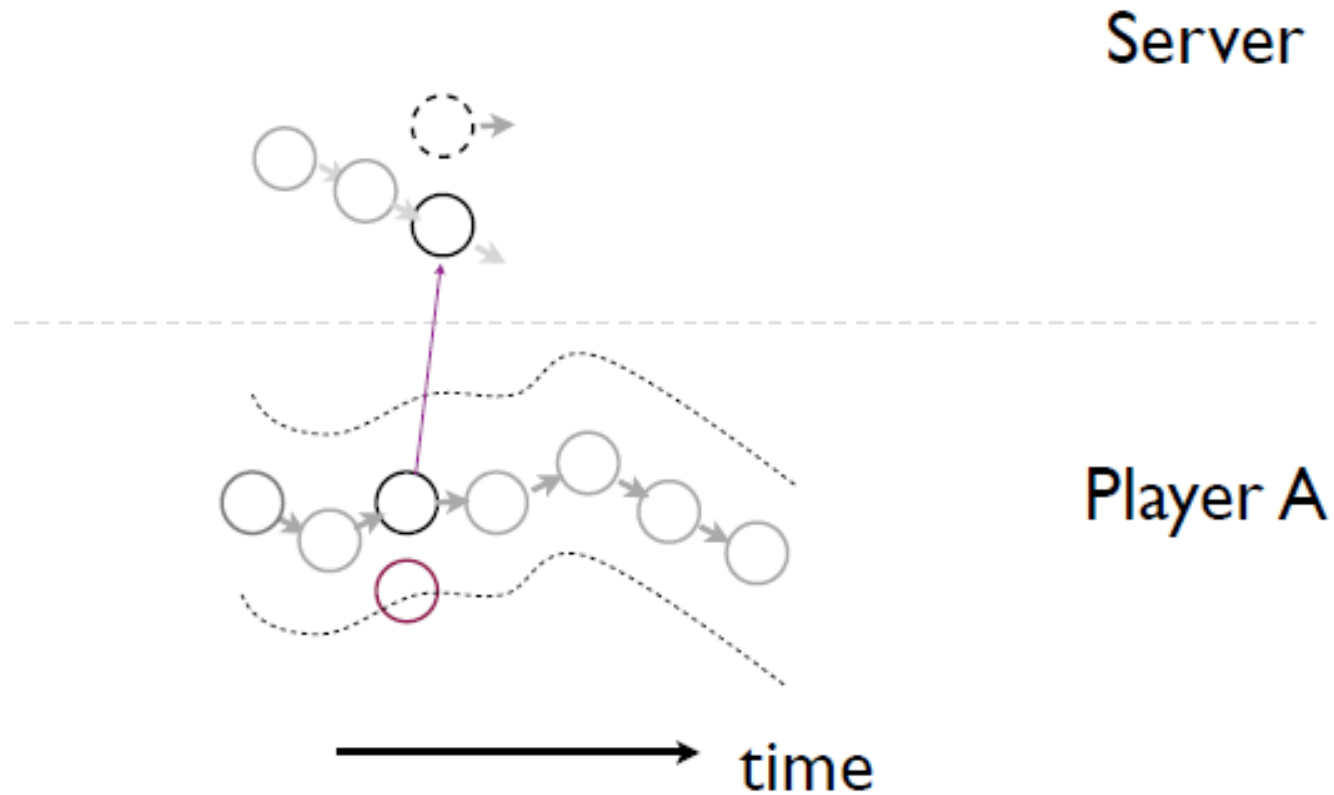
DEAD RECKONING: AN EXAMPLE



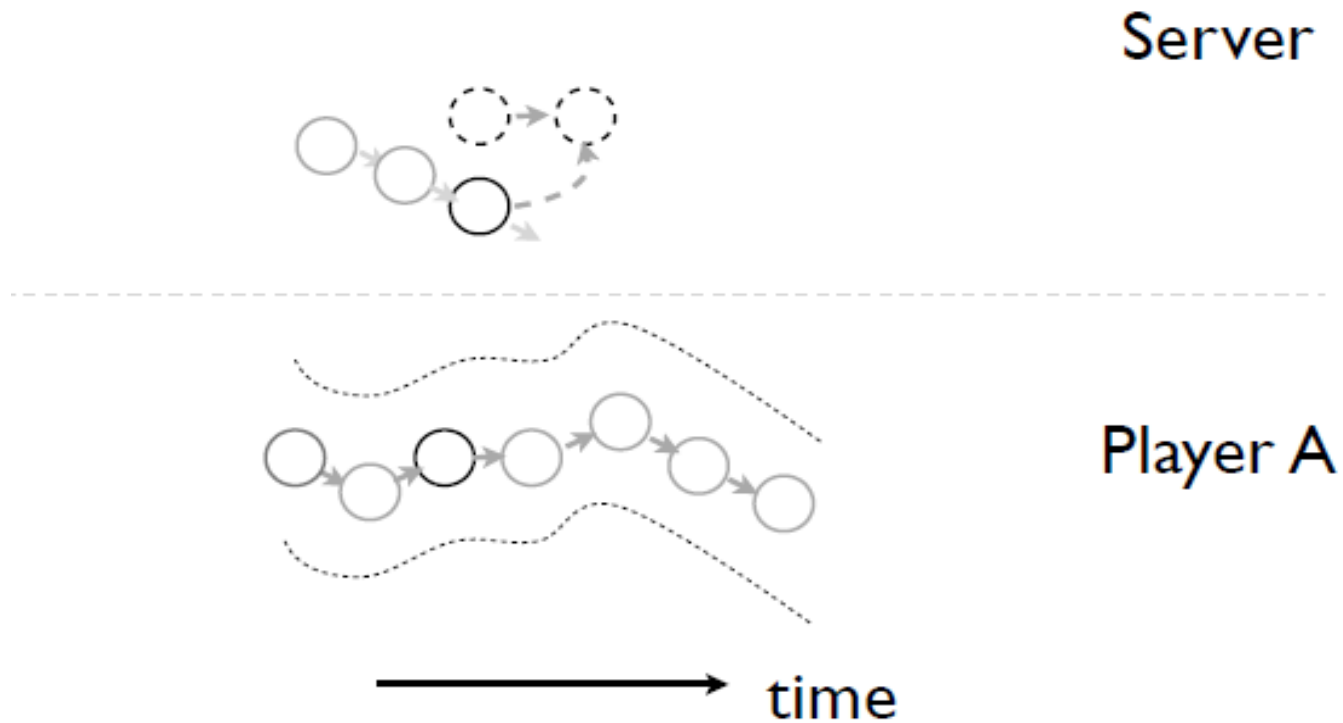
DEAD RECKONING: AN EXAMPLE



DEAD RECKONING: AN EXAMPLE



DEAD RECKONING: AN EXAMPLE



DEAD RECKONING AND CHEATING

- cheating: an unauthorized interaction with the system aimed at offering an advantage to the cheater

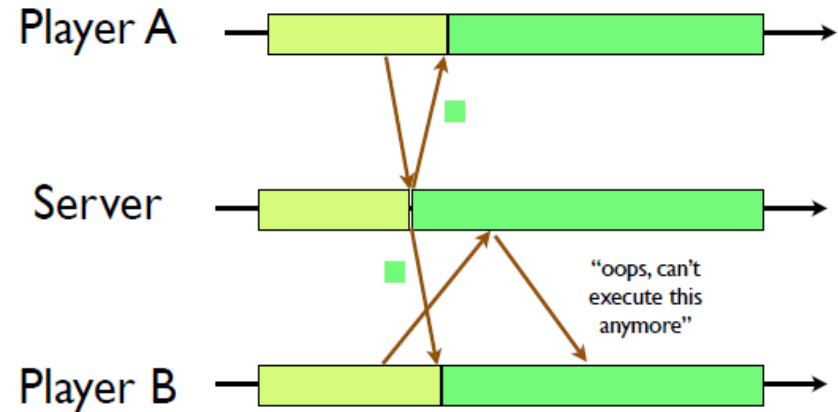
suppress correct cheat

- exploits dead-reckoning to gain advantages
- player purposely dropping some messages
 - a cheater purposefully drops $n - 1$ moves and then, having received the other players' last $n - 1$ moves, constructs a move based on this information
 - example: a sluggish player chasing an agile player
- Cannot be distinguished from an honest player with a lossy communication

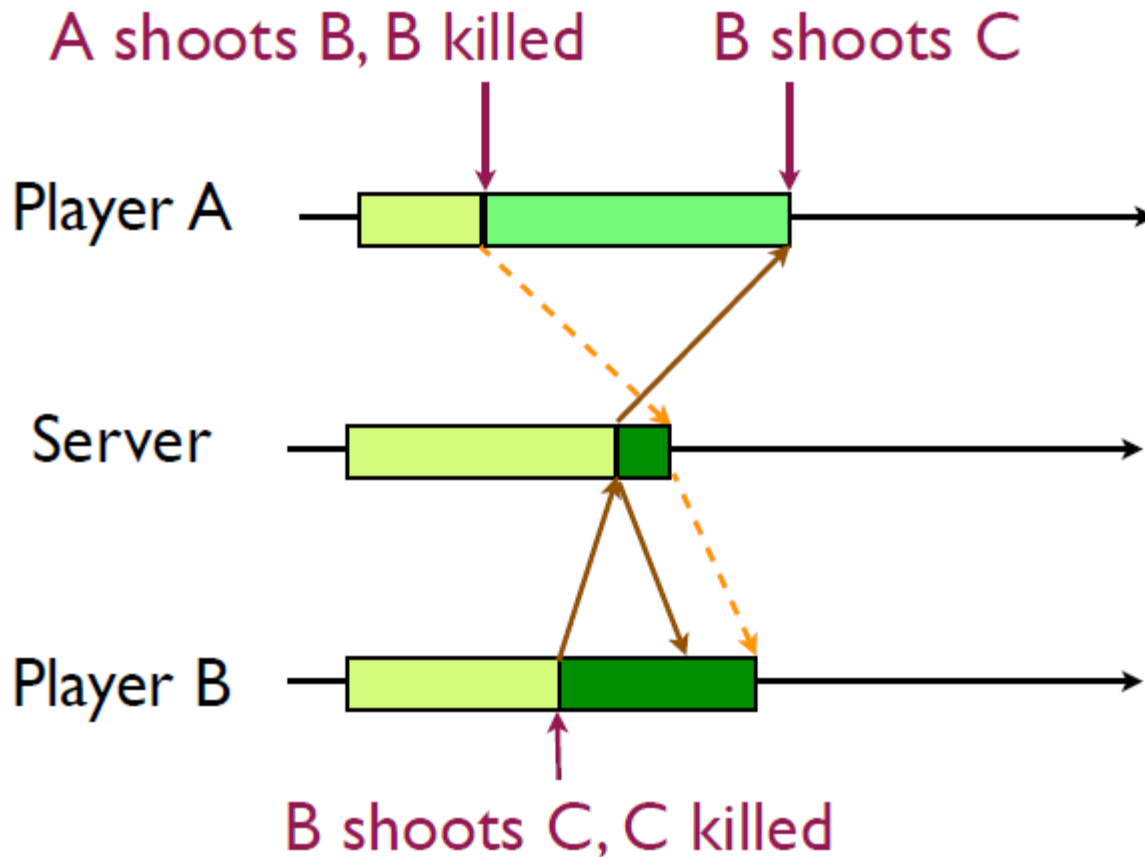
GAME CONSISTENCY

The server:

- Check the consistency of operations to avoid inconsistent ones
 - player B cannot drink a potion if it has already been drunk from A
- Unfair solution
 - Players with higher latency are disadvantaged w.r.t. others
- Alternative optimistic approach: do first, check later
 - execute actions locally and let inconsistency arise
 - server is the authority and solve inconsistencies
 - acceptable only for a class of events
 - important events like 'shoots' must be decided from the server



HOW TO KEEP A DEAD MAN FROM SHOOTING



In this case the optimistic solution does not work

Martin Mauve, how to keep a dead man from shooting

SIMULTANEITY VS. RESPONSIVENESS

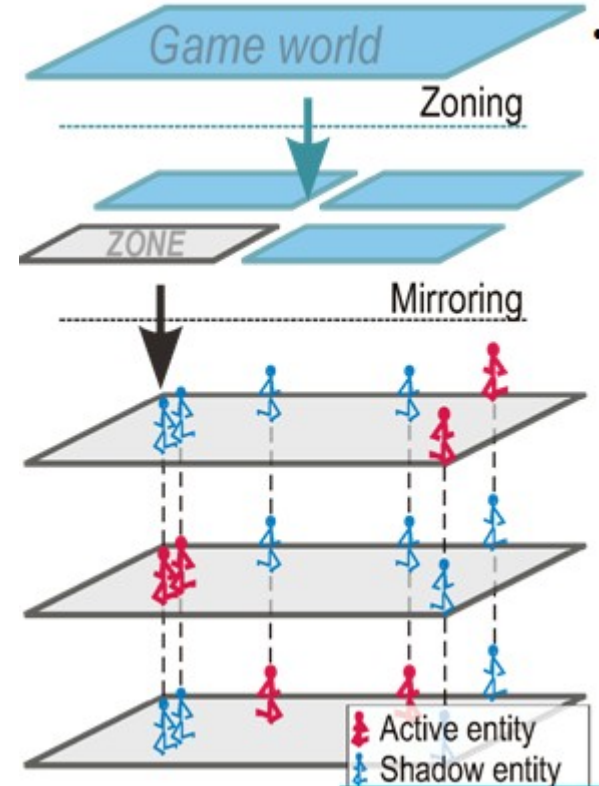
- To increase fairness and guarantee simultaneity of events, the server
 - measure the average latencies of each player
- When it must notify an event it sends the event
 - earlier to the player with higher latency
 - later to those with smaller latency
- Local Lag technique
- In this way
 - fairness and simultaneity of events increases
 - responsiveness decreases and is bounded by the player with worst latency

CLIENT/SERVER ARCHITECTURE

- most commonly used architecture in current commercial DVE
- Pro:
 - high degree of administrative control
 - ease of implementation of networking code
 - authentication
 - cheating control
 - billing
 - easy update of client code
 - persistent data stored at the server
- Cons:
 - single point of failure
 - scalability
 - cost

SERVER PARALLELIZATION TECHNIQUES

- **Zoning:** the world is split up into regions, each region is hosted by a different server
 - Second Life
- **Mirroring:** the same game world handled by different servers, each one handles a subset of the entities
 - mirrors must be kept consistent
 - high synchronization
 - players see everyone across all mirrors

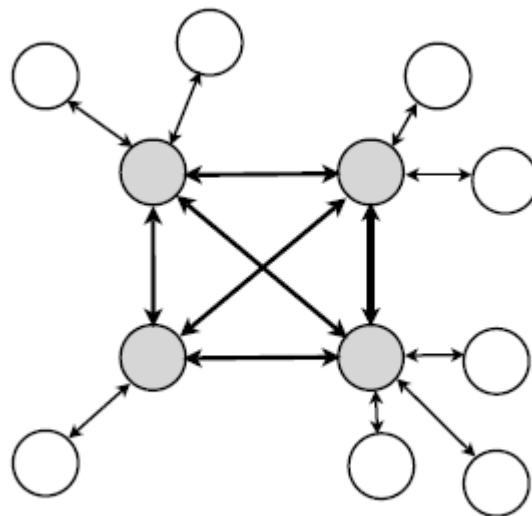


SERVER PARALLELIZATION TECHNIQUES

- **Sharding:** the world is replicated at each server; each shard contains an independent world; players go to specific shard
 - War of Warcraft
 - most MMORPG
- **Instancing:**
 - divide the virtual world into zones
 - multiple instances of the same zone, with independent states
 - last versions of WoW

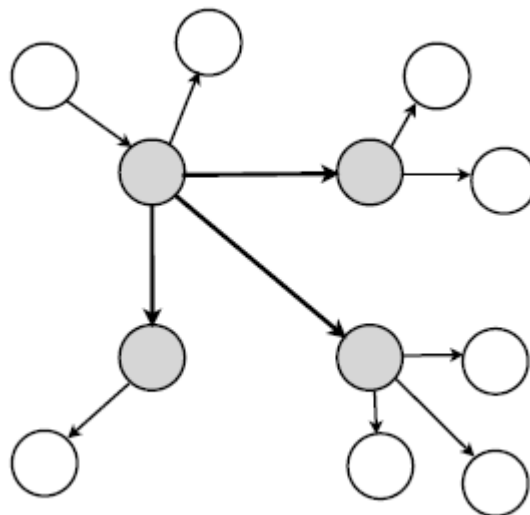
MIRRORED SERVER ARCHITECTURE

- Multiple servers, each **replicating the complete game states**, serve clients from different geographical regions
- Servers connected by high speed network to reduce synchronization latency
- Each client connects to one server



MIRRORED SERVER ARCHITECTURE

- Updates from a client are sent to its server which forwards them to the other servers
- The servers forward the updates to all relevant clients

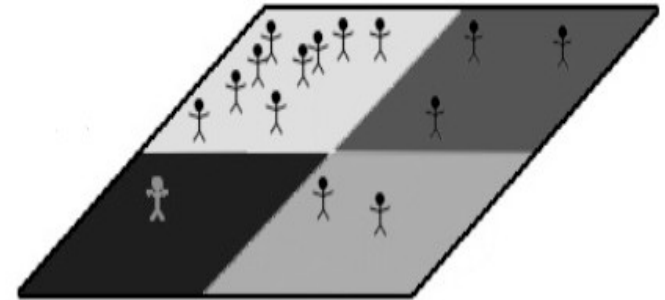


MIRRORED SERVER ARCHITECTURE

- Advantages
 - Increases scalability
 - No single point of failures
 - Lower latency server chosen by the clients
 - Synchronization between servers is more easy since
 - servers are trusted
 - clocks may be synchronized
 - IP multicast may be easily exploited
- Disadvantages
 - latency for clients updates increase w.r.t. classical client server solution
 - scalability is still an issue: servers maintain the whole game state

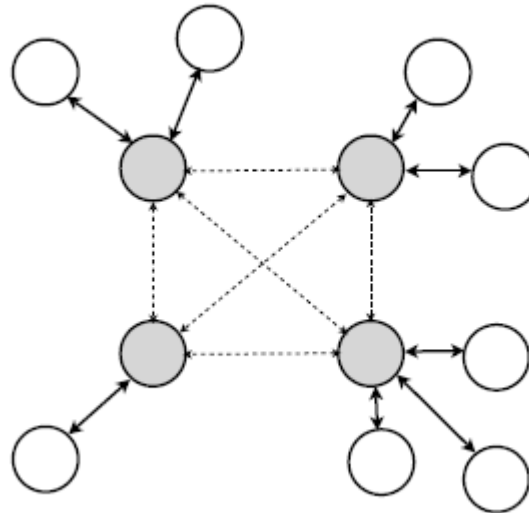
HYBRID SOLUTIONS: ZONING

- The Virtual world is partitioned (cell/zones) and distributed over multiple servers
 - One server in-charge of a region of the virtual world
 - Reduction of load for each server
- Partitioning may be
 - static: may still lead to overloaded servers
 - dynamic: problem of cell/zones resizing
- Cell size is critical for system behaviour
 - interaction with interest management: cell size is not smaller than AOI
 - too few cells leads to overloaded cells
 - too much cells leads to frequent clients handoff.



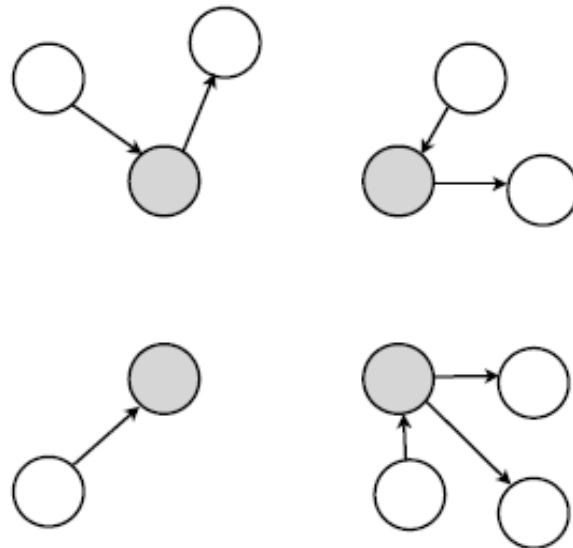
HYBRID SOLUTIONS: ZONING

- A client connects to the server serving its current region and is handed-off to another server when it moves to another region
- Interaction is admitted only between players located in the same region
- If cell size is not too small, communication among servers is less frequent than in the mirrored solution



HYBRID SOLUTIONS: ZONING

- Updates of a client are sent to its server
- The server forwards them to all the clients in the same region
- No communication between players of different regions

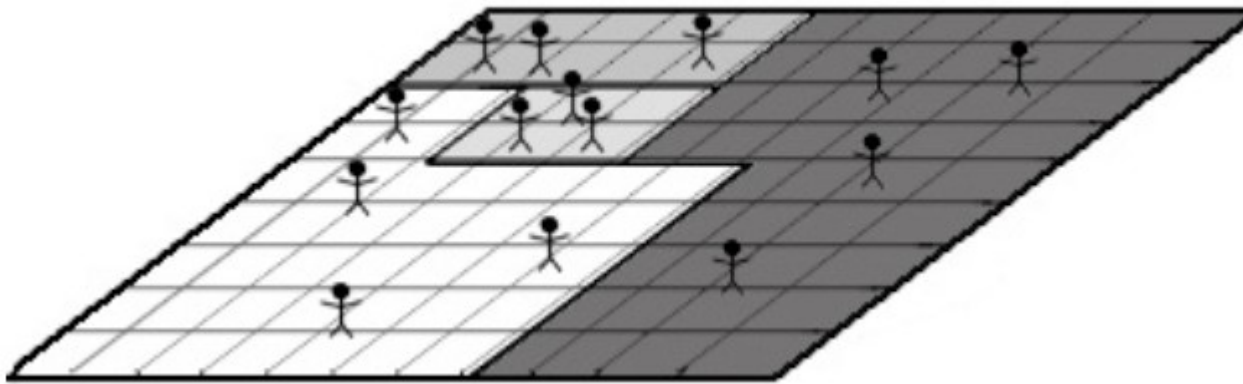


HYBRID SOLUTIONS: ZONING

- Pro
 - reduces inter-server communication
 - an higher degree of scalability: each server manages a single region
- Cons
 - single point of failure for each region
 - partition is not transparent to the player: passing from a region to another one is not seamless
 - game design constraints
 - definition of a set of rules to pass from a region to another one
 - connection by flights, special ports, tele-porting
 - transition accompanied on the screen by a load clock or standard animation video

ZONING: MICROCELLS

- zones are partitioned into micro cells
 - micro cells may be transferred across servers
 - load may be balanced at run time
 - zones dynamically change their size



ZONING: SHARDING

- Partitioning of the virtual world
 - Instantiation of one server cluster (shard) for partition
 - Shard share account data
 - Users may be routed to different shards, according to shard load
 - Zoning within shard possible
- Implemented by World of Warcraft
- Pros
 - Balancing by routing of login
- Cons
 - Shards are completely independent

HYBRID SOLUTION: GAME INSTANCES

- Instance: a simplified version of replication
 - special area (a dungeon) such that a new copy of it is generated for each group, or for a predefined number of players entering that area
 - each instance is completely independent from the others
 - players belonging to different instances cannot interact even if they are geographically close in the virtual world
 - reduce competition over resources: avoids players harass each other in the same dungeon, in order to get to the most valuable items
 - improve game playability
- Instancing allows resource optimization
 - reduces traffic to/from the central server
 - reduces lag and improve responsiveness
- Adopted by several recent MMOG

FROM CLIENT SERVER TO P2P

Drawbacks of centralized architectures:

- servers have to be deployed, operated and maintained
 - high cost for the game operator
- scalability problems:
 - server have to be dimensioned according to the maximum number of expected players
 - underprovisioning lead to high response time/player frustration
- limited server availability: single point of failures
- fairness problems: players experience different latencies

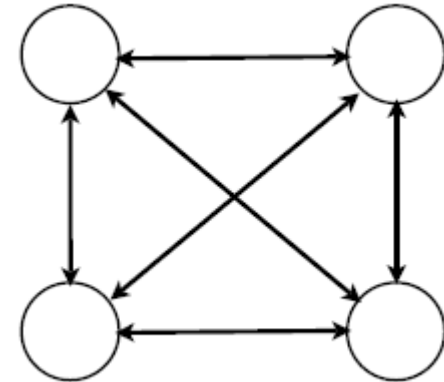
These drawbacks are strong incentives to investigate **alternative decentralized P2P architectures**

P2P SOLUTIONS FOR MMOG

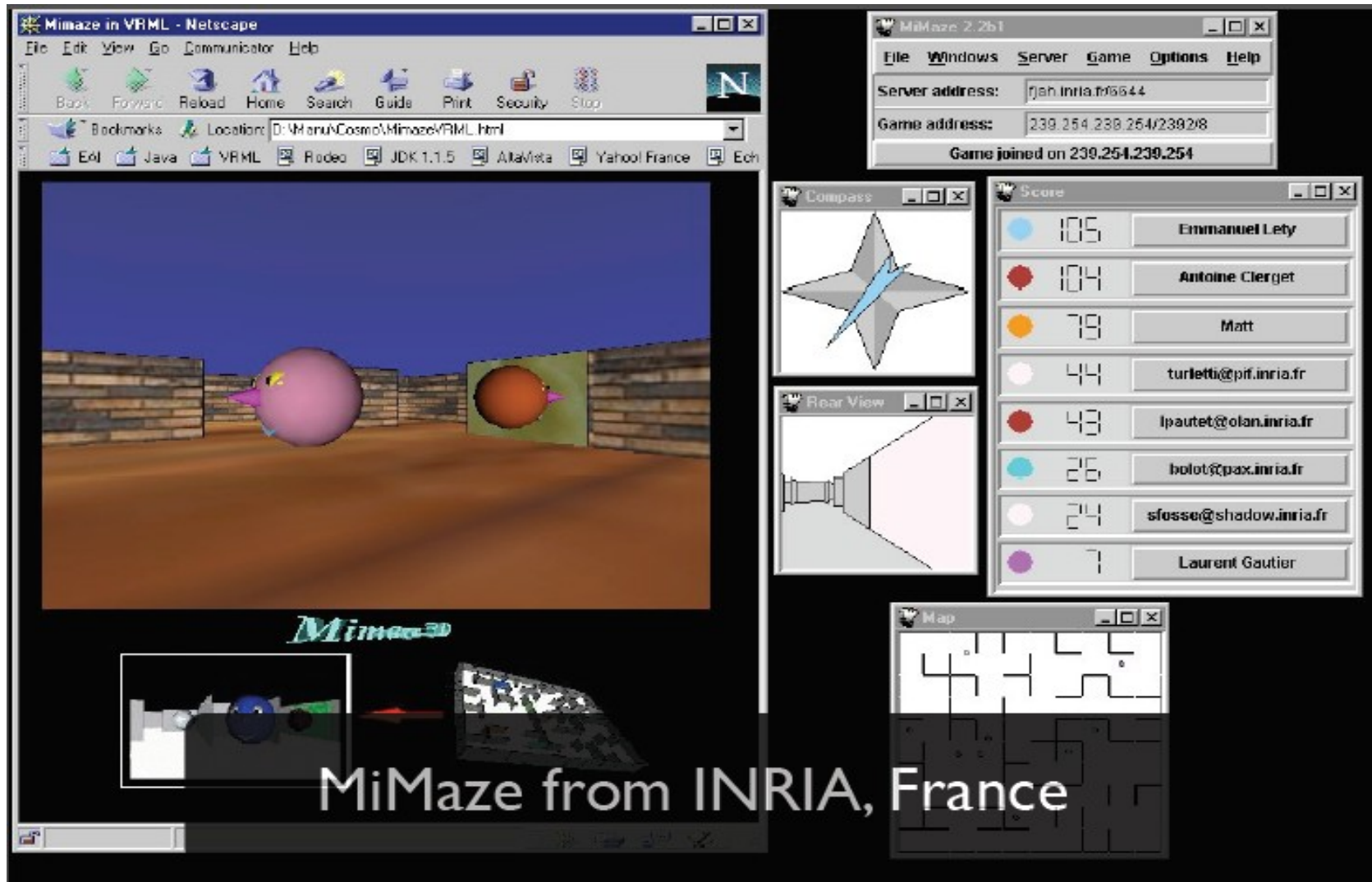
- Reduces deployment investment: no physical resource to be provisioned
- Naturally scalable solution
 - all players benefit from the shared resource (CPU, storage and bandwidth)
- No single point of failure: no catastrophic failure if one node leaves the network
- Self organization to optimize the delay between the peers
- Suitable computational paradigm to deploy games over **infrastructure-less/mobile networks**
 - user with intermittent connectivity, or connectivity with a small sets of devices
 - heterogeneous devices
 - "mixed reality games", where players physical location impacts the location of the avatar in the game

P2P SOLUTIONS FOR MMOG: CHALLENGES

- P2P overlay definitions
 - Routing strategy
 - AOI-cast distributed mechanisms
- management of distributed storage
- consistency and persistence of game states
- delay optimization between peers
- protection against cheating
 - no trusted centralized authority



MIMAZE: A MULTIPLAYER DISTRIBUTED GAME



P2P SOLUTIONS FOR MMOG

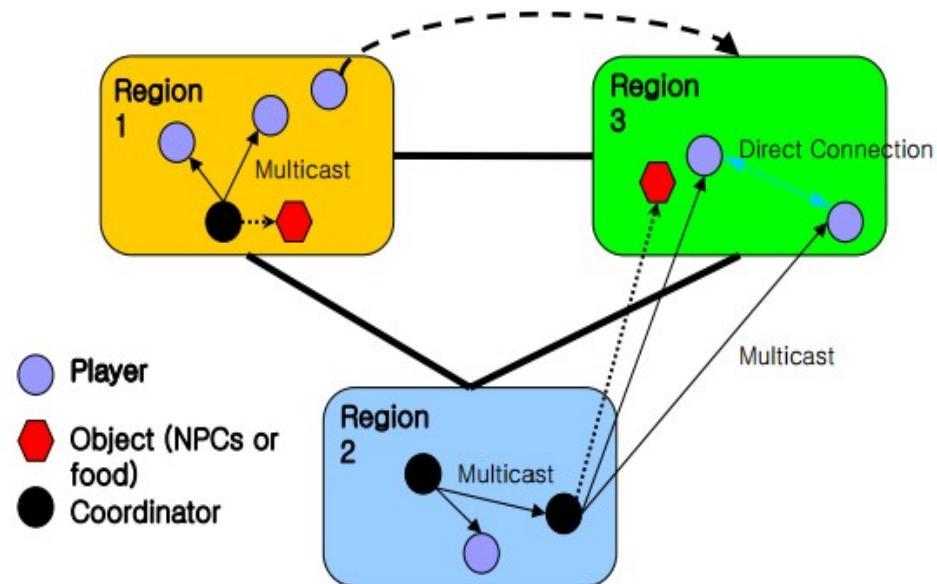
- Mimaze: first multiplayer game with a distributed architecture
 - exploits IP multicast
- After that, several research proposal in the last 10 years:
 - DHT based
 - Gossip based
 - Voronoi/Delaunay based
- Due to limited time we will investigate
 - SIMMUD, a DHT solution
 - A proposal from my research group, based on Delaunay overlays

DHT BASED MMOG: SIMMUD

- Players contribute memory, CPU cycles and bandwidth for the shared game state
- Persistent user state is centralized
 - Example: payment information, character
 - Transactional consistency
- Exploits interest management
 - limited amount of state a player has access to
- The implementation based on:
 - **Pastry** distributed Hash Table
 - **Scribe** application level multicast support built on top of Pastry

SIMMUD: A DHT BASED MMOG

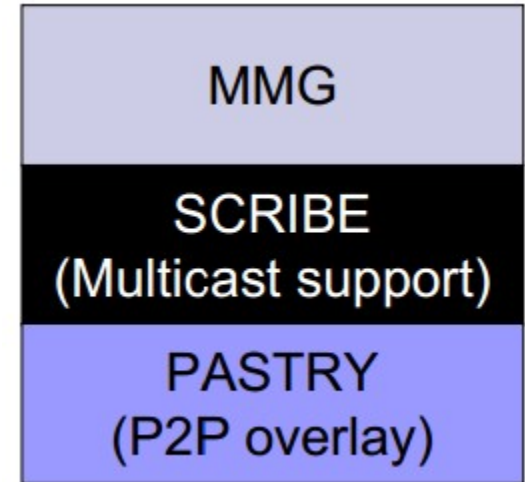
- Divide entire game into several regions
 - hash region name into P2P key space
- Players in the same region form an interest group and change group when going from region to region
- A single coordinator per region
 - one of the players
 - distribution server of the map
 - resolves conflict updates
 - send state updates relevant to the group
 - root of the multicast tree



Reference: [INFOCOM 2004]: B. Knutsson, H. Lu, W.Xu, B. Hopkins

SIMMUD: A DHT BASED MMOG

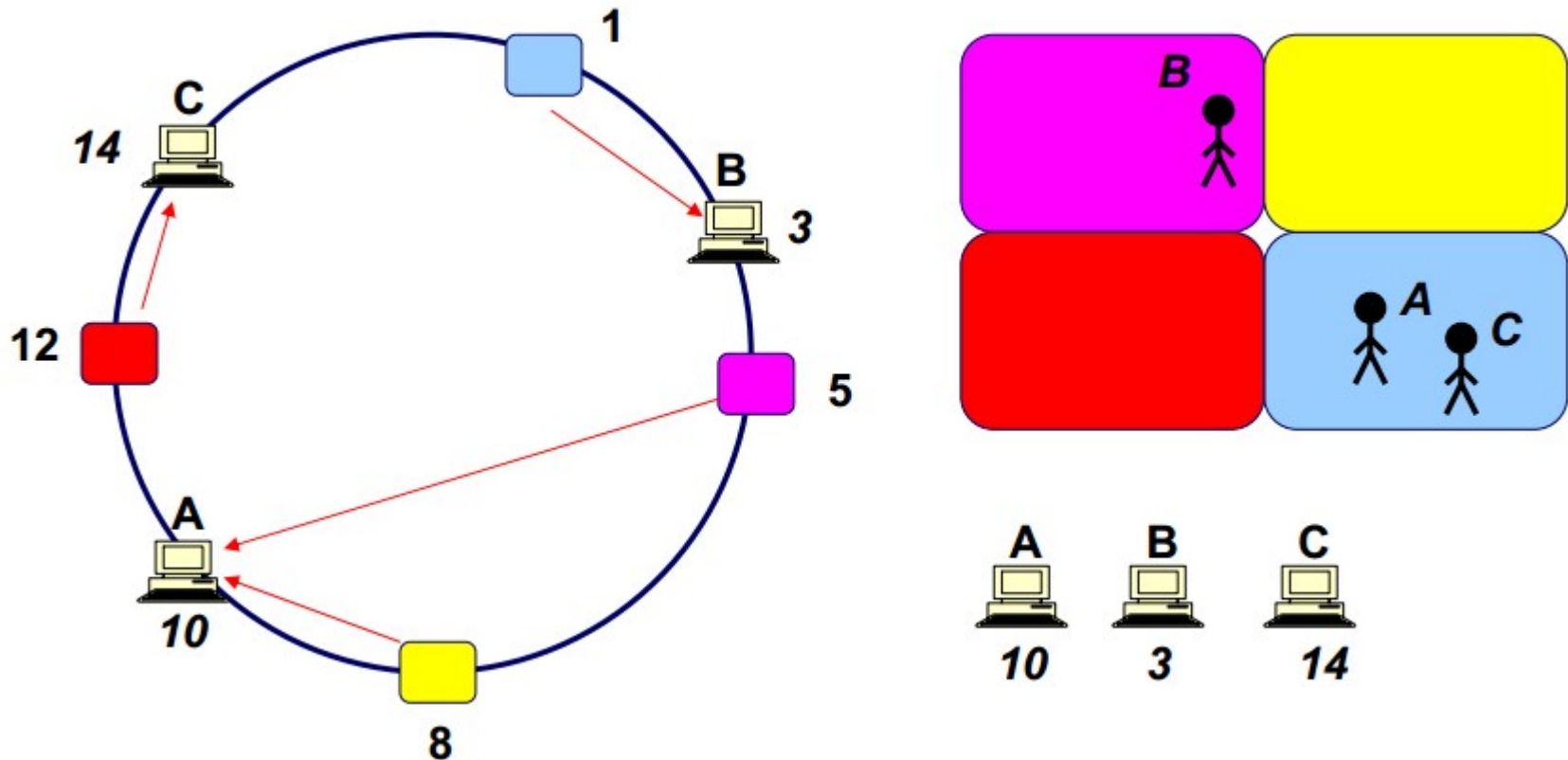
- P2P overlay
 - scale up and down dynamically
 - self organizing decentralized system
- Scribe: Multicast support defined on top of Pastry: the coordinator exploits this DHT-based multicast to notify state update to all the peers of a region
- Scribe: a multicast tree associated with a group is formed by the union of the Pastry routes from each group member to the group ID's root, which also serves as the root of the multicast tree. Messages are multicast from the root to the members using reverse path forwarding.



SIMMUD: STATE CLASSIFICATION

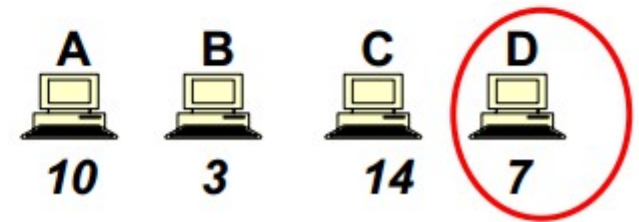
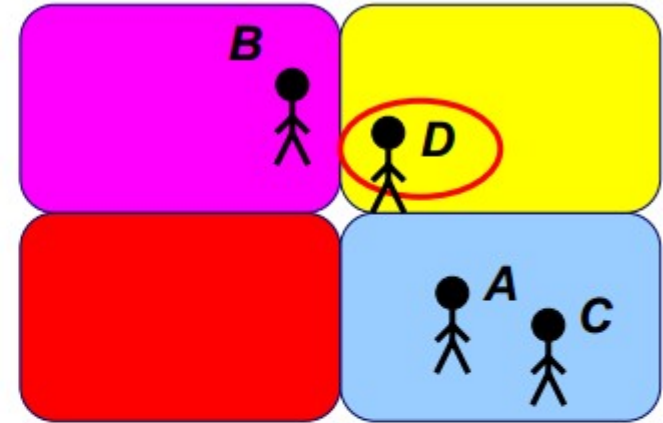
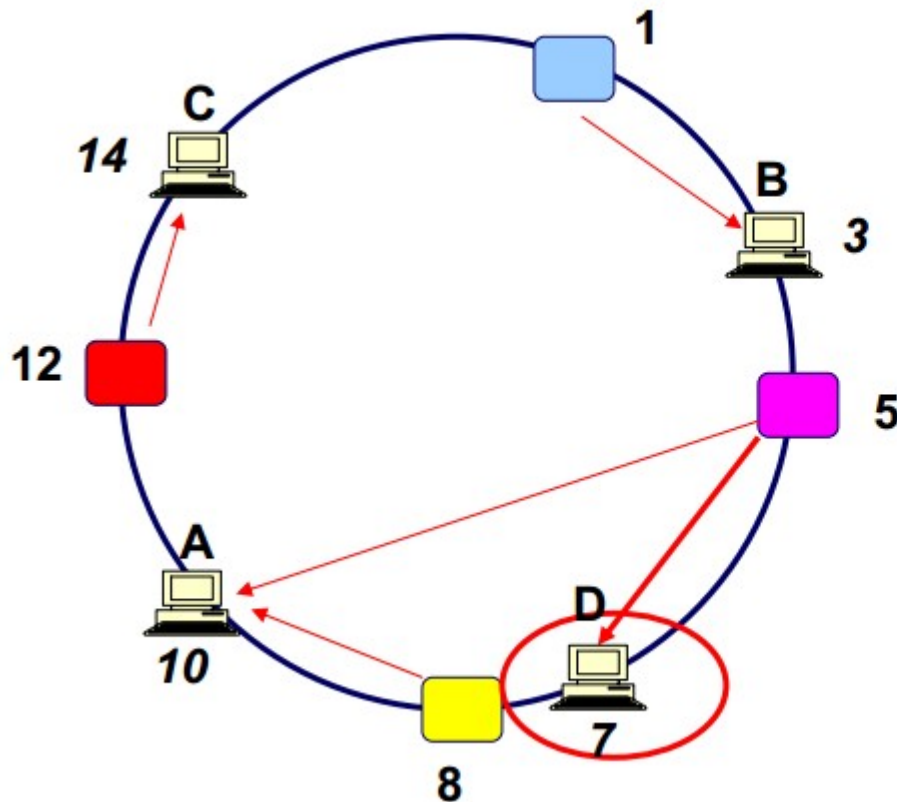
- **Player state: Single Writer - Multiple Readers**
 - Position change is most common event
 - use best effort multicast to players in the same region
 - use dead reckoning to handle loss or delay
- **Object state: Multiple Readers- Multiple Writes**
 - use coordinator based mechanism for shared objects
 - coordinator resolves conflicts and hold up-to date values
- **Maps: Multiple Reader, No writers**
 - maps are considered read-only because they remain unchanged during the game-play
 - If they are created online and inserted in the system dynamically, dynamic map elements are handled as objects

SIMMUD: MAPPING REGIONS TO COORDINATORS



Regions and nodes (players) mapped to the same logical space
A region is managed by the successor node in the key space

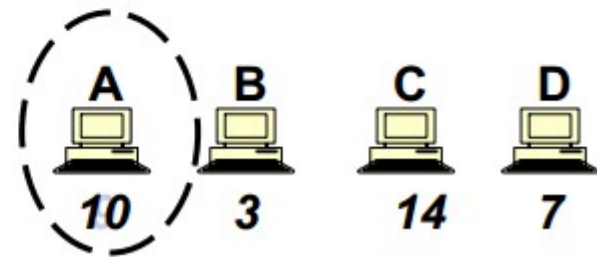
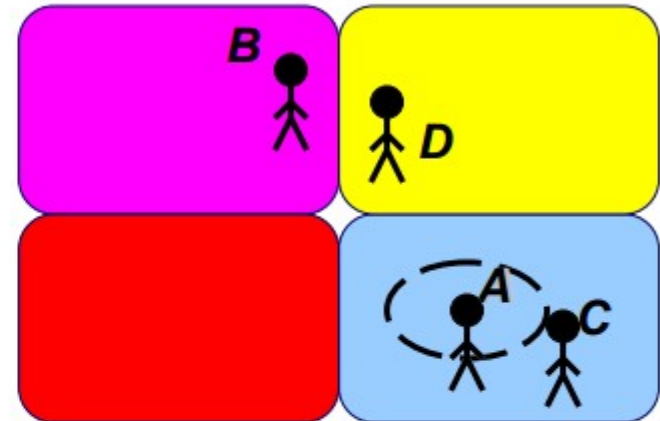
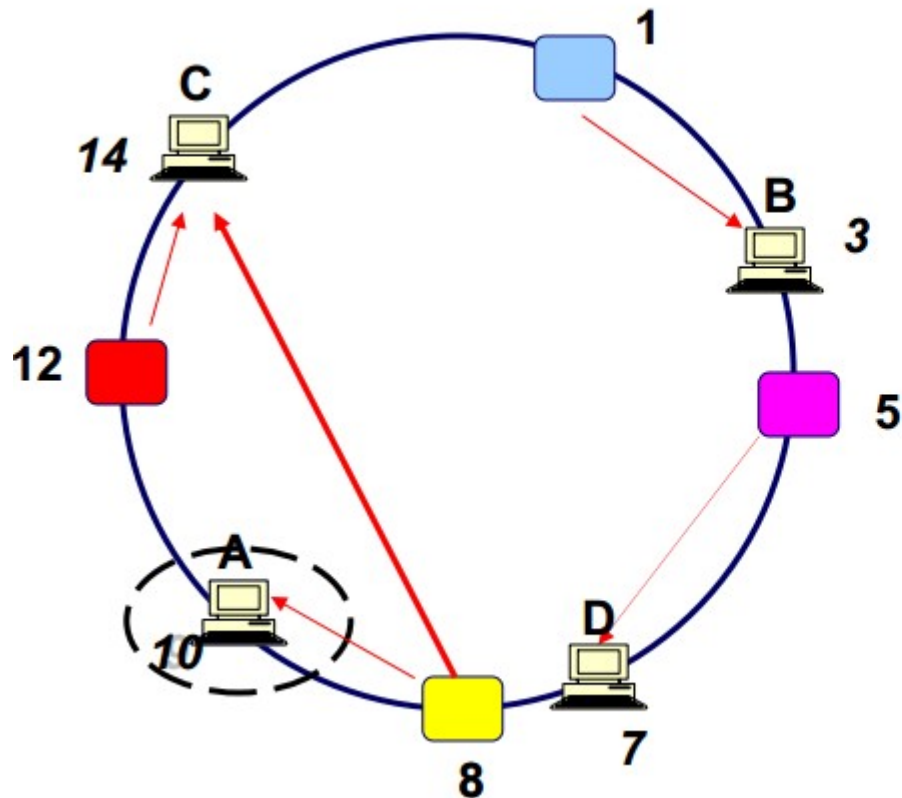
SIMMUD: NODE JOIN



Player D on node 7 joins the game

Exploit DHT functionalities to define the new region/node assignment

SIMMUD: NODE LEAVE

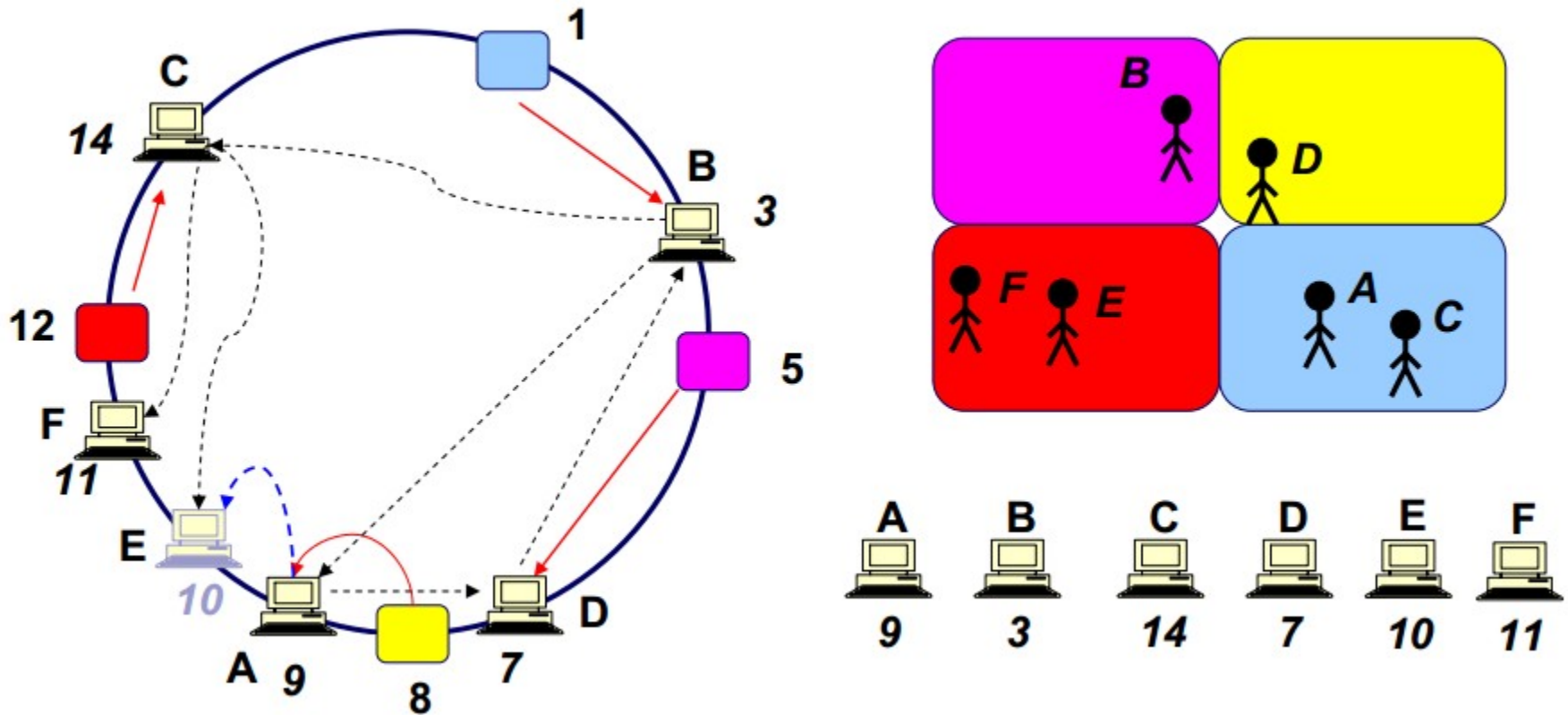


Player A on node 10 leaves the game

Region 8 is remapped to node C

DHT functionalities are exploited to define the new mapping regions/peer

SIMMUD: INTERACTION BETWEEN PLAYERS



Black arrow: logical interaction between coordinator of a region and players

Real communication is implemented via application-level multicast

Information is replicated on successive nodes

Except node E and F each node is both coordinator and player

DHT BASED COORDINATOR

- Because of the random mapping, the coordinator of a region is unlikely to be a member of the region,
- Random mapping implies lack of locality but....:
 - reduces the opportunities for cheating by separating the shared objects from the players that access them. Second,
 - no handing off the coordinator when the corresponding player leave the region
 - coordinator hand-offs when a player either joins or leaves the game.

SHARED STATE REPLICATION

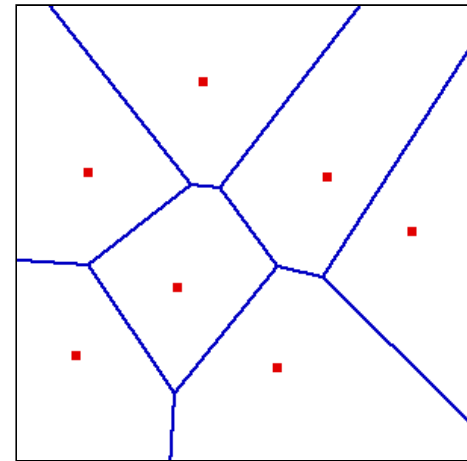
- Failure detected through regular game events
- Keep at least one replica of the region at all time
- Replica kept at node N which is the next closest to the key K of the region
- When a node T which is closer to K than its current coordinator N joins the DHT
 - forward messages to N until all state of region K is transferred
 - takes over as coordinator and N becomes a replica

DELAUNAY BASED OVERLAYS FOR MMOG

- Consider a set $S = \{s_1, \dots, s_n\}$ of n distinct points (sites) of the plan. Each site corresponds to a peer
- $\text{dist}(p, q)$ = the euclidean distance between points
- Voronoi Tessellation of S = Partition of the plane into n regions, where each region, $V(s_i)$:
 - ♦ corresponds to a site in $s_i \times S$
 - ♦ includes all the points q such that
$$\text{dist}(q, s_i) \leq \text{dist}(q, s_j) \text{ for } i \neq j$$

Voronoi neighbours = Sites whose Voronoi regions' borders overlap

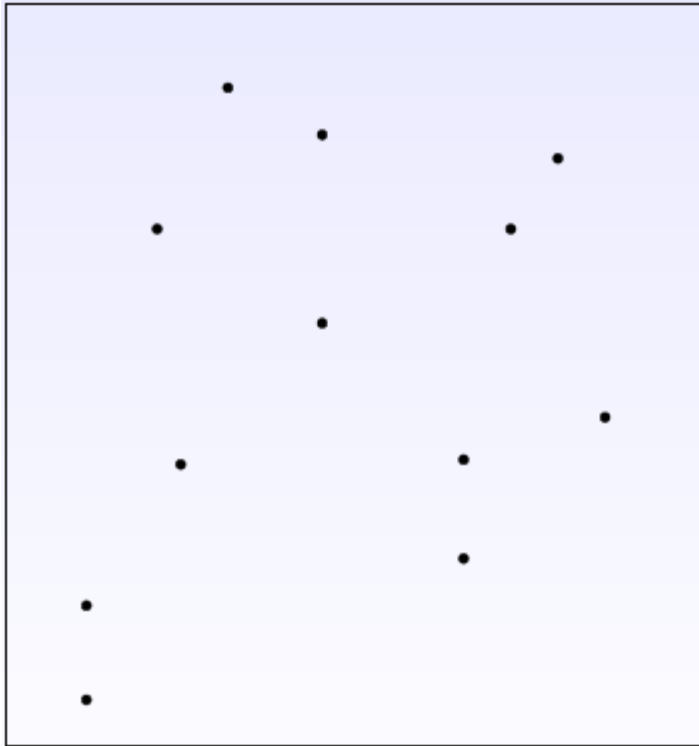
- Delaunay Triangulation = Graph defined by connecting sites which are Voronoi neighbours



DELAUNAY BASED OVERLAYS FOR MMOG

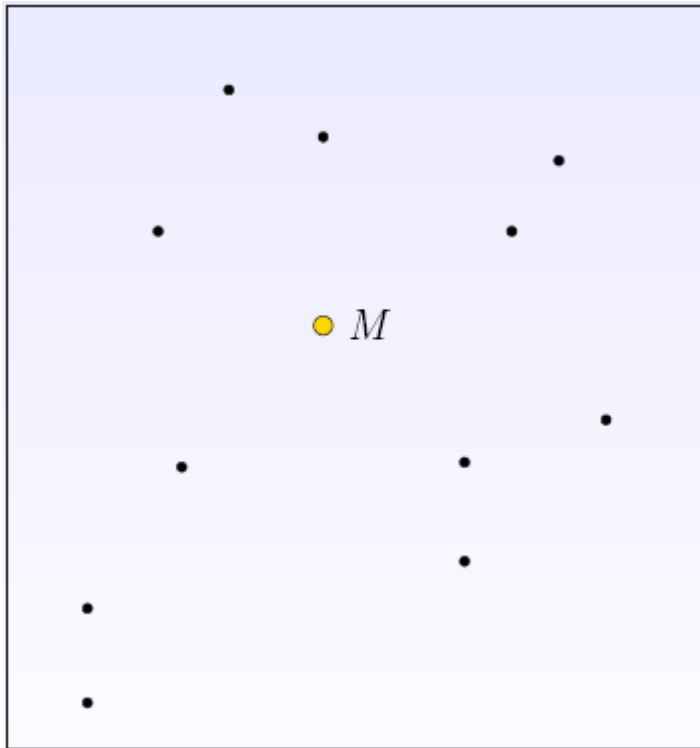
- An alternative solution is based on dynamic P2P overlays
- Each peer connects to those which are closer to it in the virtual space
- Overlay structure modified dynamically as the peer moves
- Requires a notion of closeness in the virtual space
- Voronoi Diagrams and Delaunay triangulation may be exploited
 - Well known structures in computational geometry
 - Centralized algorithm for the construction of Delaunay are well known
 - Distributed and Efficient algorithm have been recently been proposed

A VORONOI TESSELLATION



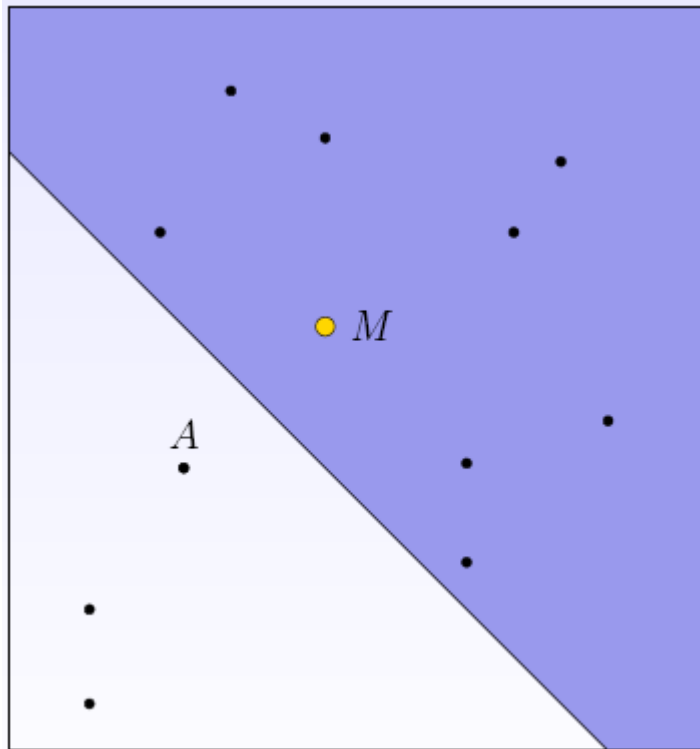
- A set of sites in \mathbb{R}^2

A VORONOI TESSELLATION



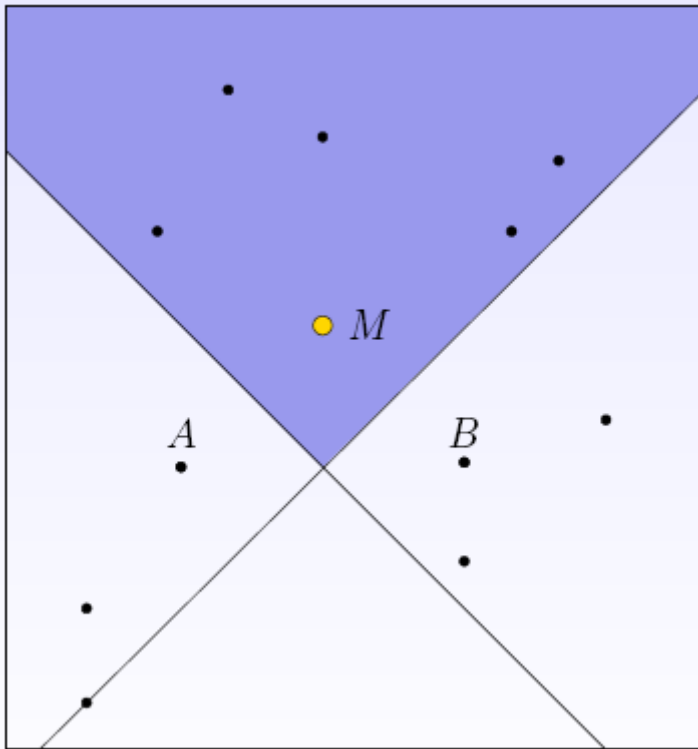
- A set of sites in \mathbb{R}^2
- Consider a site at point M

A VORONOI TESSELLATION



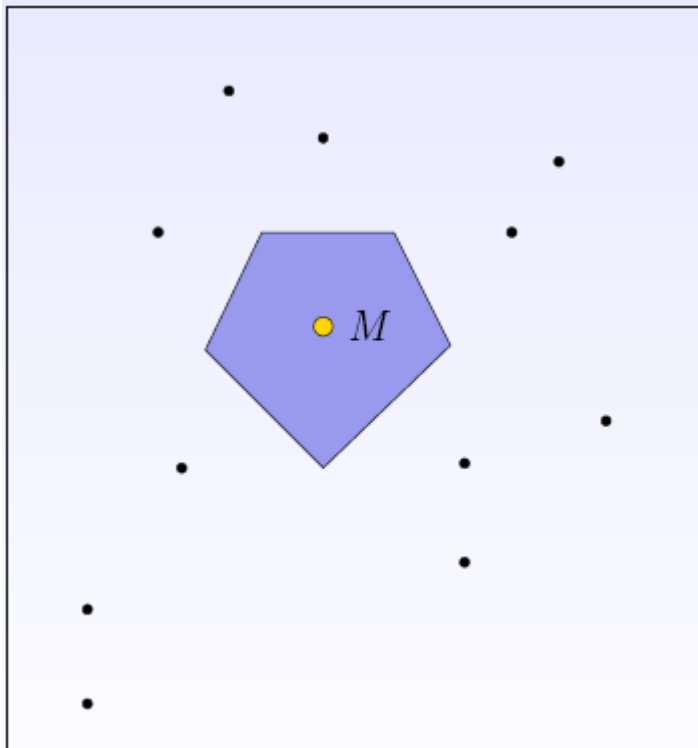
- A set of sites in \mathbb{R}^2
- Consider a site at point M
- Region of points closer to M than to A

A VORONOI TESSELLATION



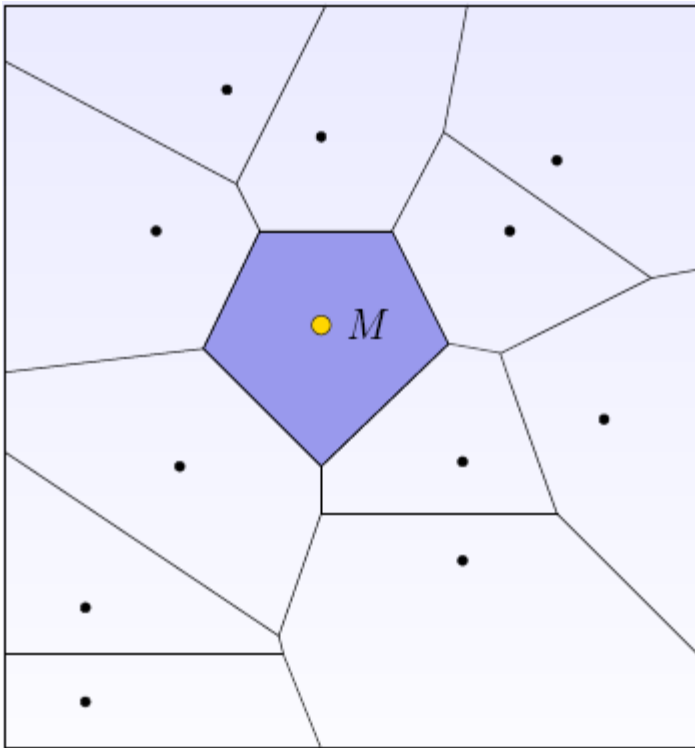
- A set of sites in \mathbb{R}^2
- Consider a site at point M
- Region of points closer to M than to A
- Region of points closer to M than to A/B

A VORONOI TESSELLATION



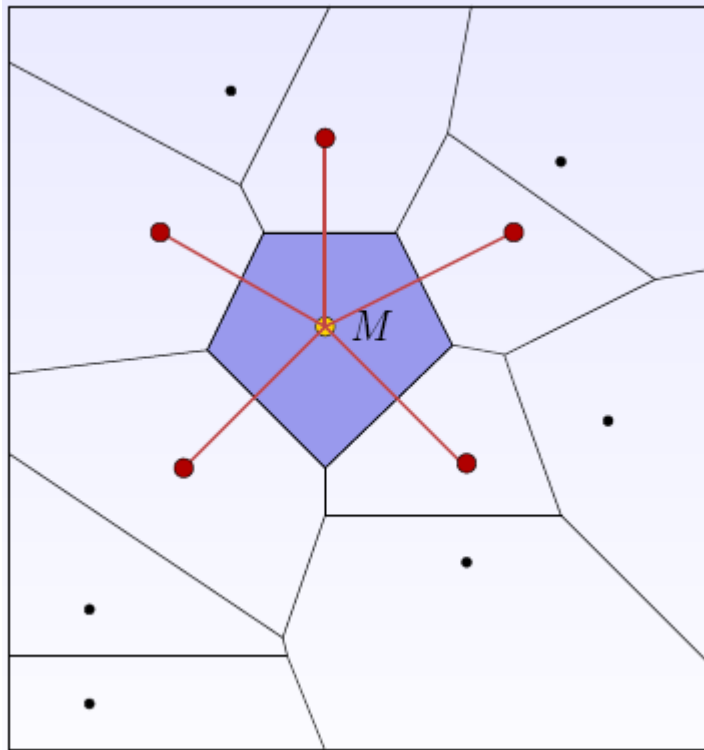
- A set of sites in \mathbb{R}^2
- Consider a site at point M
- **Voronoi Region of M :** region of points closer to M than to any other object

A VORONOI TESSELLATION



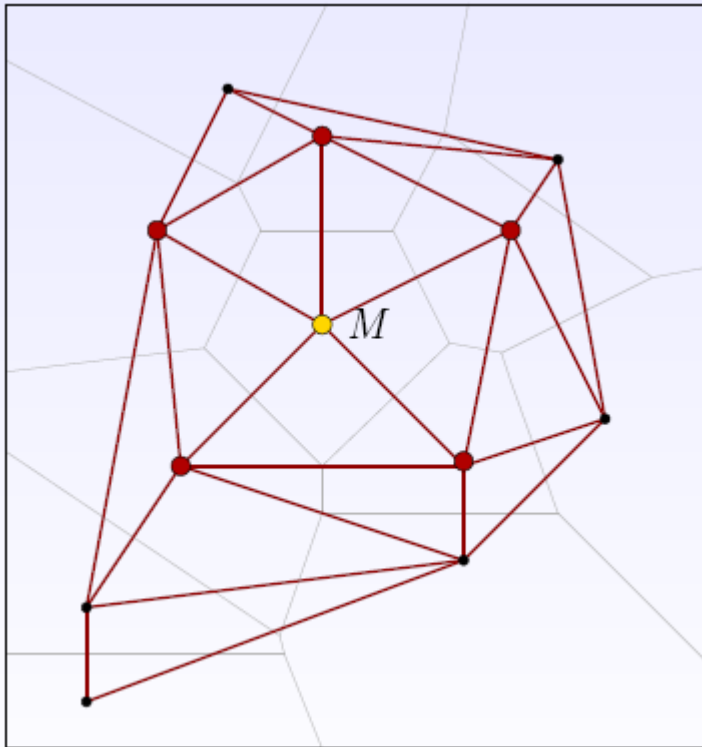
- A set of sites in \mathbb{R}^2
- Consider object at point M
- **Voronoi Region of M :** region of points closer to M than to any other object
- do the same for all objects

A VORONOI TESSELLATION



- A set of sites in R^2
- Consider object at point M
- **Voronoi Region of M :** region of points closer to M than to any other object
- do the same for all objects
- **Voronoi neighbours:** their region share a border

A DELAUNAY TRIANGULATION



- A set of sites in \mathbb{R}^2
- Consider site M
- **Voronoi Region of M** : region of points closer to M than to any other object
- do the same for all objects
- **Voronoi neighbours**: their region share a border
- **Delaunay triangulation**: links Voronoi neighbours

DELAUNAY BASED P2P MMOGs

- The position of each peer is exploited to define a P2P overlay where links correspond to Delaunay triangulation
 - Delaunay links guarantee **overlay connectivity**
- Requires an **efficient distributed algorithm** for the construction of the dynamic construction Delaunay overlay
 - neighbour test based approaches
- The overlay may be exploited to propagate player positions (heartbeats)
- Each peer P notifies its positional updates to its Delaunay neighbours
 - each neighbour propagates the heartbeats received by P to other peers it knows which are located inside the AOI of P
- **Pass the word mechanism'**. Peers become acquainted of each other through their Voronoi neighbours

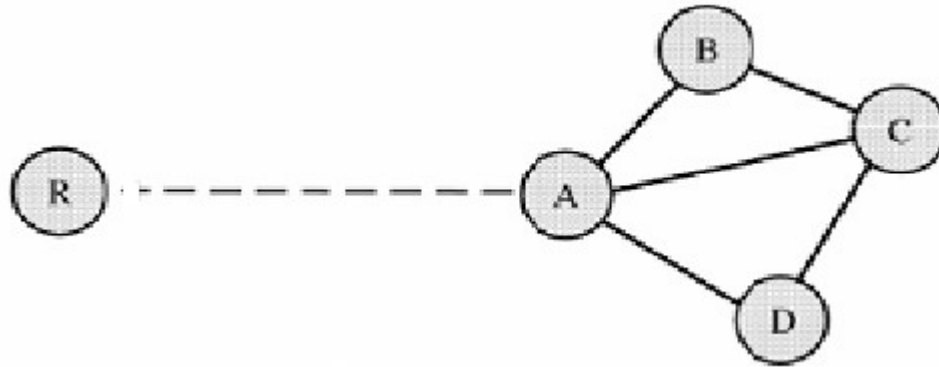
AOI-CAST ROUTING

The AOI of a peer P may include

- no Voronoi neighbours of P, for instance P is in a uninhabited region of the DVE.
 - In this case the Delaunay links guarantee connectivity
- a superset of the Voronoi neighbours of P
 - requires the definition of a proper mechanism for notifying each event of P to any peer belonging to its AOI
- AOI-cast:
 - **application level multicast** constrained within the AOI of a peer
 - requires the definition of a proper **routing strategy**
 - **compass routing**: exploits Delaunay properties to compute a spanning tree covering the peers in the AOI

COMPASS ROUTING

Exploits the topological properties of the Delaunay triangulation to define a spanning tree over the AOI



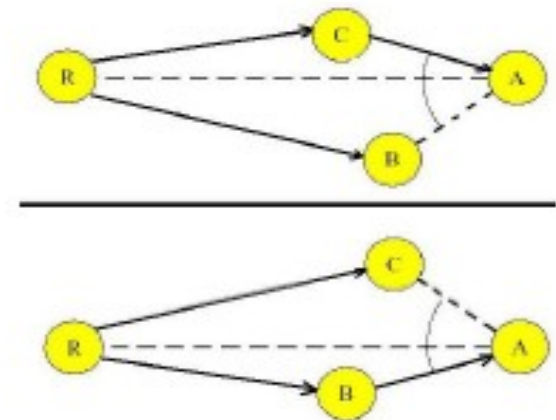
- Root of the spanning tree peer R generating the heartbeat
- a recursive spanning tree construction process based on angles is started at R
- Angle argument:
 - C is a child of A in the spanning tree rooted at R because the angle RCA is smaller than RCB and RCD
- AOI-cast based on the construction of the spanning tree

AOI CAST ROUTING

- **Crowding Scenario:** a huge amount of peers are located in the AOI of P
 - peers are attracted by the same object (a potion or a sword,...) or gather to fight each other
 - a large number of routing hops may be required by the AOI-cast routing algorithm when a crowding scenario occurs
 - responsiveness of the DVE is reduced due to large notification delays
- A set of **direct connections** between a peer and a subset of other peers in its AOI may be added to the Delaunay links
 - these connections acts as 'short-cuts' to reduce the notification delay
- Trade off: a large number of direct connections
 - reduces the notification delay
 - increases the bandwidth requirement of each peer

INCONSISTENCIES IN COMPASS ROUTING

- Latency may introduce inconsistencies in the local views of the peers
- Two peers may have a different perception of a common neighbour, due to the network delays in the heartbeat notification
- In the upper part of the figure
 - local view of B
- In the lower part:
 - local view of C
- Neither B nor C propagates the hertbeat to A



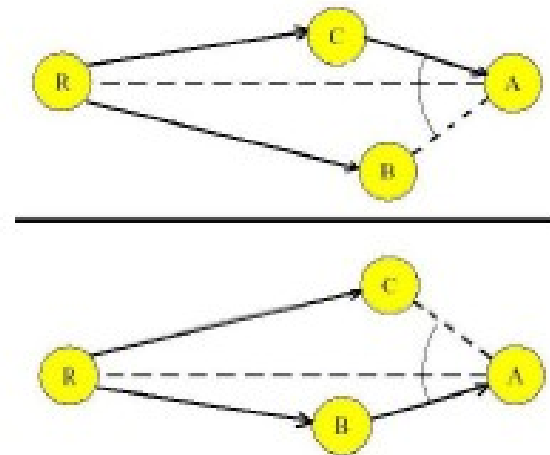
An incorrect behaviour of compass routing may be generated by an inconsistent local view

notifications replication
notification loss

TOLERANCE BASED COMPASS ROUTING

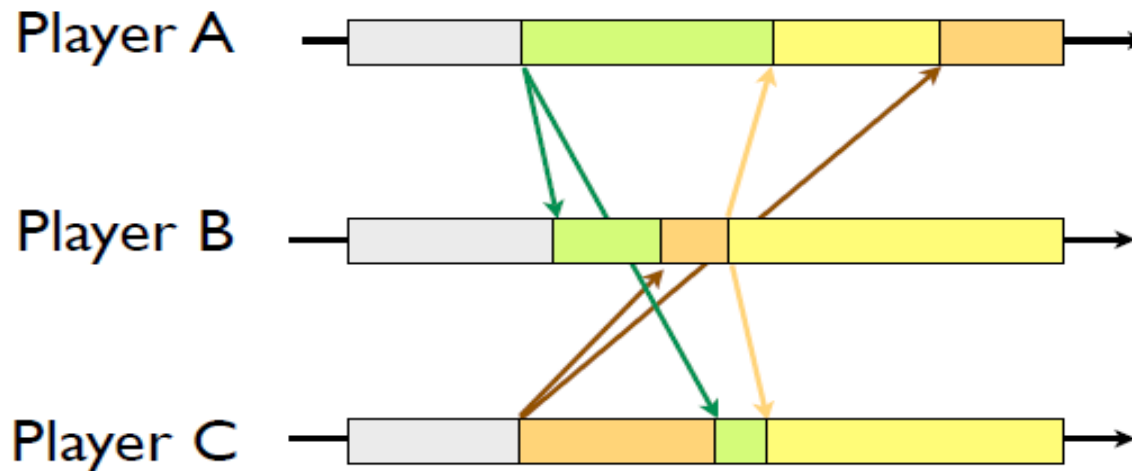
Tolerance based compass routing exploits a tolerance threshold T so that:

- if the difference between the angles considered by compass routing is smaller than the T , send the message
 - introduces a larger number of messages, but inconsistencies decrease
 - threshold = tolerance angle
-
- consider the local views of C
 - if the difference between the angles CAR and RAB is smaller than the threshold, send the message to A
 - B and C may send the same notification to A



P2P GAMING: CHALLENGES

- Without a central server, the game can easily get into inconsistent states
- Order of received messages may be wrong



Approaches:

Global clock synchronization (NTP protocol)

Bucket synchronization: make the game a turn-based game with very fast turns

CLOUD GAMING

- Games are stored and executed in real time on a distant server
- Game command and Audio/Video streams are exchanged very quickly between the client and the server allocated on the cloud
- The clients feel like they are playing locally
- Moving the processing task to the cloud turns almost any device which decodes Audio/Video Stream to a gaming device, such TV, tablets, PC or smart phones
- Clients can start a game at their home on their TV, pick it up on their Ipad if travelling, and finish it on their laptop when getting destination
- Model: very dumb (or thin) client very smart server

CLOUD GAMING: GaiKai

- A recent proposal: GaiKai (japanese for Open Ocean) <http://www.gaikai.com/>
 - Founded in november 2008 in the Netherlands
 - Deploys gaming platform together NVIDIA
 - Recently acquired by Sony, “Sony on the clouds”
- The goal:
 - From the GaiKai site: "when video games can be accessed as easily as movies and music, we believe they will become the #1 form of entertainment in the world"
 - to give game providers the possibility to take advantage of cloud technologies to deliver superior gaming experiences to their users
 - to deliver game experience on different platform, tablets, digital TV, smartphones, Facebook or entirely embedded into we sites

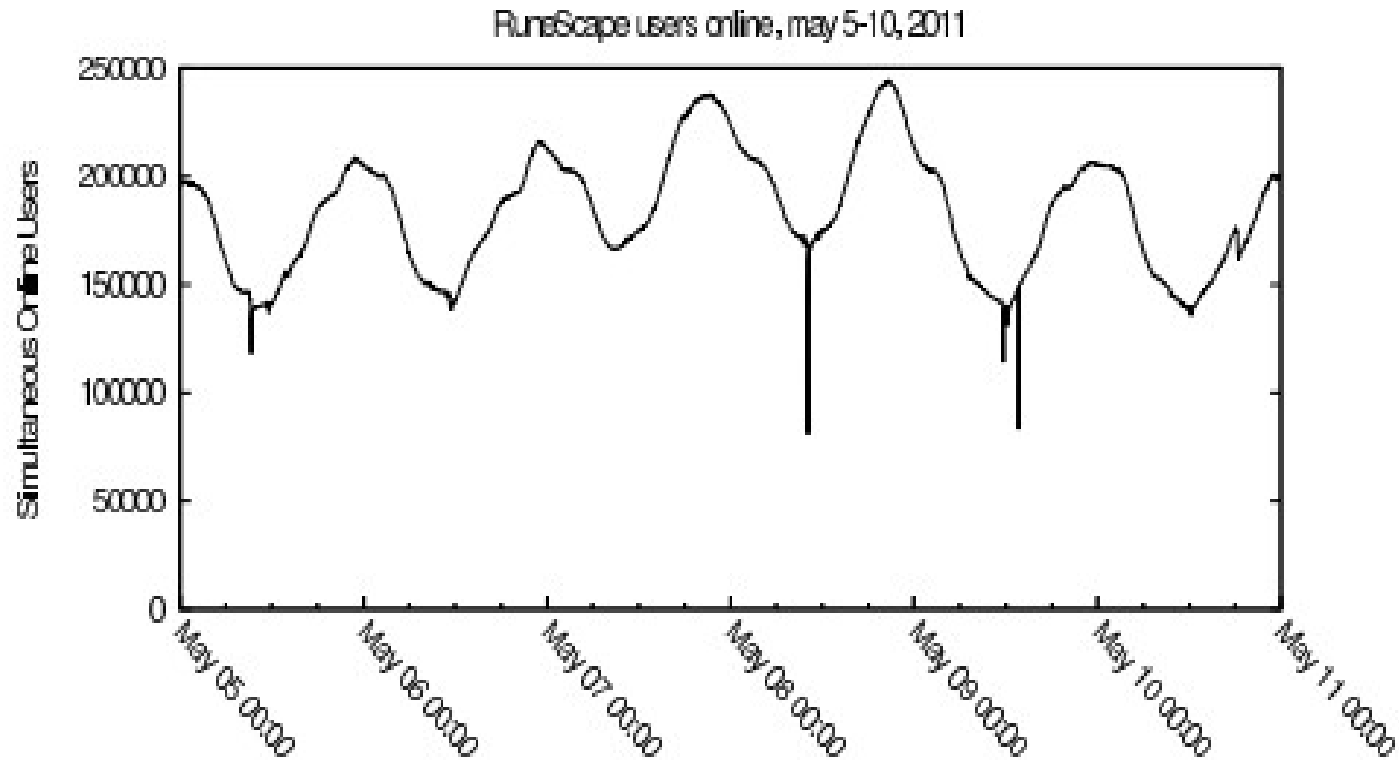
CLOUD GAMING: GaiKai

- Thin client model (smart server dumb client)
 - no download or installation for the client
 - a simple registration process gives instantaneous access to multi-player experience
 - game screen encoded on the server and decoded on the client
 - Platform as a Service as Cloud model
- **Latency reduction:** exploits nvidia technologies to encode a game frame in a single pass. Reduce latency to 10 ms, on tenth the duration of a human blink
- **Power Reduction** cost minimization
- **High Performance Video Encoding**

CLOUD BASED GAMING INFRASTRUCTURES

- MMOG exhibits high workload variability
- the response time depends on the number of online players
- over provisioning of resources to face peak load
 - worst case scenario is considered
 - enough resources to cope with this scenario are allocated
- largely suboptimal utilization of the hosting environment resources.
 - worst-case scenario rarely happens
 - allocated resources may remain unused at run time
- Infrastructure as a Service cloud model

CLOUD BASED GAMING INFRASTRUCTURES



Number of simultaneous Runescape players during the period may 5–may 10, 2011

D'Angelo, Ferretti, Marzolla Dynamic Resource Provisioning for Cloud-based Gaming Infrastructures

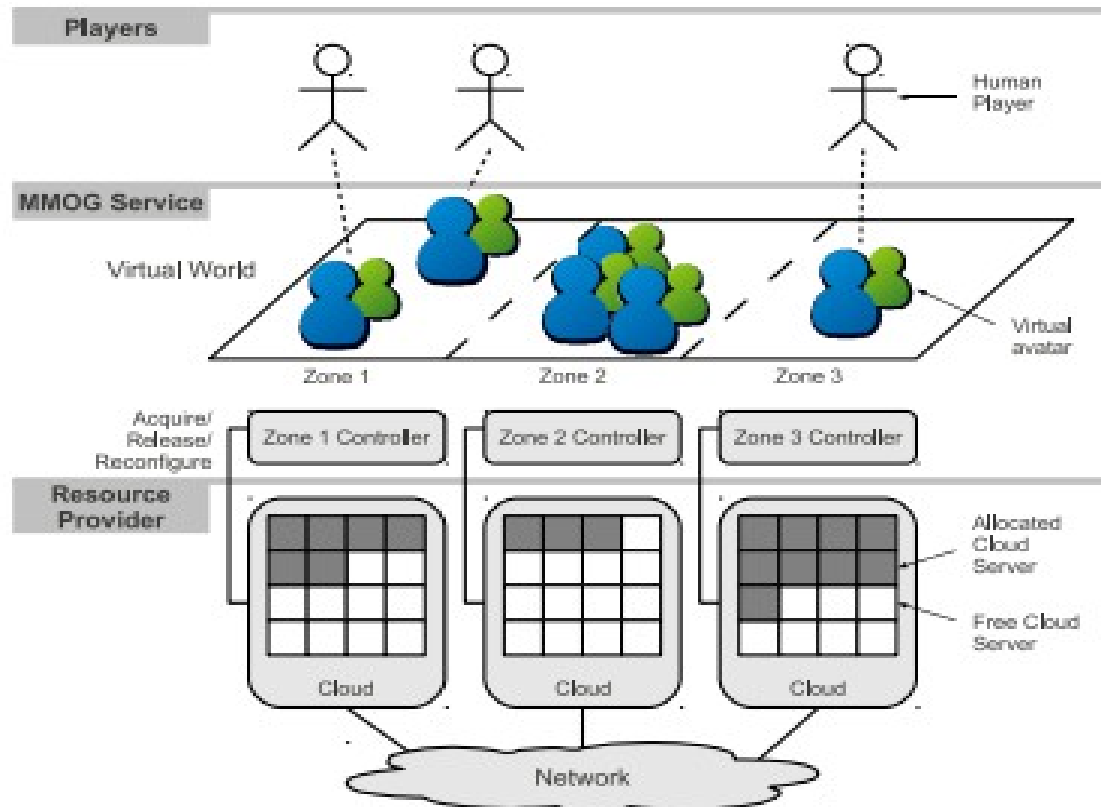
CLOUD BASED GAMING INFRASTRUCTURES

- Cloud infrastructures may solve the over-provisioning problem
 - they allow an **elastic provisioning** of computational resources
 - resource **pay-per rent model**
 - game operators may request a large set of resources during peak hours and release them when no more needed
- IaaS Infrastructure as a Service:
 - customers may run their host operating system and application on the top of the virtualization software
- Service Level Agreement guarantees
 - Quality of Service and the corresponding cost

CLOUD BASED GAMING INFRASTRUCTURES

- for MMOG, the Response Time of the application determines the QoS
- an average load of the game server executed over the VM is considered to define QoS
- a provisioning mechanism is required
 - to acquire a larger number of machines during peak loads
 - to release machines in order to cut down the service cost, when they are not needed
- dynamic provisioning of resources implies monitoring application performance, according different metrics
 - analytical load models
 - fast prediction algorithms

CLOUD BASED GAMING INFRASTRUCTURES

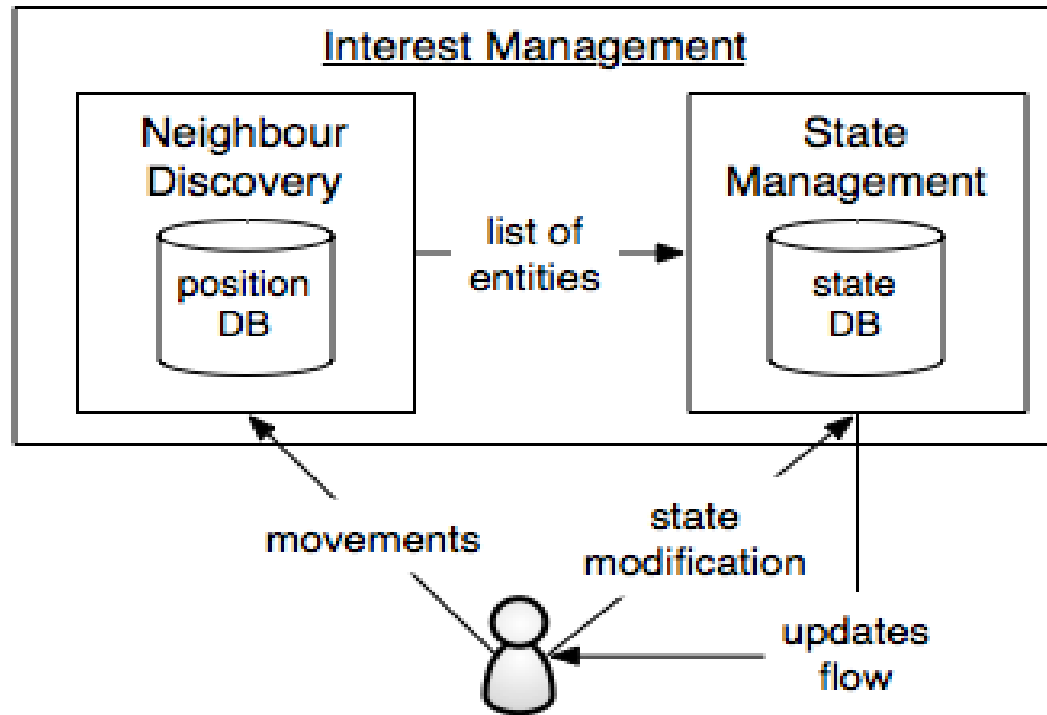


D'Angelo, Ferretti, Marzolla Dynamic Resource Provisioning for Cloud-based Gaming Infrastructures

CLOUD BASED GAMING INFRASTRUCTURES

- Zone controller periodically monitor the system
 - collect several system statistics
 - trigger the provisioner when the response time deviates from a given threshold
- Tools for system monitoring
 - queueing network performance models
 - prediction models
 - autoregressive
 - moving average
 - exponential smoothing
 - neural networks

A NOVEL APPROACH TO DISTRIBUTED INTEREST MANAGEMENT



Combination of Cloud and P2P to support DVEs in an inexpensive and QoS-aware fashion (developed in my research group)

- Neighbour Discovery
 - Epidemic fully decentralized network
 - Spatial DHT with cloud nodes
 - Idea: *Use “Wisdom of the crowd when possible”, Cloud otherwise*
- State Management
 - A regular DHT with Cloud and P2P nodes
 - Idea: *Let the cloud manage the peak load and the basic services*

Carlini, Coppola, Ricci,

Flexible Load Distribution for Hybrid Distributed Virtual Environments,

Future Generation Computing and Systems

August 2013

CONCLUSIONS

- Development of support for MMOG is an active research area
 - a widespread application
 - A complex applications integrating networking, AI, graphics
 - several challenges: state consistency and persistence, cheating prevention, P2P overlay, security
- Current architectures follow the client/server paradigm
 - a lot of research effort for P2P architectures
 - Overlay. Consistency in absence of a central coordinator
 - P2P techniques may also be exploited to define distributed servers
 - cloud elastic solutions ideal for MMOG resource provisioning