Simulazione 2009 – 2010

Lezione 7:

Introduzione all'Online gaming

Gabriele D'Angelo

<gda@cs.unibo.it>

http://www.cs.unibo.it/gdangelo/

Corso di Laurea in Informatica

Università di Bologna

Argomenti della lezione

- Networked and multiplayer games
- Minimi cenni storici
- Classificazione ed architettura
- Modello astratto per i networked games
- Principali requisiti e problematiche
- Architetture di comunicazione
- Bibliografia

Networked and Multiplayer Games

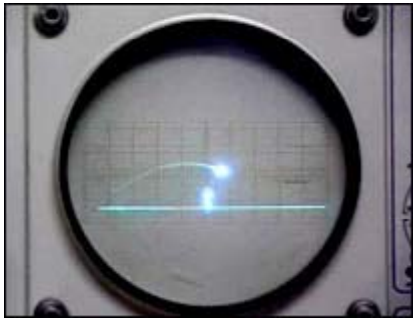
- In questa lezione e nelle successive tratteremo giochi che sono **networked** e **multiplayer**

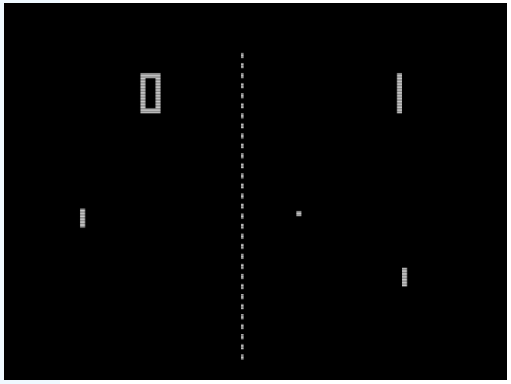
  Cosa significano esattamente questi due termini?

- **Networked** = è presente una rete di comunicazione che collega due o più computer
- **Multiplayer** = sono presenti più giocatori, ma non stanno necessariamente giocando in contemporanea (ad esempio i giocatori si alternano nel gioco usando lo stesso computer)
- Attenzione: molti giochi multiplayer non sono networked (es. giochi split-screen) e allo stesso modo, i giochi networked non sono necessariamente multiplayer

Giochi Multiplayer: minimi cenni storici

- Il "**tennis for two**" (1958), inventato da William A. Higinbotham viene considerato il primo esempio di gioco multiplayer
- Usando un oscilloscopio, veniva simulato in modo molto primitivo il gioco del tennis
- Visto l'uso di un oscilloscopio non può essere considerato il primo esempio di gioco multiplayer su computer
- Questo primato va invece a "**Spacewar!**", programmato su un PDP-1. Per i curiosi: http://spacewar.oversigma.com
- Molti, della mia generazione, hanno avuto come primo gioco il **Pong**: http://www.xnet.se/javaTest/jPong/jPong.html





Giochi Networked e Multiplayer: minimi cenni storici

- Nel 1993 la id Software produce "**DOOM**", un First-Person Shooter (FPS) di grandissimo successo
- Era possibile giocare fino a 4 giocatori, in modo cooperativo o competitivo (death-match)
- I computer dei giocatori, tutti nella stessa LAN, comunicavano attraverso il protocollo IPX e attraverso un topologia peer-to-peer
- Ogni giocatore risultava un peer indipendente che ogni 1/35 di secondo riceveva gli aggiornamenti di stato dagli altri giocatori e spediva i suoi aggiornamenti
- L'implementazione era basata su una serie di broadcast ethernet: pessima scelta progettuale! Perché?

Online gaming: classificazione ed architettura

- **Networked game**: un gioco che comprende più di un host che comunica attraverso una rete
- Ovviamente sono possibili tutta una serie di architetture, con i rispettivi vantaggi e svantaggi
- La natura della rete di comunicazione (es. LAN vs. Internet) ha un impatto fondamentale sulla scelta dell'architettura
- Nel nostro caso non siamo interessati ai **Turn-based Multiplayer games** (ad esempio il gioco degli scacchi) quanto piuttosto ai **Real-time Multiplayer games**
- In particolare ci riferiremo ai **Massively Multiplayer Online games (MMO),** ovvero in grado di supportare contemporaneamente centinaia di migliaia di giocatori

Online gaming: classificazione ed architettura

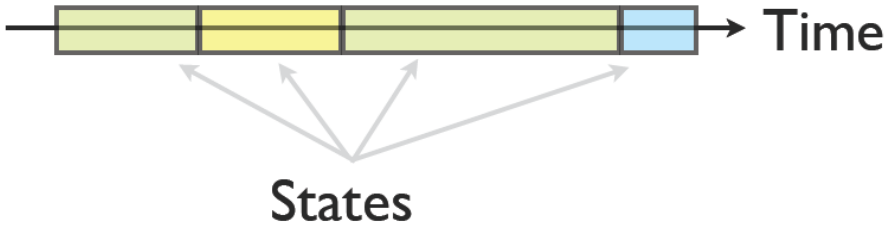
- Nell'ambito dei **Massively Multiplayer Online games (MMO)** abbiamo un'ulteriore vastissima specializzazione:
  - Massively Multiplayer Online Role Playing games (MMORPG)
    - ad esempio: World of Warcraft (Blizzard), attualmente il più diffuso con 10-11 milioni di abbonati, pari al 60% del mercato
  - Massively multiplayer online first-person shooter (MMOFPS)
  - Massively multiplayer online real-time strategy (MMORTS)
  - ...

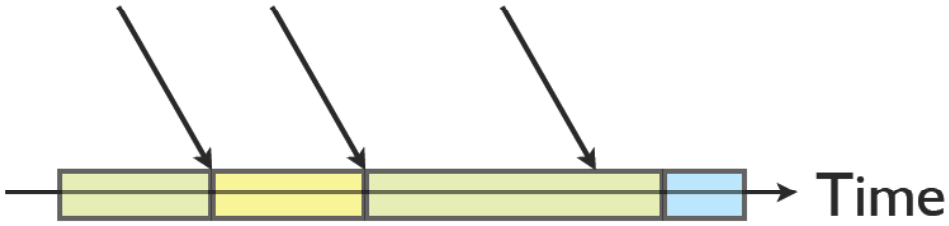
Non solo gaming

- Esistono anche altre applicazioni che condividono molte delle caratteristiche tipiche dei Massively Multiplayer Online games
  - **Military Training Simulators**
  - **Networked Virtual Environments**
    - ad esempio: Second Life (Linden), Google Lively
- Con le dovute peculiarità queste applicazioni condividono molte delle tecnologie utilizzate per l'implementazione di giochi su Internet e a loro volta molte delle problematiche tipiche della simulazione parallela e distribuita: ad esempio la *sincronizzazione* ed il *data distribution management*

Un modello astratto per i networked games

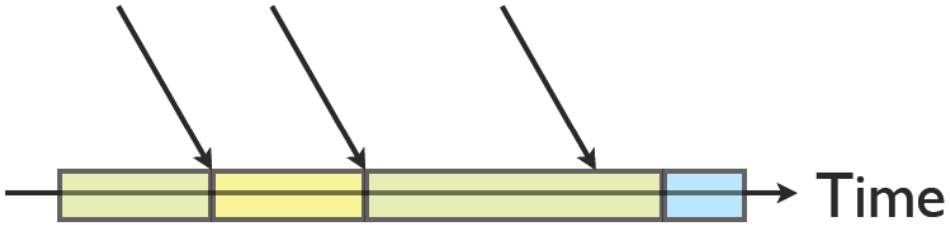
- Come abbiamo visto una simulazione distribuita, eseguita su un host, può essere vista come una **sequenza di stati** che cambiano nel tempo, in reazione ad **eventi**
- È possibile estendere questo modello anche all'ambito dei networked multiplayer games?

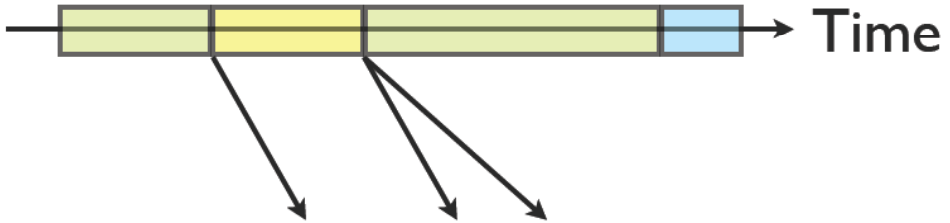




Un modello astratto per i networked games

- Gli eventi che producono un cambiamento di stato possono essere “interni” all’host. Ad esempio un input dovuto all’utente, un timer
- Oppure “esterni”, quindi dovuti all’interazione con altri host





Requisiti

- L'architettura di gioco dovrebbe soddisfare tutta una serie di requisiti, che sono fondamentali per garantire
  - **responsivity**: capacità di reagire (prontamente) alle azioni del giocatore
  - **consistency**: tutti i giocatori "coinvolti" hanno una *visione consistente dello stato di gioco*. Non significa che hanno esattamente la stessa visione (es. a causa della loro posizione) ma che queste sono in accordo tra di loro
  - **cheat proof**: barare può essere difficile, ma non necessariamente impossibile

Requisiti

- **fairness**: il gioco dovrebbe essere equo *nel trattare tutti i giocatori allo stesso modo*. Come è possibile in presenza di forti disparità hardware o di prestazioni della rete di comunicazione (es. modem vs. banda larga)?
- **resource efficient**: l'obiettivo è quello di minimizzare il quantitativo di banda utilizzata, il tempo CPU e la memoria, sia lato client che, a maggior ragione, server
- **scalability**: l'infrastruttura a supporto del gioco deve essere in grado di gestire un numero crescente di giocatori e quindi di scalare per dimensione, con l'aggiunta di nuove risorse

Requisiti

- **robust**: impedire i guasti è impossibile, quello che si può fare è creare architetture che siano in grado di reagire ai guasti minimizzando il loro impatto
- **simple**: la semplicità è un requisito fondamentale, più o complessa l'architettura più sarà complicato rispettare il time-to-market prefissato

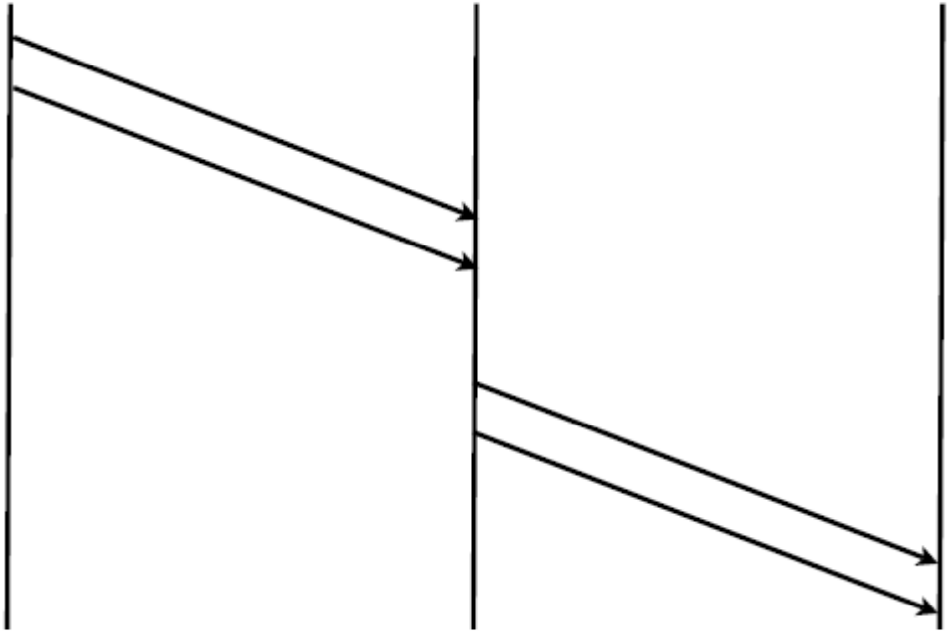
Alcuni problemi

- **rete**: Internet è una rete best-effort, i pacchetti arrivano in ritardo ed eventualmente non arrivano per niente
- **scalabilità**: creare architetture scalabili è difficile e costoso
- **guasti**: all'ordine del giorno
- **cheating**: tentare di barare è insito nella natura umana

Iniziamo questa nostra panoramica concentriamoci sulle problematiche relative alla rete, nel seguito vedremo alcuni ulteriori aspetti tipici delle altre problematiche

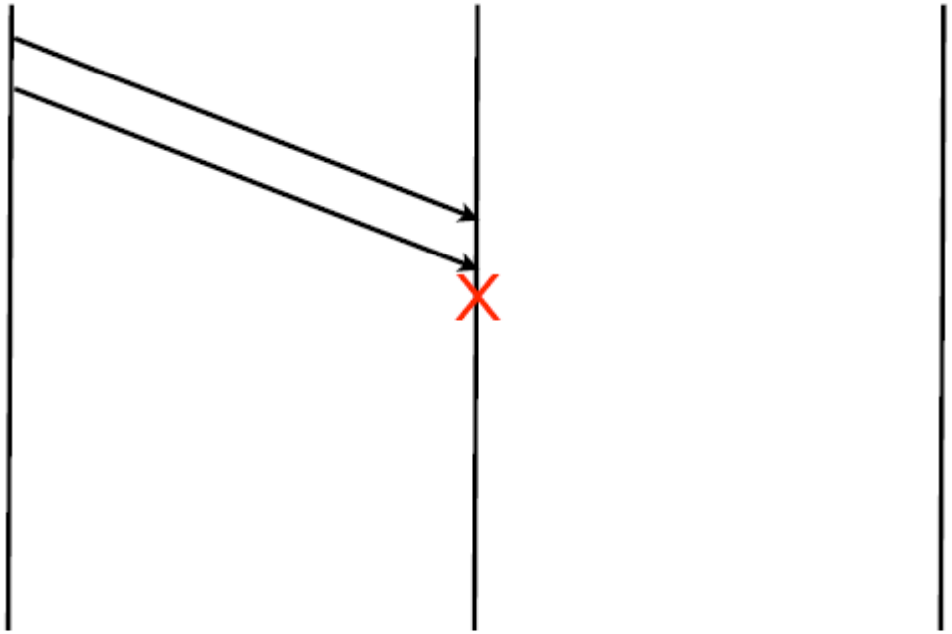
Alcuni problemi: rete

- I pacchetti che rappresentano gli eventi di gioco o gli aggiornamenti di stato impiegano un certo tempo per raggiungere il destinatario
- Questo quantitativo di tempo è **non predicibile** a priori



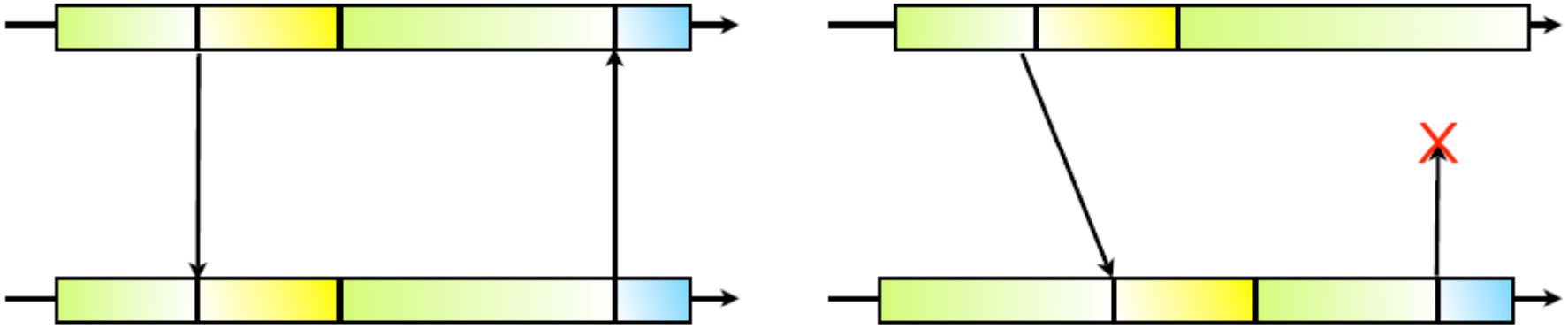
Alcuni problemi: rete

- I pacchetti vengono spesso **persi**, in **ordine diverso** rispetto alla spedizione o ancora peggio risultano **corrotti**
- Come faccio a capire se un pacchetto è corrotto? Si tratta di una modifica introdotta in volo o volontaria?



Alcuni problemi: rete

- In un mondo ideale, gli aggiornamenti di stato dovrebbero raggiungere **istantaneamente** tutti gli altri giocatori
- In questo modo tutti gli stati di gioco sarebbero sempre consistenti
- Ovviamente, un'assunzione di questo tipo è **totalmente irrealistica**
- Ritardi e perdite di pacchetti portano a **stati non consistenti**



Alcuni problemi: cheating

- I giocatori sono estremamente competitivi e cercano in ogni modo di avvantaggiarsi rispetto agli avversari
- In una tipica architettura di Internet gaming è possibile intervenire su tutta una serie di punti:
  - **hardware** del proprio computer
  - **software**
  - trasmissioni sulla rete di **comunicazione**
- Non sempre è facile distinguere il cheating da strumenti che invece sono del tutto legittimi

Alcuni problemi: cheating

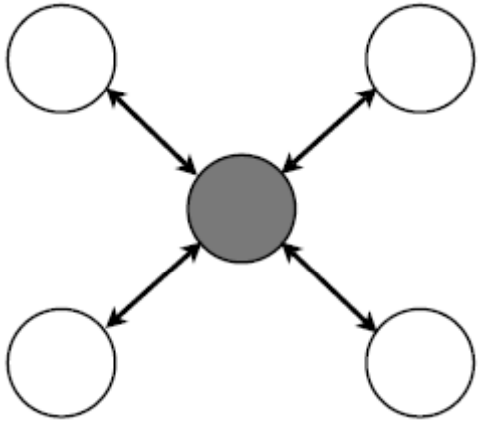
- L'uso di periferiche "avanzate" è un caso di cheating?
- Inoltre, esistono dei prodotti (es. connettività ADSL) appositamente pensati per giocatori. Ad esempio con tempi di latenza ridotti rispetto alla media. Anche questo è quindi un caso di "indebito vantaggio"?

Architetture di comunicazione

- Fino ad ora abbiamo visto alcune delle problematiche relative ai giochi su Internet e i principali requisiti che le varie implementazioni dovrebbero soddisfare
- Nell'ambito dei networked games ovviamente le **architetture di comunicazione** rivestono un ruolo fondamentale
- Nei prossimi ludici vedremo una serie di architetture tipiche di comunicazione e analizzeremo **vantaggi** e **svantaggi** di ciascuna di esse

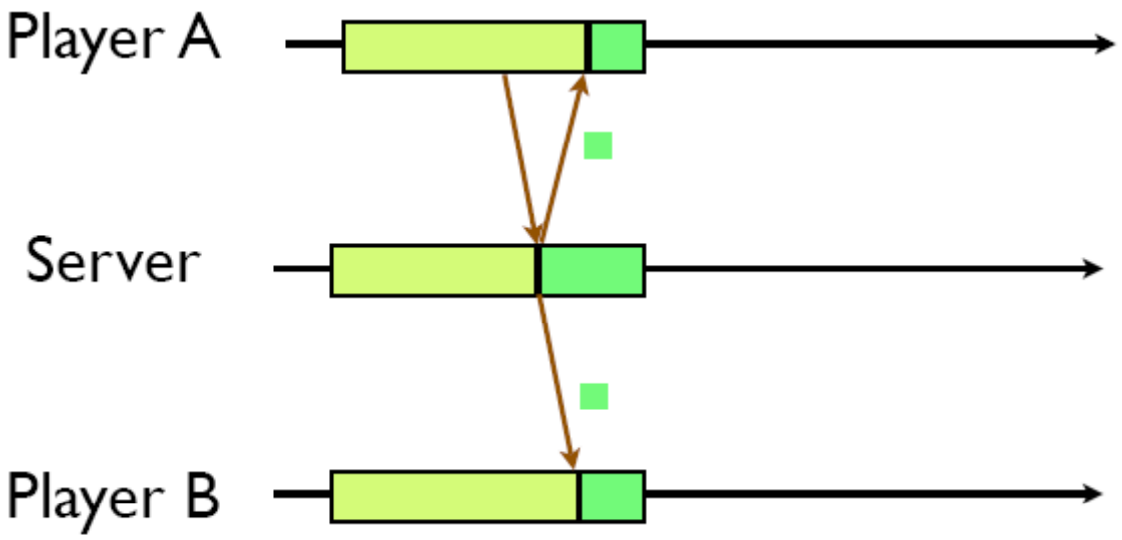
Architetture di comunicazione: centralizzata

- **Centralizzata**
  - tipica architettura client-server, dove è presente un unico server centralizzato. I giocatori sono rappresentati dai client
  - *vantaggi*: facilità dell'implementazione
  - *svantaggi*: prestazioni, scalabilità



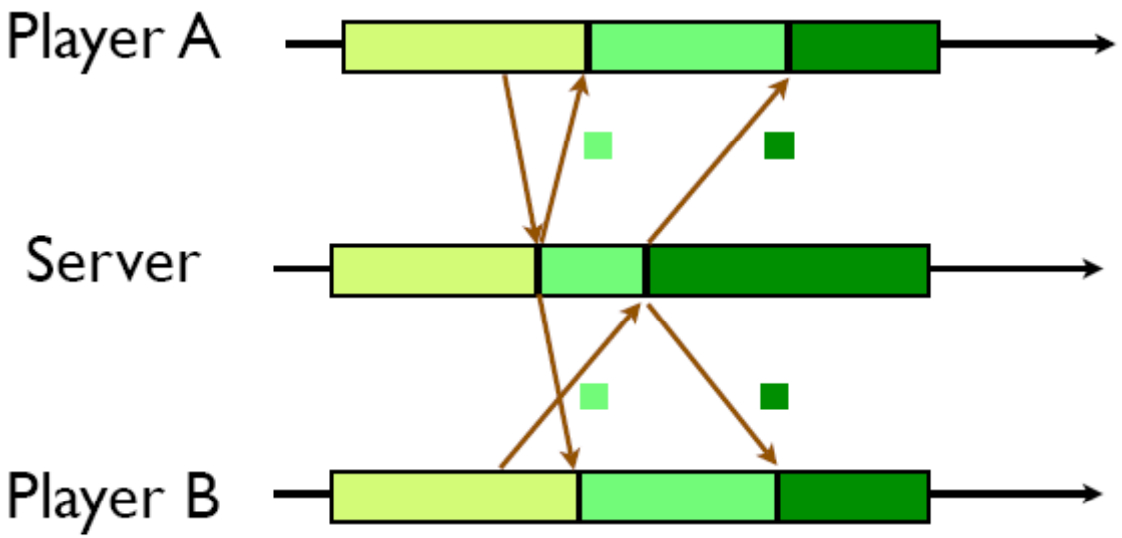
Architetture di comunicazione: centralizzata

- **Esempio di funzionamento:**
  - i client inviano i loro comandi al server centralizzato
  - il server calcola il nuovo stato e propaga a tutti i client lo stato
  - *il giocatore A, spedisce un comando al server*



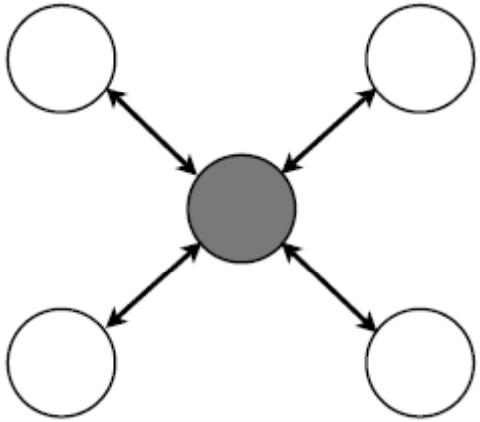
Architetture di comunicazione: centralizzata

- **Esempio di funzionamento (continua):**
  - i client inviano i loro comandi al server centralizzato
  - il server calcola il nuovo stato e propaga a tutti i client lo stato
  - *il giocatore B, spedisce un comando al server*



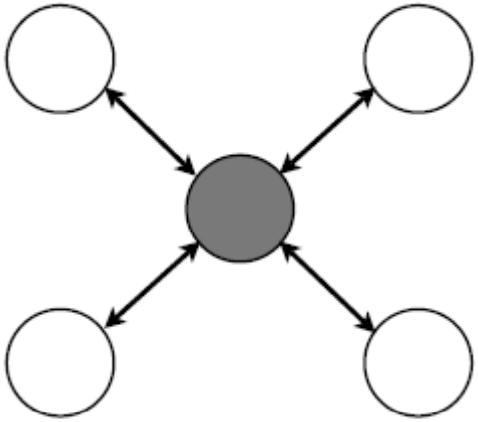
Architetture di comunicazione: centralizzata

- **Centralizzata**
  - è possibile implementare un'architettura di questo tipo in almeno due modi diversi:
    - **smart server, dumb clients**
    - **dumb server, smart clients**



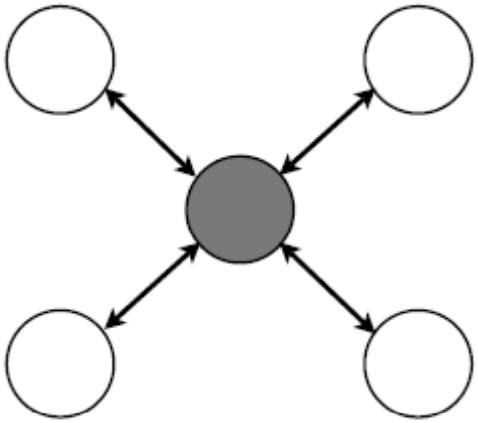
Architetture di comunicazione: centralizzata, smart server

- **smart server, dumb clients**
  - *ruolo del server:*
    - mantenimento dello stato
    - elaborazione del gioco
    - notifica dei client
    - risoluzione dei conflitti
  - *ruolo dei client:*
    - semplice visualizzatore
  - **vantaggi e svantaggi?**



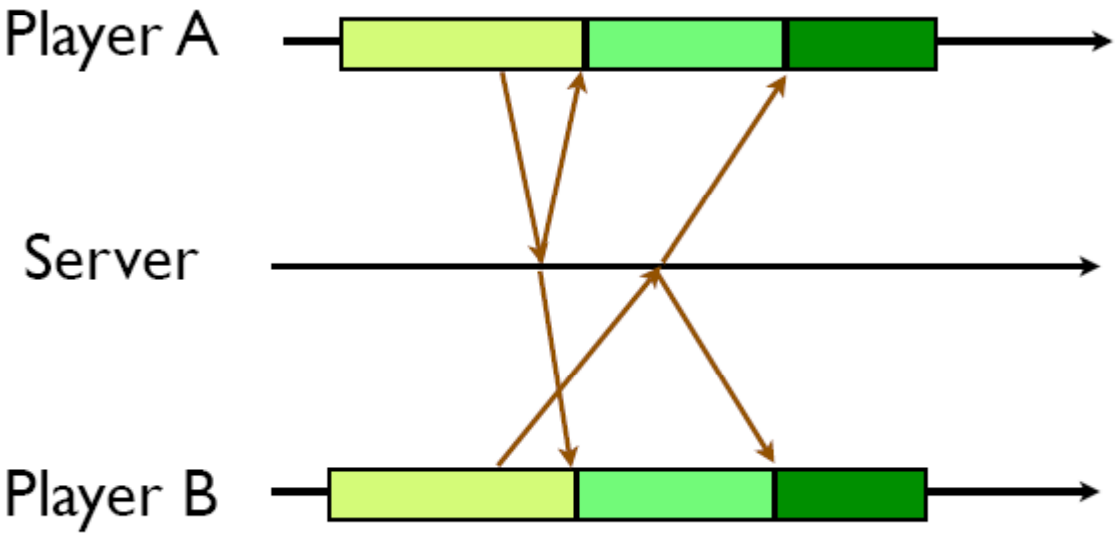
Architetture di comunicazione: centralizzata, smart clients

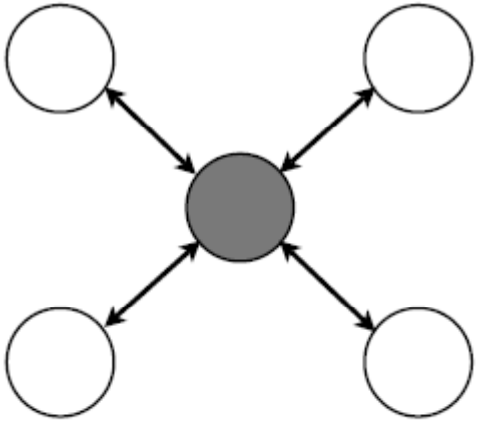
- **dumb server, smart clients**
  - *ruolo del server:*
    - notifica dei client
    - risoluzione dei conflitti
  - *ruolo dei client:*
    - mantenimento dello stato
    - elaborazione del gioco
  - **vantaggi e svantaggi?**



Architetture di comunicazione: centralizzata, smart clients

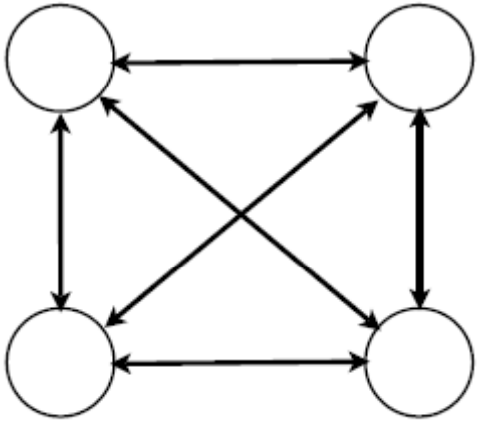
- **dumb server, smart clients**
  - secondo questo approccio, sono i client ed elaborare l'evoluzione del gioco e a determinare il nuovo stato
  - il server si limita a risolvere i conflitti e "girare" gli aggiornamenti ai client





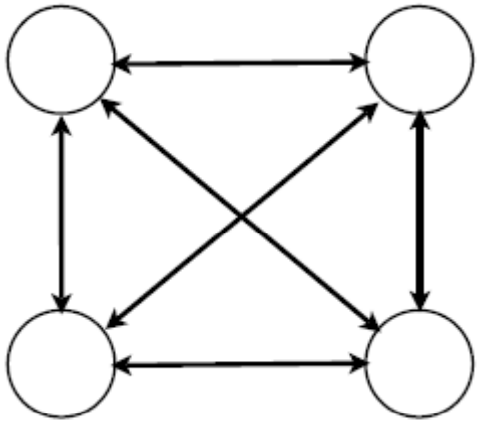
Architetture di comunicazione: peer-to-peer

- **Peer-to-peer**
  - in questo caso non esiste un punto di centralizzazione, i client devono necessariamente comunicare direttamente tra di loro per propagare i comandi ed i conseguenti aggiornamenti di stato



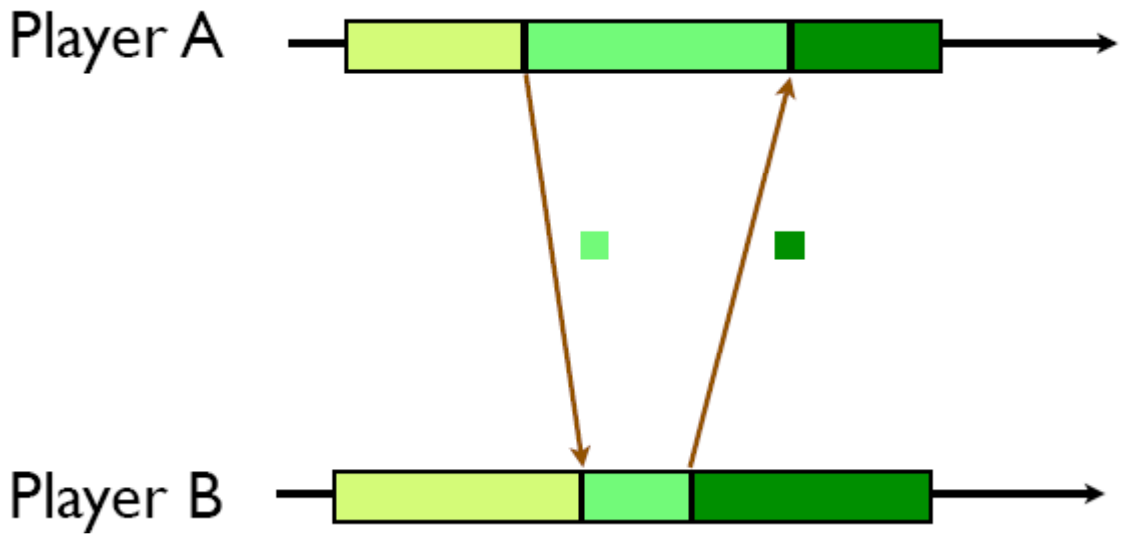
Architetture di comunicazione: peer-to-peer

- **Peer-to-peer**
  - *ruolo dei client:*
    - notifica dei client
    - risoluzione dei conflitti
    - mantenimento dello stato
    - elaborazione del gioco
  - **vantaggi e svantaggi?**
  - **chi ci salva dal cheating?**



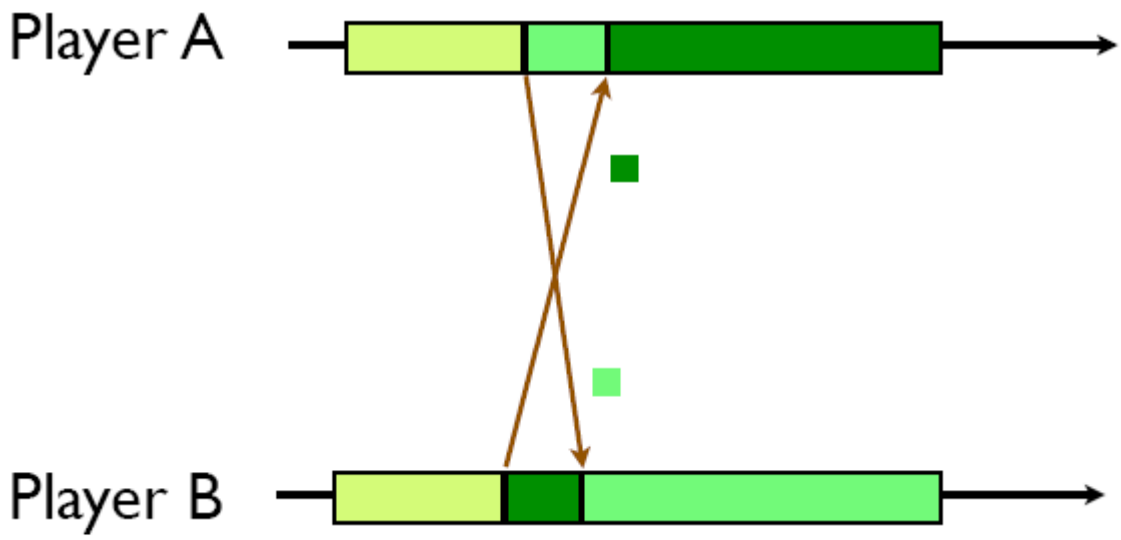
Architetture di comunicazione: peer-to-peer

- **Peer-to-peer**
  - ogni client deve gestire l'evoluzione del gioco, mantenere lo stato e propagare gli aggiornamenti agli altri client



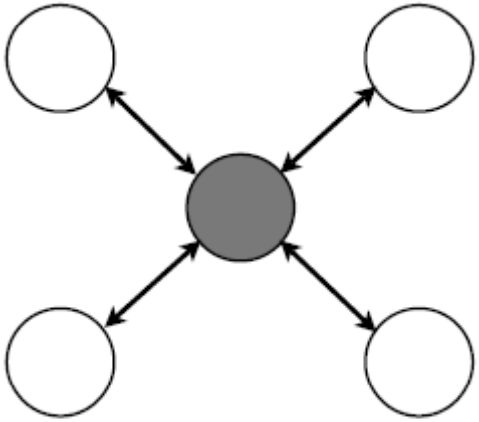
Architetture di comunicazione: peer-to-peer

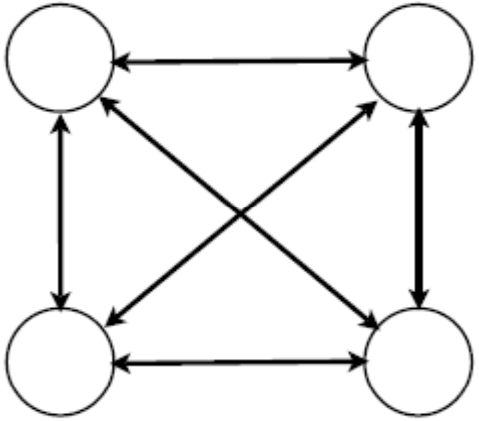
- **Peer-to-peer**
  - ogni client deve gestire l'evoluzione del gioco, mantenere lo stato e propagare gli aggiornamenti agli altri client
  - **problema: come si garantisce la consistenza?**



Architetture di comunicazione: centralizzata vs. peer-to-peer

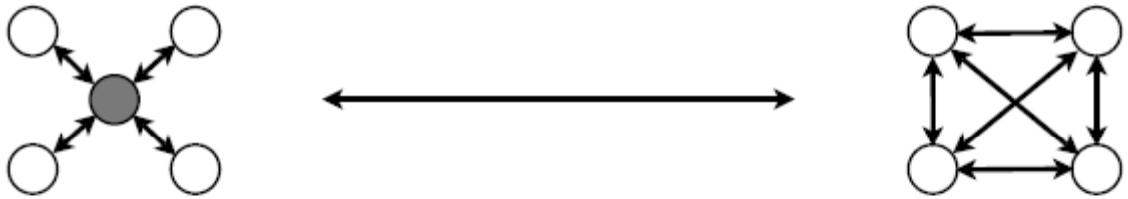
- **Architettura centralizzata vs. peer-to-peer**
  - nella pratica, l'architettura centralizzata **non offre** una **scalabilità sufficiente** per l'implementazione di giochi massivamente popolati
  - l'architettura peer-to-peer ha un potenziale di scalabilità molto elevato ma, come abbiamo visto, introduce altri problemi di difficile risoluzione





Architetture di comunicazione: ibrida

- **Architettura ibrida**
  - quello che cerchiamo è un'architettura che eviti il singolo punto di centralizzazione e che allo stesso tempo eviti un approccio totalmente decentralizzato, con i problemi che questo porta

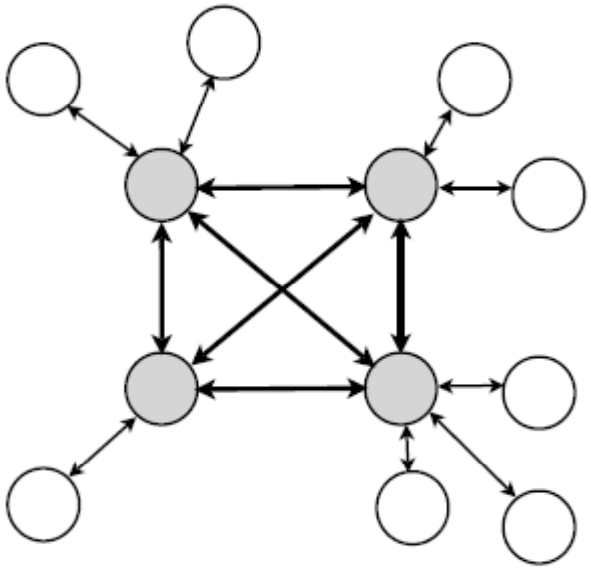


single centralized server

completely decentralized

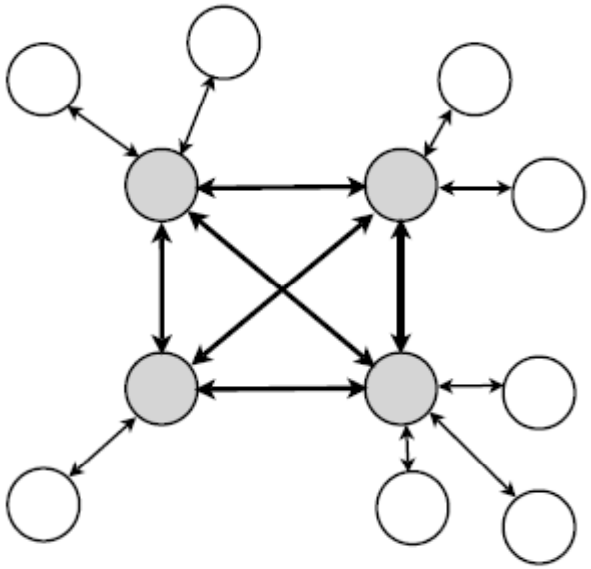
Architetture di comunicazione: mirrored servers architecture

- **Mirrored servers architecture**
  - in un'architettura di questo tipo abbiamo un **insieme di server** invece di un singolo server
  - i **server** comunicano tra di loro secondo uno schema **peer-to-peer**
  - **ogni server replica totalmente lo stato del gioco**
  - ogni **client** risulta collegato ad un **solo server**, scelto seguendo un "apposito" algoritmo di selezione



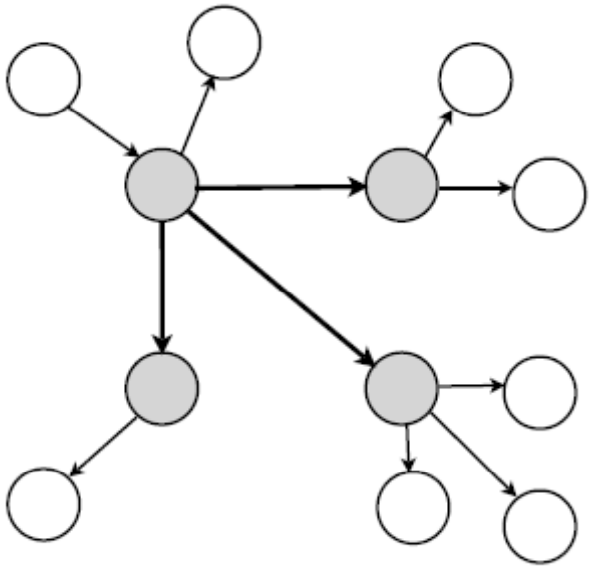
Architetture di comunicazione: mirrored servers architecture

- **Mirrored servers architecture**
  - comunemente si assume che i server siano collegati tra loro per mezzo di una rete a bassa latenza ed alta velocità, con un buon livello di affidabilità
  - i client sono collegati ai server attraverso le usuali reti di telecomunicazione (es. PSTN, ADSL, fibra ottica ...)



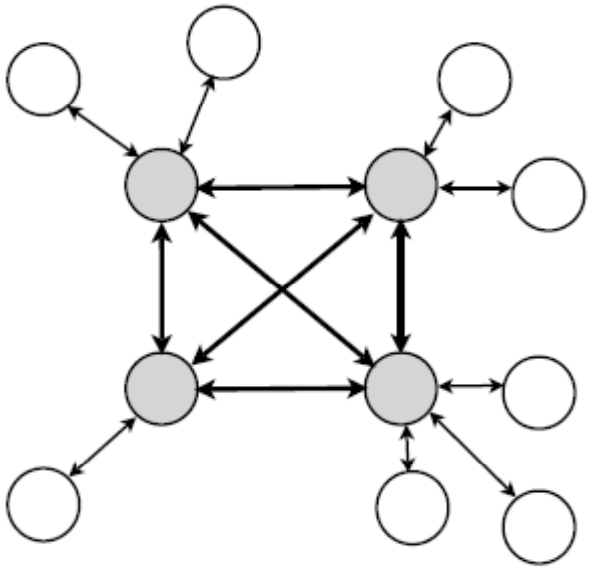
Architetture di comunicazione: mirrored servers architecture

- **Mirrored servers architecture**
  - in questa architettura, ogni server raccoglie i comandi generati dai giocatori (client) **di cui si fa carico**
  - il server elabora gli aggiornamenti di stato
  - ultimata la fase precedente, propaga a tutti gli altri server gli eventuali aggiornamenti di loro interesse, in modo che questi possano farli **pervenire ai loro client**
  - tutto questo deve ovviamente avvenire in maniera consistente e sincronizzata



Architetture di comunicazione: mirrored servers architecture

- **Mirrored servers architecture**
  - la sincronizzazione ed in generale il coordinamento tra i server è relativamente semplice in quanto:
    - i server sono abitualmente considerati come "**trusted**"
    - la rete è a **bassa latenza**
    - l'infrastruttura può essere ad alta affidabilità (**high availability**)
    - in alcuni casi è possibile usare **IP-multicast**



Architetture di comunicazione: mirrored servers architecture

- **Vantaggi**
  - **affidabilità**: non è più presente un single-point of failure, nel caso di guasto/malfunzionamento di un server, i client possono scegliere un altro dei server replicati (ma non è detto che questa riallocazione sia trasparente)
  - **scalabilità**: rispetto all'approccio centralizzato a singolo server, questa architettura può allocare un numero maggiore di giocatori contemporanei. Ogni server accetta un numero di giocatori che dipende da vari fattori, tra i quali la disponibilità di banda e tempo di calcolo

Architetture di comunicazione: mirrored servers architecture

- **Vantaggi**
  - **latenza**: rispetto al caso peer-to-peer, è possibile progettare ed implementare degli schemi che minimizzino la latenza di comunicazione tra il client ed il server di riferimento. *Normalmente questa scelta come viene effettuata?*
- **Svantaggi**
  - **latenza**: se consideriamo la latenza rispetto agli altri giocatori, la presenza di un'operazione aggiuntiva di forwarding (nei server) aumenta, anche in modo significativo, il tempo necessario per la propagazione dei comandi

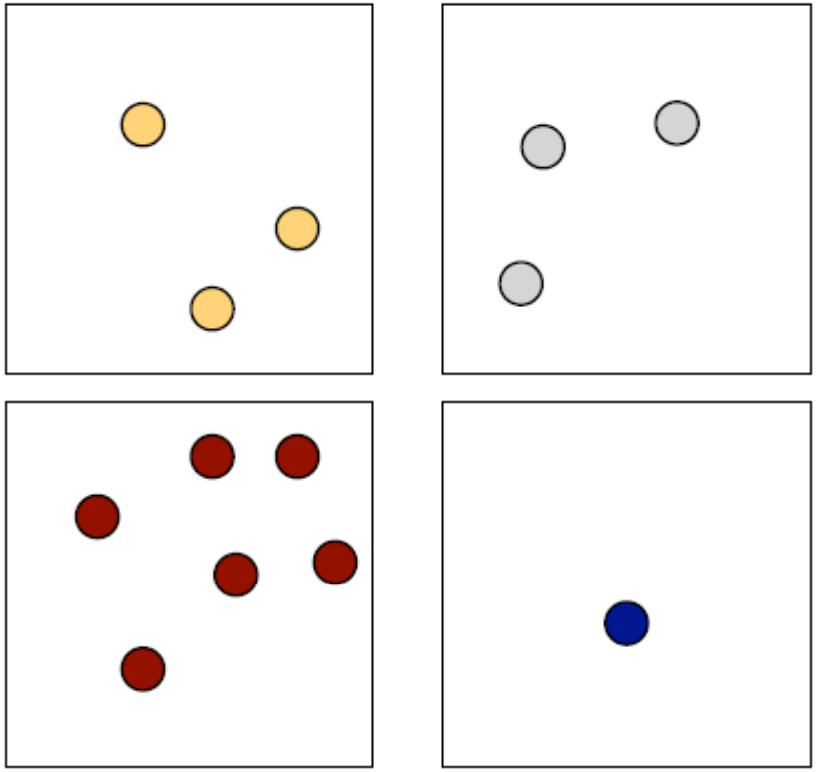
Architetture di comunicazione: mirrored servers architecture

- **Svantaggi**
  - **scalabilità**: se ogni server mantiene localmente una copia integrale dello stato di tutto il mondo simulato, la *scalabilità del sistema è limitata dalle risorse di ogni singolo server*. Ed è necessario anche considerare il costo in termini di comunicazione necessario per mantenere tutti i server coordinati, può aumentare in maniera non lineare all'aumentare del numero dei server coinvolti.

    Necessità di minimizzare lo stato gestito localmente così come la propagazione delle informazioni che non sono necessarie

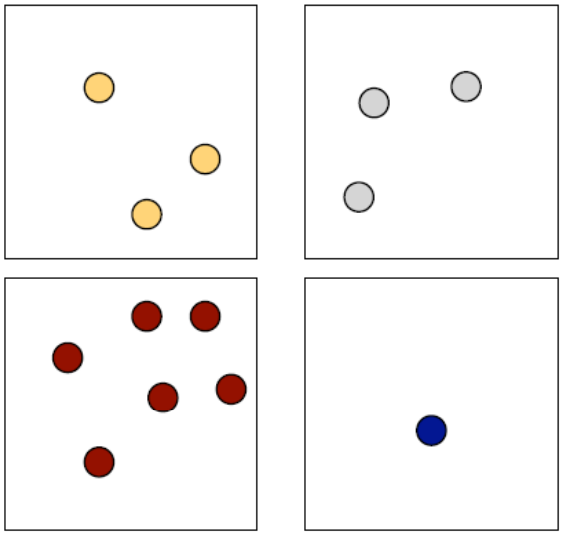
Architetture di comunicazione: zoned servers architecture

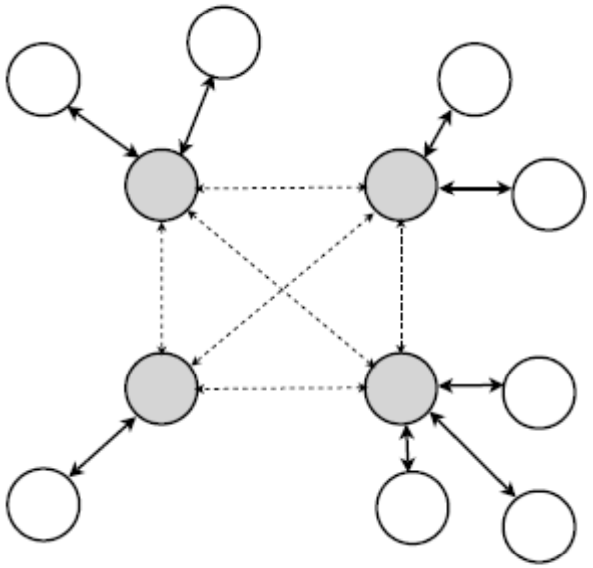
- **Zoned servers architecture**
  - una delle prime soluzioni proposte divide l'area di gioco in una serie di **zone distinte** che sono **totalmente indipendenti** tra di loro
  - in questo modo, ogni server può occuparsi di **una sola delle zone** di gioco e quindi si prende carico dei **soli giocatori che si trovano in quella zona**



Architetture di comunicazione: zoned servers architecture

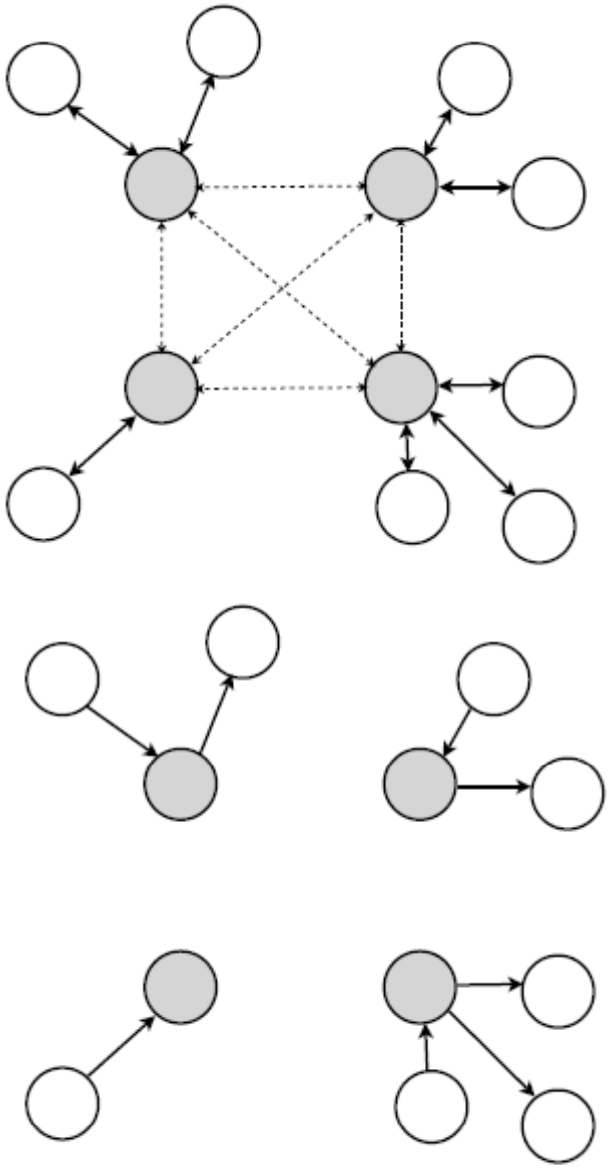
- **Zoned servers architecture**
  - i giocatori ovviamente hanno la necessità di muoversi liberamente nell'area di gioco
  - quando il giocatore si muove al di fuori della zona gestita dal server attuale, ha la necessità di cambiare il server di collegamento, individuando il nuovo server di riferimento ed effettuato quindi un vero e proprio "**hand-off**"





Architetture di comunicazione: zoned servers architecture

- **Zoned servers architecture**
  - in questo modo si riducono le comunicazioni tra i server, ogni server è responsabile solo di una parte dello stato
  - il compito del server di zona è quindi quello di raccogliere i comandi dei suoi giocatori e di “girare” questi messaggi ai soli giocatori nella sua area che risultano interessati

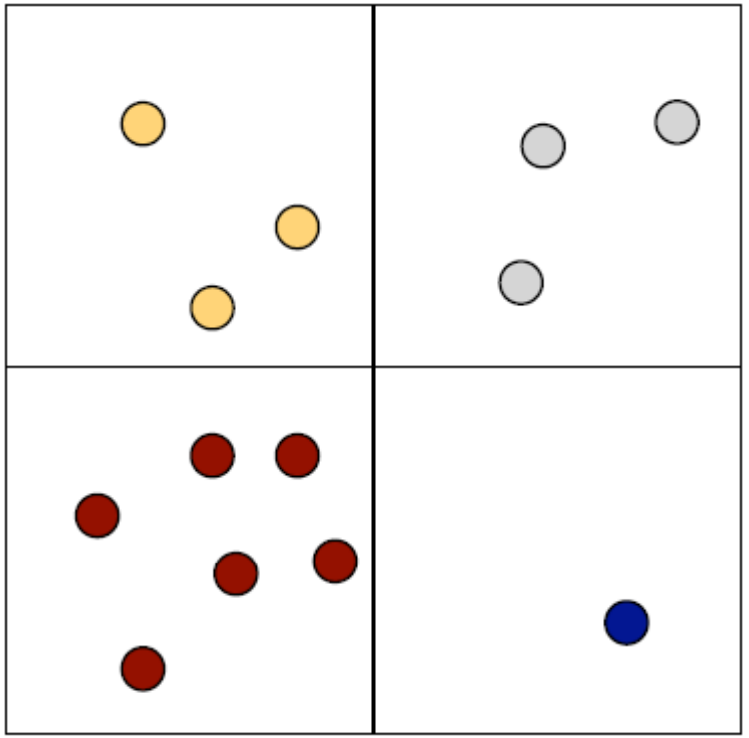


Architetture di comunicazione: zoned servers architecture

- **Zoned servers architecture**
  - **vantaggi:**
    - grazie alla suddivisione dello stato, ogni server può gestire un numero maggiore di giocatori, *incrementando la scalabilità dell'architettura*
  - **svantaggi**:
    - nell'ambito di ogni zona abbiamo un single point of failure. Eventualmente è possibile prevedere server di replicazioni interni alla stessa zona
    - il design del gioco è **pesantemente limitato** dal vincolo che le aree sono totalmente indipendenti tra di loro (es. è come se si trattasse di più giochi distinti tra di loro)

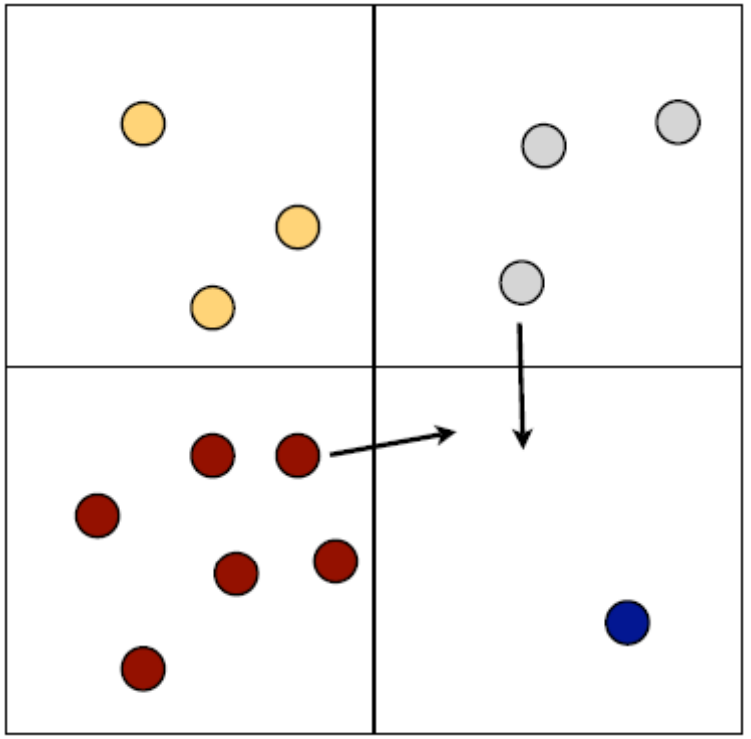
Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - quello che vogliamo è invece dividere l'area di gioco in una serie di regioni confinanti tra di loro
  - ogni server è responsabile della gestione di una o più regioni
  - in pratica un'**unica area di gioco viene suddivisa in regioni**



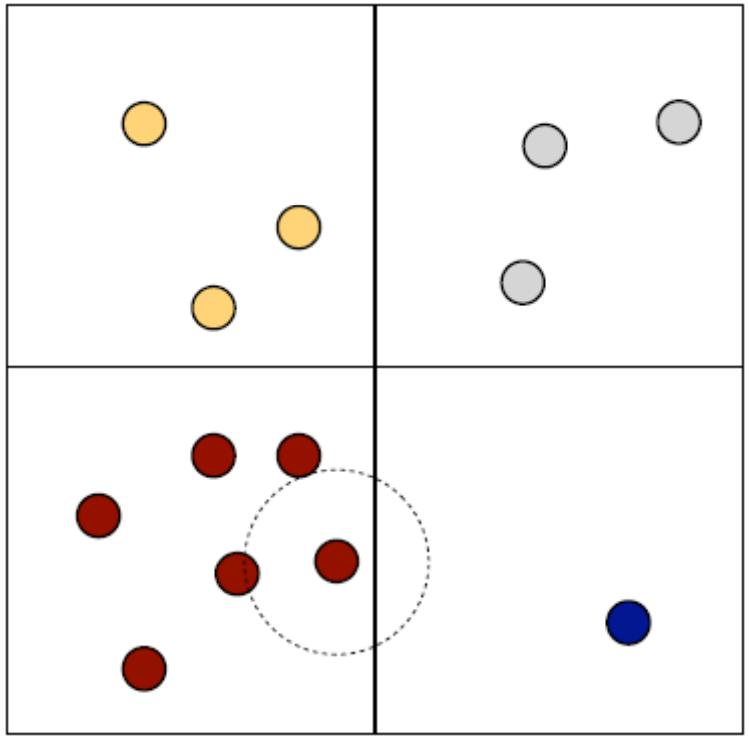
Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - in questo modo, il movimento di un “**avatar**” provoca il **passaggio da una regione all’altra**
  - il “mondo simulato” è uno unico ed il passaggio, almeno teoricamente è del tutto “trasparente”



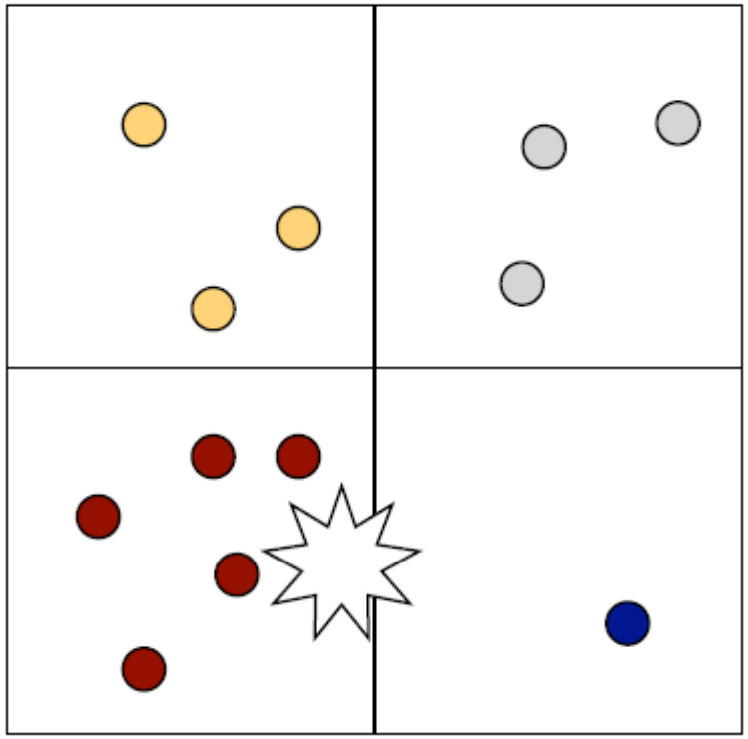
Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - i giocatori sono in grado di “vedere” cosa avviene attorno a loro, anche se si trovano ai margini della regione
  - è compito dei server gestire il trasferimento di informazioni necessario per implementare questa visione di gioco



Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - è comune che eventi che avvengono in una regione, interessino anche una o più regioni vicine (ad esempio una deflagrazione)
  - il server di riferimento della regione avrà l'obbligo di propagare tutte le informazioni necessarie agli altri server coinvolti, che a loro volta le gireranno ai loro client interessati



Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - **vantaggi**:
    - il **paradigma di gioco risulta molto più naturale** e soprattutto non risulta vincolato da scelte che sono prettamente tecniche
    - almeno teoricamente, il **partizionamento in regioni è del tutto trasparente ai giocatori**, che non si accorgono di nulla quando passano da una regione all'altra. Oppure se gli eventi che ricevono sono "locali" (originati nella stessa regione in cui si trovano) o "remoti" (originati nelle regioni vicine)

Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - **svantaggi**:
    - per implementare questo meccanismo è necessario prevedere la **migrazione dei giocatori**. Quando un giocatore cambia la sua regione di appartenenza deve essere attivato un meccanismo di “hand-off” dal server della vecchia region a quello della nuova region di destinazione
    - il meccanismo di migrazione deve prevedere il trasferimento dello stato del giocatore, così come gli eventuali aggiornamenti ed i messaggi eventualmente “in volo”. Il tutto in modo trasparente

Architetture di comunicazione: region servers architecture

- **Region servers architecture**
  - **svantaggi**:
    - cosa succede in presenza di **densità non uniformi** dei giocatori all'interno dell'area di gioco?
    - in che modo **ripartisco** l'area di gioco in regioni, e come **assegno** le regioni ai vari server?
    - in questa ripartizione, quali sono le **funzioni di costo** che devo tenere sotto controllo ed eventualmente minimizzare?

Bibliografia

- G. Armitage, M. Claypool, P. Branch. *Networking and Online Games. Understanding and Engineering Multiplayer Internet games.* Wiley, 2006.
- O. W. Tsang. CS4344: *Networked and Online Games*..
- J. Smed, H. Hakonen. *Algorithms and Networking for Computer Games.* Wiley, 2006.

Simulazione 2009 – 2010

Lezione 7:

Introduzione all'Online gaming

Gabriele D'Angelo

<gda@cs.unibo.it>

http://www.cs.unibo.it/gdangelo/

Corso di Laurea in Informatica

Università di Bologna