Distributed Systems

Peer-to-Peer Online Gaming

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
https://teaching.dahahm.de

16.07.2024

What is an Online Game?

Game Data

Which type of data needs to be shared among participants of an online game?

Game Data

- Immutable game data (play world, rules. . .)
- Object states
- Actions
- User input
- . . .

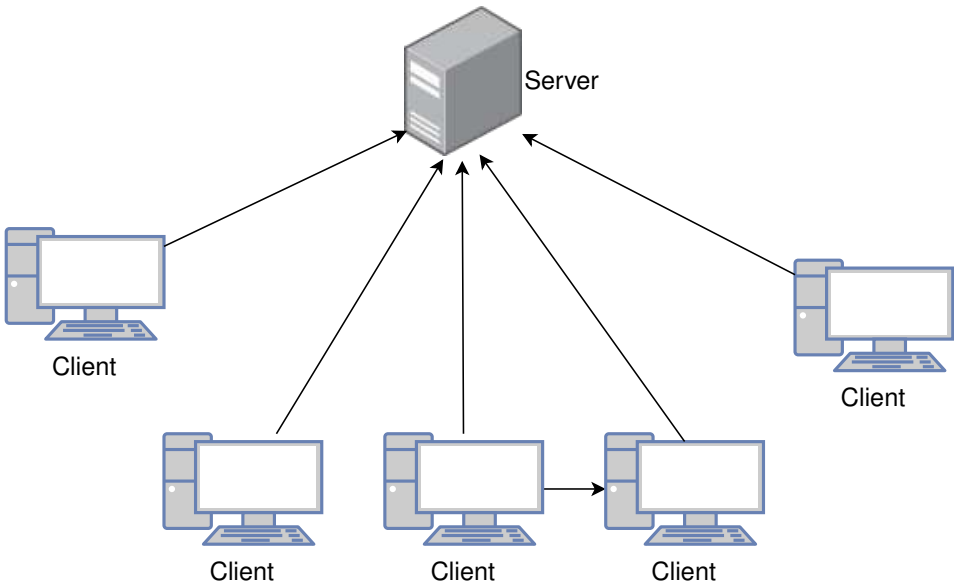
Agenda

- Architectures for Distributed Gaming
- Update Strategies

Agenda

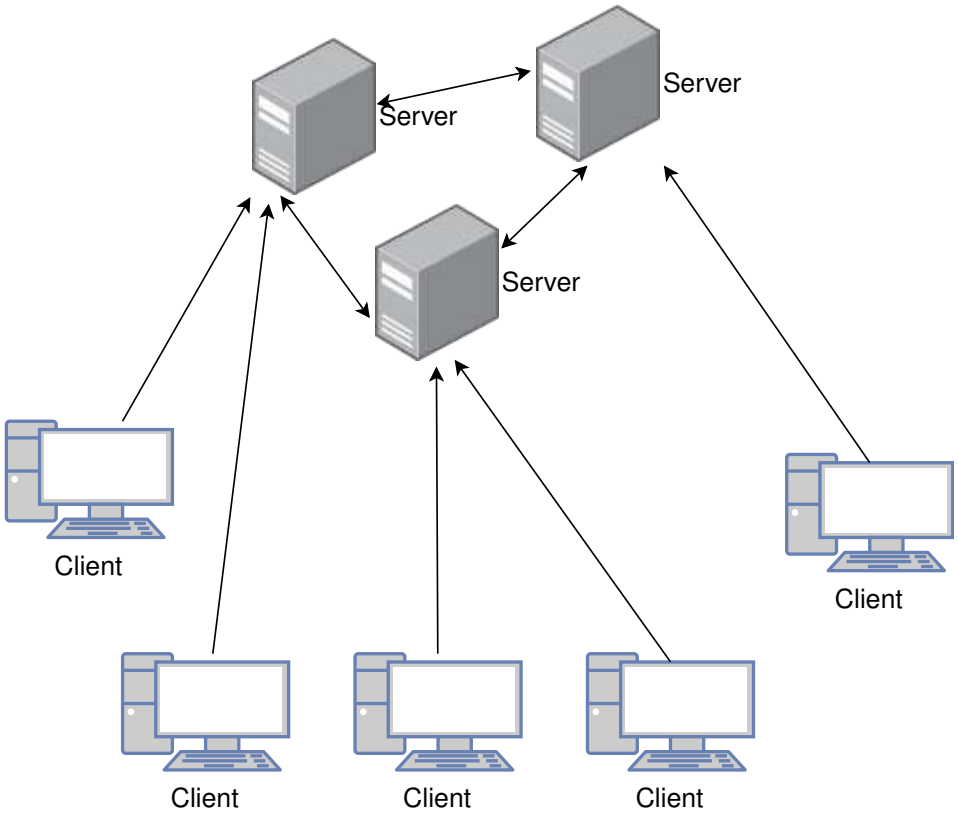
- Architectures for Distributed Gaming
- Update Strategies

Client-Server



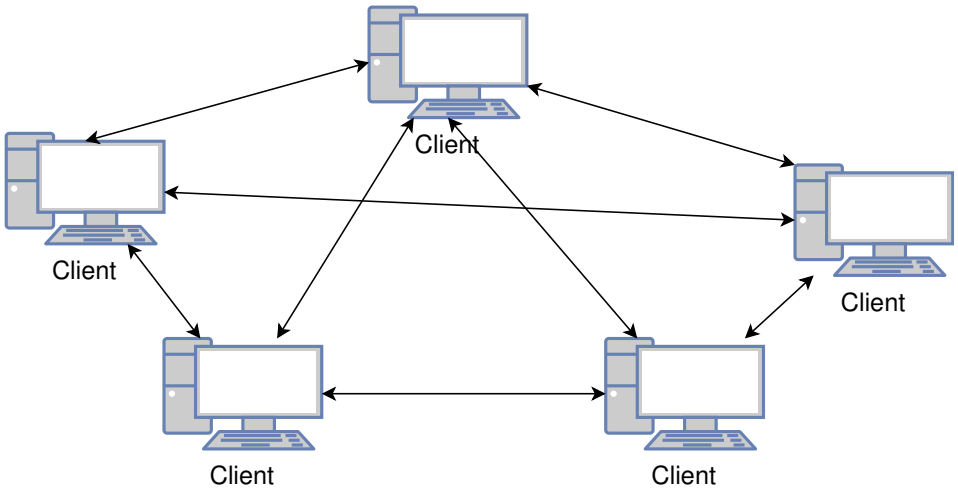
- Game is hosted in a data center
- Different software for server and client
- Centralized solution
- Subcomponents:
  - account-management
  - partitioning of the game world
  - monitoring
  - persistence

Multi-Server



- Several servers
- Redundant data storage
- Reduced distance between client and server $\Rightarrow$ reduced latency
- Dynamic solutions:
  - replication
  - proxy-Server

Peer-to-Peer



- No explicit servers
- Data exchange between adjacent peers
- Every peer is hosting part of the game world
- Dynamic partition of the game world

Thin vs. Fat Client

Thin Client

- Only the server holds and modifies the game state
- Part of the state is propagated to the clients upon connection
- Server sends updates of game state changes to clients
- Client transmits action requests to the server
- Actions are handled in their order of arrival and results are propagated to all affected clients

Fat Client

- Clients managing their own objects ($\rightarrow$ no one else can modify them)
- Server manages chronological sequence with time stamps and transmits changes to the other clients
- Local game states may vary, due to latency
- Chronological sequence may be inconsistent (local changes take effect before global ones in contradiction to their chronological order)

What are the advantages and drawBacks for a Thin Client Solution?

Thin Client Solution

- **Advantages**
  - Central management of game state
    - $\Rightarrow$ Consistent game state
    - $\Rightarrow$ No conflicts
    - $\Rightarrow$ Persistent system to store the state
  - Easier cheating protection
- **Drawbacks**
  - High server load
  - Potential high latency
  - Client processing power not utilized

Design Choices

- How many participants can play the game?
- What kind of participants can take part?
- What needs to be exchanged?
- Which data can be accessed by whom? Which permissions do the participants have?
- What are the timing requirements?

Protocol Information

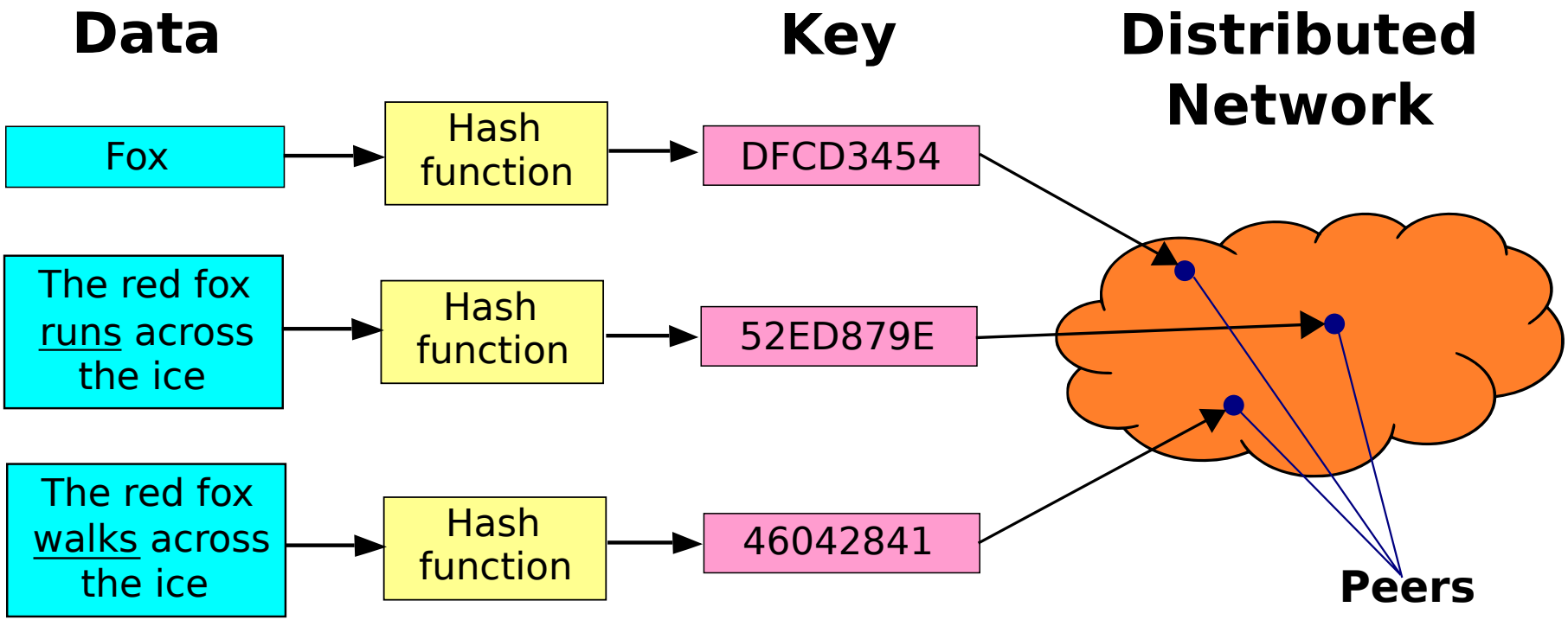
- **Object attributes**: (Action Result Protocol)
  - protocol sets the current parameter value of a game entity
    (`set hit points for player "hax0r" t to 96`)
  - protocol sends relative changes
    (`reduce hit points for player "hax0r" by 100`)
- **Actions**: (Action Request Protocol)
  - Contains only player input without direct impact on game state
  - Protocol only transfers user input $\Rightarrow$ results must be calculated on the server
    (`try to hit player "hax0r" with "axe"`)

Agenda

- Architectures for Distributed Gaming
- Update Strategies

How can we distribute data in a P2P network?

Distributed Hash Tables

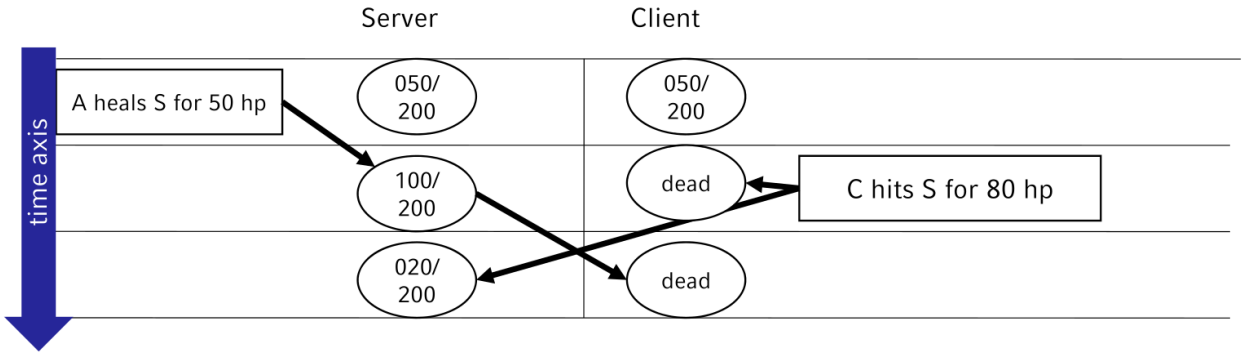


Overlay Networks

- A logical network built on top of a physical network
  - Overlay links are realized by connections of the underlying network
- Multiple overlay networks may coexist at once
  - On the same layer or on top of each other
  - Providing particular services
- Nodes are often end hosts of the underlying network
  - Acting as intermediate nodes that forward traffic
  - Providing a service, such as access to files
- Who controls the nodes providing service?
  - The party providing the service
  - Distributed collection of end users

Conflicts during decentralized computing

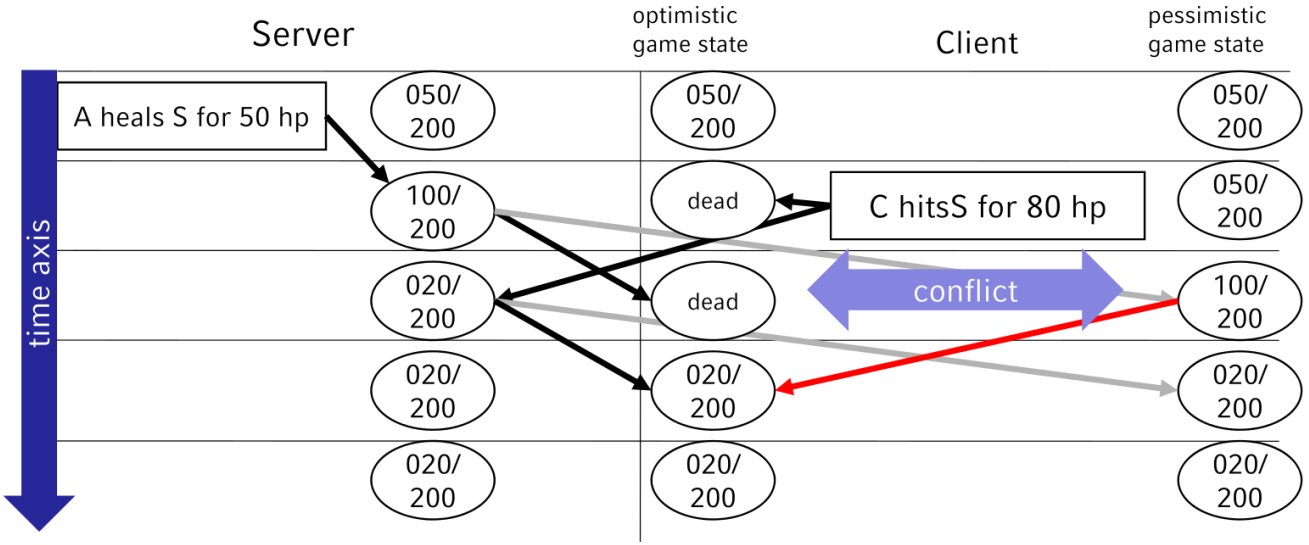
- Local changes need time to be transmitted to the network
- Actions are calculated for and executed on local game states
  ⇒ changes that predate the action may not be taken into account
- Simple solutions:
  - Client is not allowed to change local data without server acknowledgment
  - Using object protocols the server may send an update of the current game entity state.



Solution Approach

Reset local actions

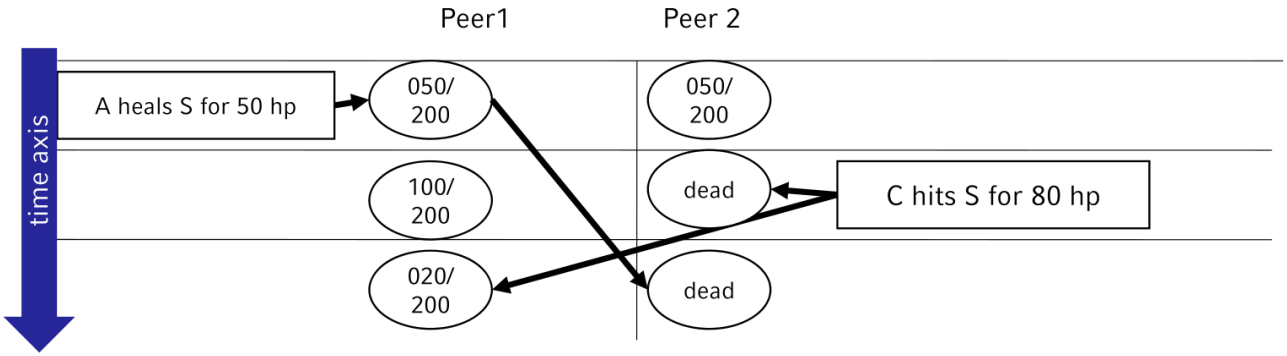
- Client has 2 game states:
  - optimistic GS (contains local changes)
  - pessimistic GS (contains actions transmitted by the server)
- On mismatch: Resetting the optimistic GS to the pessimistic GS



Local time

up until now: One server handles processing sequence

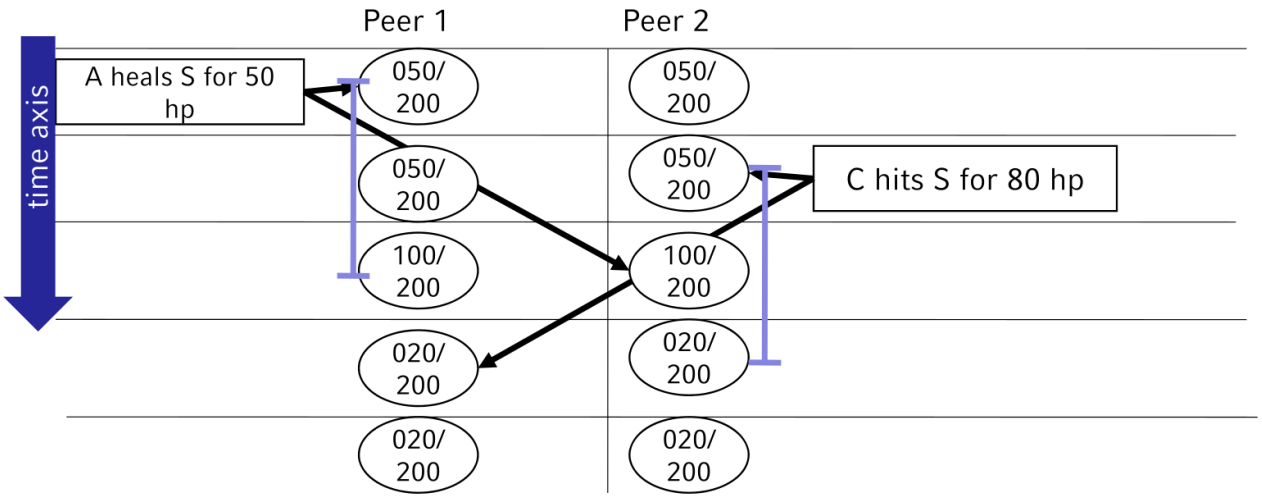
- Impossible for P2P games and multi server architecture
  - ⇒ sequence is inconclusive after arrival at server
  - ⇒ organization by local time stamps on creation
- During processing both, own and foreign changes may appear in incorrect sequence
- In case of inconsistencies game entities can be synchronized



Solutions by Local Lag Mechanism

Problem is causes by the lack of knowledge about previous actions

- **Solution:** Lag-Mechanism
  - Processing updates is delayed to allow for other actions to arrive in time
  - If this time frame is exceeded, conflict detection and reset become necessary



Server Side vs. Client Side Processing

Server side processing

- Properties
  - content accuracy is important
  - response time less important
  - chronological order is important
- Used for...
  - damage and healing
  - item pick up

Client side processing

- Properties
  - response time is crucial
  - synchronization and sequence are less important
- Used for...
  - position and movement data
  - animations and other display effects

References

- Matthias Schubert, Managing and Mining Multiplayer Online Games, LMU München

  http://www.dbs.ifi.lmu.de/cms/VO_Managing_Massive_Multiplayer_Online_Games
- Jennifer Rexford, Computer Networks, Princeton

  http://www.cs.princeton.edu/courses/archive/spring07/cos461/
- Distributed Hash Table, Wikipedia
  https://commons.wikimedia.org/wiki/File:DHT_en.svg