



# pSense

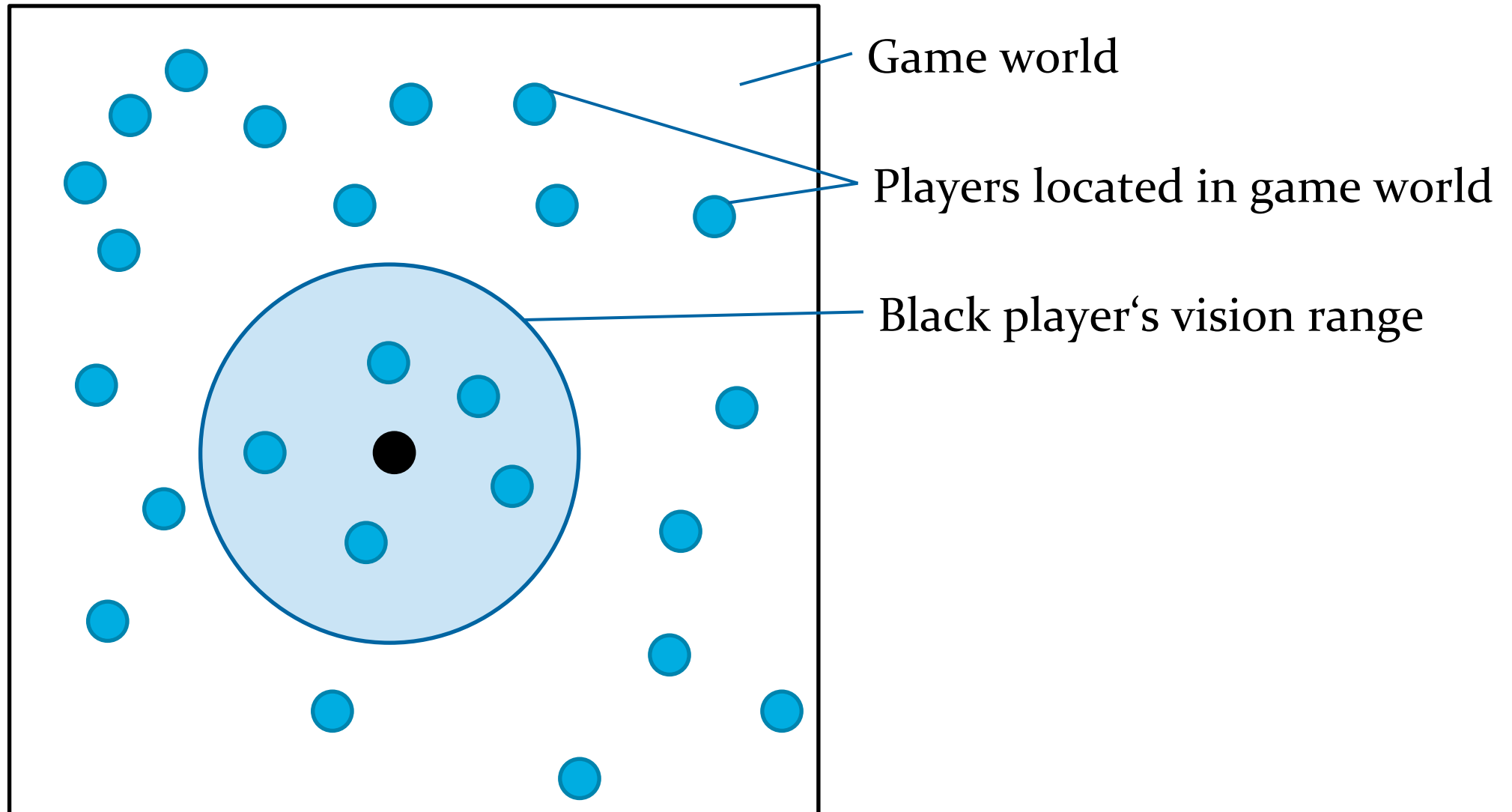
Maintaining a dynamic localized peer-to-peer structure  
for position based multicast in games

- Arne Schmieg<sub>(1)</sub>, Michael Stieler <sub>(1)</sub>, Sebastian Jeckel <sub>(1)</sub>,
- Patric Kabus <sub>(1)</sub>, Bettina Kemme <sub>(2)</sub>, Alejandro Buchmann <sub>(1)</sub>

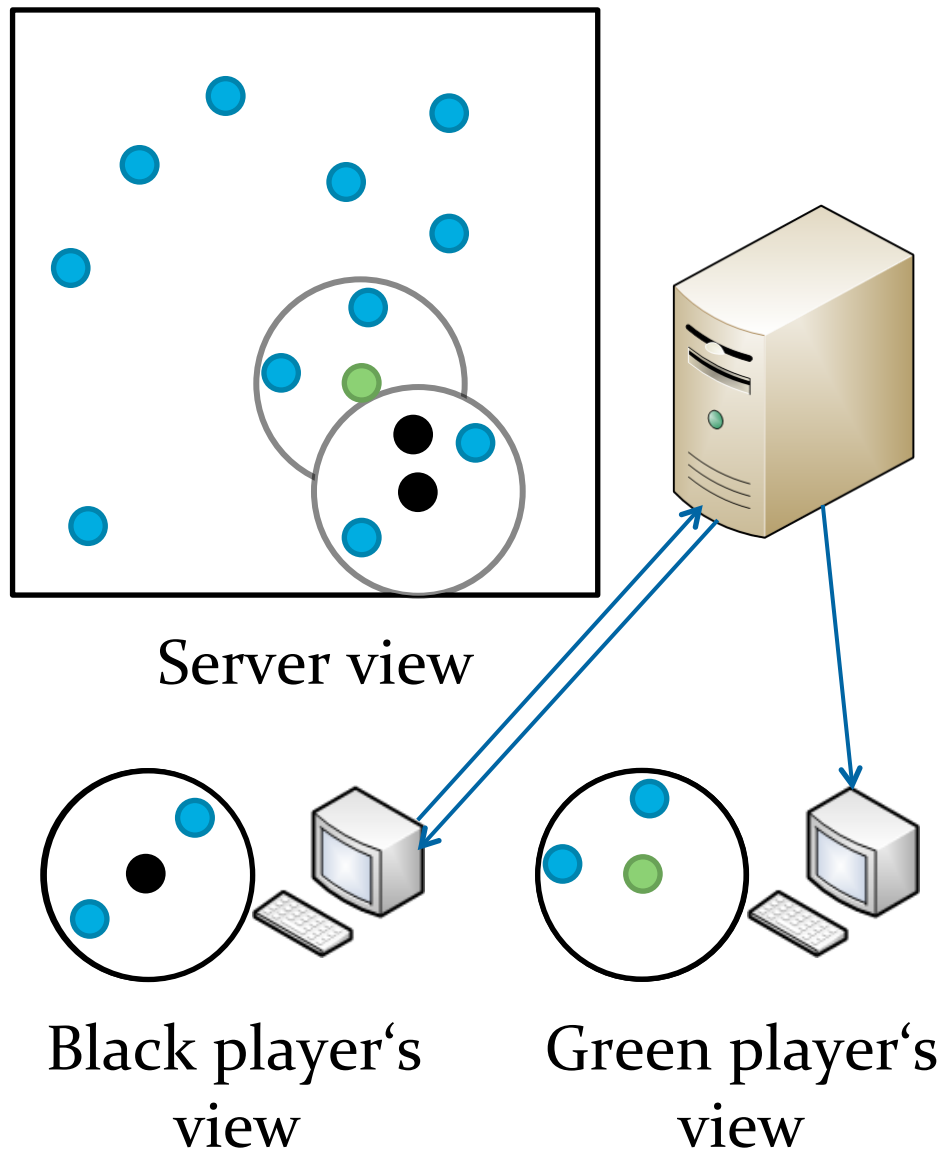
(1) Technische Universität Darmstadt

(2) Mc Gill University

# The Scenario of Online Multiplayer Games



# Traditional Gaming Client/Server Architecture



- Server maintains all information
- Clients → Players
- Players only see players in vision range
- Players send movements and other actions to server
- Server forwards actions to players in vision range
  - Interest management
- Delay: always 2 hops
- Problem: Every update is forwarded to every player in vision range by the server!

# Pros and Cons of the Client/Server Architecture



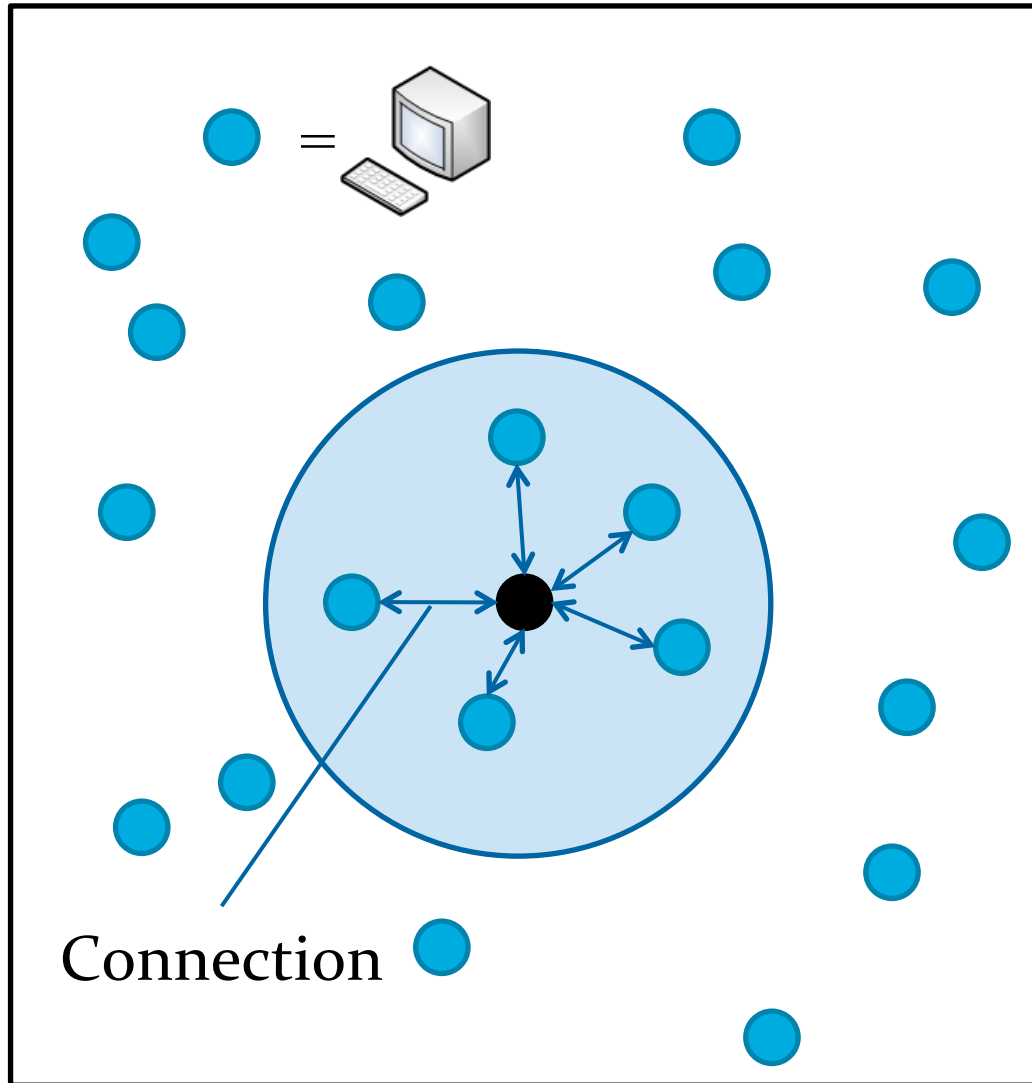
## Pro

- Simple
- Server has global knowledge:  
interest management easy

## Contra

- Calculation of interest  
management expensive
  - Many connections to server
  - High bandwidth  
requirements at server
  - Many messages from and to  
server
- Possible bottleneck and single  
point of failure

# Our Approach: P2P Architecture



## Principles

- Players are peers
- Players are connected to some other players
- Each player sends its actions directly to a subset of other players
- Actions reach interested players directly(1-hop)

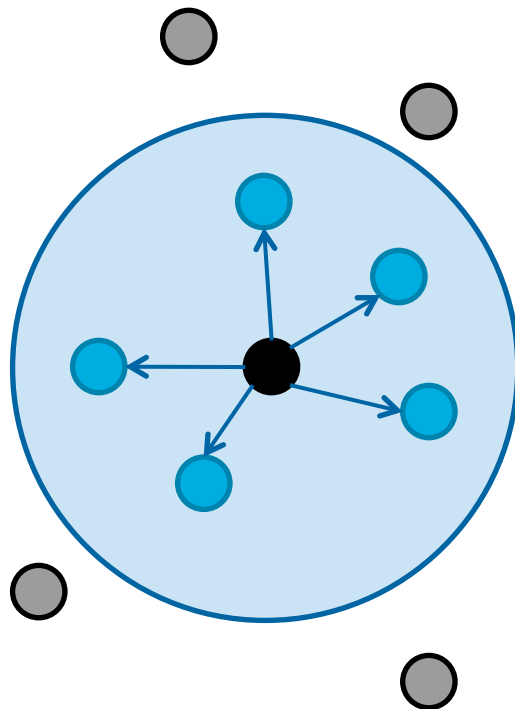
## Goals

- All players in vision range should get actions fast
- Players outside vision range should not get actions
- Fast adjustment when players move around
- No network partitioning

# Applying a P2P algorithm

- Many P2P systems exist
  - Usually developed with different goals
    - Often optimized for file distribution, search or reliability
    - Not suitable for multiplayer context
- Selection of network type
  - Structured network
    - Can be optimized for a specific problem
    - But: Often high maintenance cost (bad in highly dynamic networks like multiplayer games)
  - Unstructured network
    - Low maintenance cost
    - But: Often higher latency and / or required bandwidth
- **Can we exploit advantages of both?**

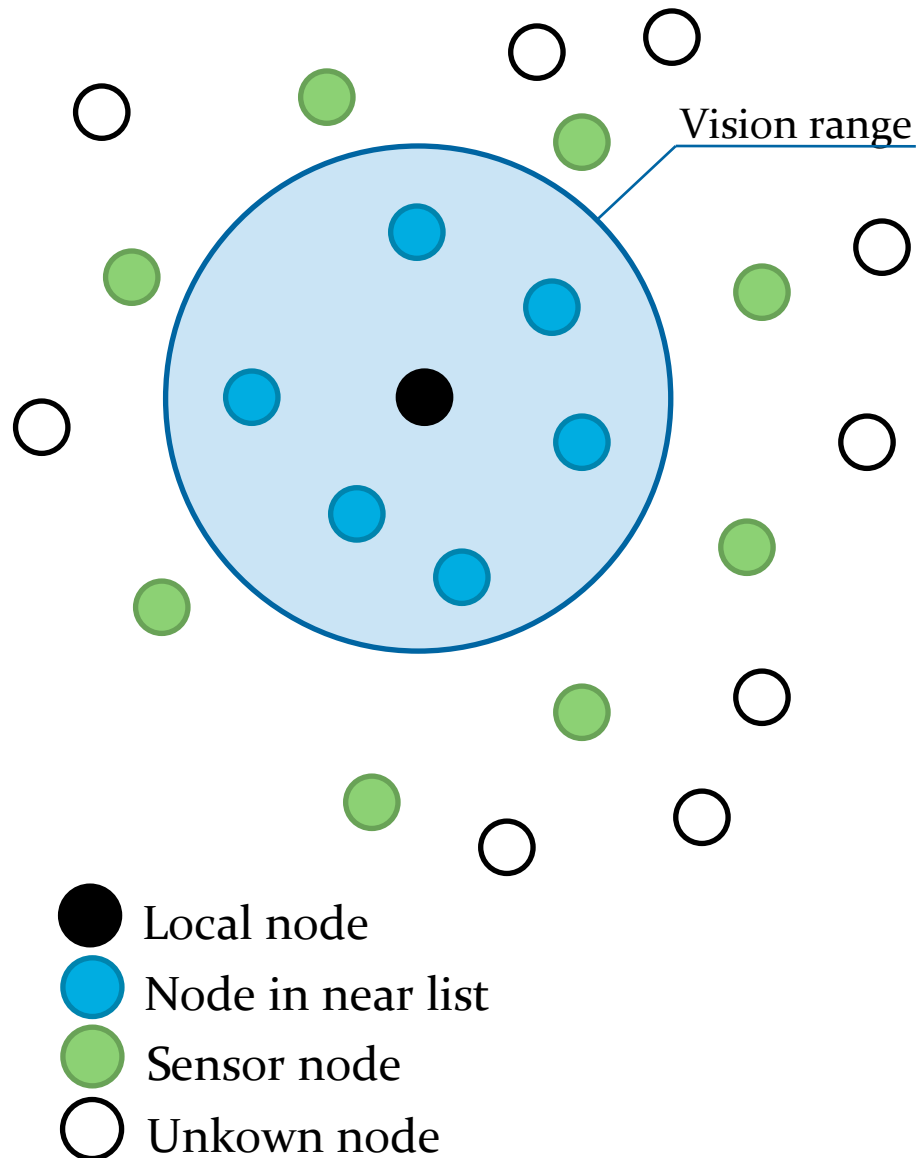
## Near Nodes – Nodes in Vision Range



- Node outside vision Range
- Node in vision Range
- Position update

- Whenever we move, we send the update to all nodes in vision range
- They are informed within 1 hop about our actions
- Very fast (1 hop)
- Maintenance
  - We maintain a list of all nodes we know in vision range
  - When a node leaves the vision range we kick it out of the near node set
  - Detecting nodes that enter vision range solved through **sensor nodes**

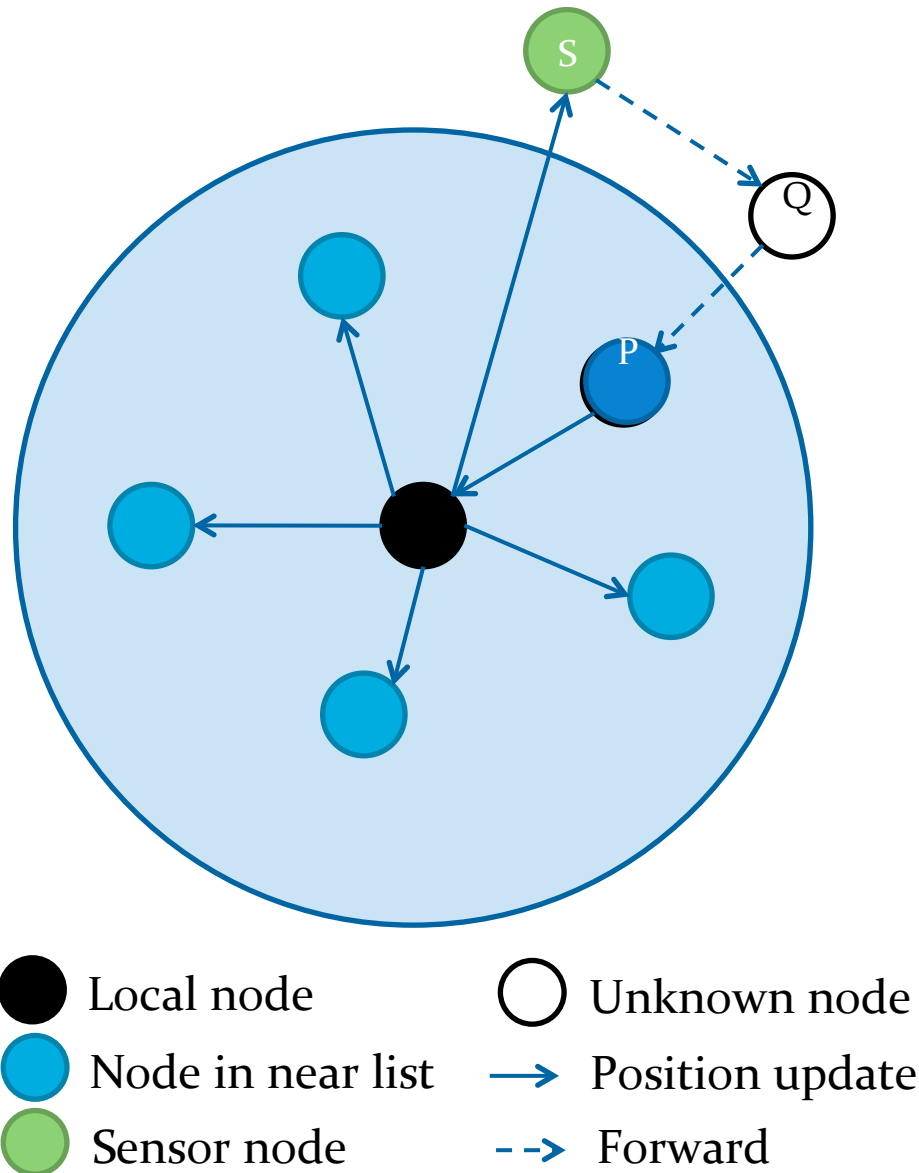
# Sensor Nodes



## Basic Idea

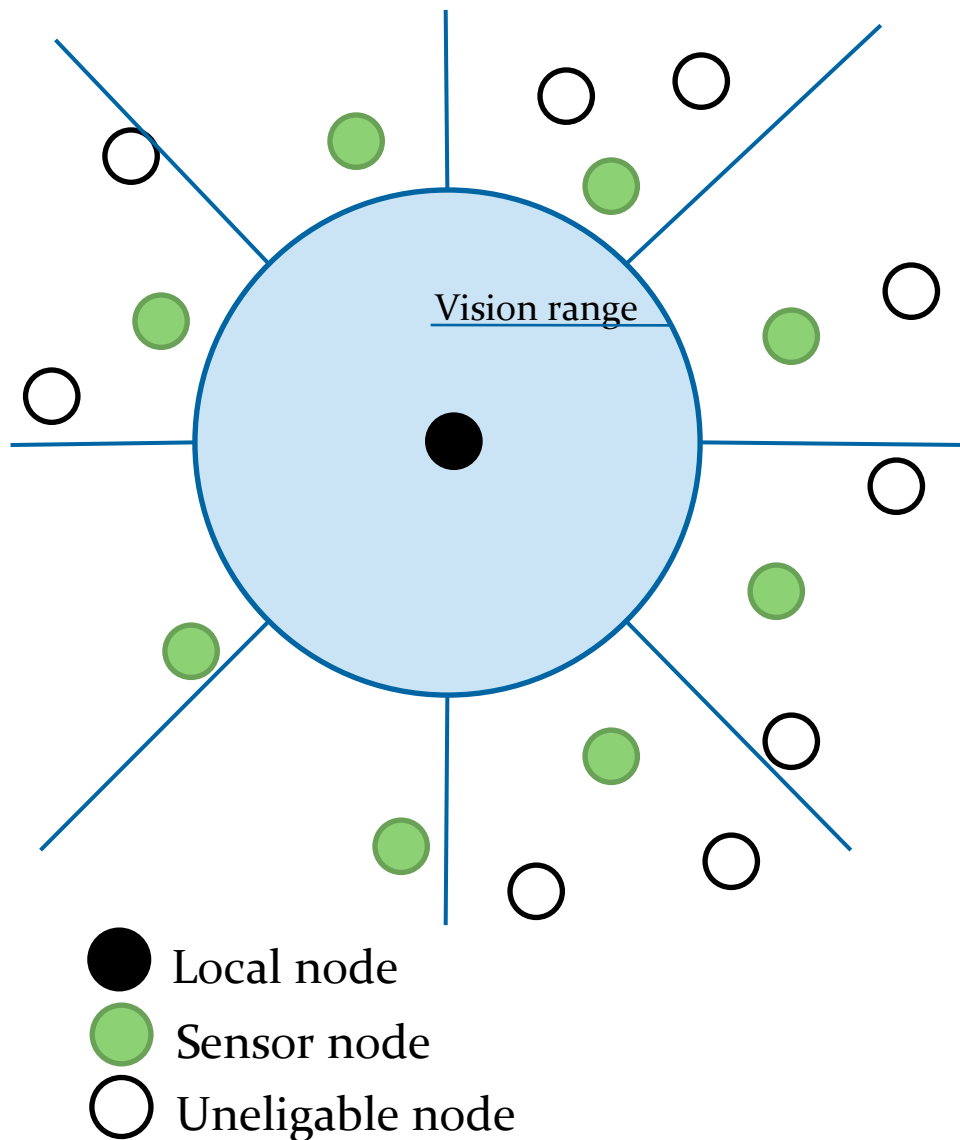
- Keep a set of nodes outside of vision range (called sensor nodes)
- A node sends its actions regularly to its sensor node
- Sensor nodes help finding new nodes that approach or enter vision range
  - Forward messages to nodes they know
- Sensor nodes help to keep system connected

# Sensor Nodes – Detecting New Near Nodes



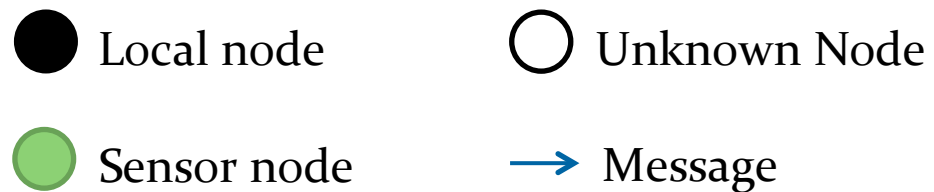
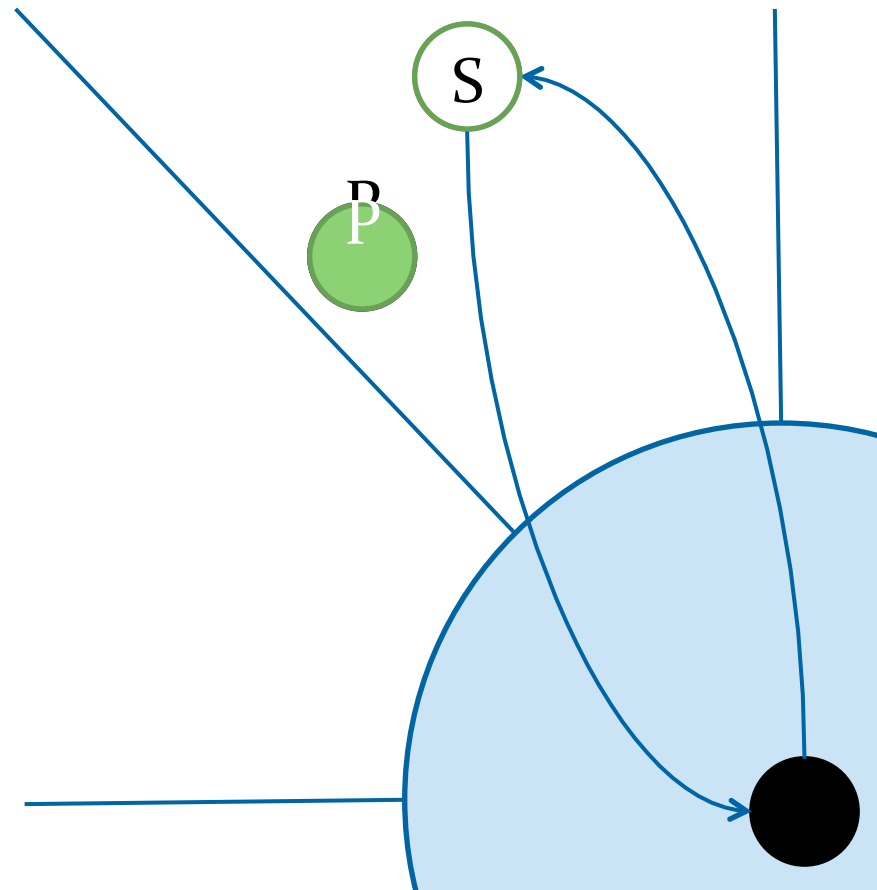
1. Black sends position update to near nodes
2. Sends position update to sensor nodes
3. Sensor node forwards to Q
4. Q forwards to P
5. Now P knows black player and sends it's own position update to black player
6. P is in black's near node list

# Sensor Node Selection



- Different strategies possible
- We applied a simple but effective approach
- Player's view is divided into  $n$  equally sized angle sections
- The nearest node in each angle section that is outside the vision range is chosen

# Sensor Node Maintenance



1. Black sends sensor request to S
2. S knows better sensor P and sends suggestion
3. P replaces the old sensor node

# Additional Solved Challenges



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Handling limited bandwidth
- Joining and leaving the game

# Evaluation

- The algorithm was tested using P2P simulator “Peersim”
- Round based simulation
- Different parameters varied
  - Size of the game world
  - Number of players
  - Movement type of the nodes
    - Random movement
    - Hot spot movement

# Evaluation

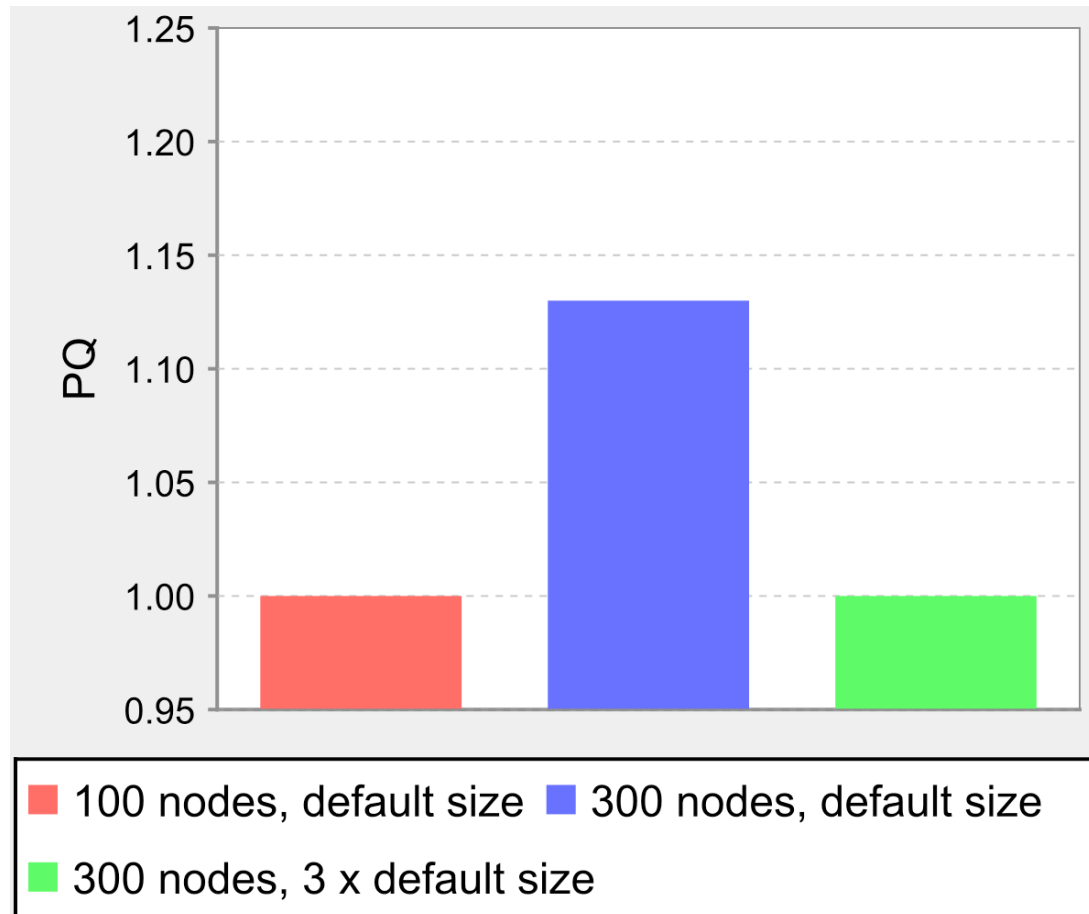
## PositionQuality (PQ)

- Artificial metric
- Used to evaluate freshness of position updates
- Near players are more important than far players
- Higher values are worse
- Perfect value is 1, for client/server it's about 1.4

## Standard Parameters used

- Default game area 1000 X 1000
- Vision radius 200 (→sight = 25% of default game area!)
- Outbound bandwidth 5 KByte/round
- Simulation rounds 500
- Game mode random

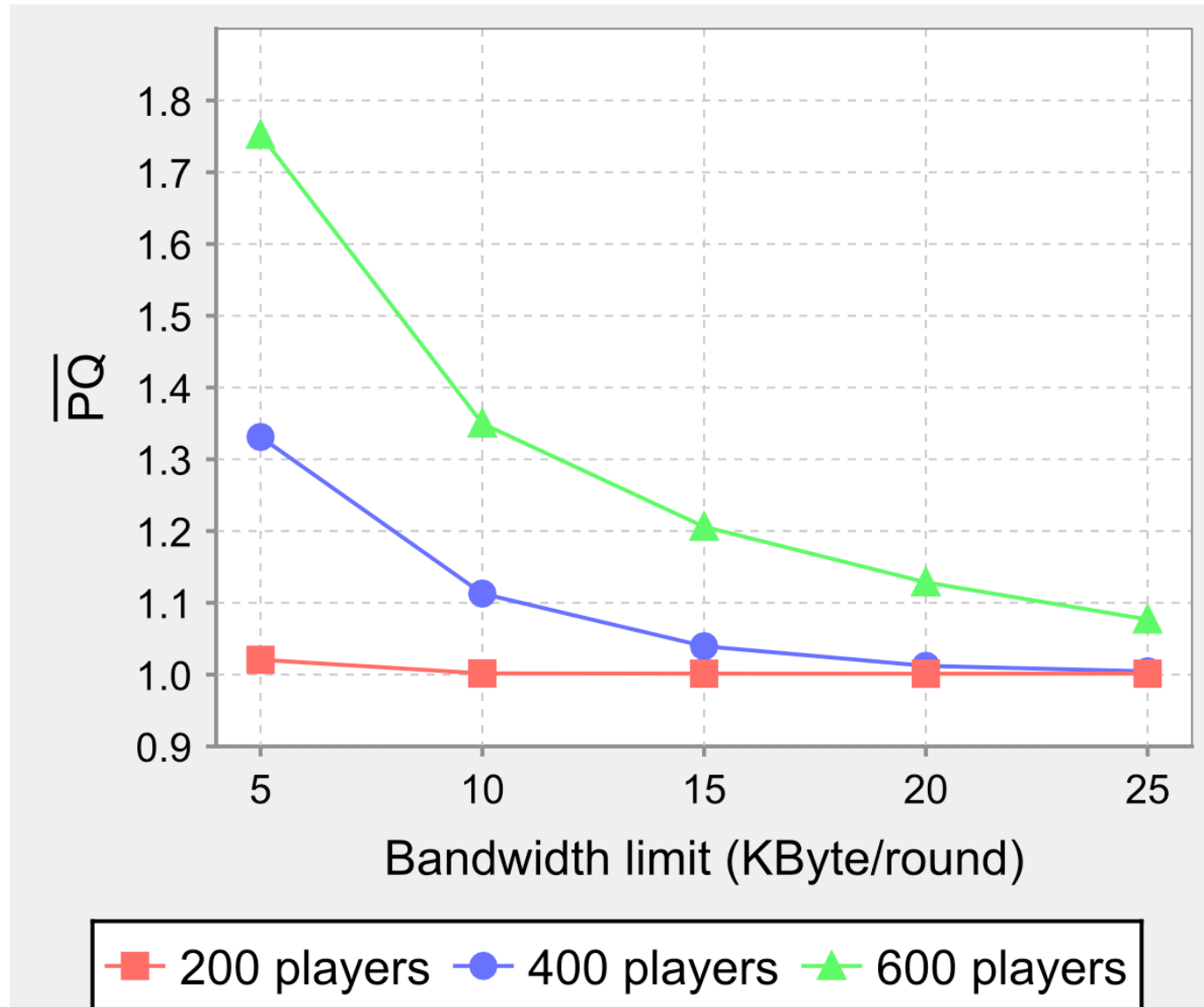
# Statistics: Scalability



- In all cases PQ is very low
- Quality does not depend on number of players in the game or game size
- Quality depends on number of players in the vision range

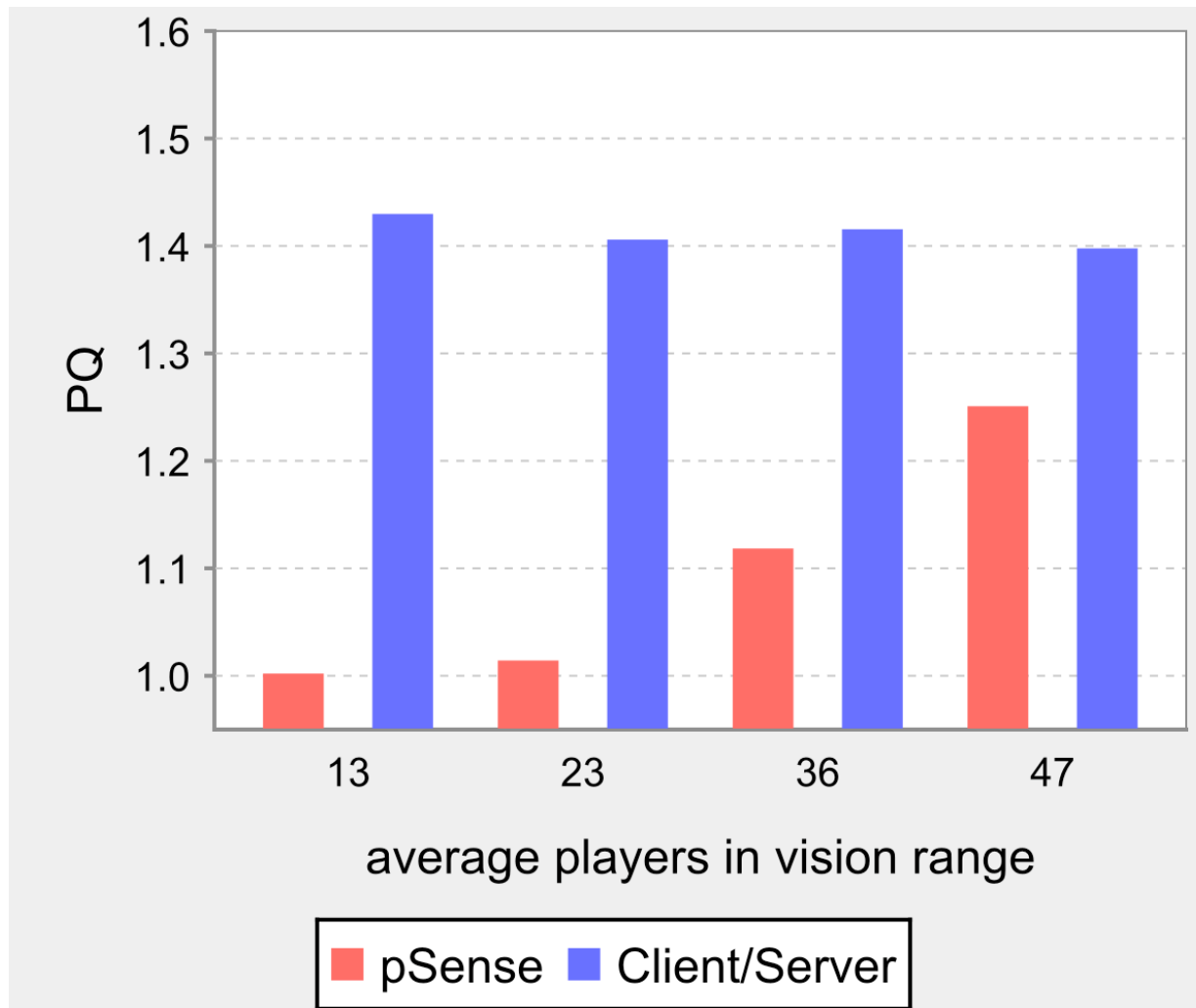
Outbound bandwidth: 5KByte / round

# Bandwidth vs Quality



- More players in vision range require a higher bandwidth
- Even with a low bandwidth the quality is good (<1.8)

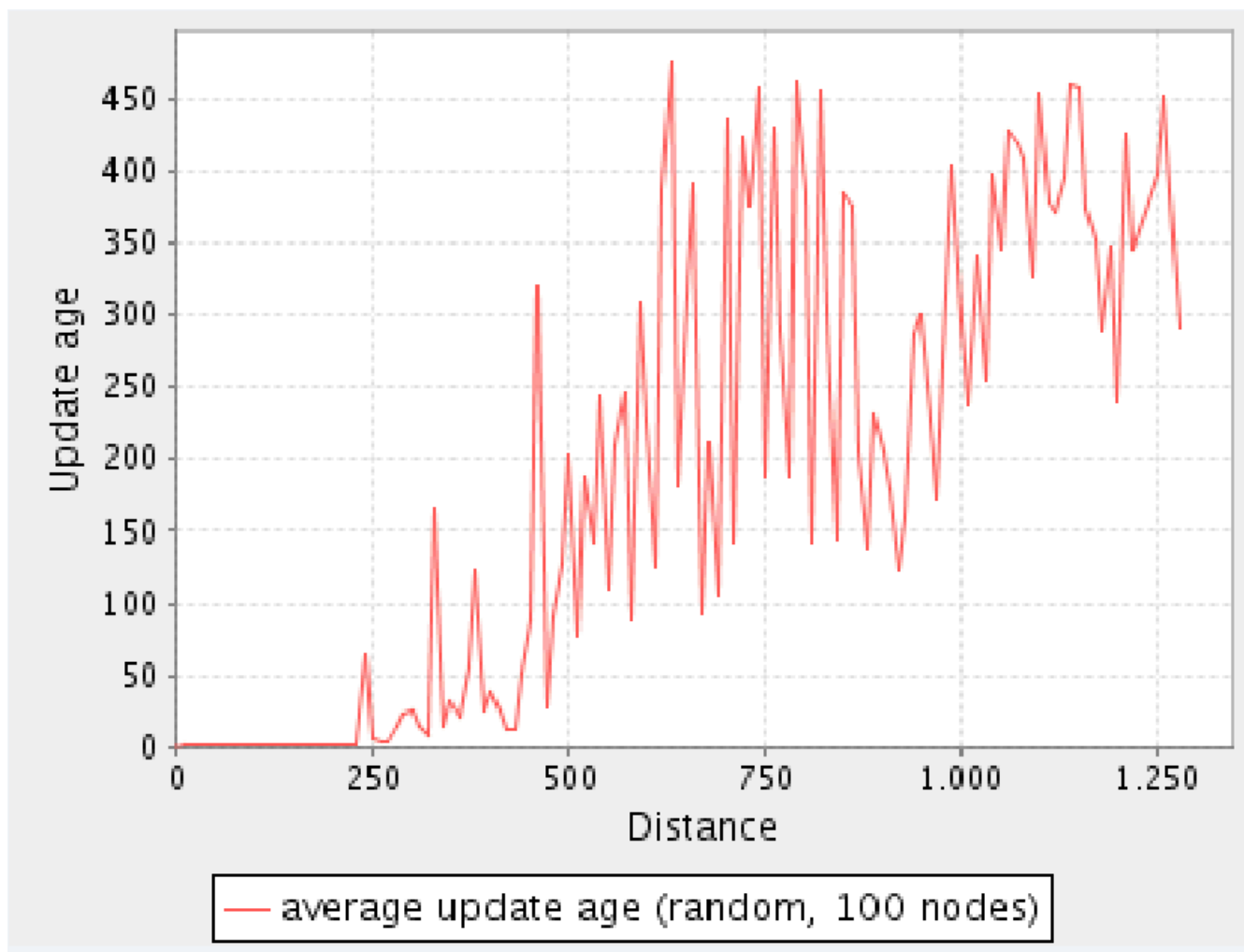
# Statistics: Comparison to Server Based Systems



- Better results than client server

Outbound bandwidth: 5kByte / round

## Statistics: Knowledge of Players Regarding their Distance



- Update age in vision range is close to one
- Outside it is very high

## Conclusion & Future work

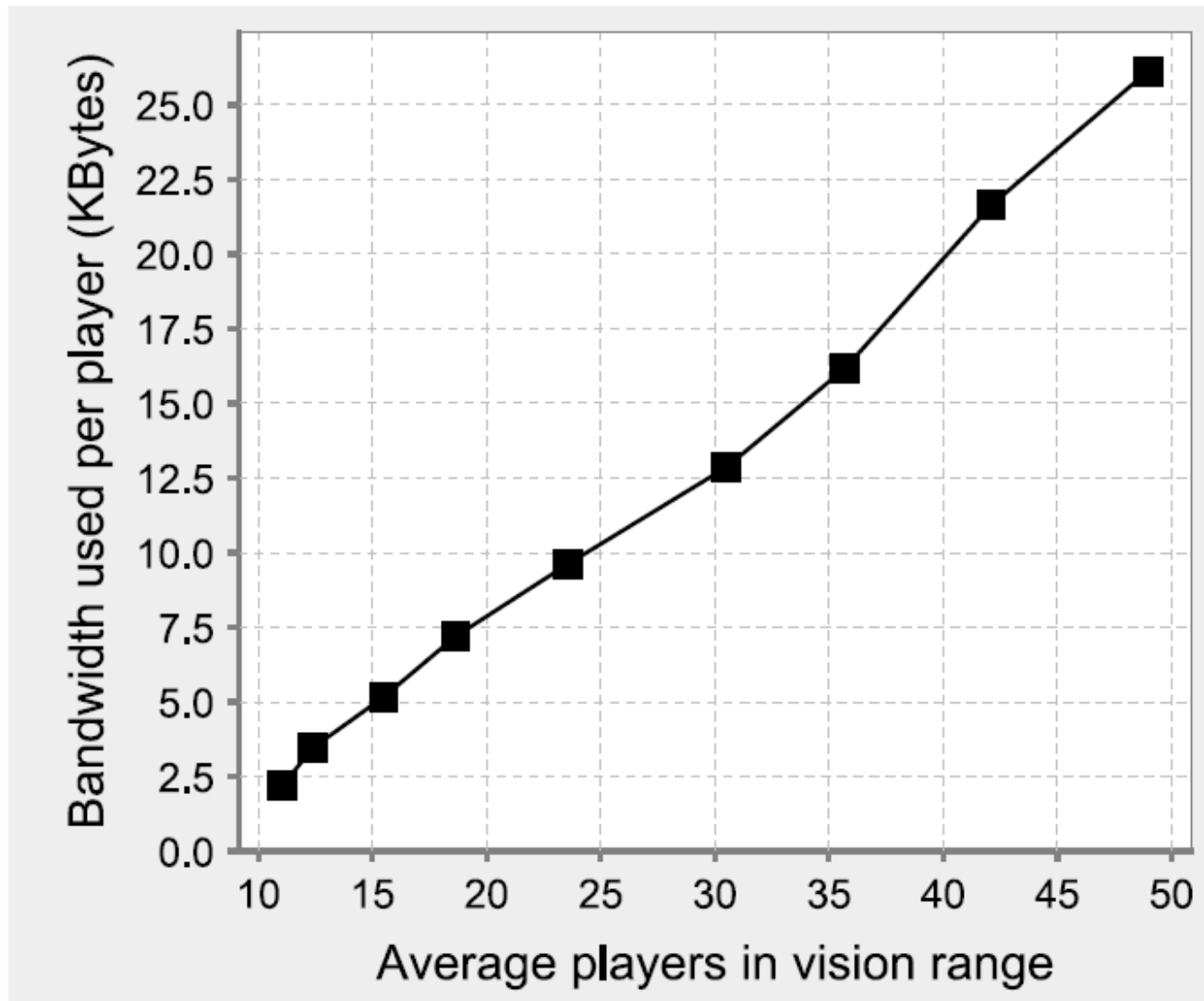
- Conclusion
  - Position-based multicast algorithm suitable for multiplayer games
  - P2P based, no server needed
  - Scales only with number of players in vision range
  - Flexible bandwidth requirements
- Future Work
  - Sensor node selection
  - Different kind of events
  - Crashing peers
  - Cheating



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Thank you for your attention!

# Appendix: Bandwidth Requirement



- Linear growth of Bandwidth needed for perfect update age over average amount of nodes in vision range