Area-Based Gossip Multicast

Christian Seeger, Bettina Kemme
Patric Kabus, Alejandro Buchmann

McGill
TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

- Motivation
  - Massively Multiplayer Games
  - Gossip-Based Broadcast
- Area-based Gossip Multicast (AreaCast)
  - Idea
  - Example
  - Summary
- Experimental results
  - Fanout configuration
  - Gossip configuration
  - Scalability

Motivation: MMGs

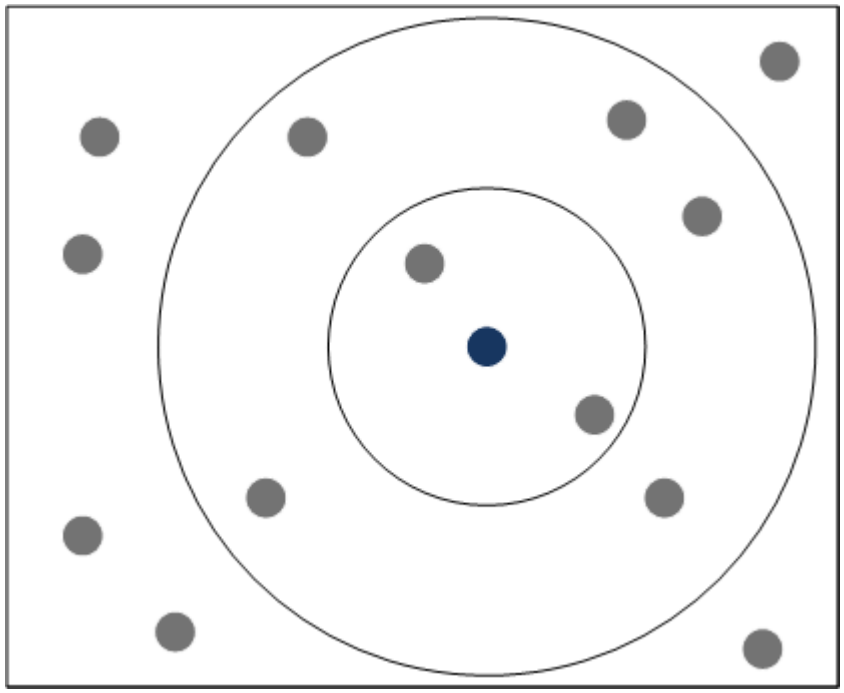
McGill

TECHNISCHE UNIVERSITÄT DARMSTADT

- Communication in massively multiplayer games (MMGs)
  - Players permanently move

  →Position updates are sent continuously

  - A simple broadcast is not scalable
    - Position update broadcast(s) of
      - 1 player
      - 2 players
      - 3 players



- Standard server-based MMGs define special ranges
  - Updates are only sent to players inside a range
  - But: global knowledge is needed → a server has to keep track of all players

Motivation: Gossip-based Broadcast

McGill
TECHNISCHE UNIVERSITÄT DARMSTADT

- Gossip-based broadcast
  - Peer-to-peer approach
  - Nodes send messages to a random subset of nodes in their views
  - Messages contain own and foreign information

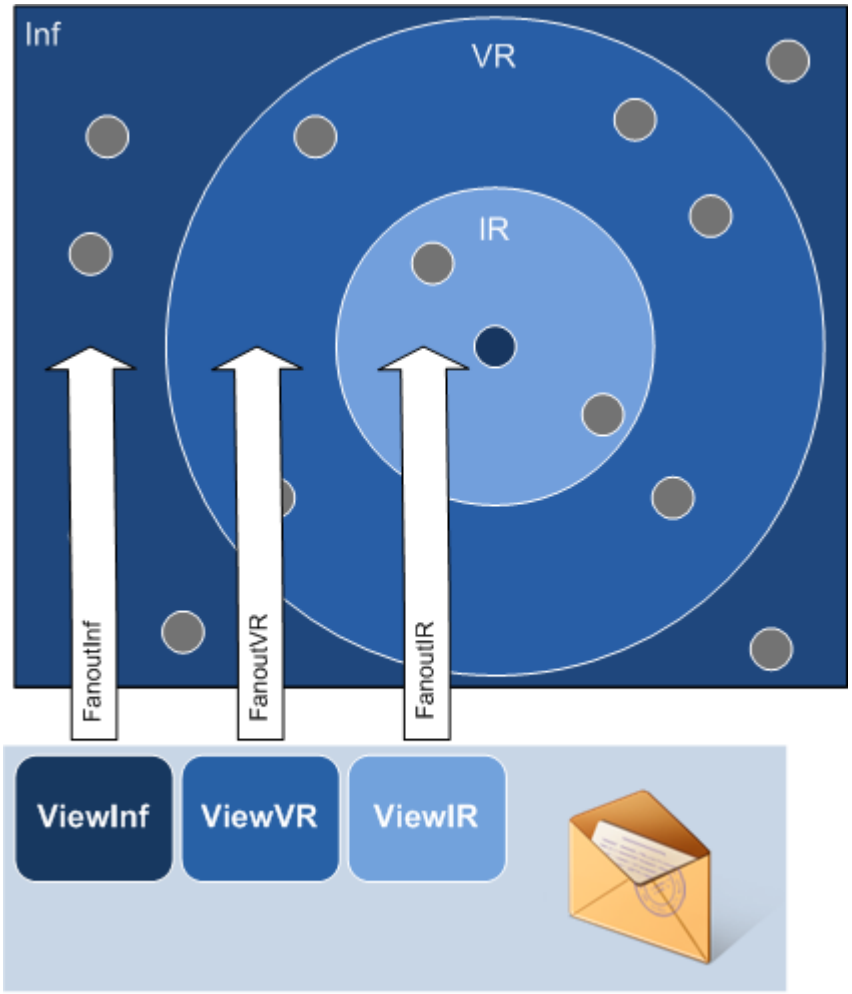
Pro	Contra
-Large groups -High reliability -Robust -Peer-to-Peer	-High redundancy -Overload of an individual node -Too slow for player interaction

Idea: Multicast with 3 Views

- Interaction range (IR)
- Vision range (VR)
- Outside the vision range (Inf)

- Every node has 3 different views
  - Each view contains a set of nodes
  - Nodes are added to a specific view with respect to their position
  - Updated by incoming game information

→Possibility to send messages to nodes in the specific ranges (multicast)



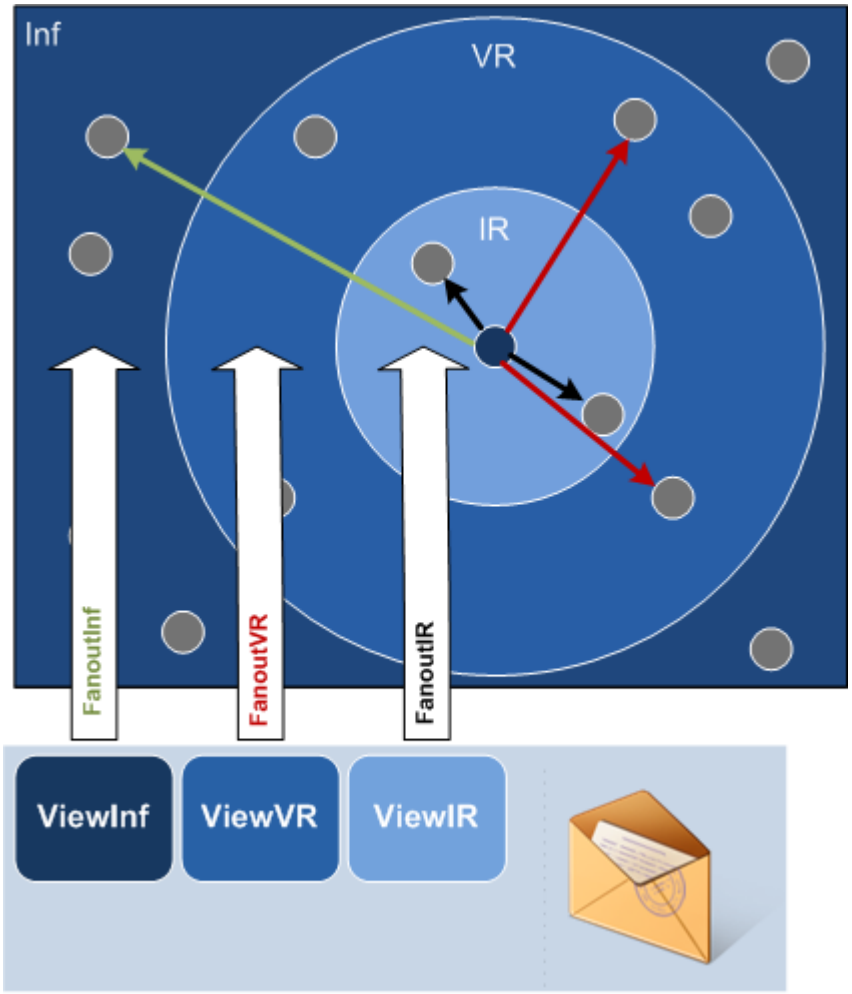
Idea: Multicast with 3 Views

McGill

TECHNISCHE UNIVERSITÄT DARMSTADT

- Nodes within a view are selected randomly, but
- The number of selected nodes is fixed
  - Example:
    - FanoutIR = 2
    - FanoutVR = 2
    - FanoutInf = 1

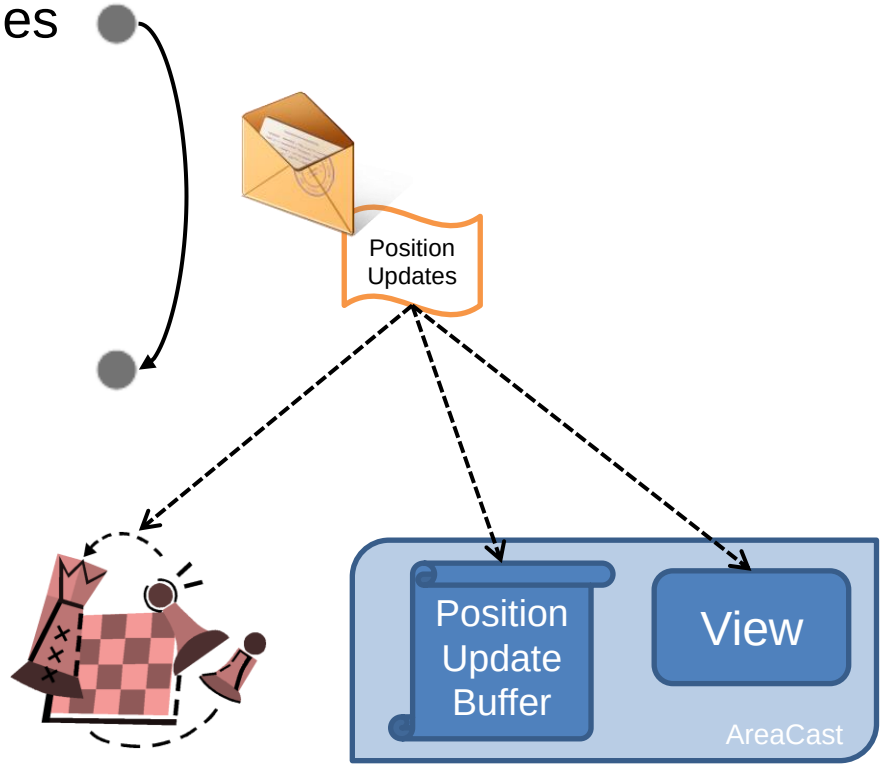
  (Later: fanout-2-2-1)



→ Area-based Gossip Multicast (AreaCast)

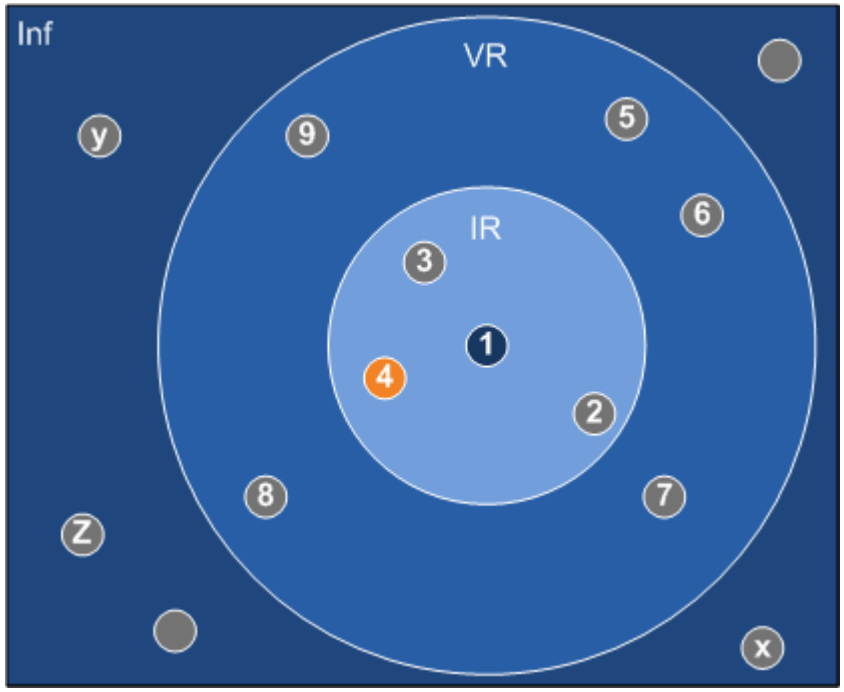
Idea: Message Emission / Reception

- Nodes frequently emit gossip messages
- A gossip message contains
  - Local position update
  - Plus: position updates of foreign nodes
- Incoming position updates are
  - Delivered to the application
  - Stored in the position update buffer
  - Used for view updates
- Old position updates will be deleted from
  - Position update buffer
  - View



AreaCast Example

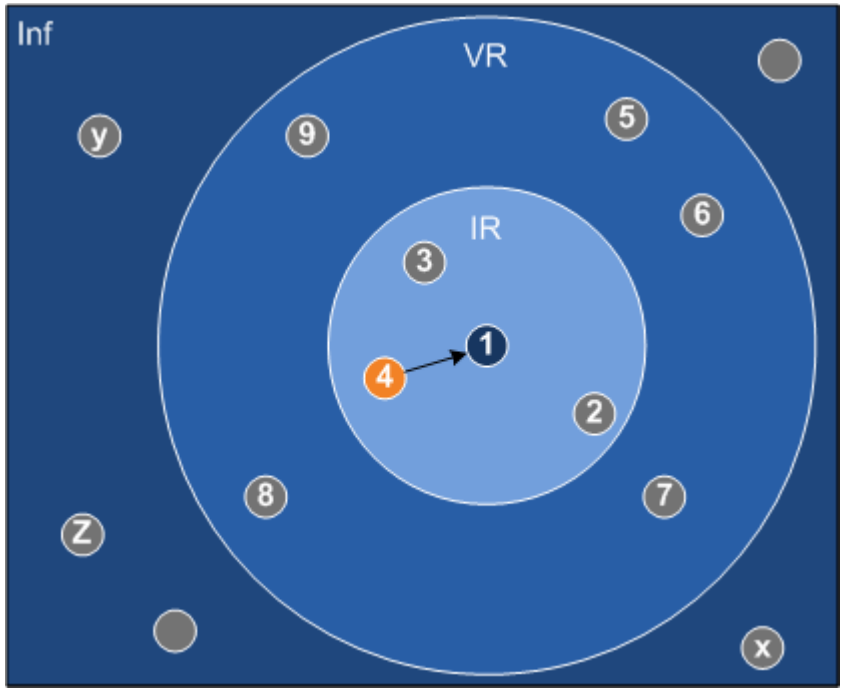
- Node 1 is the local node
  - Nodes in IR and VR are well-known
  - In Inf only some nodes are known
  - *FanoutIR* = 2, *fanoutVR = 2*, *fanoutInf* = 1
- The player of node 4 enters the IR of the player of node 1
- How will the position of node 4 be propagated?



ViewIR	ViewVR	ViewInf
• Node 2 • Node 3	• Node 5 • Node 6 • Node 7 • Node 8 • Node 9	• Node x • Node y • Node z

AreaCast Example

- In this example node 4 enters the IR of node 1 within one round
  - Normally nodes need more rounds to enter another node's IR
- To simplify, we just observe IR and VR
  - Inf is just important to keep the network connected
  - IR and VR are more important for position update emission

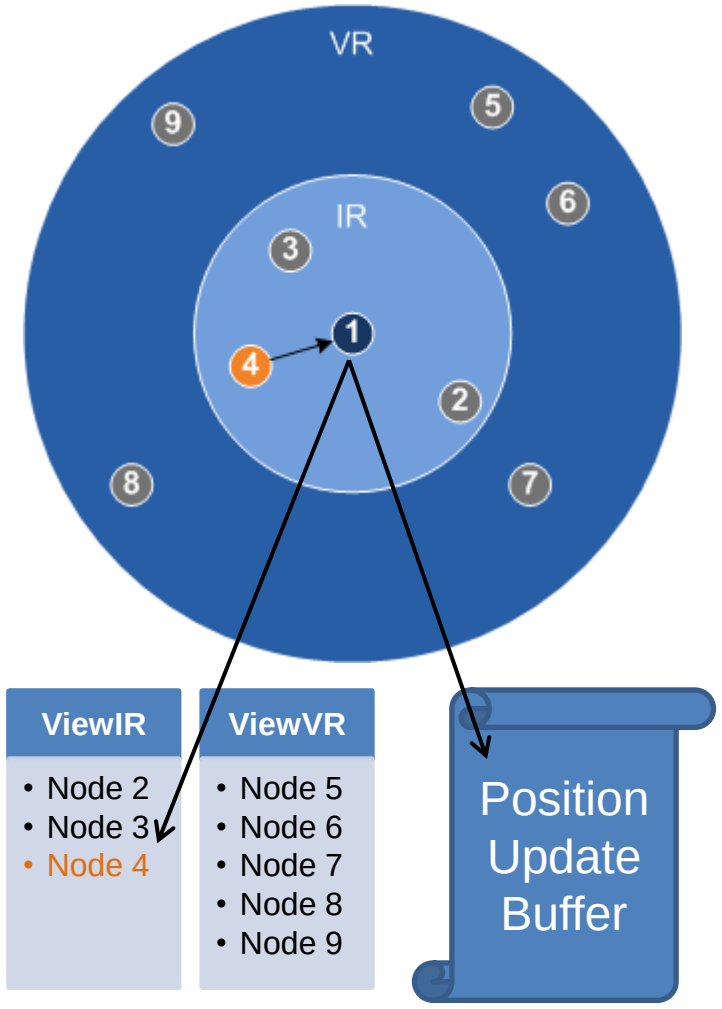


ViewIR	ViewVR	Vi
• Node 2 • Node 3	• Node 5 • Node 6 • Node 7 • Node 8 • Node 9	• N • N • N

Position Update Buffer

AreaCast Example – Round 0

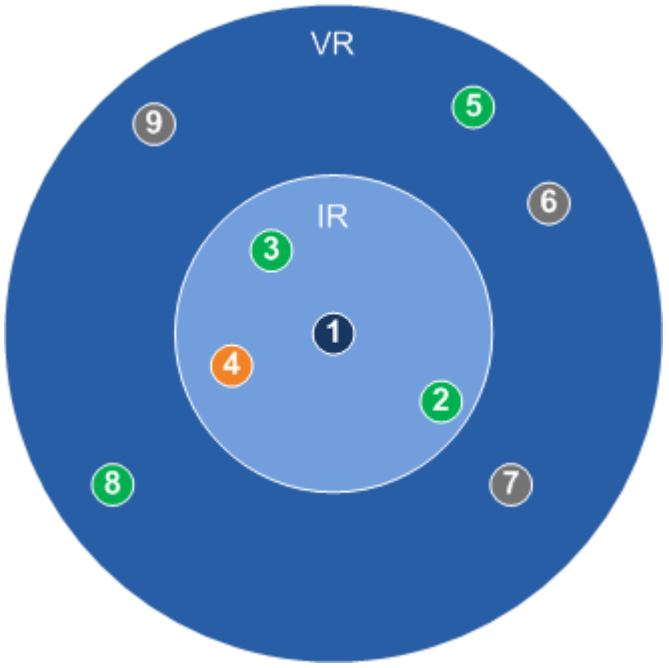
- Node 1 receives gossip message from node 4
  - Extracts all position updates
  - Delivers new position updates to the application
  - And:
    - Adds node 4 to viewIR
    - Adds position update to position update buffer



AreaCast Example – Round 1

- Node 1 – gossip emission

1. Randomly select *n* position updates from position update buffer
2. Randomly select *fanout* nodes from views
   - 2 nodes from viewIR (fanoutIR = 2)
   - 2 nodes from viewVR (fanoutVR = 2)
3. Send gossip message to selected nodes



ViewIR	ViewVR
• Node 2	• Node 5
• Node 3	• Node 6
• Node 4	• Node 7
	• Node 8
	• Node 9

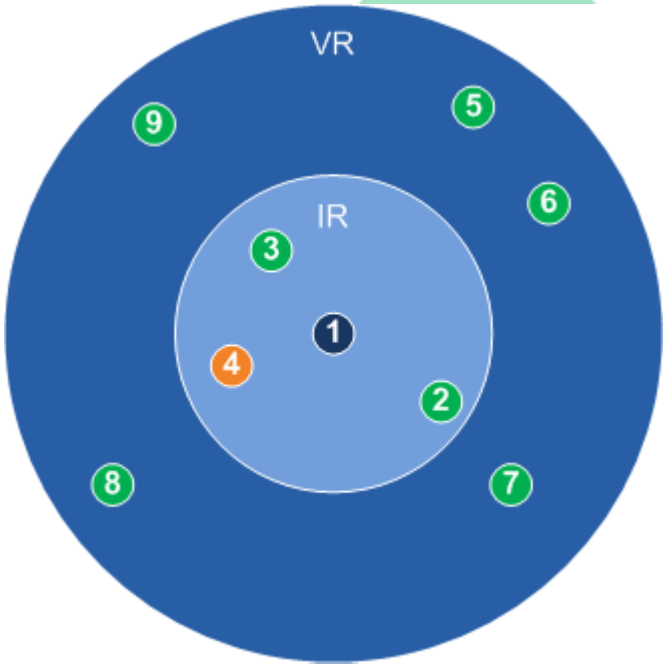
Position Update Buffer

AreaCast Example – Round 2

- Node 1 – gossip emission
  1. Randomly select *n* position updates
  2. Randomly select *fanout* nodes from views
  3. Send gossip messages to selected nodes

- Do node 6 and node 7 know node 4's position?
  - They are in the IRs of node 2 / 5
  - Node 2 / 5 got the position one round before
  - → They'll inform node 6 / 7 within this round

  - For a larger buffer than n: $P(node_4) = n / BufferSize$

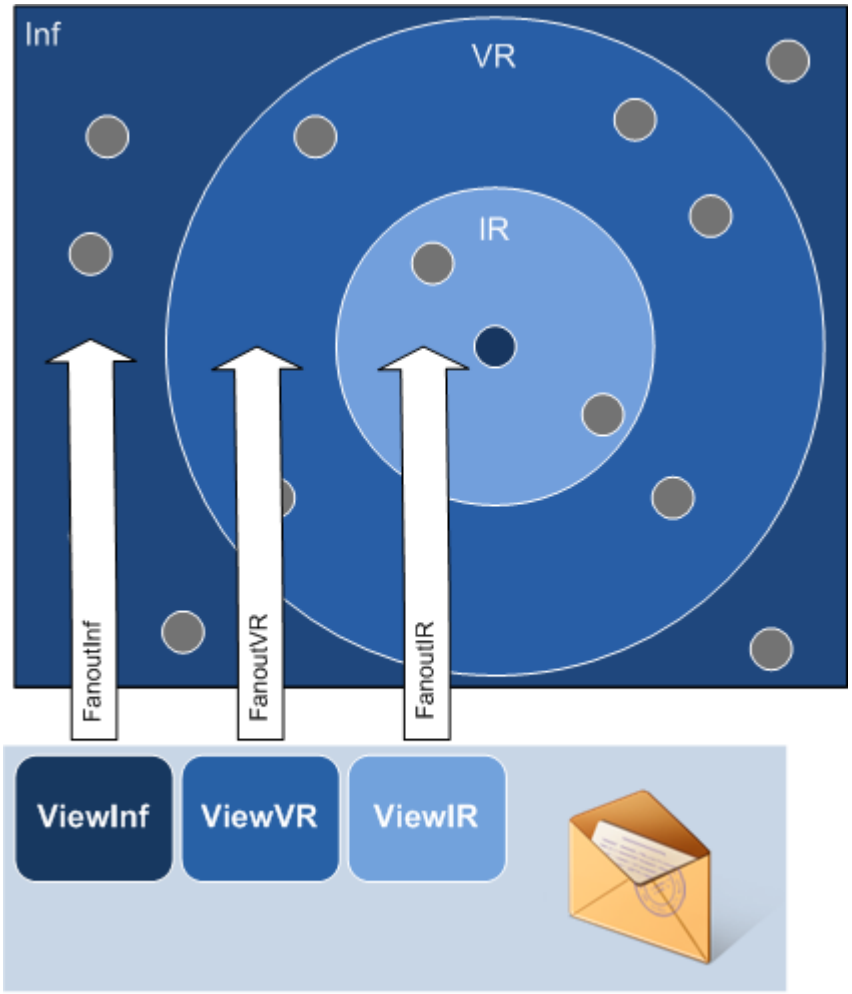


ViewIR	ViewVR
• Node 2	• Node 5
• Node 3	• Node 6
• Node 4	• Node 7
	• Node 8
	• Node 9

Position Update Buffer

AreaCast Summary

- Nodes frequently emit gossip messages
  - A gossip message contains local and foreign position updates
  - Gossip messages are sent to different ranges: *IR, VR, Inf*
  - Number of selected nodes per range: *fanoutIR, fanoutVR, fanoutInf*

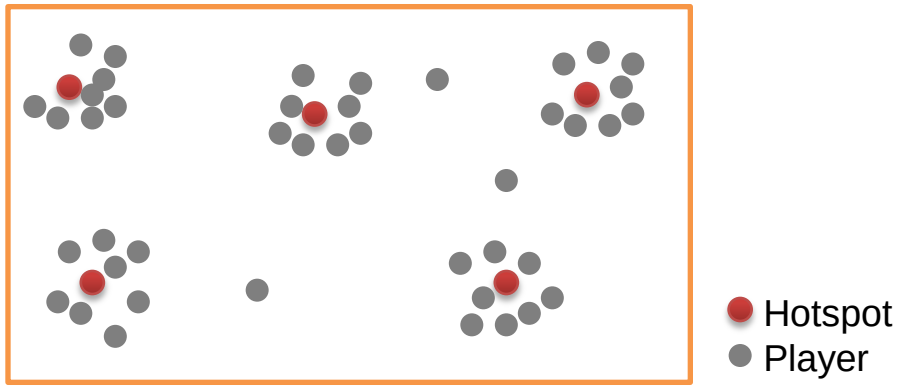


Evaluation: Play Modes

McGill
TECHNISCHE UNIVERSITÄT DARMSTADT

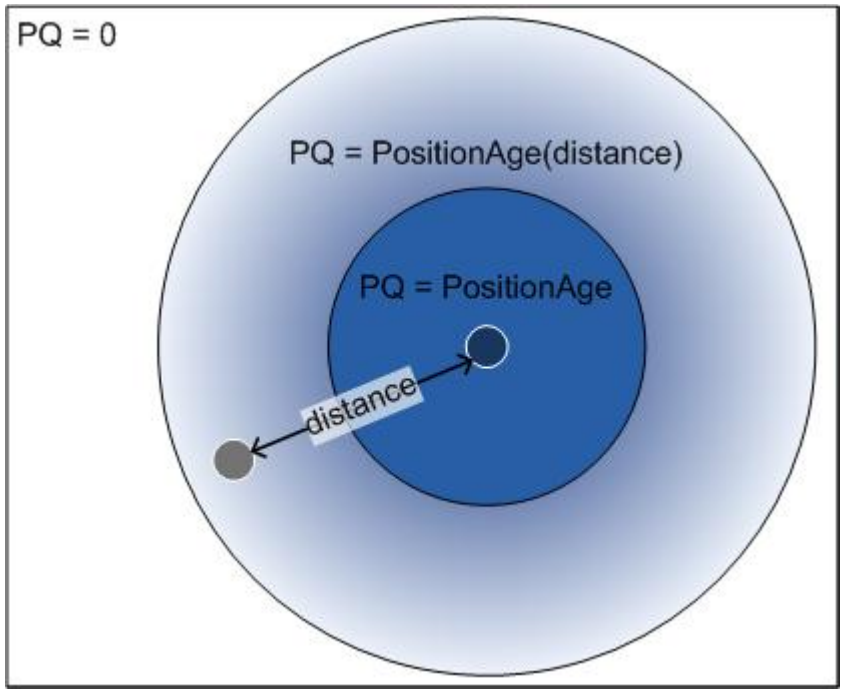
- Random play mode
  - Each player moves into a random direction
  - Direction changes with a certain probability

- Hotspot play mode
  - Additional points on the field (hotspots)
  - Each player chooses one and moves to it
  - After a random exposure time the player moves to another hotspot



Evaluation: Protocol Quality

- *Protocol quality (PQ)* is determined by
  - The age of position information a node has about another node
  - The distance between those nodes
    - *IR*: PQ is equivalent to the age
    - *VR*: the importance of the information decreases with the distance
    - *Inf*: not considered



- *PQ1Max*: worst *PQ* of a node in a given simulation round
- *PQ2*: average of all *PQ1* values up to a given simulation round

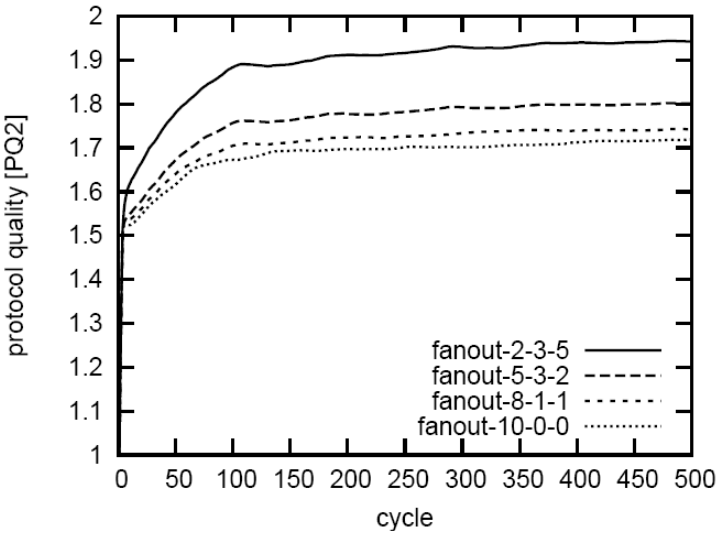
- Desired PQ2 value: 1 < PQ2 < 2 (server client system: PQ2 ~ 1.5)

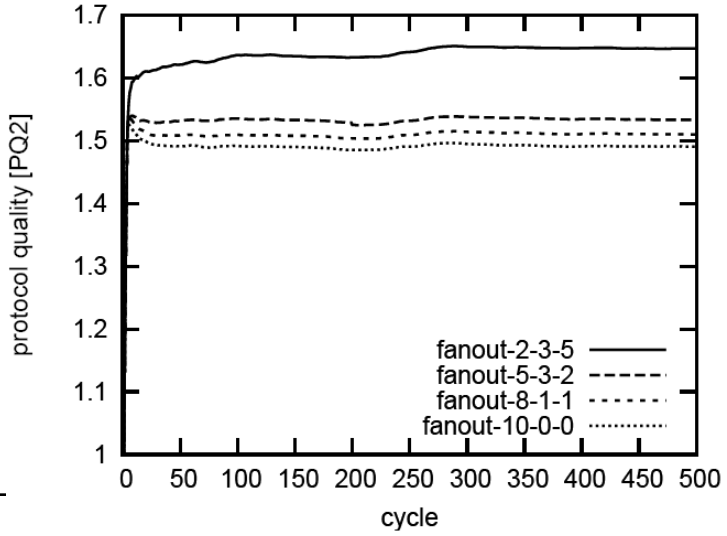
Evaluation: Fanout Configuration

- **Overall fanout of 10**
- PQ2 values of different fanout configurations in
  - Hotspot mode (upper figure)
  - Random mode (lower figure)

- Results
  - →The more messages are sent within IR, the better the protocol quality
  - →Best performance with the whole fanout at the interaction range

Configuration:
100 players
60 updates per gossip message
3 rounds maximum update age





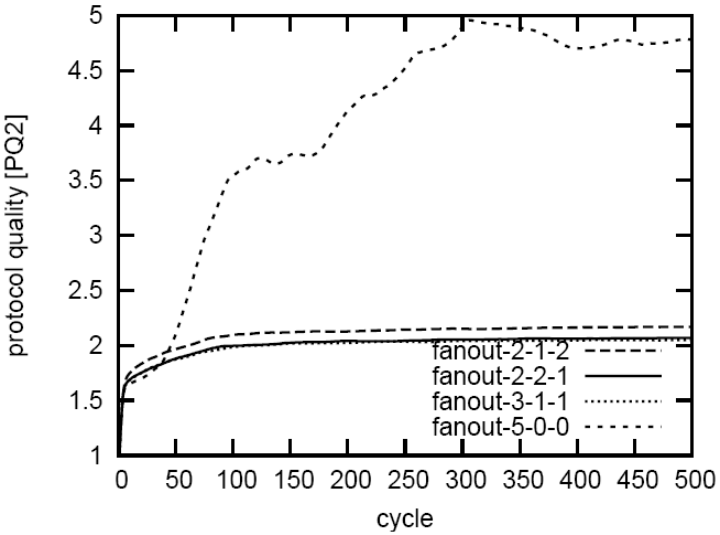
Evaluation: Fanout Configuration

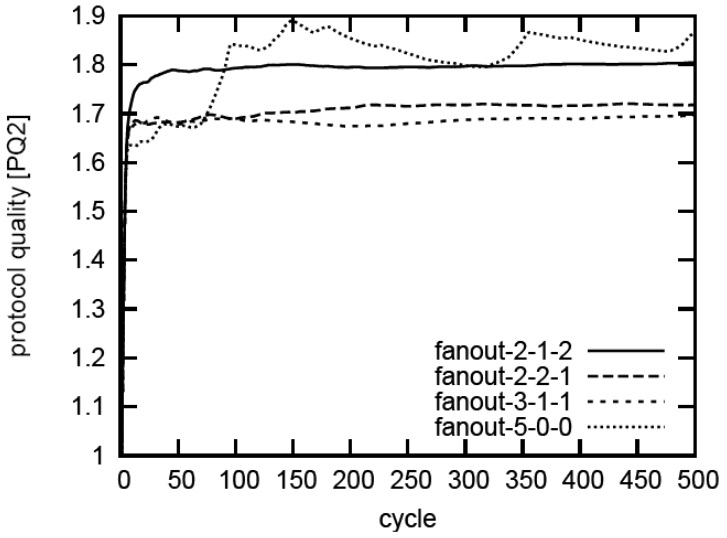
- **Overall fanout of 5**
- Compared with a fanout of 10
  - Same characteristics (except *fanout-5-0-0)*
  - Little worse results
- For *fanout-5-0-0* the protocol quality breaks down and becomes erratic

- In contrast to fanout-10-0-0, fanout-5-0-0 provides the worst protocol quality

→ Why?

Configuration:
100 players
60 updates per gossip message
3 rounds maximum update age





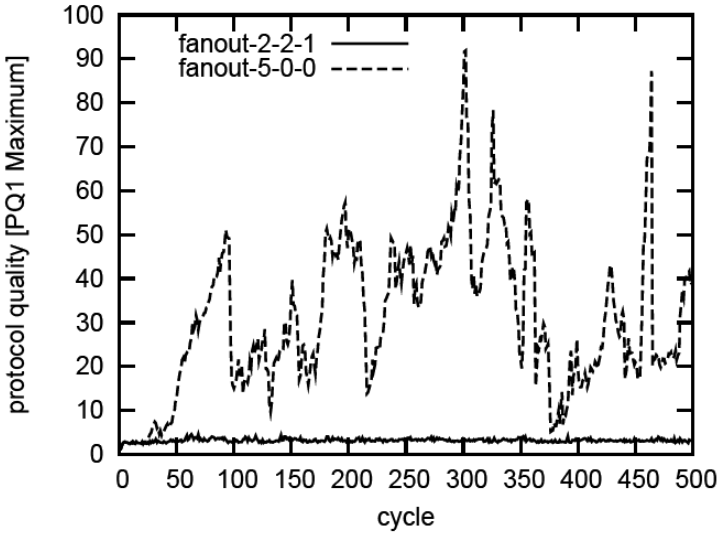
Evaluation: Fanout Configuration

McGill

TECHNISCHE UNIVERSITÄT DARMSTADT

- Comparison of PQ1Max values
- Hotspot mode

- Very high peaks of the *fanout-5-0-0* values
  - Indicates that nodes temporary do not know other nodes in their IR / VR
  - Reason for the worse PQ2 values



Configuration:
100 players
60 updates per gossip message
3 rounds maximum update age

- Results

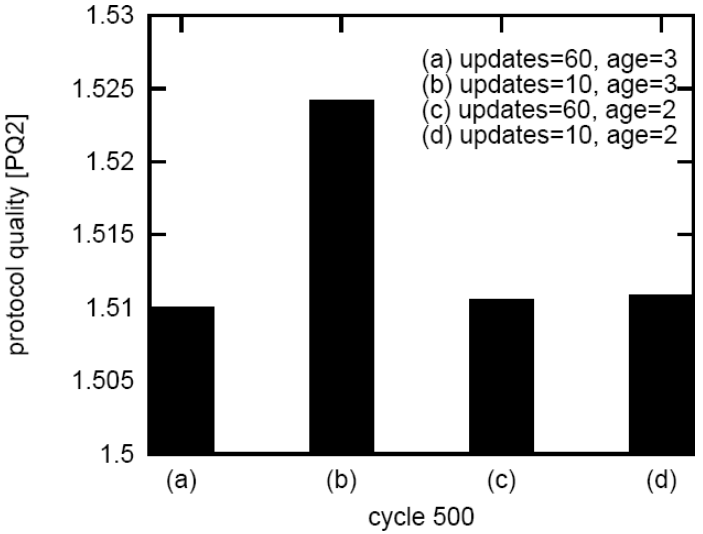
→*FanoutInf* could be required to keep the network connected

Evaluation: Gossip Configuration

McGill

TECHNISCHE UNIVERSITÄT DARMSTADT

- For a lower bandwidth demand we tested AreaCast with a reduced gossip size
  - Number of updates per gossip: **60 /** 10
  - Number of hops an update may travel: **3** / 2
  - Fanout-8-1-1 configuration, random mode



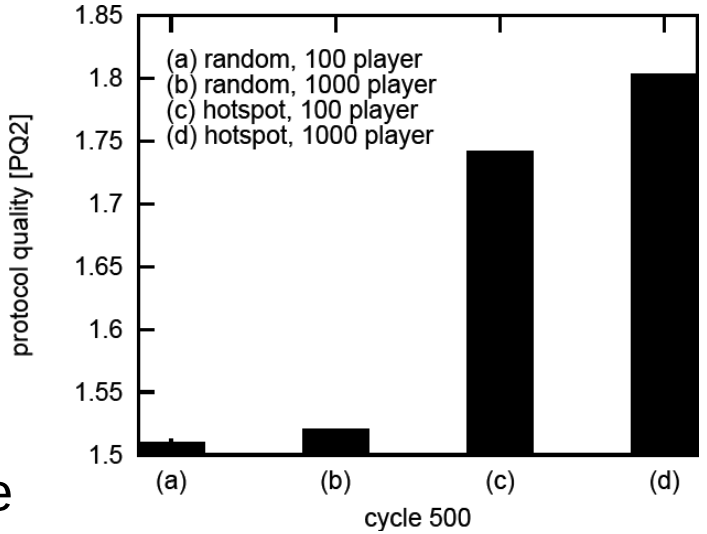
  - Generally small differences in PQ2 performance
  - Nearly the same results with 10 updates (d) as with 60 updates (a, c)
  - (b) provides slightly worse results
    →Obsolete updates congests the gossips

- Results

  →Just a few updates per gossip achieve a good performance

Evaluation: Scalability

- Comparison between 100 and 1000 players
  - Game size for 1000 players is 10-fold the standard game size
  - Fanout-8-1-1 configuration
  - 60 updates, 3 rounds



- Random mode: nearly the same performance
- Hotspot mode: slightly worse results with 1000 players

  →Players need longer to change the hotspot, so it takes longer to adjust their views

- Result

  → Good scalability properties

Conclusions and Future Work

McGill
TECHNISCHE UNIVERSITÄT DARMSTADT

- Area-Based Gossip Multicast (AreaCast)
  - Fast and truly distributed message dissemination
  - Continuously changing neighborhoods while keeping network connected
  - Low network congestion
  - Good scalability
  - Good performance

- Open topics
  - Optimized view management
  - Dynamic fanout decisions and forwarding strategies
  - Cheating detection
  - Conflict resolution for special events (e.g. picking up objects)

THANK YOU FOR YOUR ATTENTION!