Tecellate

Wireless Network Simulator

From the Future!

Tim Henderson & Steve Johnson

The Problem

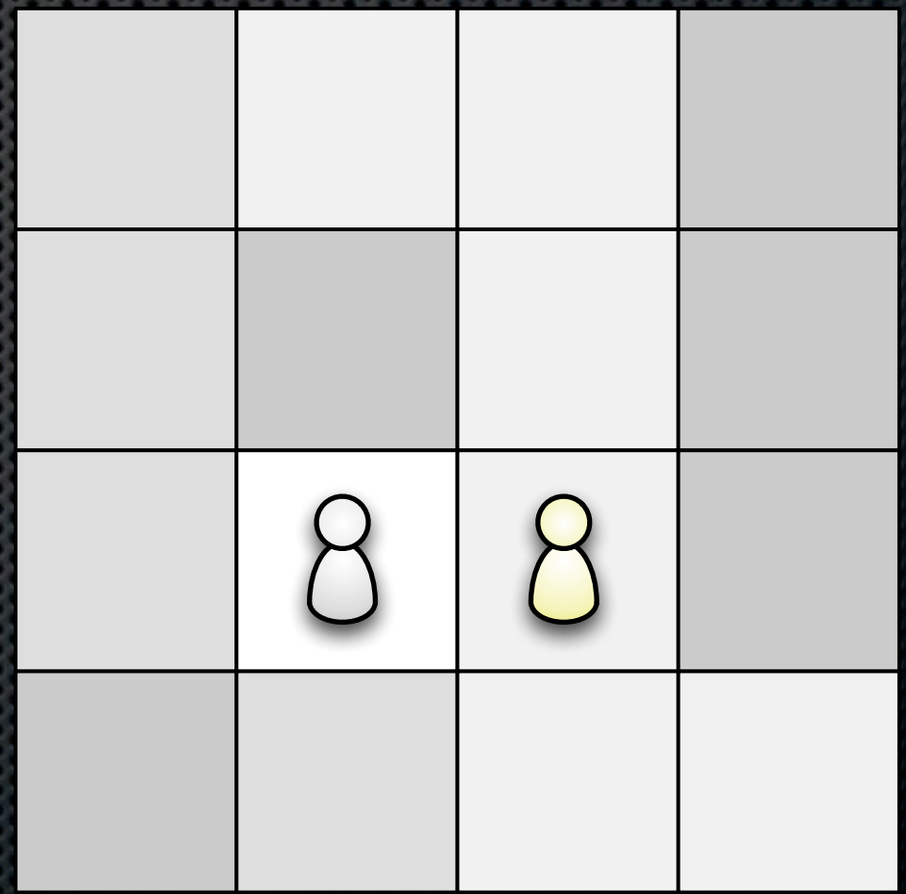
- You are a brilliant computer networks researcher
- Wireless networks are increasingly important
  - MANETs/Ad Hoc
  - Tethering
  - Cell phones
- How do you simulate algorithms in development?

Our Solution

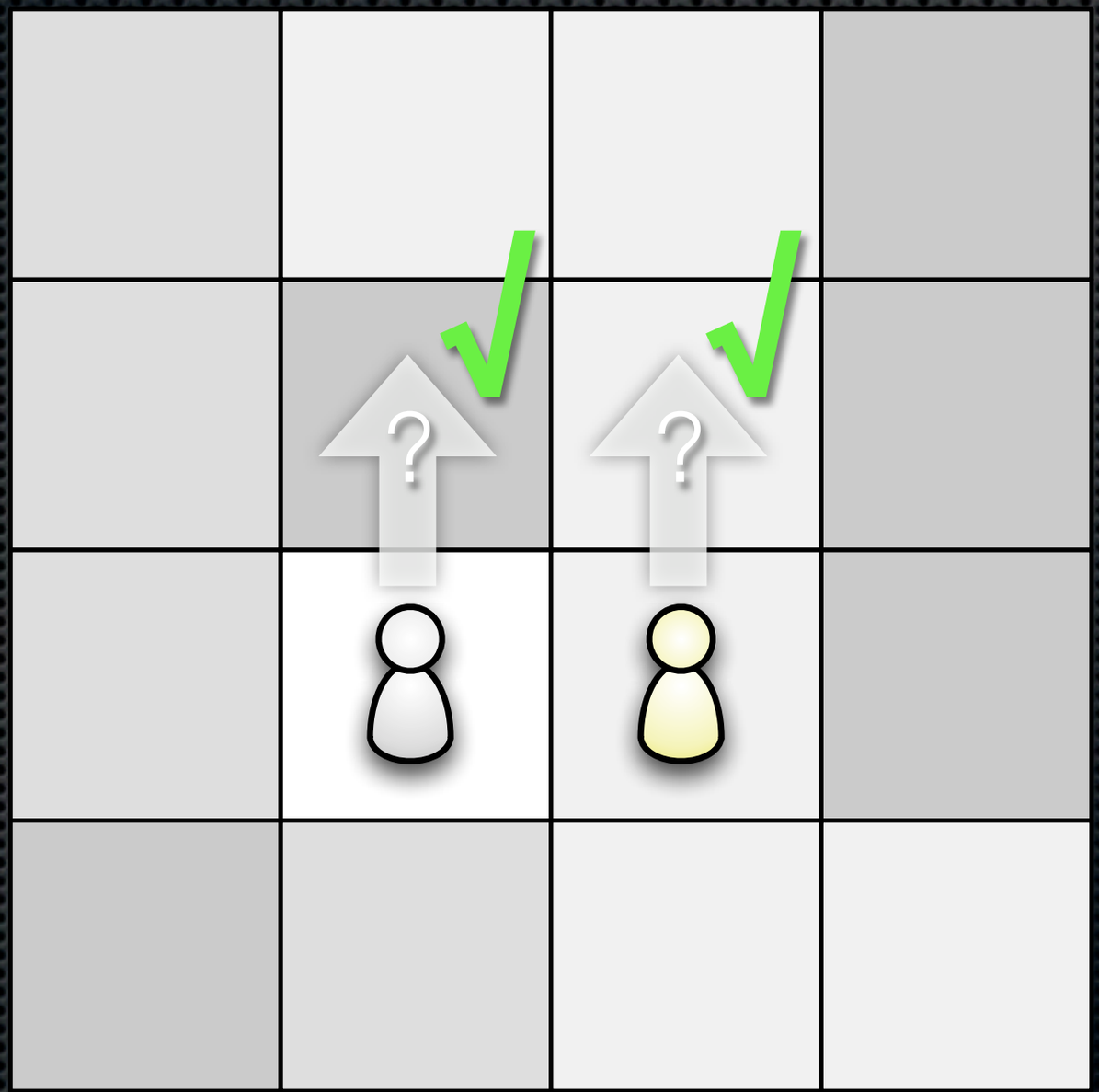
- Halfway between a probabilistic model and a physics simulation
- Highly scalable to allow efficient testing of massive scenarios
- Applications in both networks and artificial intelligence

The Environment

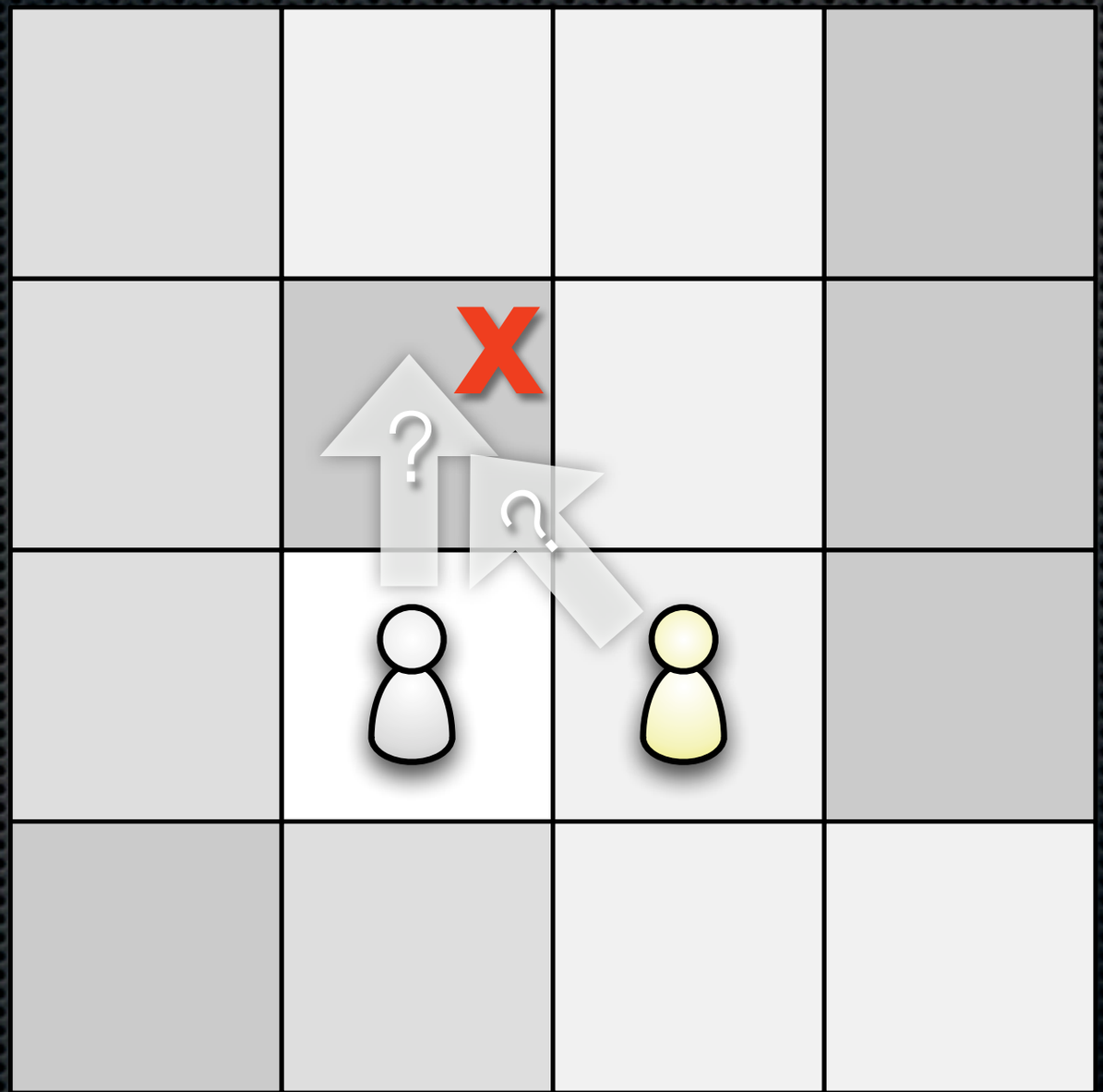
- Turn-based grid world inhabited by agents
- Each turn, agents may:
  - Move to nearby unoccupied square (if terrain allows)
  - Listen on a frequency
  - Broadcast on a frequency
  - Eat
  - Die of starvation



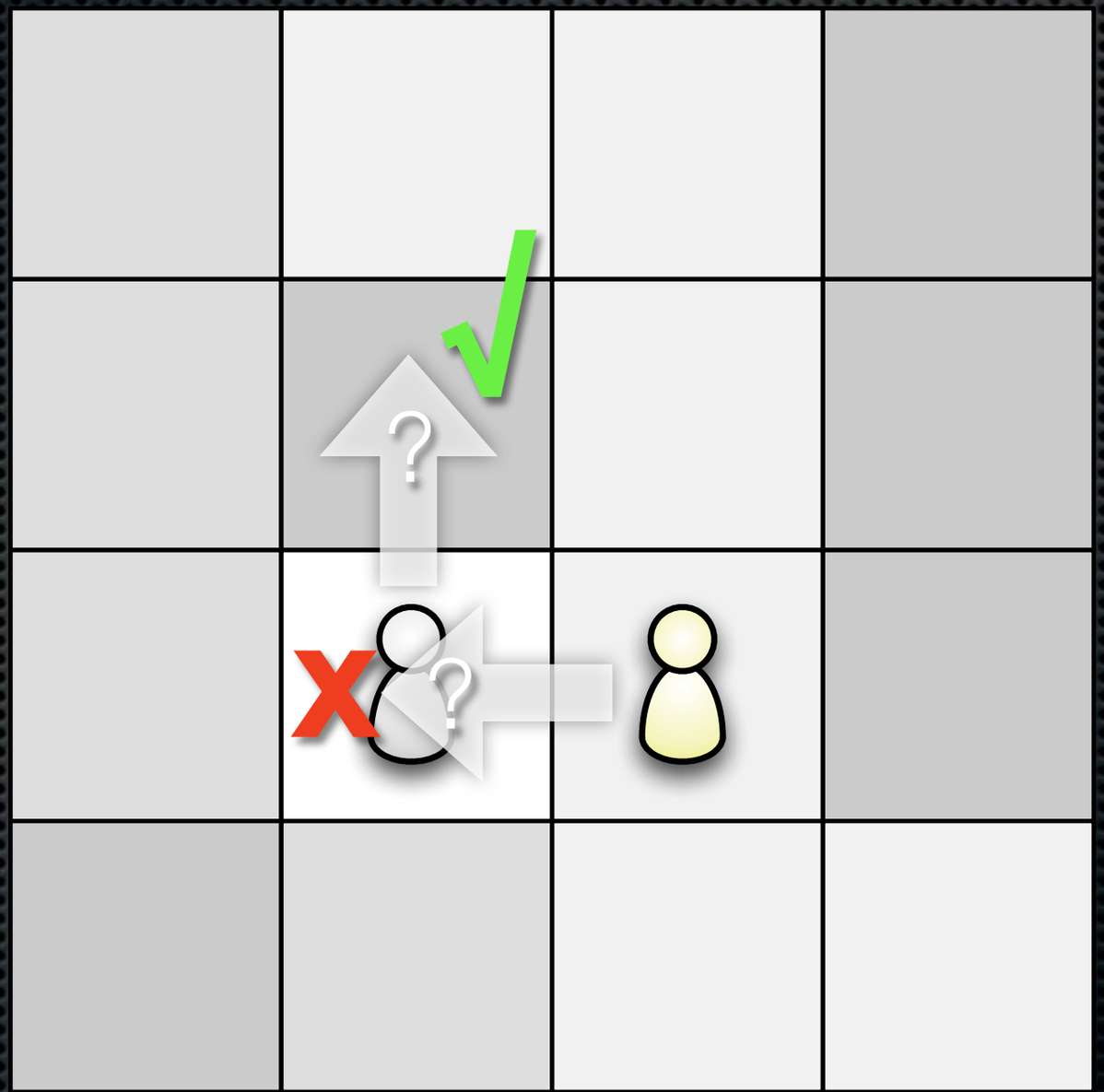
The Environment: Movement



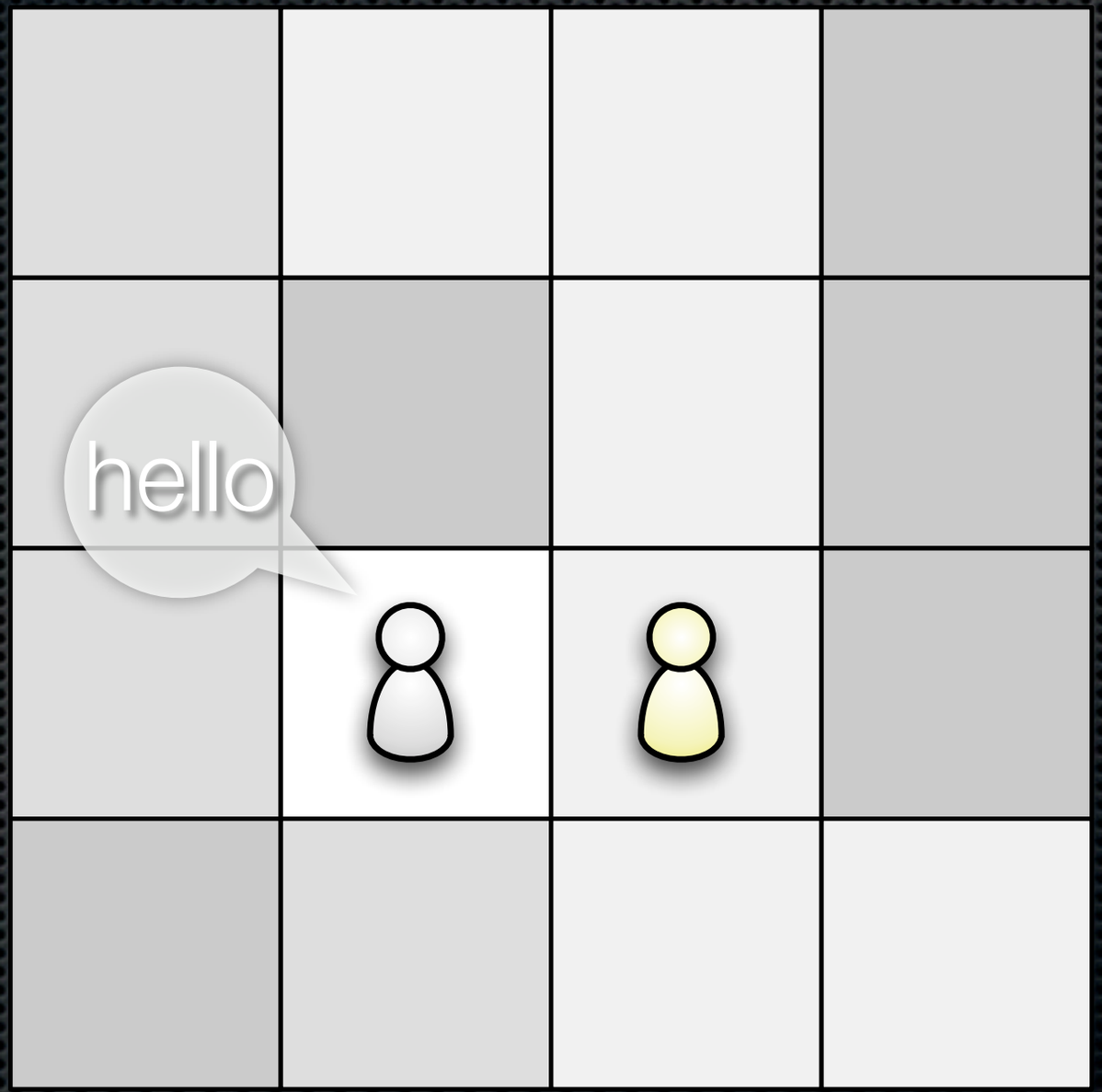
The Environment: Movement



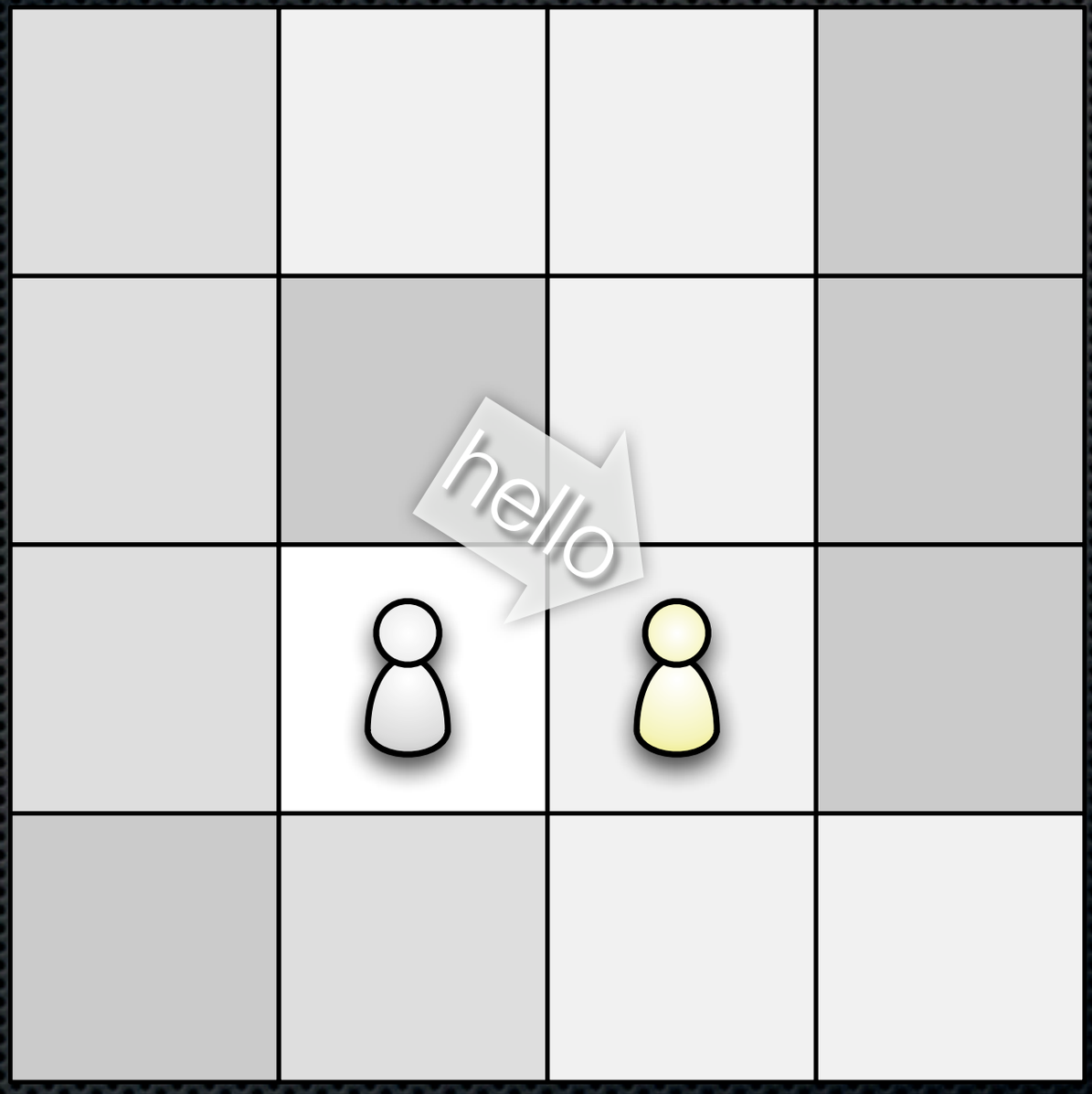
The Environment: Movement



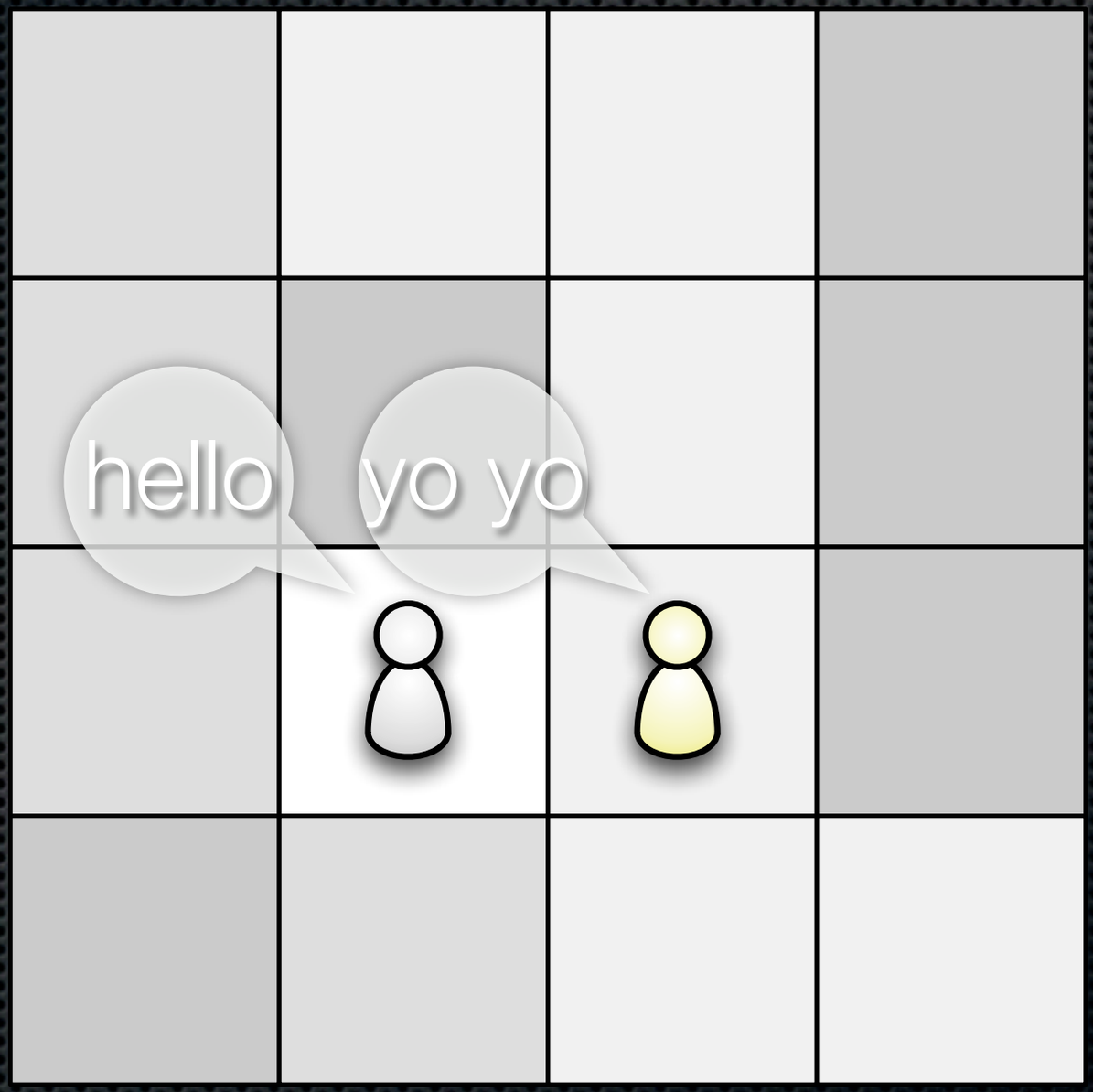
The Environment: Communication



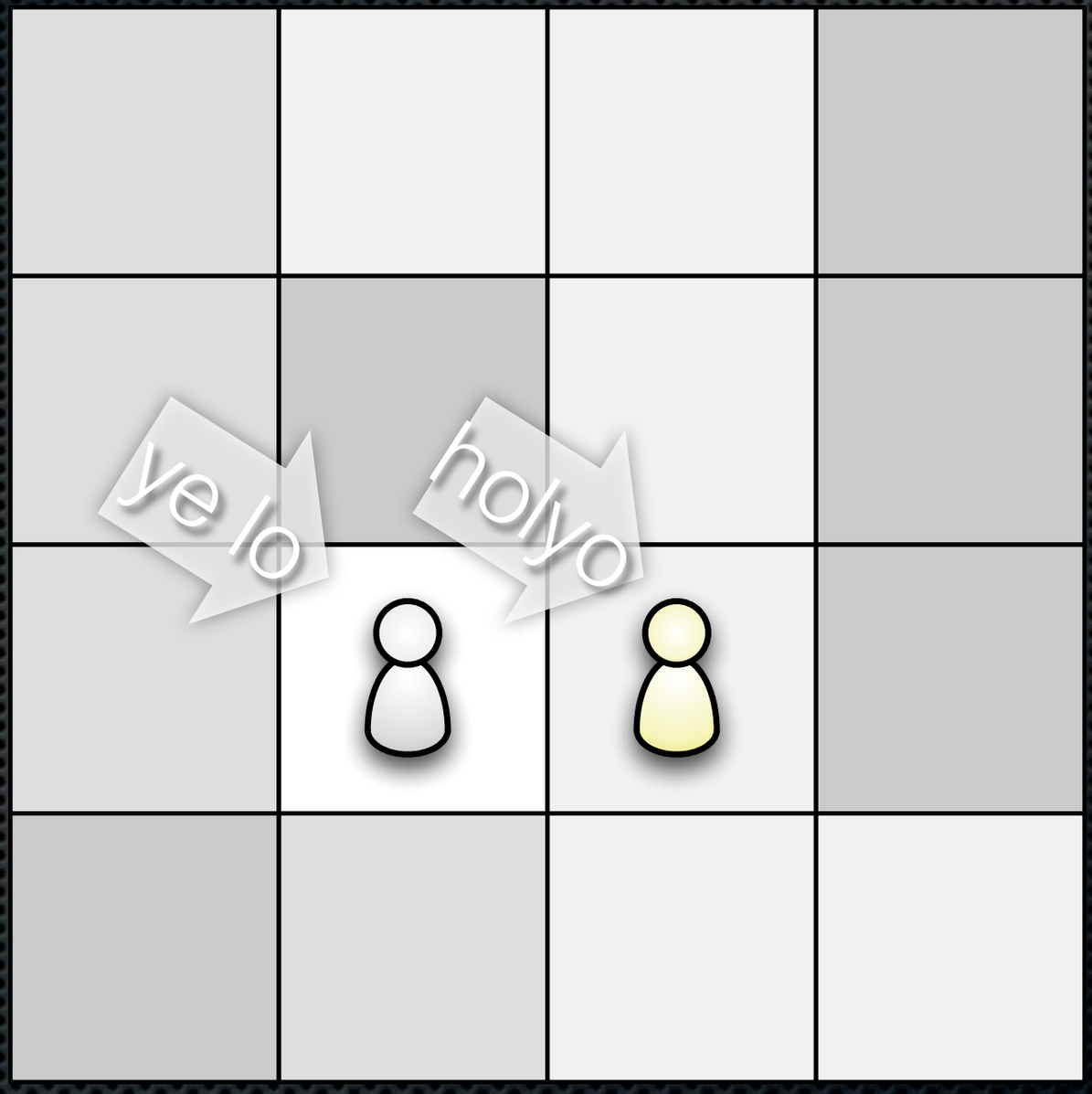
The Environment: Communication



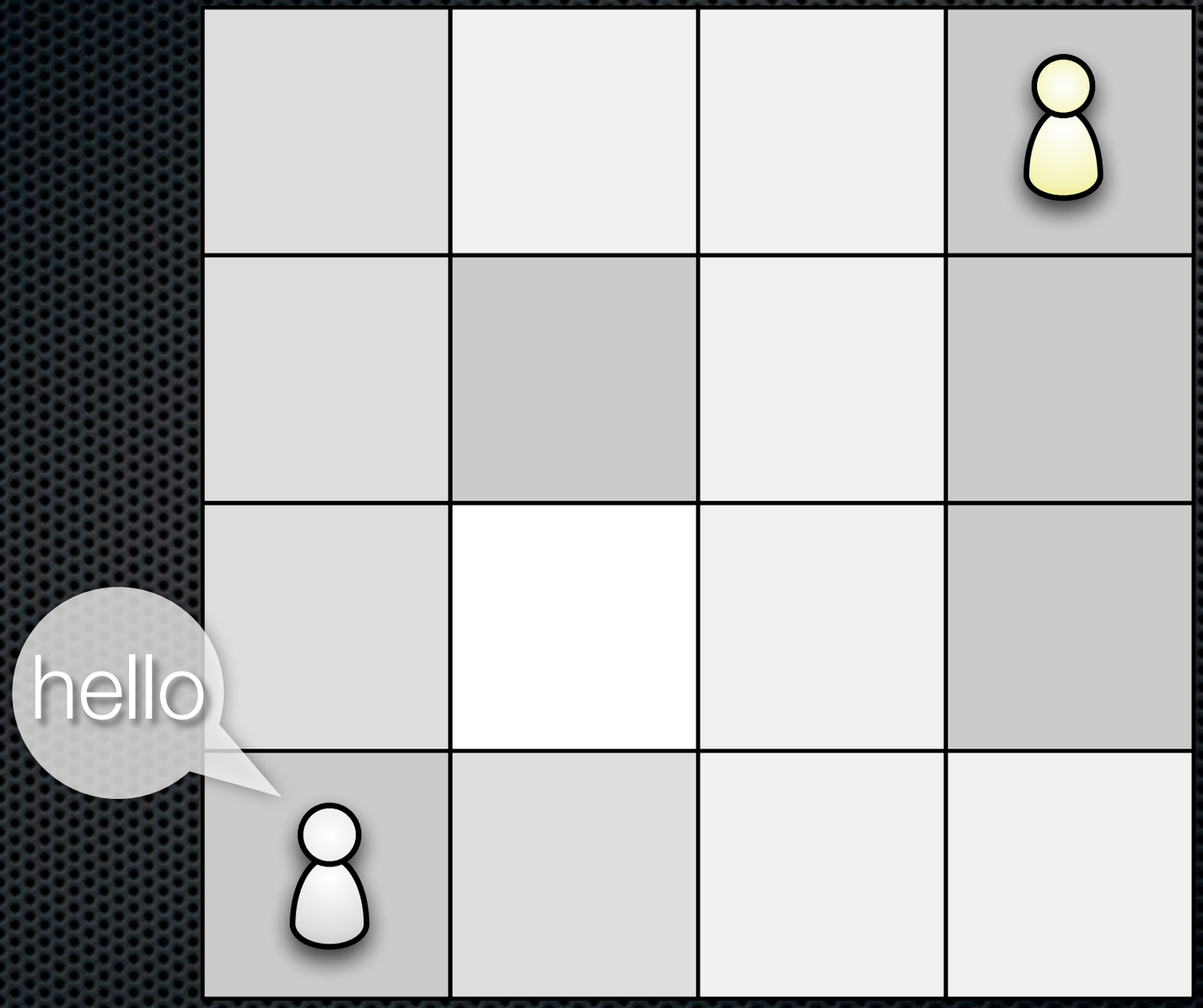
The Environment: Communication



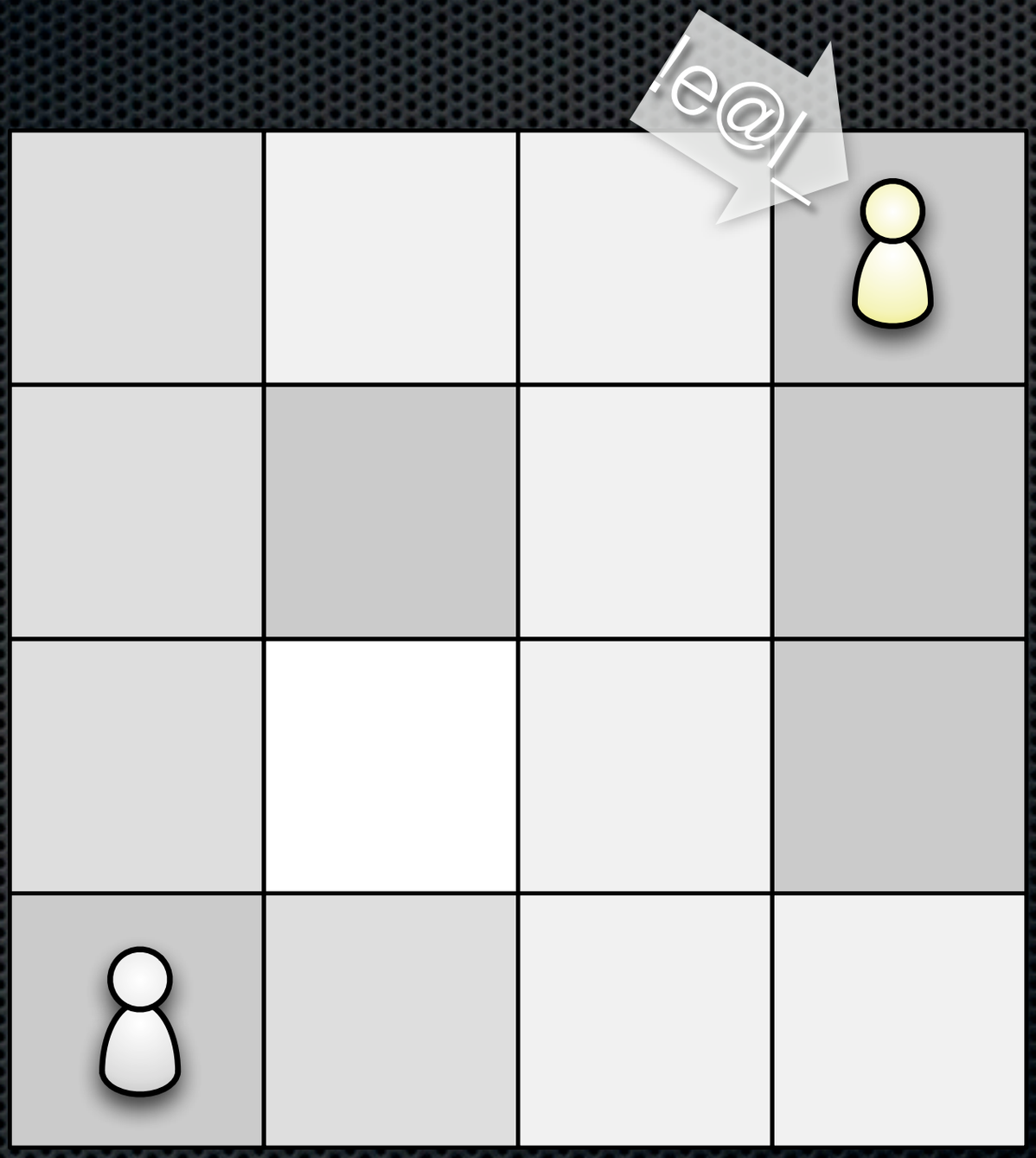
The Environment: Communication



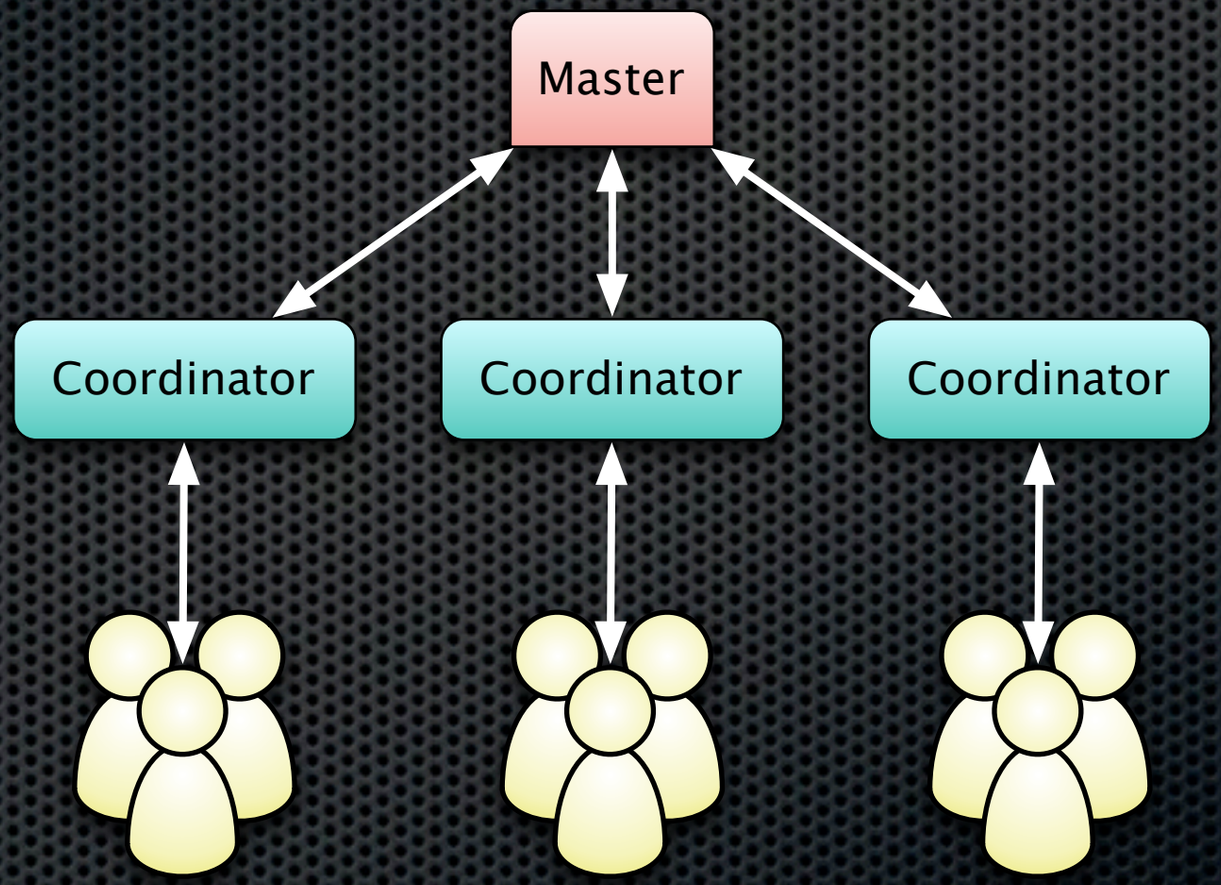
The Environment: Communication



The Environment: Communication



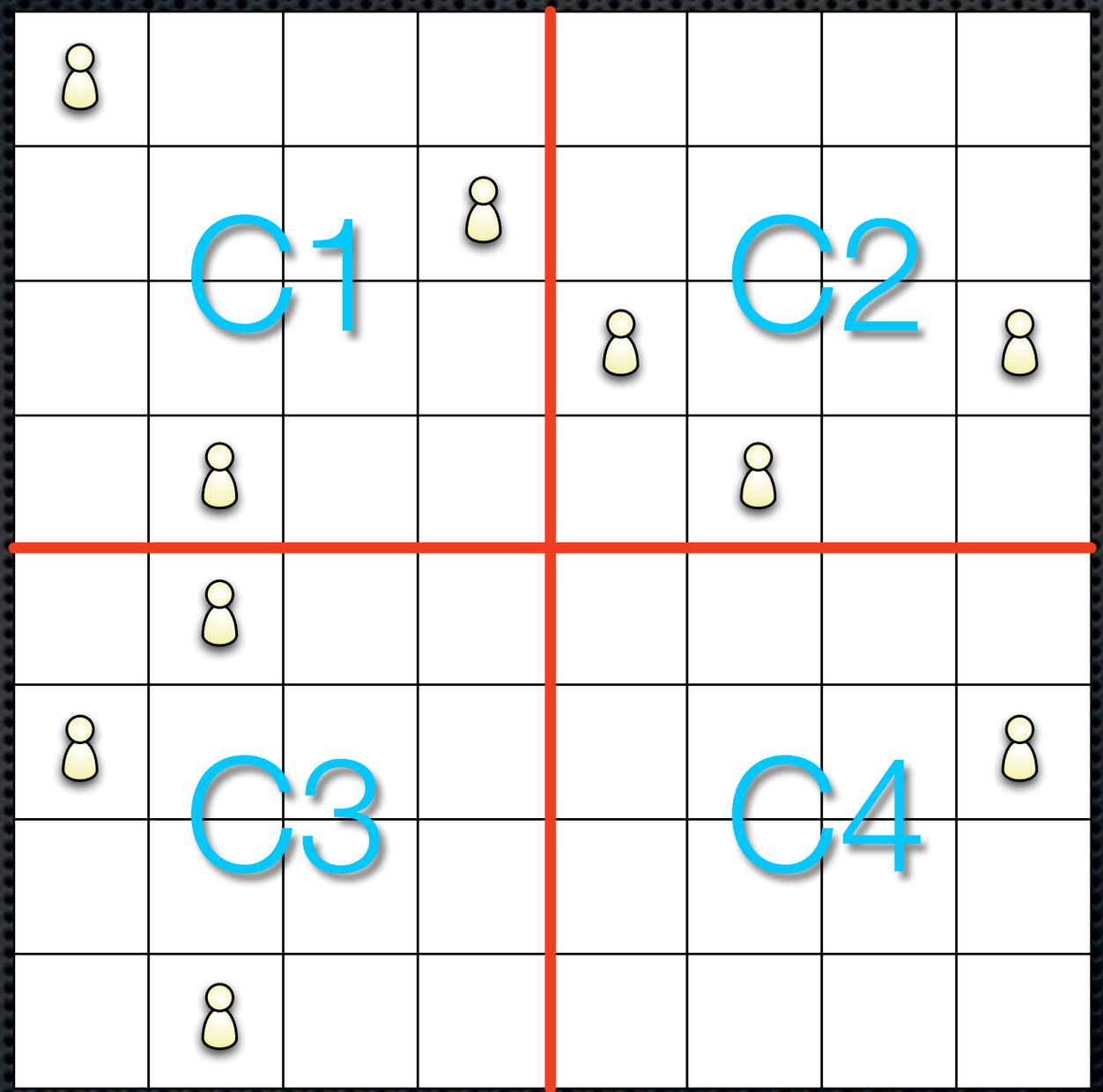
Architecture



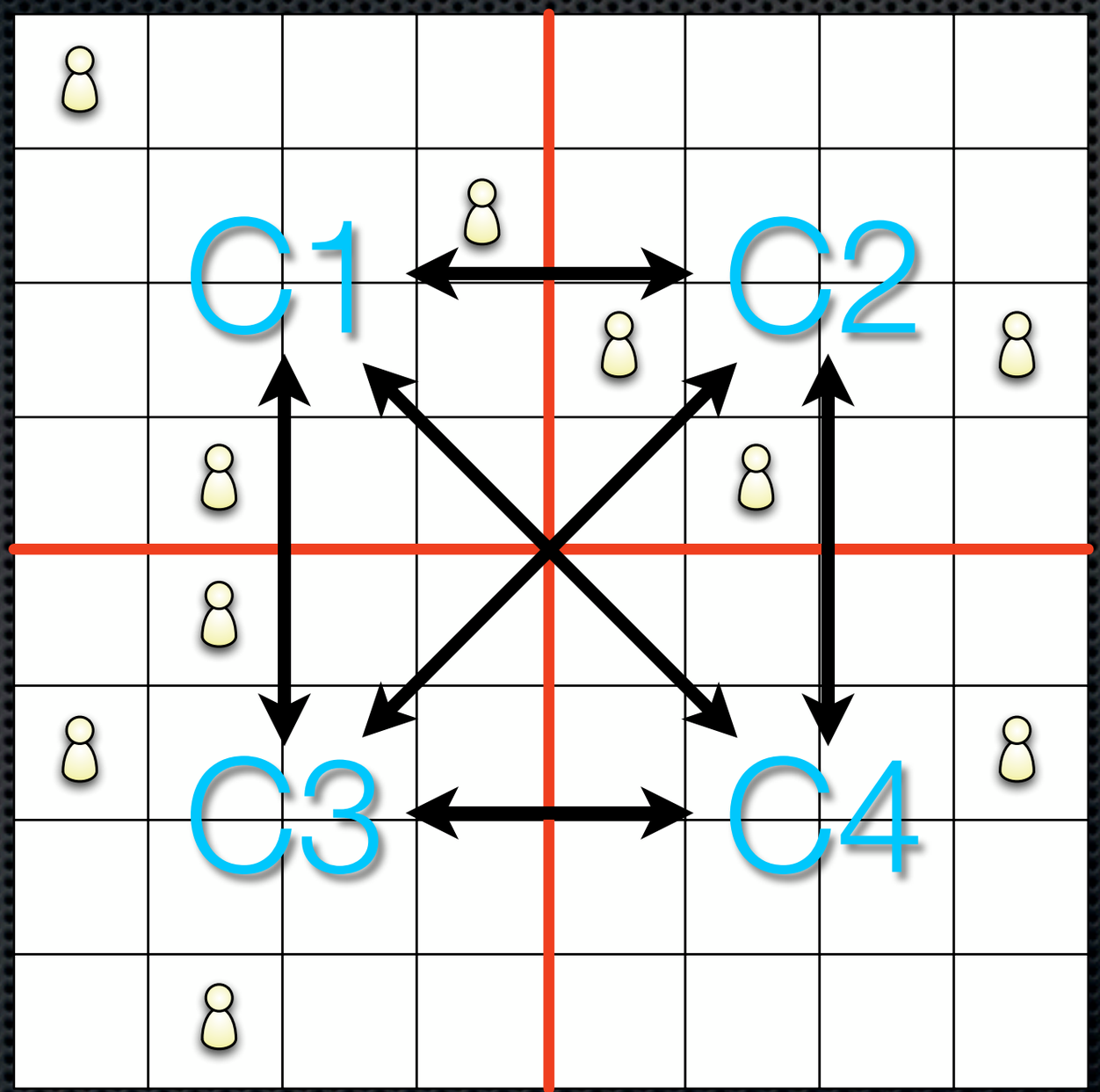
Architecture

- Master informs everyone how to connect and collects data
- Coordinators process discrete grid sections and control agents in those sections
- Agents make decisions about how to move, listen, speak, eat, etc.
- These components can be run in the same process or separate processes, communicating via TCP or memory, depending on scalability requirements

Architecture

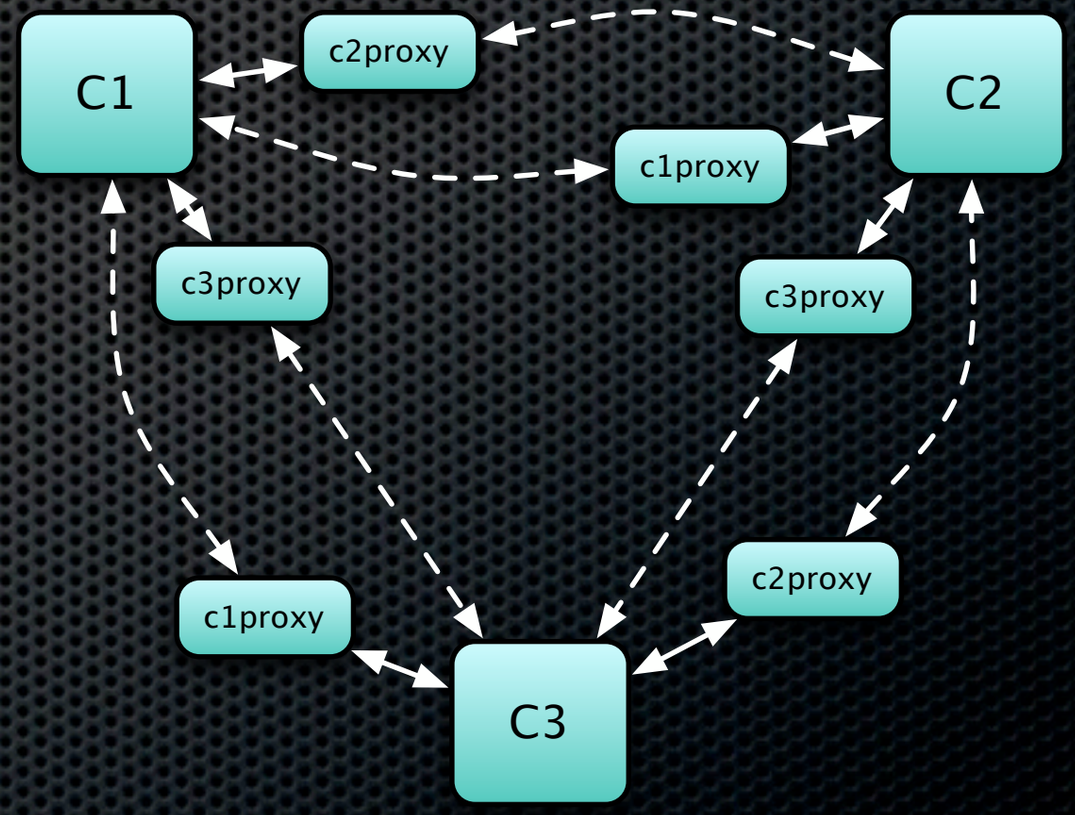


Architecture



Architecture: Coordinators

- Each turn, coordinators exchange relevant agent positions and messages
- Information used to process a single turn in lock step



Architecture: Coordinators

- Two main kinds of threads
  - Processing loop to get data from neighbors and apply rules based on agent move requests
  - RPC server to serve data requests from neighbors (one of these per neighbor for a max of 8)
  - Processing loop only advances when data requests have been served for all neighbors

Architecture: Agents

- Two threads per agent
  - RPC server at agent's coordinator to answer perception queries and accept commands
    - Move, Broadcast, Listen, etc
- One main thread for agent execution
  - Next turn, migrate to another coordinator

Architecture: Master

- Necessary if multiple processes are used
- Reads configuration file, establishes connections with all coordinator and agent processes, sends configuration, and starts simulation
- To do: report results

Agent Migration

- Coordinator A notices that an agent X has moved outside its grid area; it should belong to coordinator B
- A sends a “Migrate” message to X with B’s IP & port
- When B requests next turn’s data from A, A tells B to take over X, but A still tells its neighbors where X is
- B listens for a connection from X; when it is made, B gets X’s decision for the next turn
- A forgets about X

Issues

- Slow over TCP
- Various functions and constants need tweaking to mirror reality more accurately
- Internals are complex and require more documentation
  - ~4,900 LOC

The Future

- Noise function should not depend solely on distance
- Energy gathering not implemented
- Add more game-like qualities (just for fun)
- Algorithm/architecture optimizations
  - Fun fact: each run has ~1 + 11*C + 3*A threads

Validation: Basic IP stack

- Implemented a network stack up to IP layer as a library for agents
- Layer 1: Physical layer modeled by simulator
  - Simulating radio waves
  - Interface is sending and receiving bytes
  - Simulated corruption due to interference and distance

Validation: Basic IP stack

- Layer 2: Link Layer
  - Sending/receiving from direct neighbors
    - Packets with Command, To, and From fields (with CRC32 hash for error checking)
    - Broadcast/record “hello” to announce presence and establish direct neighbors

Validation: Basic IP stack

- Layer 3: IP Layer
  - Exchange routing tables
  - Distance vector routing
  - Send and receive messages across all reachable agents

Validation: Results

- Usable simulation for wireless mesh networks
- Error rate for any given message is ~17%
- Time to establish neighbors is ~20 turns
- Time for 8 nodes configured in a line (o-o-...-o) to exchange routing tables is ~500 turns

Validation: Future Work

- More complete, stable IP stack
- Start testing mobile networks (agents in network tests did not move)
- Implement TCP/UDP
- Implement existing link layer wireless network protocols such as 802.11, 3G, etc