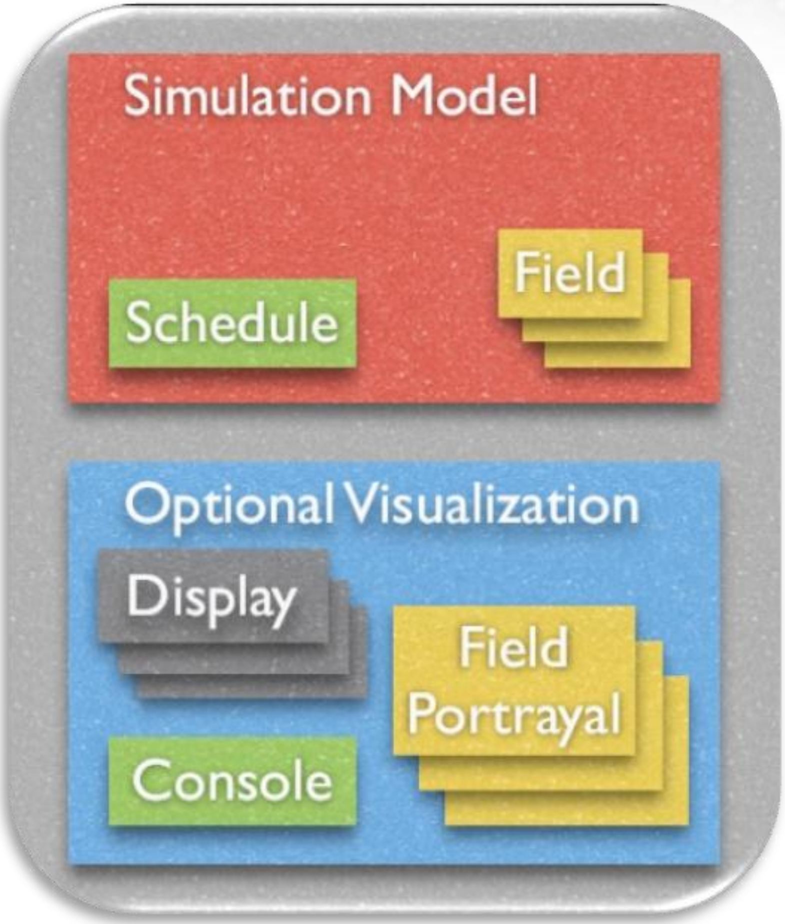
D-MASON: the state of the art

contains works by (in alphabetic order):
Michele Carillo, Gennaro Cordasco, Rosario De Chiara,
Fabio Fulgido, Ada Mancuso, Dario Mazzeo, Francesco Raia,
Vittorio Scarano, Flavio Serrapica, Carmine Spagnuolo,
Luca Vicidomini, Mario Vitale

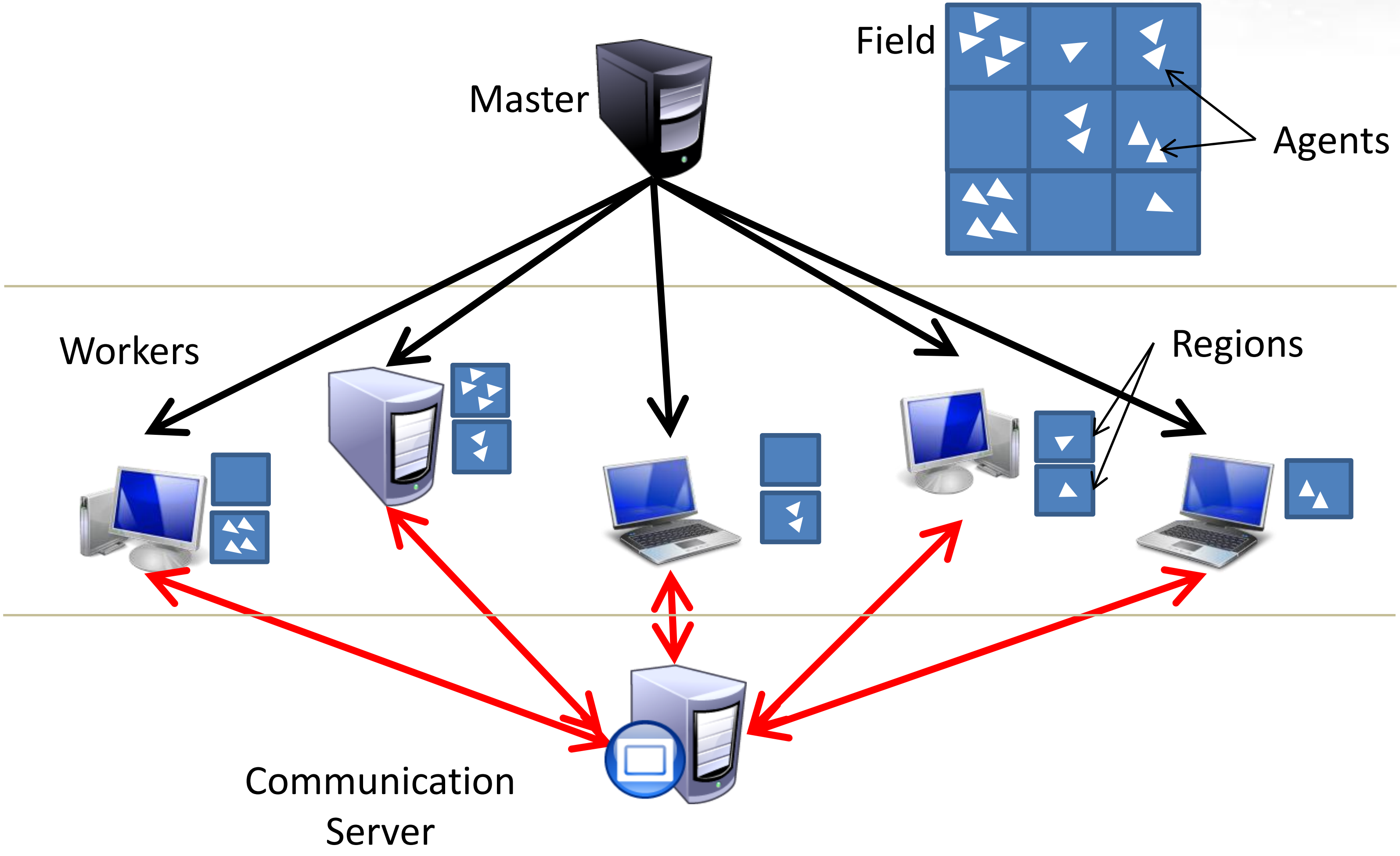
MASON AND D-MASON

Why MASON ?

- MASON is recognized to be expressive and efficient
- MASON structure: clear separation between Simulation and Visualization
- Back compatibility with simulations already present in the framework



D-MASON


Field
Master
Agents
Workers
Regions
Communication
Server

TECHNICAL ISSUES

D-MASON Issues

Work Partitioning

Synchronization

Communication

Reproducibility

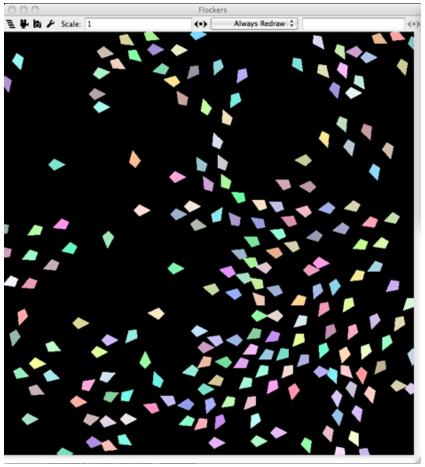
Agents vs Space Partitioning

- Agents Partitioning assigns a fixed number of agents to each available worker
  - Self balanced
  - Requires an all to all communication
- Space Partitioning partitions the simulation space into regions. Each region is assigned to a worker which is in charge of simulating all the agents belonging to the region
  - A small amount of communication is required
  - Agents can migrate
  - Load balancing is not guaranteed

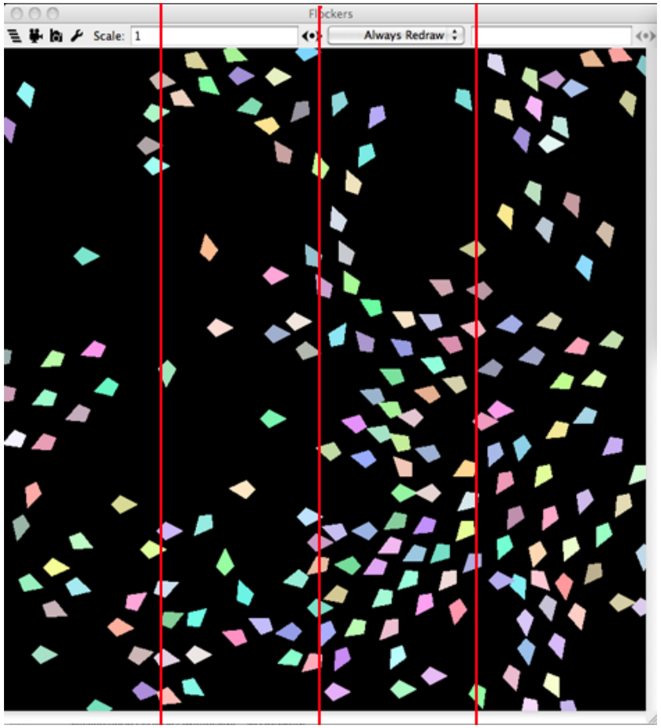
D-MASON: Space Partitioning

- D-MASON allows to partition the field into regions
- Neighboring regions communicate before each simulation step

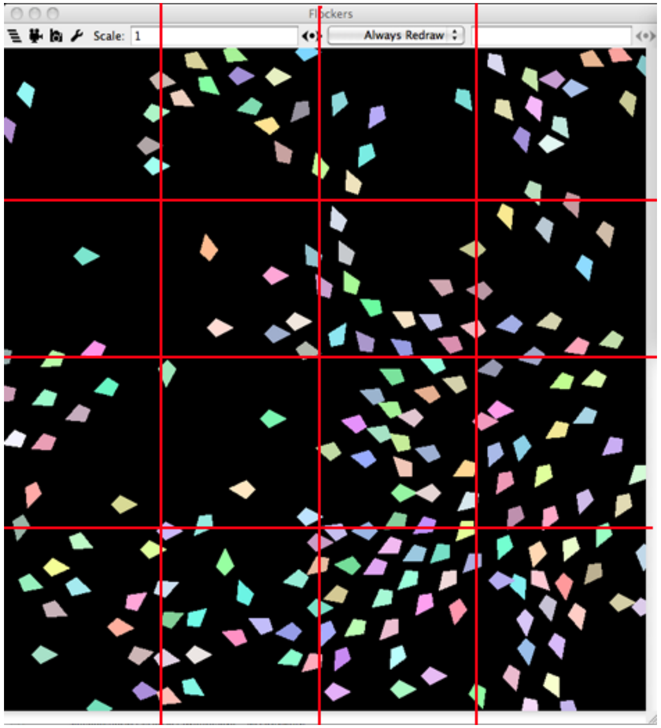
MASON



D-MASON 2DY

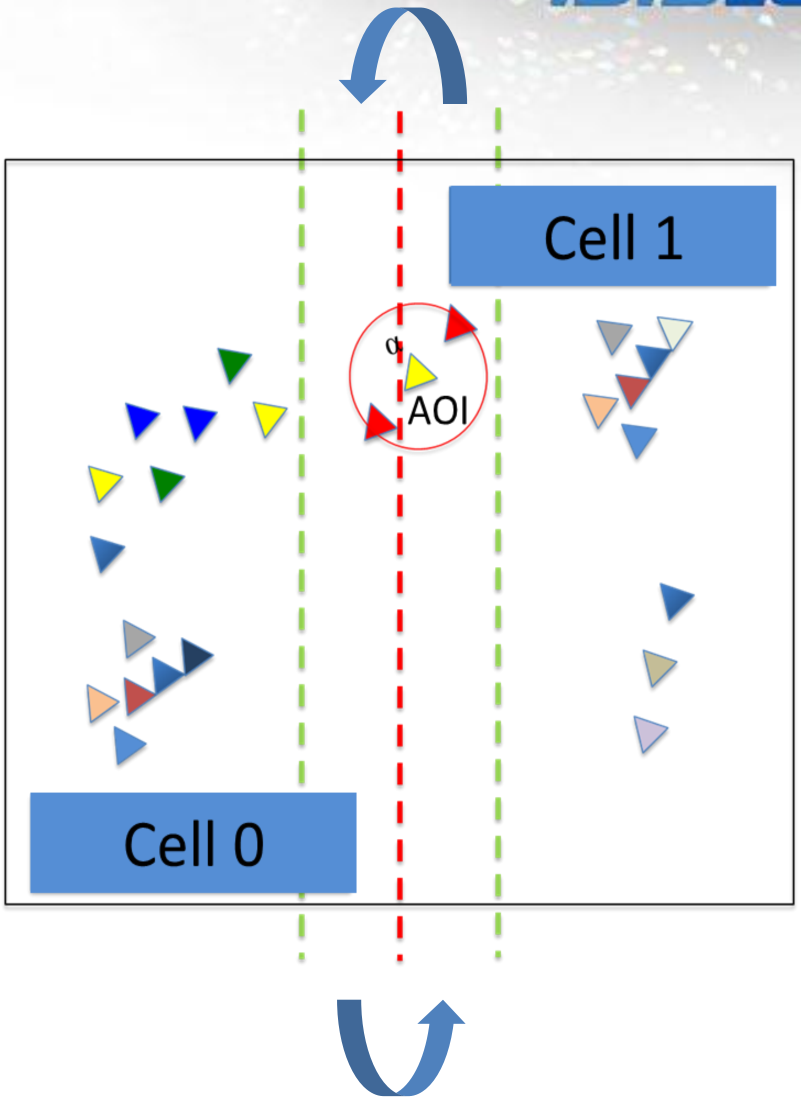


D-MASON 2DXY



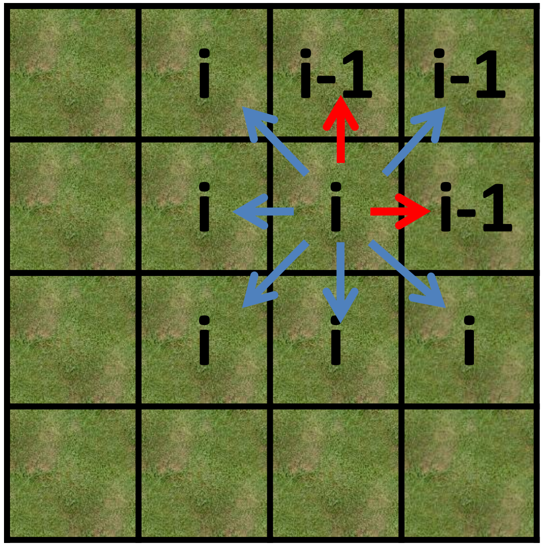
D-MASON: Field Partitioning

- A portion of each region is exchanged between neighbor workers before each simulation step
- The size of this portion depends on agents area of interest (AOI)

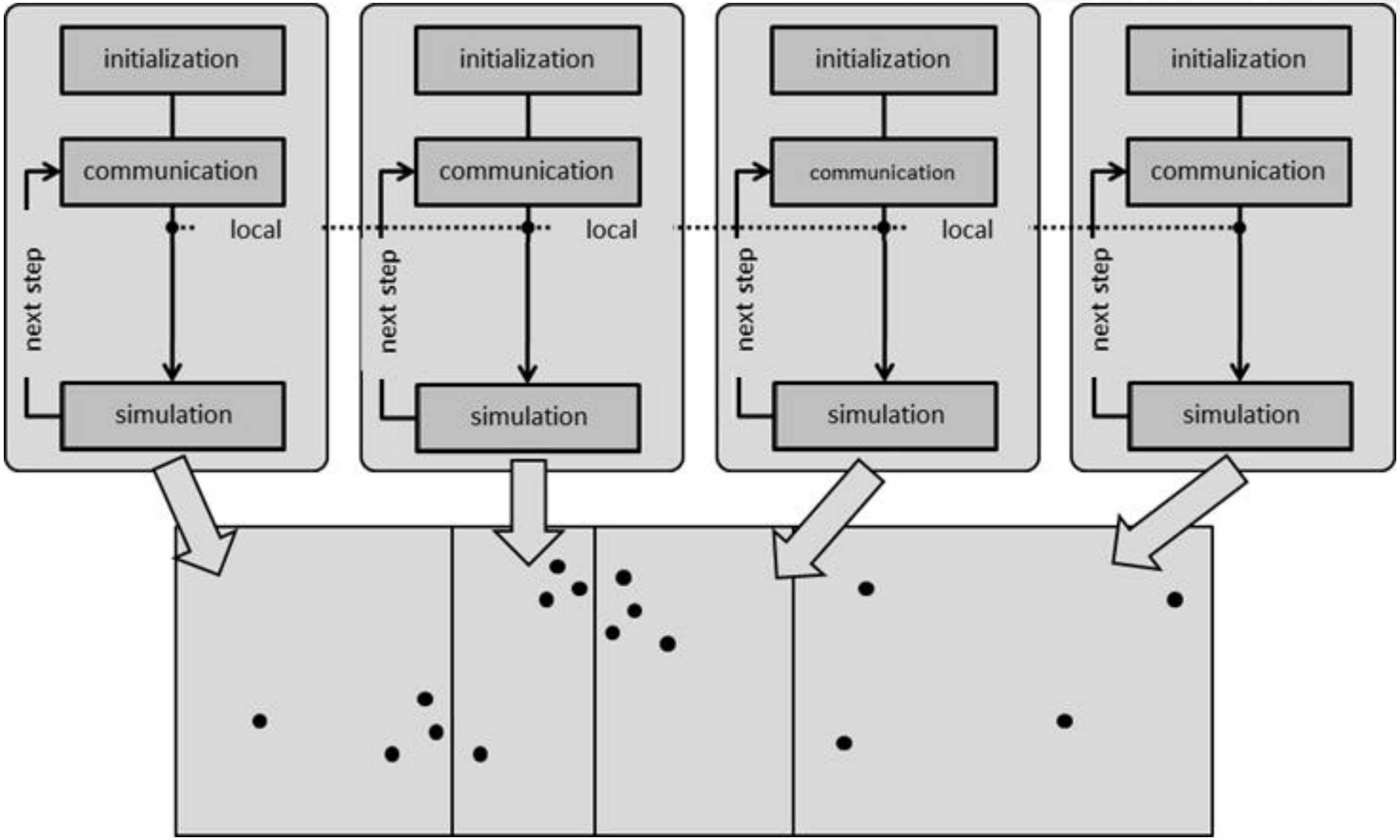


D-MASON: Synchronization

- Local synchronization:
  - The step i of region r is computed by using the states i − 1 of r’s neighborhood
  - The step i of a region cannot be executed until the states i − 1 of its neighborhood have been computed and delivered.
- No central coordinator
- Simulation speed ≈ slowest region speed

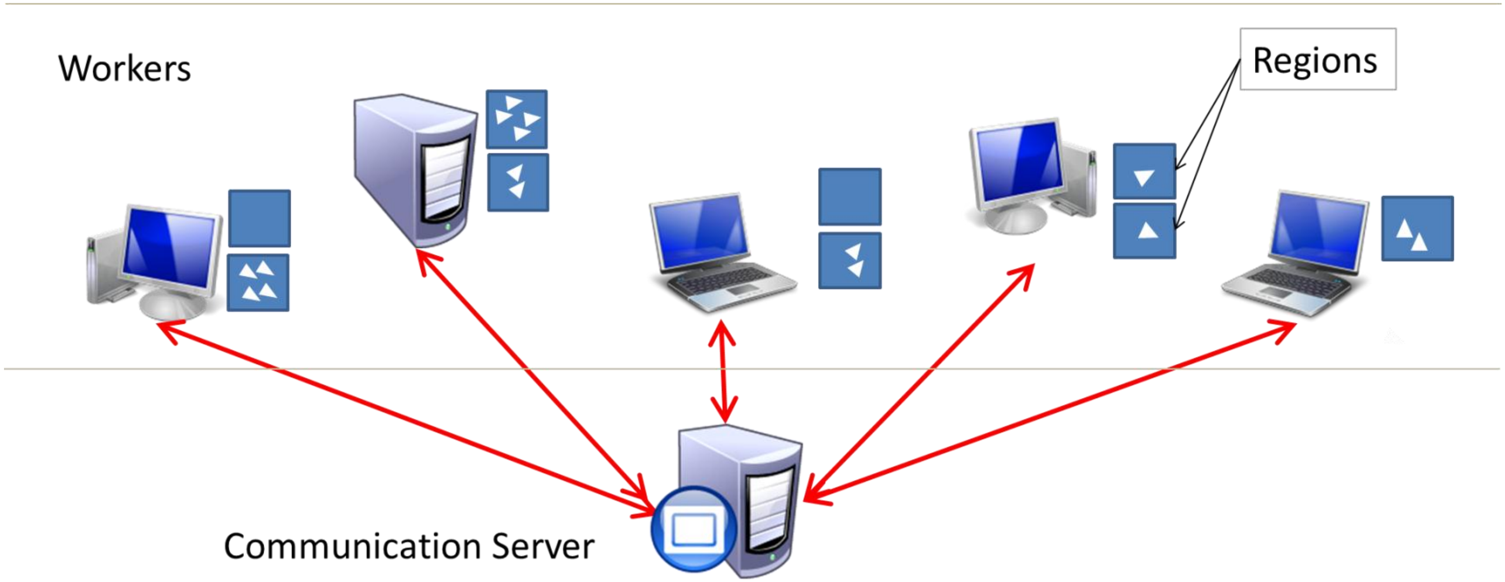


D-MASON


initialization
communication
local
next step
simulation
initialization
communication
local
next step
simulation
initialization
communication
local
next step
simulation
initialization
communication
next step
simulation

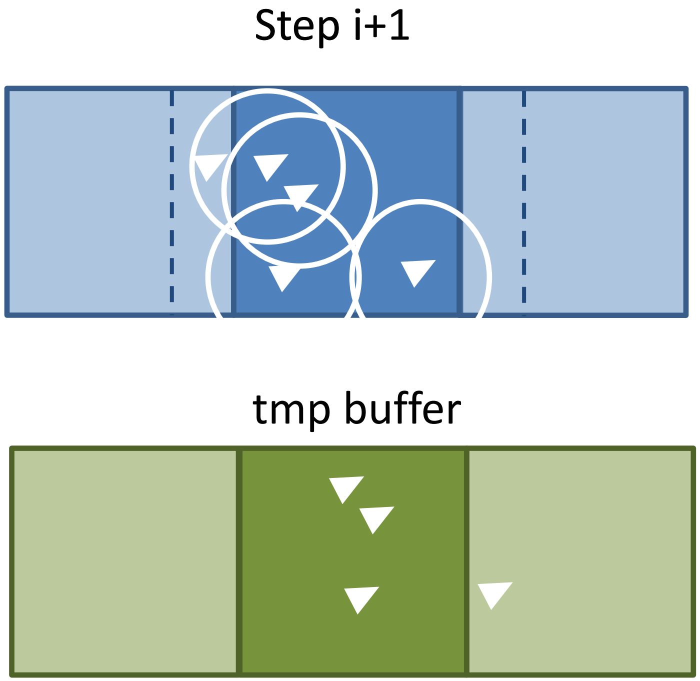
D-MASON: Communication

- D-MASON uses the publish–subscribe design pattern to propagate agents state information
  - Current version of D-MASON uses Java Message Service (JMS) for communication between workers



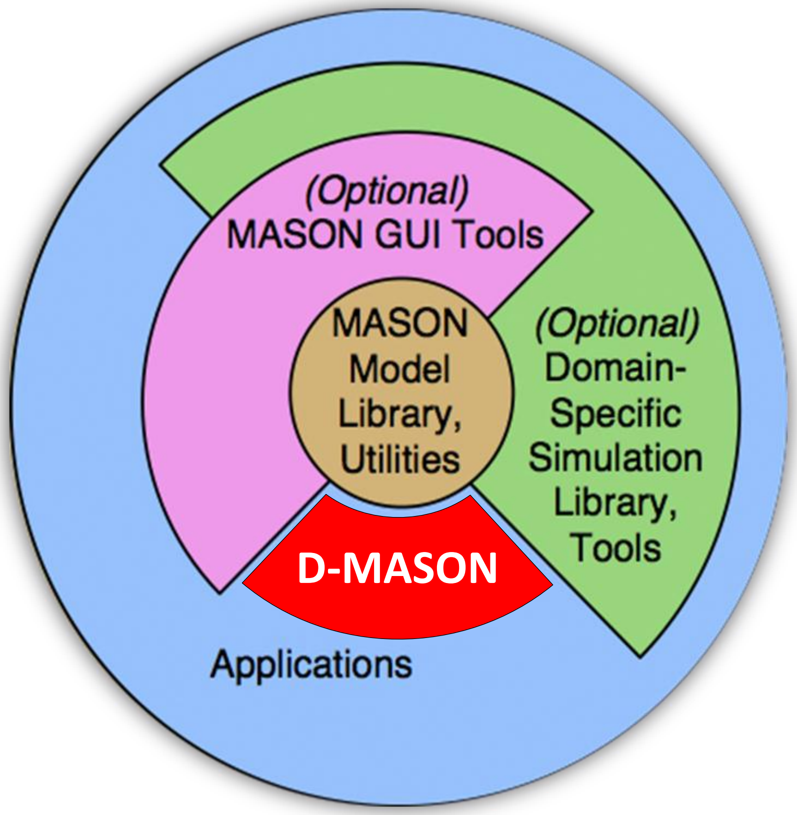
D-MASON: Reproducibility

- Agents evolve simultaneously
  - each simulation step can be executed in parallel overall the agents
  - the order in which agents are scheduled does not affect the reproducibility of results
  - Neighbors’ updates are always executed in the same order



ARCHITECTURE

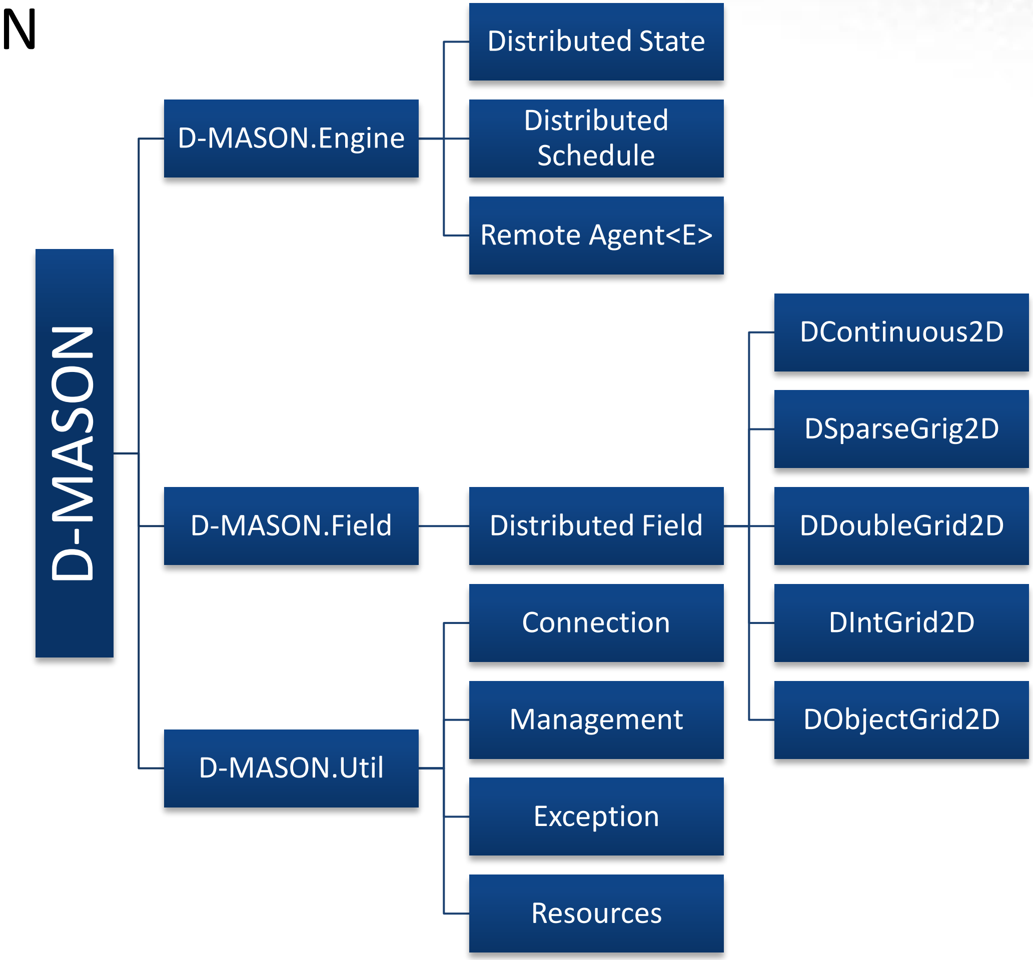
D-MASON Architecture



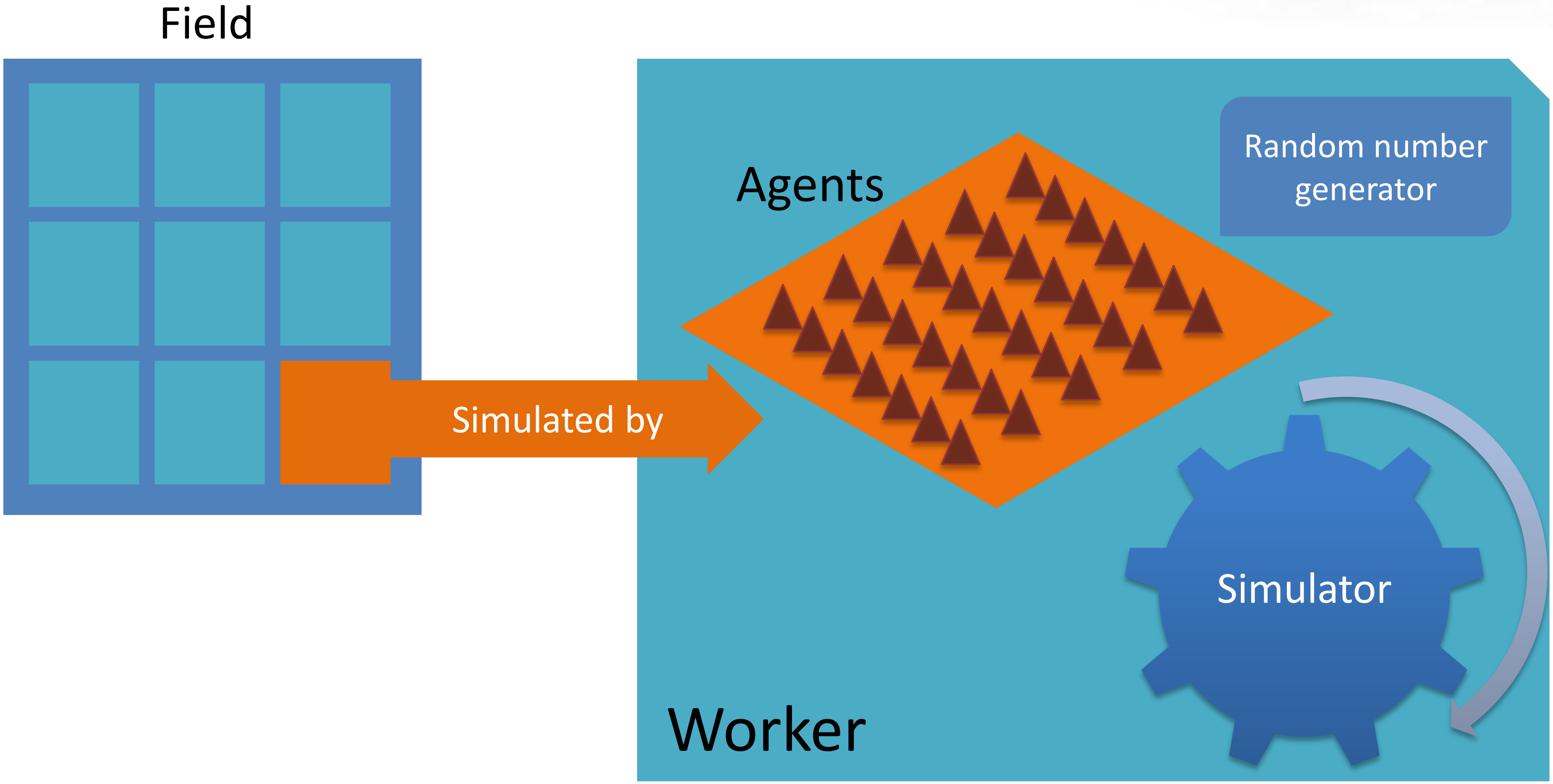
- D-MASON is a new layer which extends the MASON simulation layer.
- The new layer does not alter in any way existing layers

D-MASON Architecture

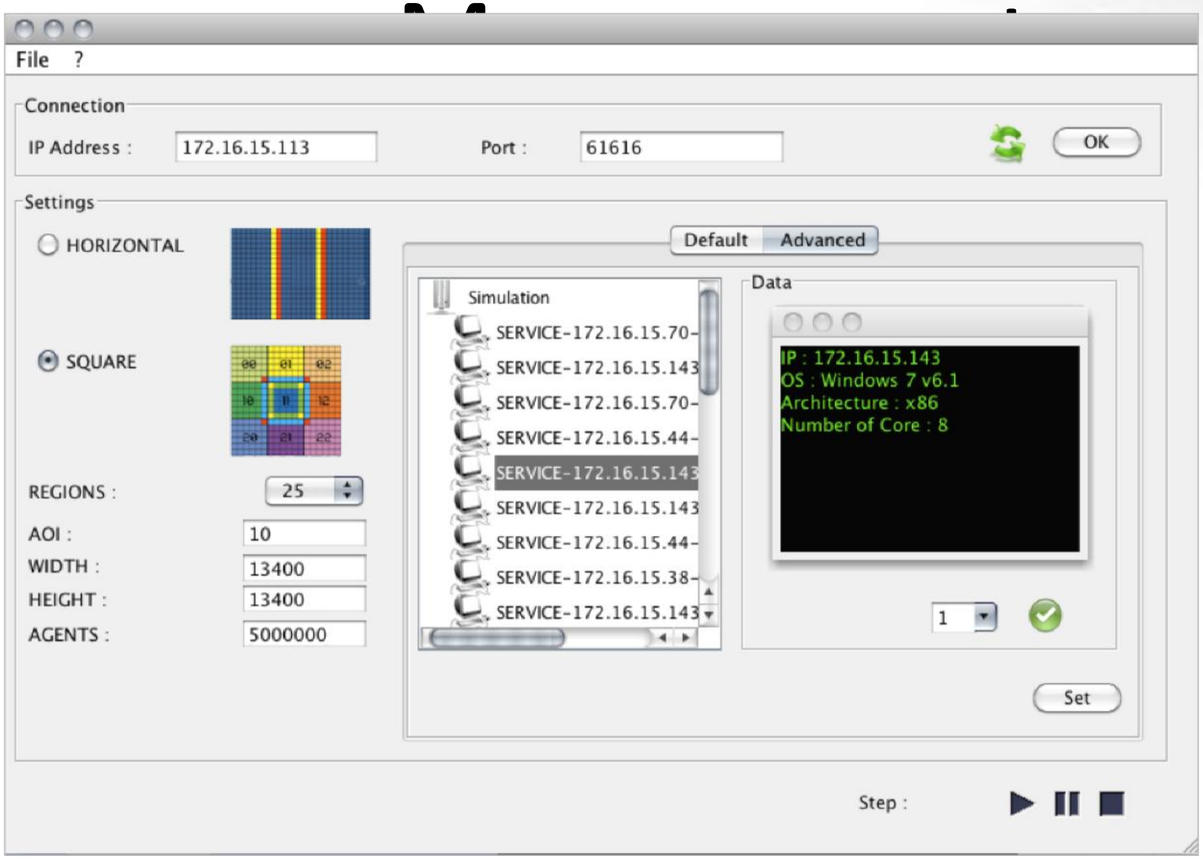
The same structure as MASON



D-MASON Worker



D-MASON: System



Connection

IP Address : 172.16.15.113

Port : 61616

Settings

○ HORIZONTAL

● SQUARE

REGIONS : 25

AOI : 10

WIDTH : 13400

HEIGHT : 13400

AGENTS : 5000000

Simulation
- SERVICE-172.16.15.70-
- SERVICE-172.16.15.143
- SERVICE-172.16.15.70-
- SERVICE-172.16.15.44-
- SERVICE-172.16.15.143
- SERVICE-172.16.15.143
- SERVICE-172.16.15.44-
- SERVICE-172.16.15.38-
- SERVICE-172.16.15.143

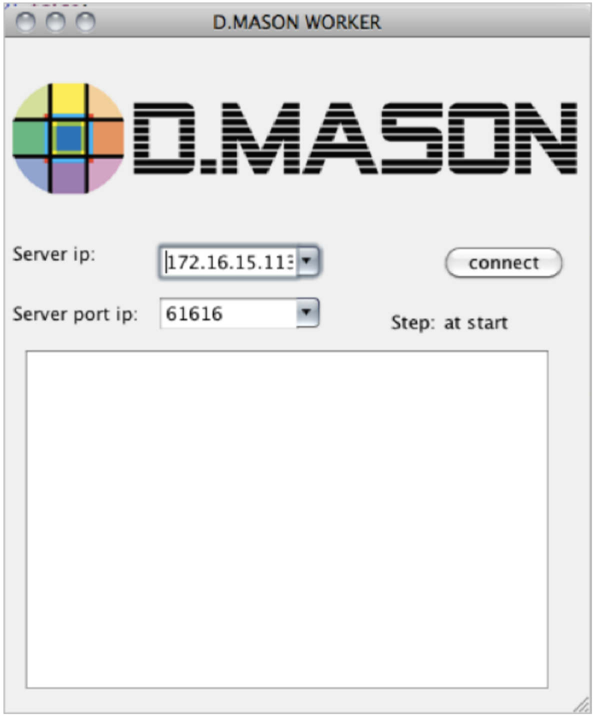
Data

```
IP : 172.16.15.143
OS : Windows 7 v6.1
Architecture : x86
Number of Core : 8
```

1

Master Console

Worker



Server ip: 172.16.15.113

Server port ip: 61616

Step: at start

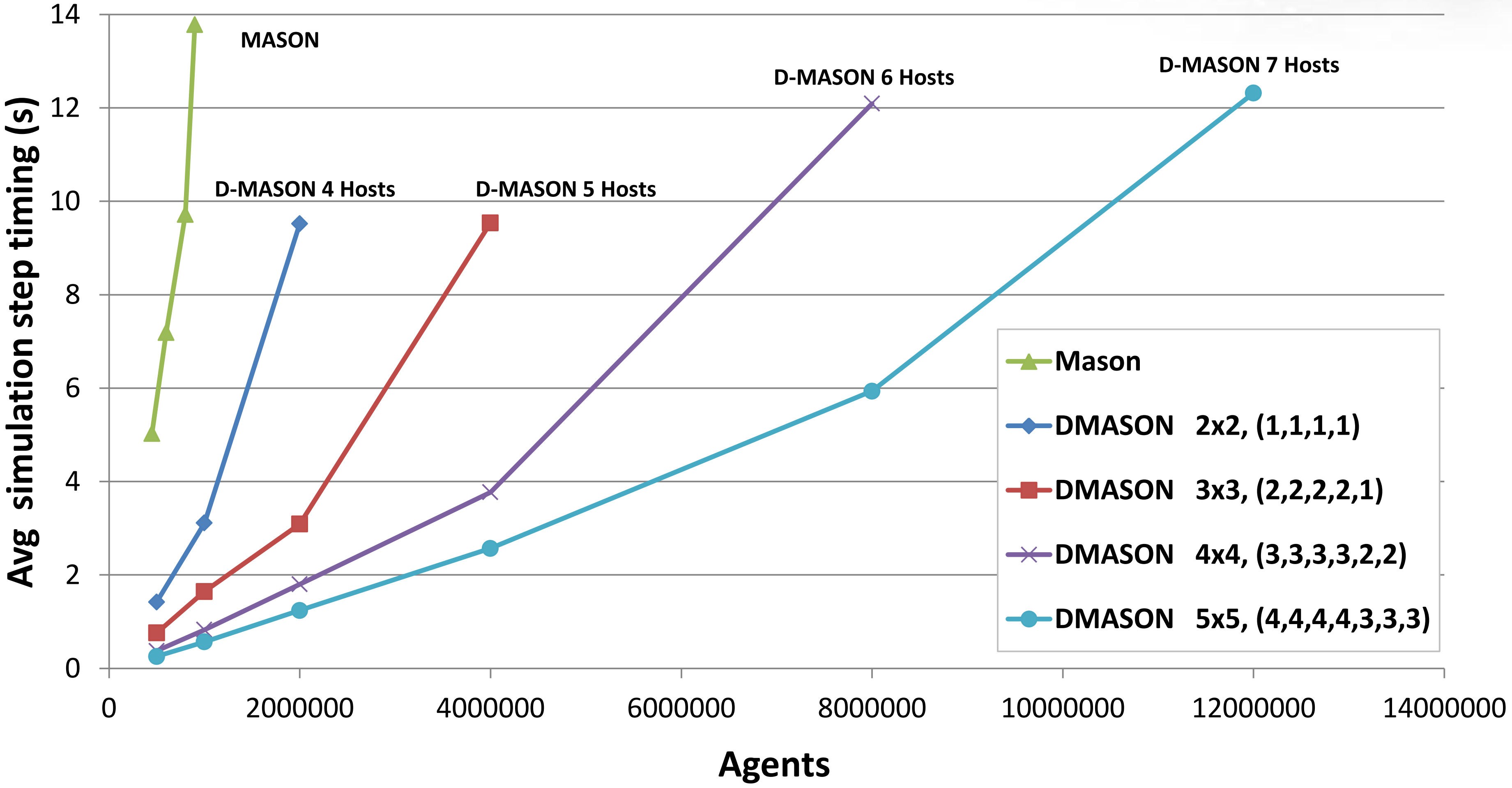
PERFORMANCES

Testing D-MASON

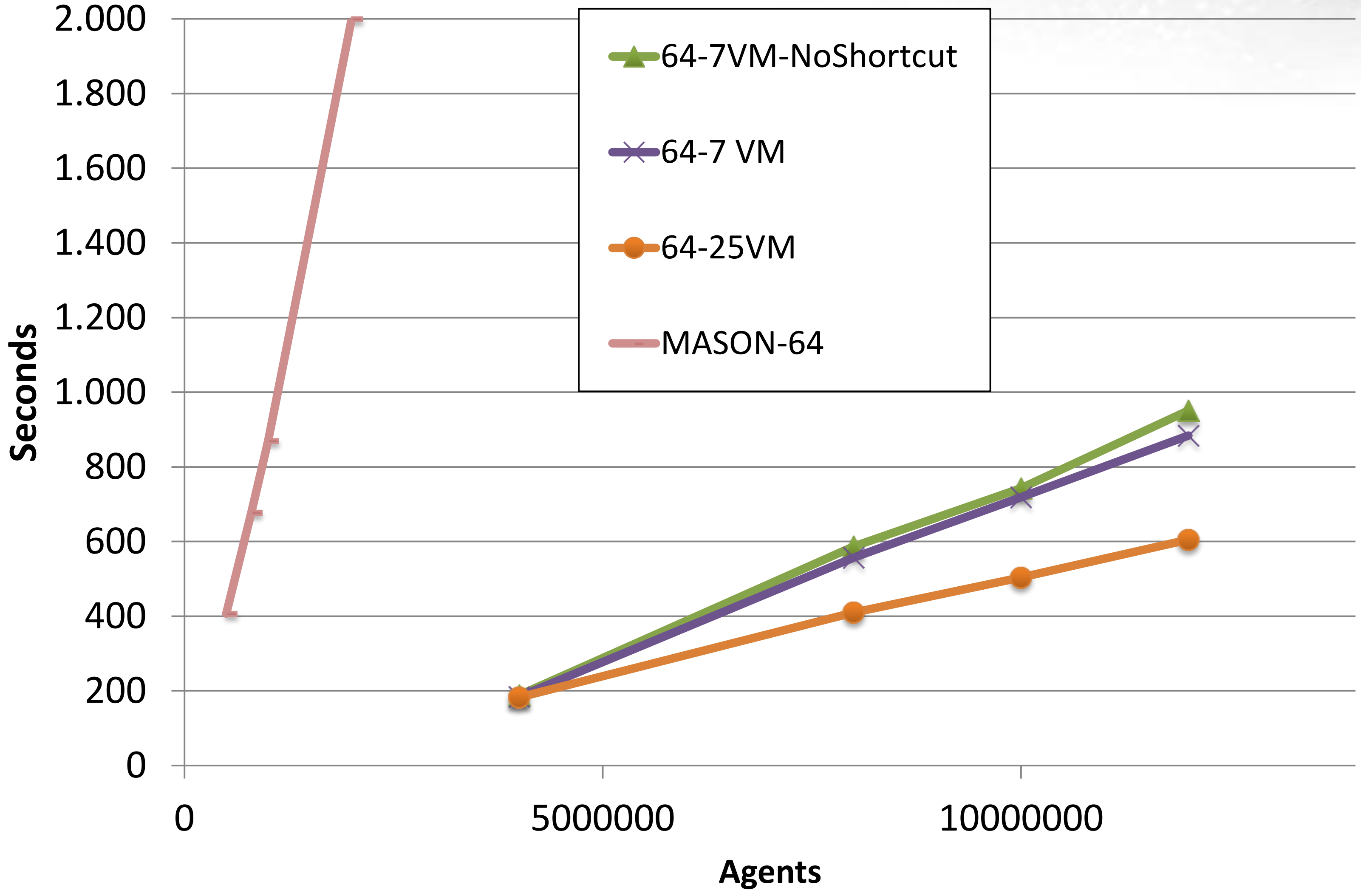
- We want to assess that D-MASON…
  - …is able to run simulations that are impractical or impossible to execute with MASON
  - …is scalable
  - …allows to develop reproducible simulations (independently from the number of workers)
  - …allows to exploit multicore CPU
  - …allows to exploit heterogeneous hardware

D-MASON: homogeneous hosts

7 HOST : Intel i7-2600 4x3,4GHz con HT 8GB RAM JVM 32 bit



D-MASON: Test


64-7VM-NoShortcut
64-7 VM
64-25VM
MASON-64
Seconds
2.000
1.800
1.600
1.400
1.200
1.000
800
600
400
200
0
0
5000000
10000000
Agents

CURRENT WORK

Three areas

Agents features

Management

Load balancing

Agents features

- Finding what is really needed is tricky
- We adopt «scenario driven» approach
  - We parallelize a new simulation and find out what is still missed in D-MASON
- Because of the distributed nature of D-MASON many MASON common mechanisms have to be re-designed
  - E.g. the teleport, calculating the number of agents currently alive etc…

Agents features

Examples of functionalities that we want to offer at framework level:

- Teleport
  - Enables agents to modify simulation fields outside agents’ area-of-interest
- Inter-region communication
  - Allows sharing information among agents located in different regions

Agents features

Examples of functionalities that we want to offer at framework level (continued):

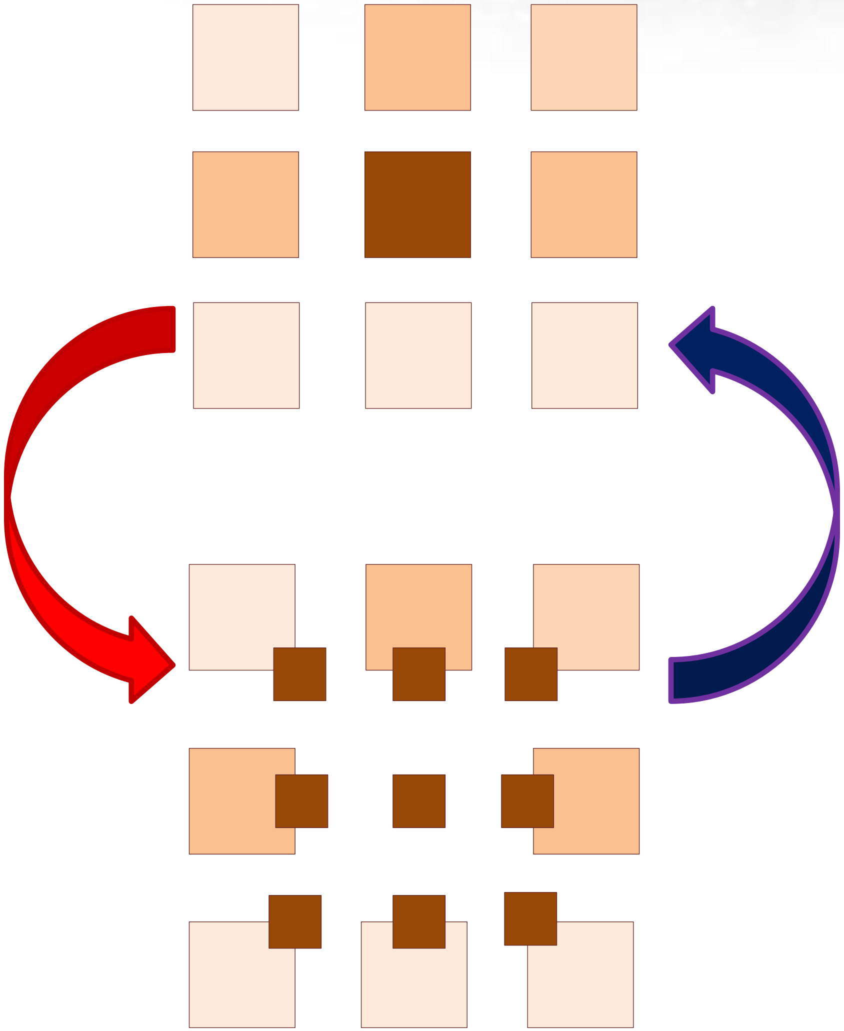
- Agents that can breed and die
  - This is complex because of swapping agents between workers
- Calculate the average age of agents
  - Display it in the Global Inspector

Load Balancing

- Integration of 2DXY Load Balancing in version 1.5 of D-MASON
  - 2DXY: Field 2D and division of work as a grid
- Implementation of Load Balancing on 2DY
  - 2DY: Field 2D and division of work cutting the field on the X

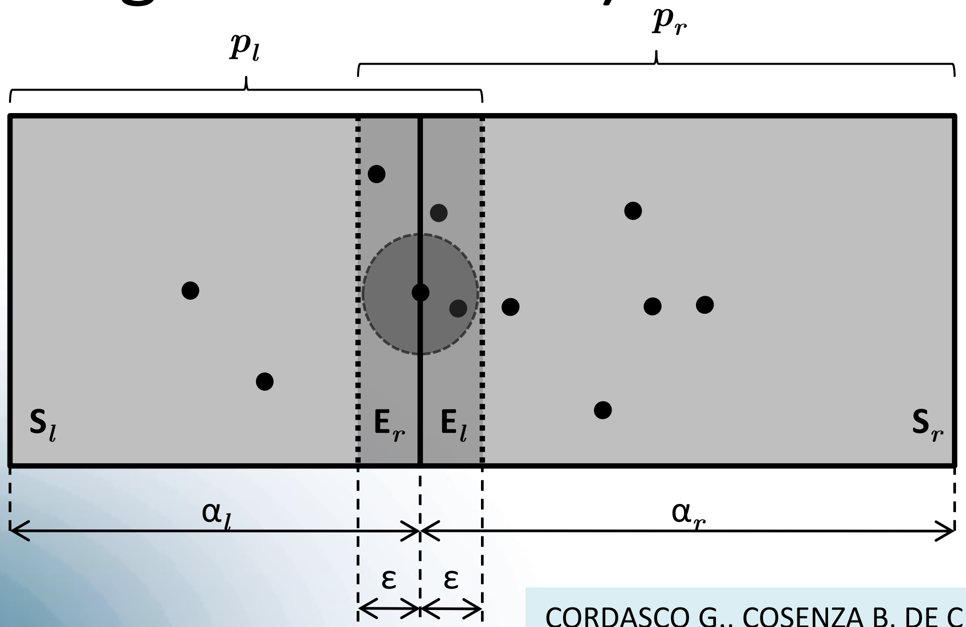
Load Balancing 2DXY

- 2DXY the field is partitioned in a grid (axis aligned squares partitioning of a 2D field)
  - Example: the worker at the center of the map became overloaded
  - The load balancing splits it into nine pieces that are assigned to nighbors
  - Once the situation cools down the pieces merge at the original worker



Load Balancing 2DY

- We intend to implement a Load Balancing mechanism for the 2DY case (Y-axis aligned partitioning of a 2D field)



CORDASCO G., COSENZA B, DE CHIARA R., SCARANO V Distributed Load Balancing for Parallel Agent-based Simulations. In Proc. of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2011). Ayia Napa, Cyprus, February 09 - 11, 2011.

Load balancing

System Management

- Peer auto reconnection
  - Automatic reconnection when the communication server (CS) is restarted after a failure
- Restart simulation
  - Restart simulation without restart the CS (e.g. launch another simulation)
- Batch test
  - Simulation parameters going to be a range with an increment step and then will be generated a series of test dependent on the combination of this parameters
- Deploy simulation
  - Switch from hard-coded simulation to a jar simulation deployed to workers

D-MASON Current Works

- Global Inspector, Charts, Real-time modification of the parameters
  - The Global Inspector displays simulation-related information, allows for chart plotting and parameters modification

FINE