Multiagent Simulation on StarBED

Deploying Agent-based Traffic Simulation on StarBED

Rafik HADFI <rafik@itolab.nitech.ac.jp>
Nagoya Institute of Technology, Japan

October 20th, 2015

Summary

1. Motivations
2. Architecture
3. The “agent” paradigm
4. MAS on StarBED stack
5. Traffic simulation

Motivations

- Agent-based, Large-scale, Distributed Simulation of Complex Systems
- Case: traffic simulation
- Simulation of complex behaviors using an agent-based model. These behaviors include:
  - Mobility
  - Driving
  - Traffic synchronization
  - Sensor networks

Motivations

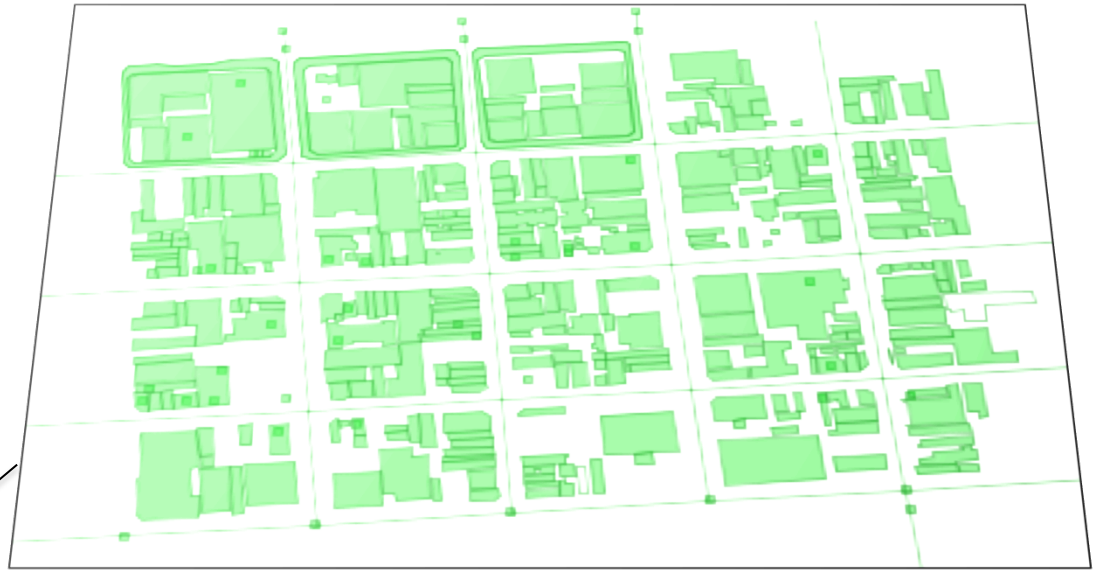
- A generic framework for programmable agent-based components
  - Multilayered, with different levels of abstractions referring to different simulation layers
  - Reusable API for both distributed and non-distributed simulations
  - Generation of realistic simulation data

- A testbed for general purpose Computational Intelligence
  - coordination
  - collaboration
  - automated negotiation
  - etc.

Architecture

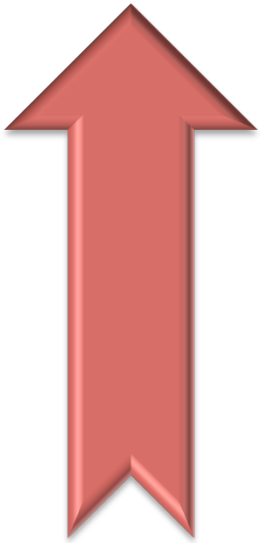
- Multilayered architecture built on a map

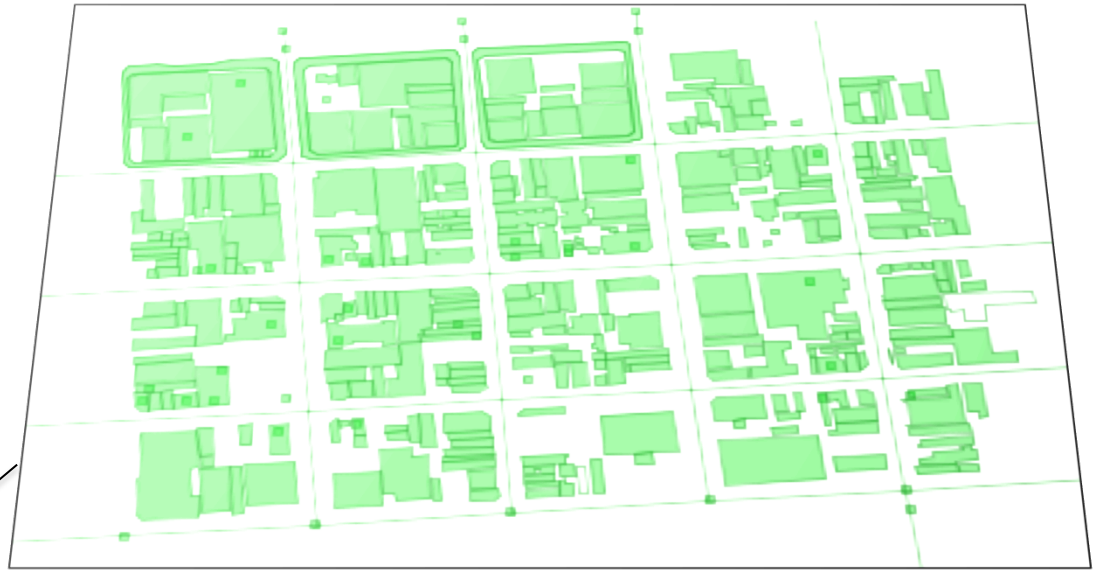
Architecture



Geographical Layer
OSM

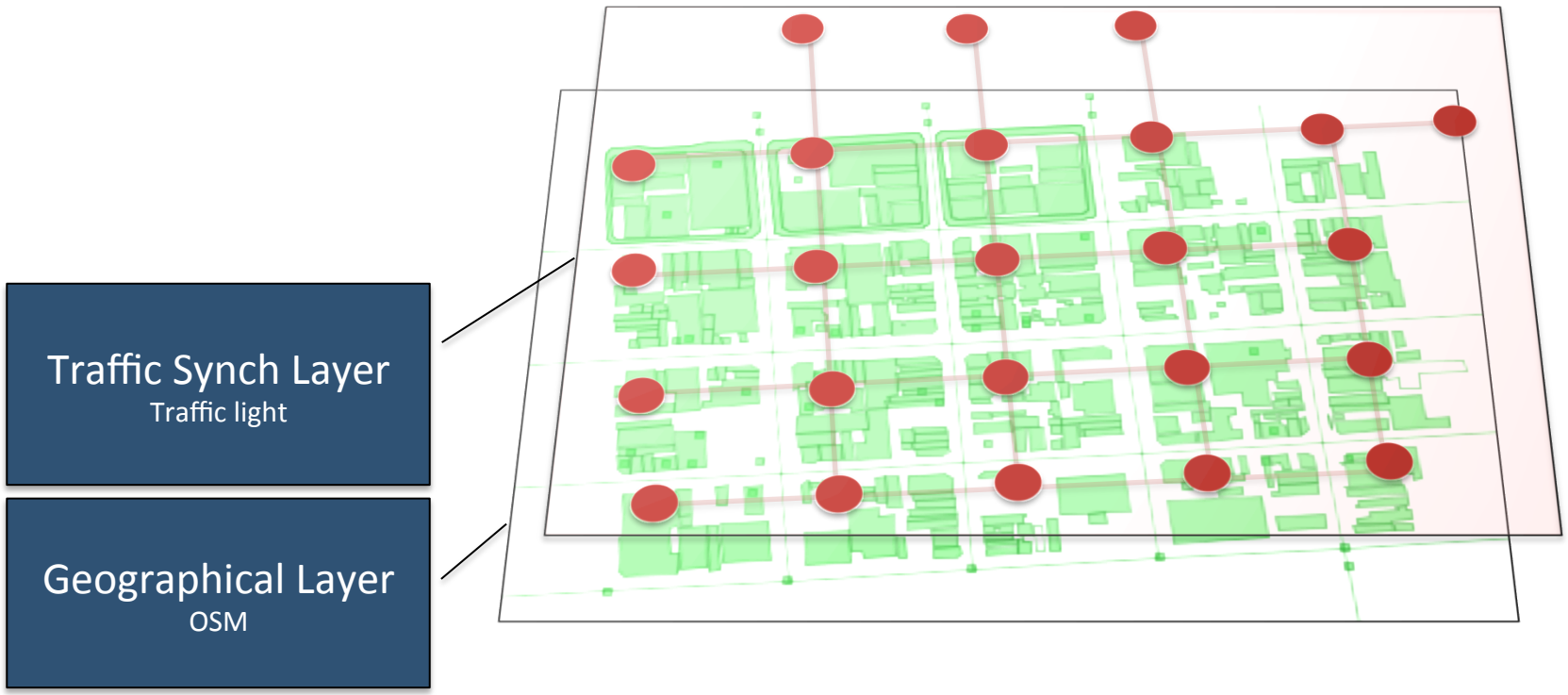
Architecture



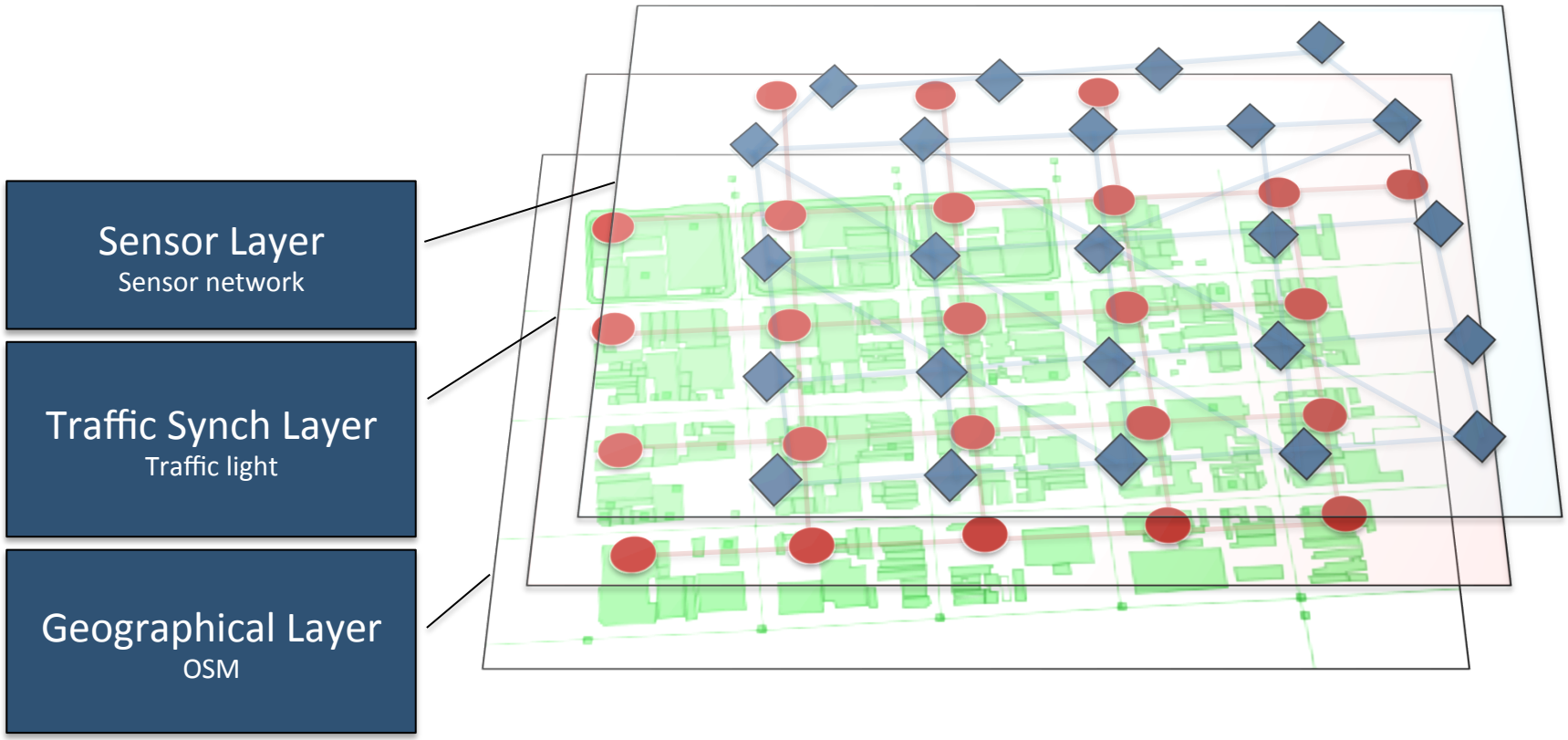


Geographical Layer
OSM

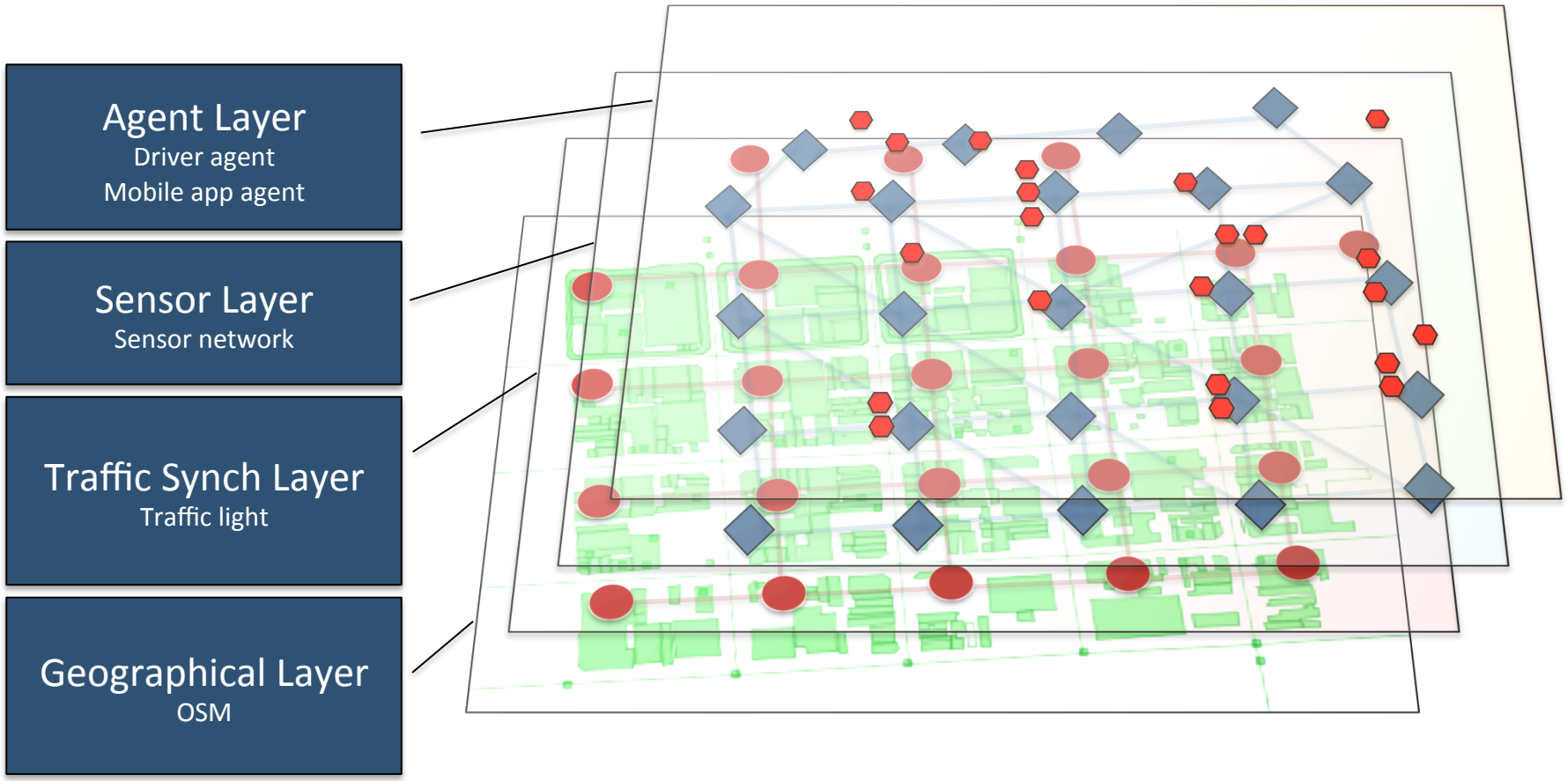
Architecture



Architecture



Architecture



Architecture

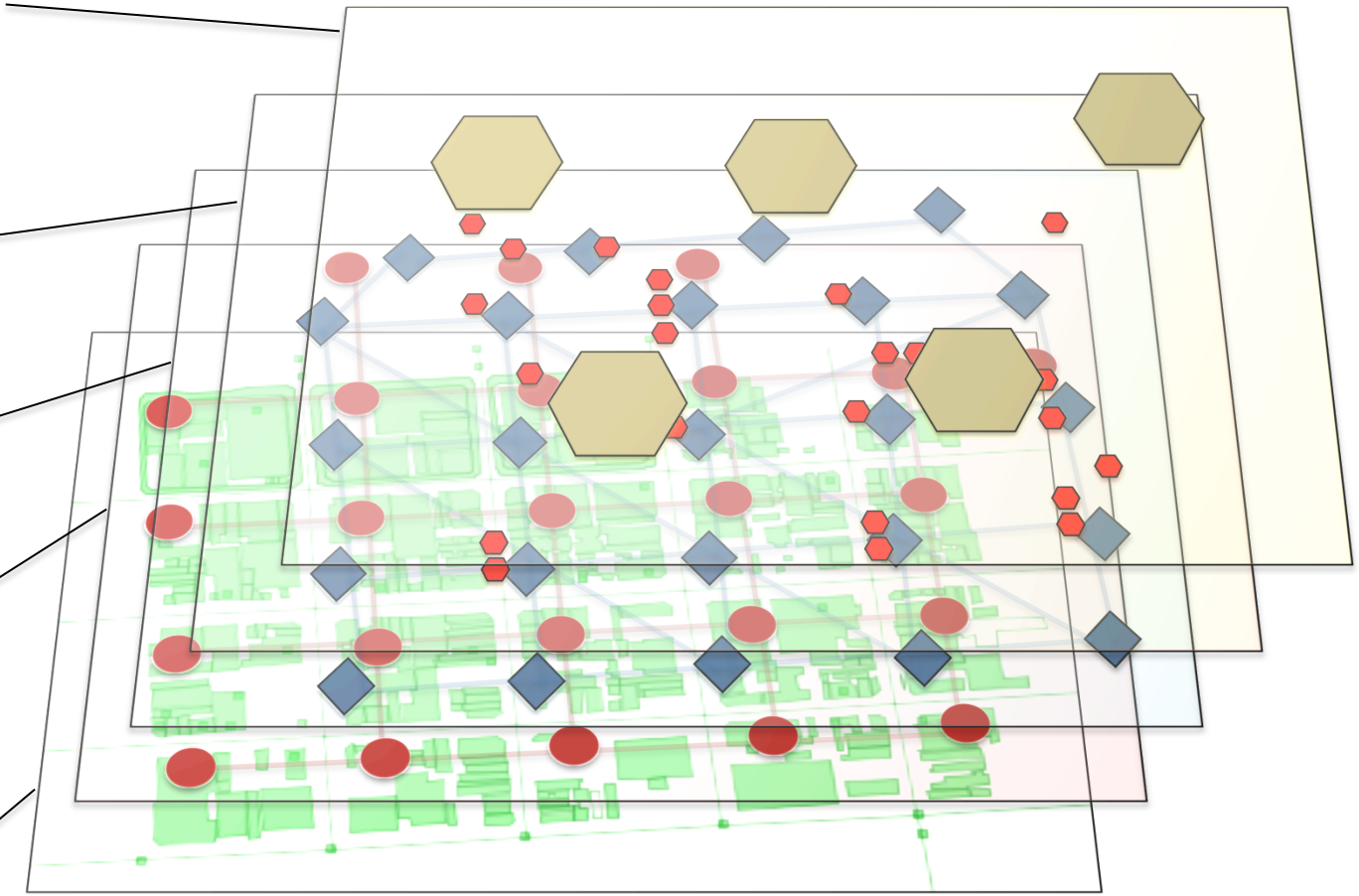
Consensus Layer
Routing
Negotiation
Cooperation
Collaboration

Agent Layer
Driver agent
Mobile app agent

Sensor Layer
Sensor network

Traffic Synch Layer
Traffic light

Geographical Layer
OSM

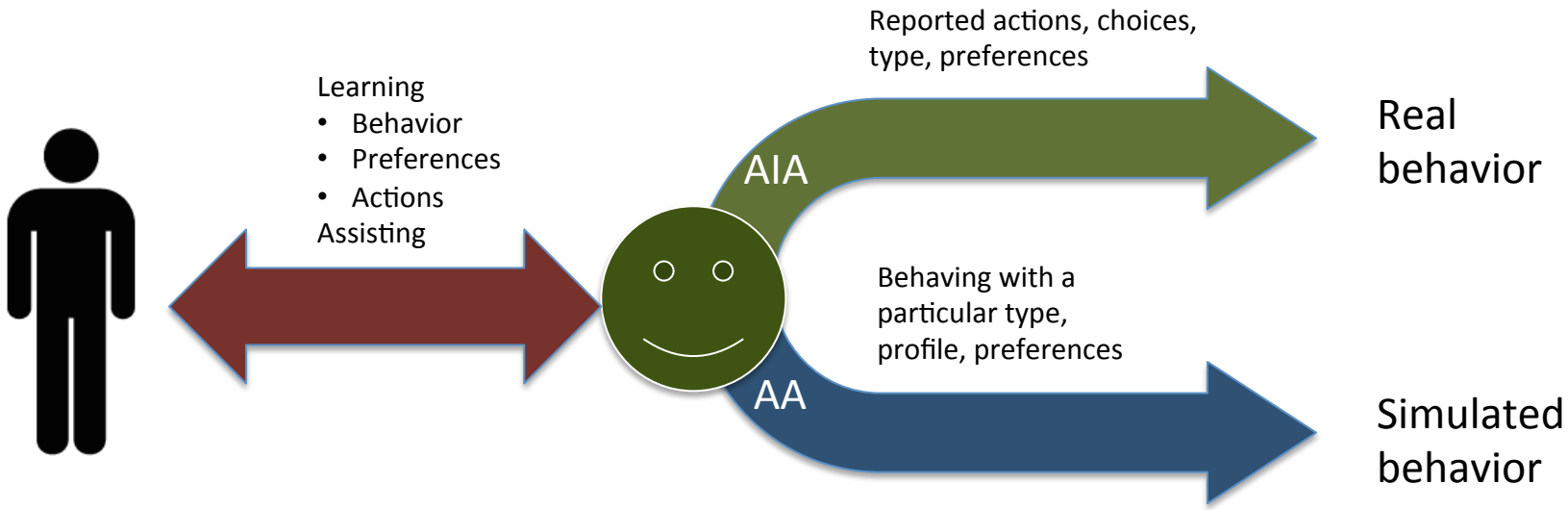


The “agent” paradigm

- Reactive, autonomous, collaborative, and goal-oriented agents
  - autonomous: parallel and distributed deployment (StarBED)
- Microscopic (autonomous) and macroscopic (collective) simulation
- An agent capable of
  - Reproducing realistic mobility
  - Learning of preferences, behaviors (peers modeling)
  - Elicitation
  - Optimization
  - etc.

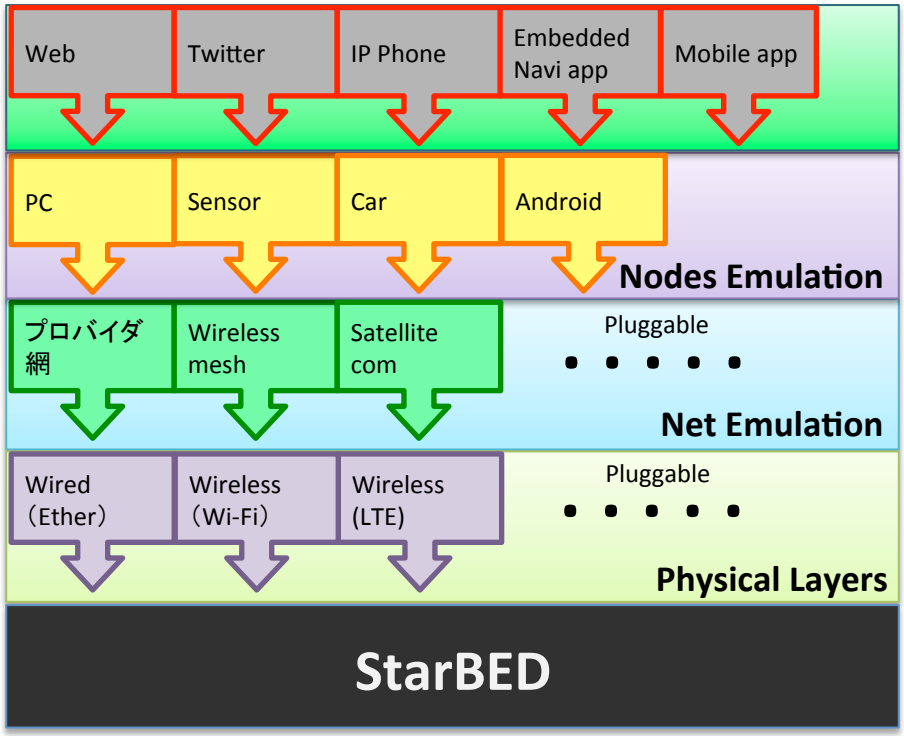
The “agent” paradigm

- An agent can operate in two different ways:
  - Autonomous Interface Agent (AIA) assisting & interacting with humans
  - Autonomous Agent (AA) to simulate the human

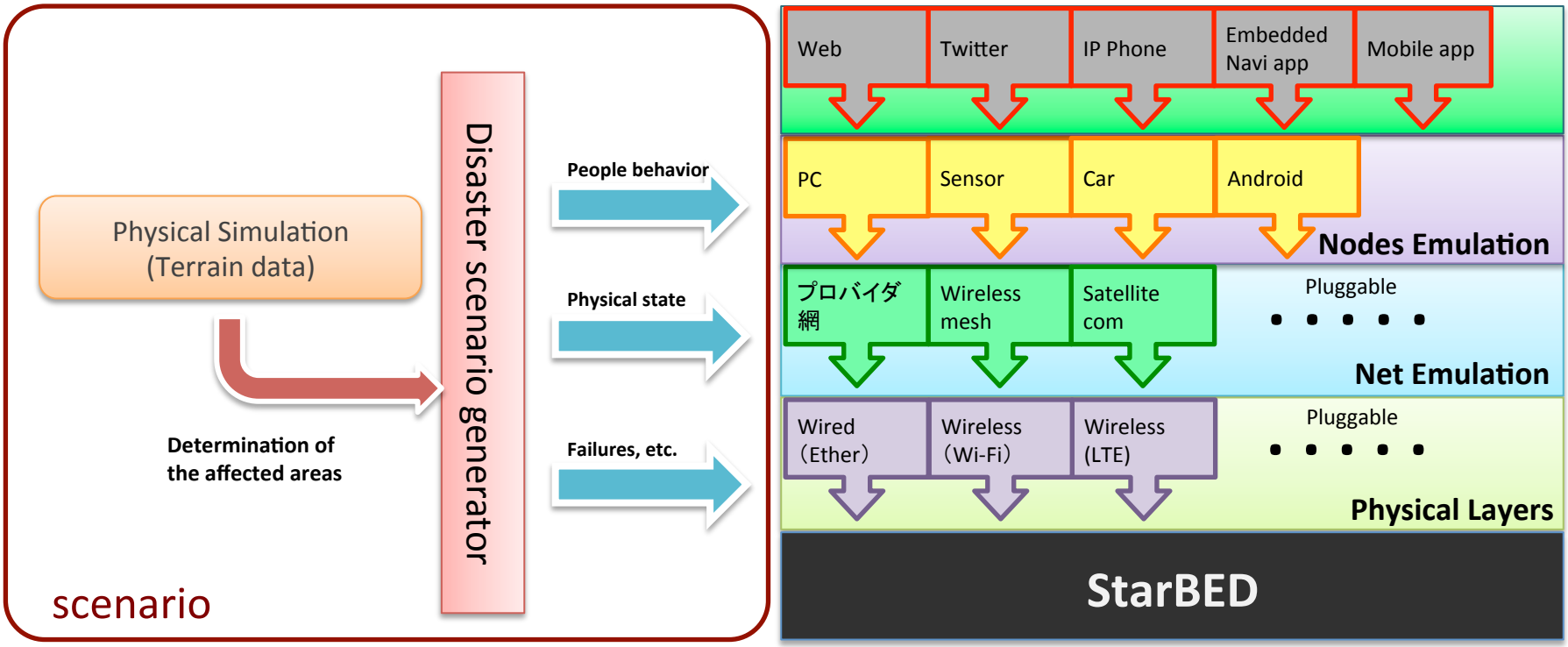


MAS on StarBED stack

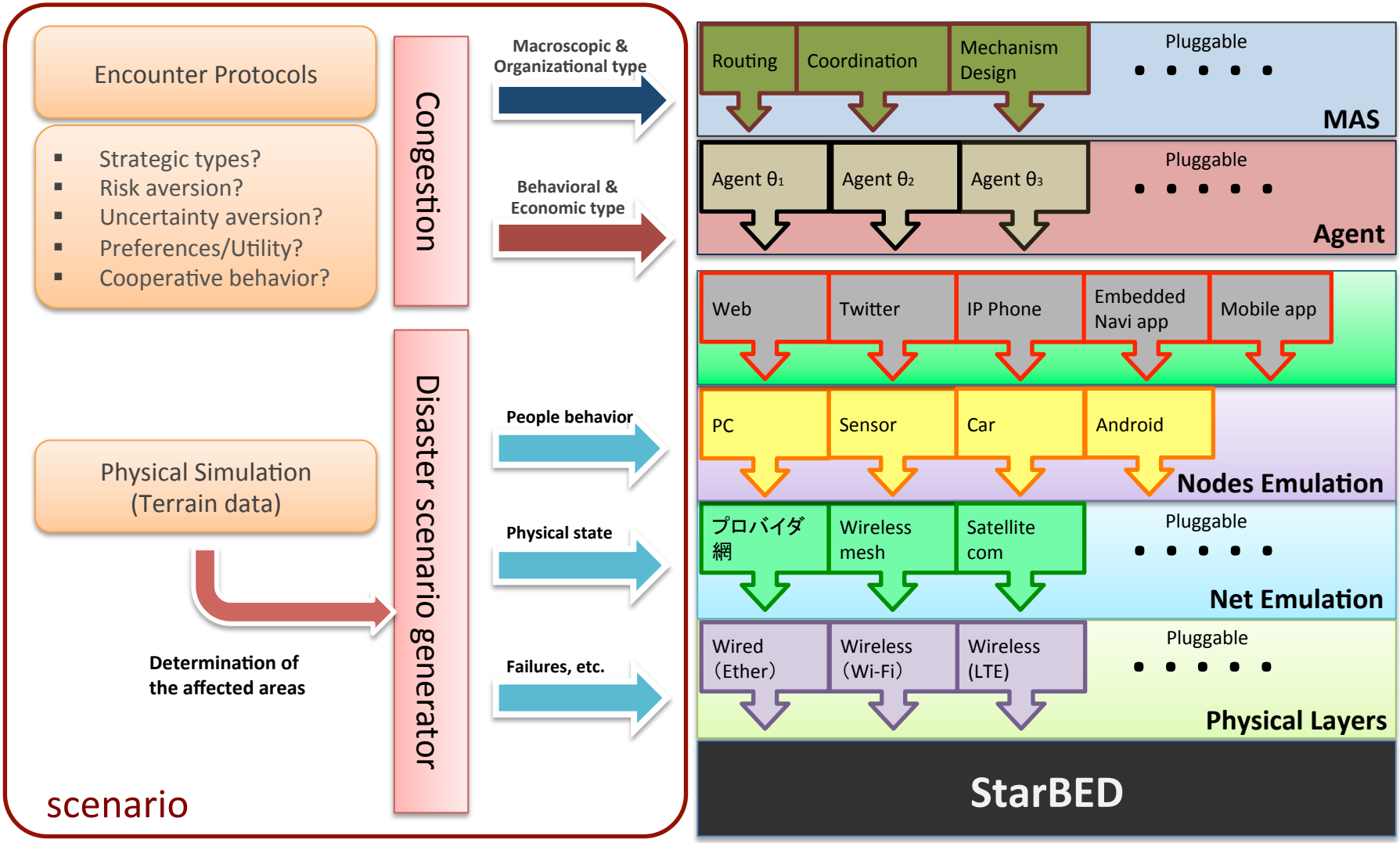
MAS on StarBED stack



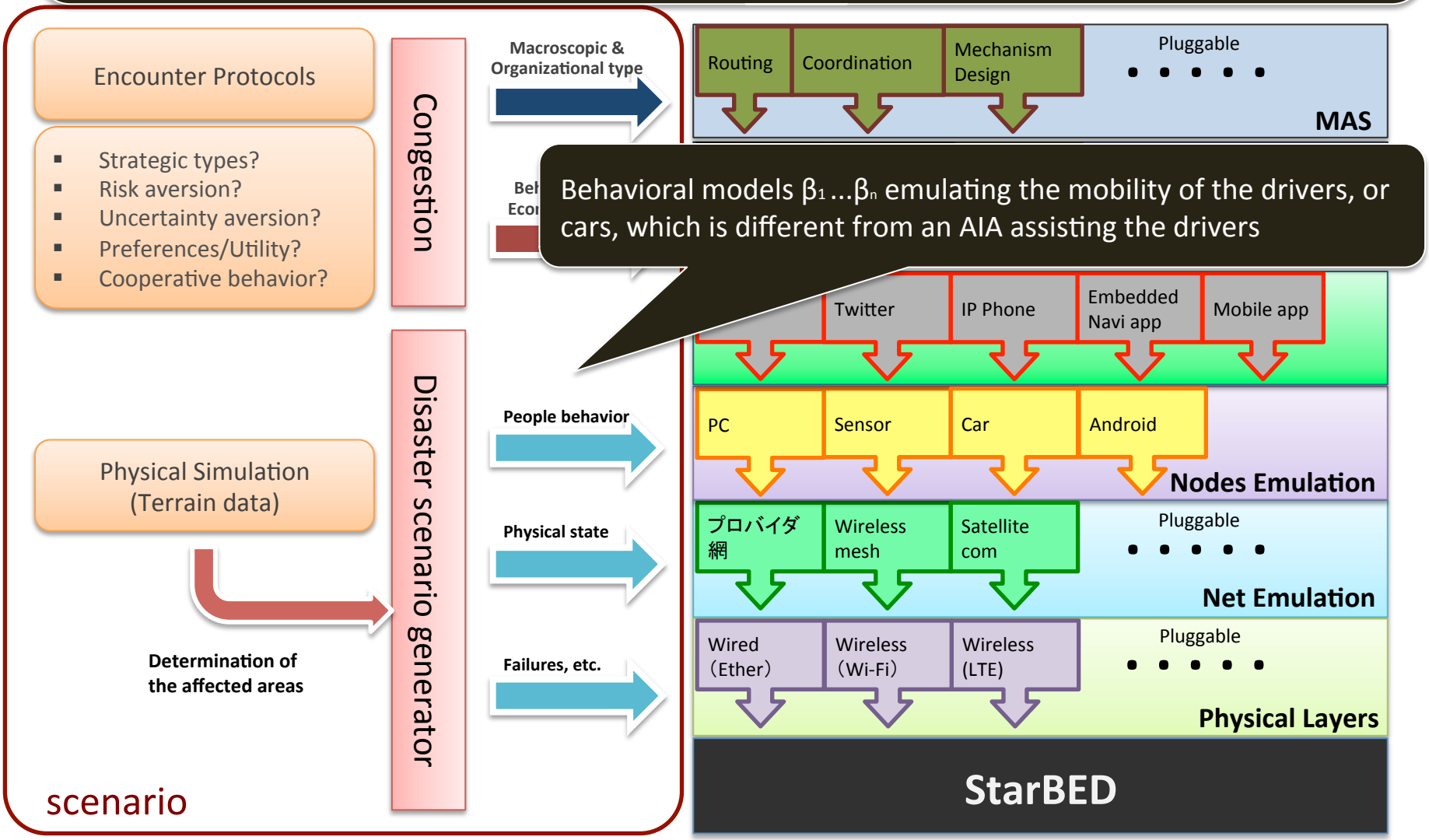
MAS on StarBED stack



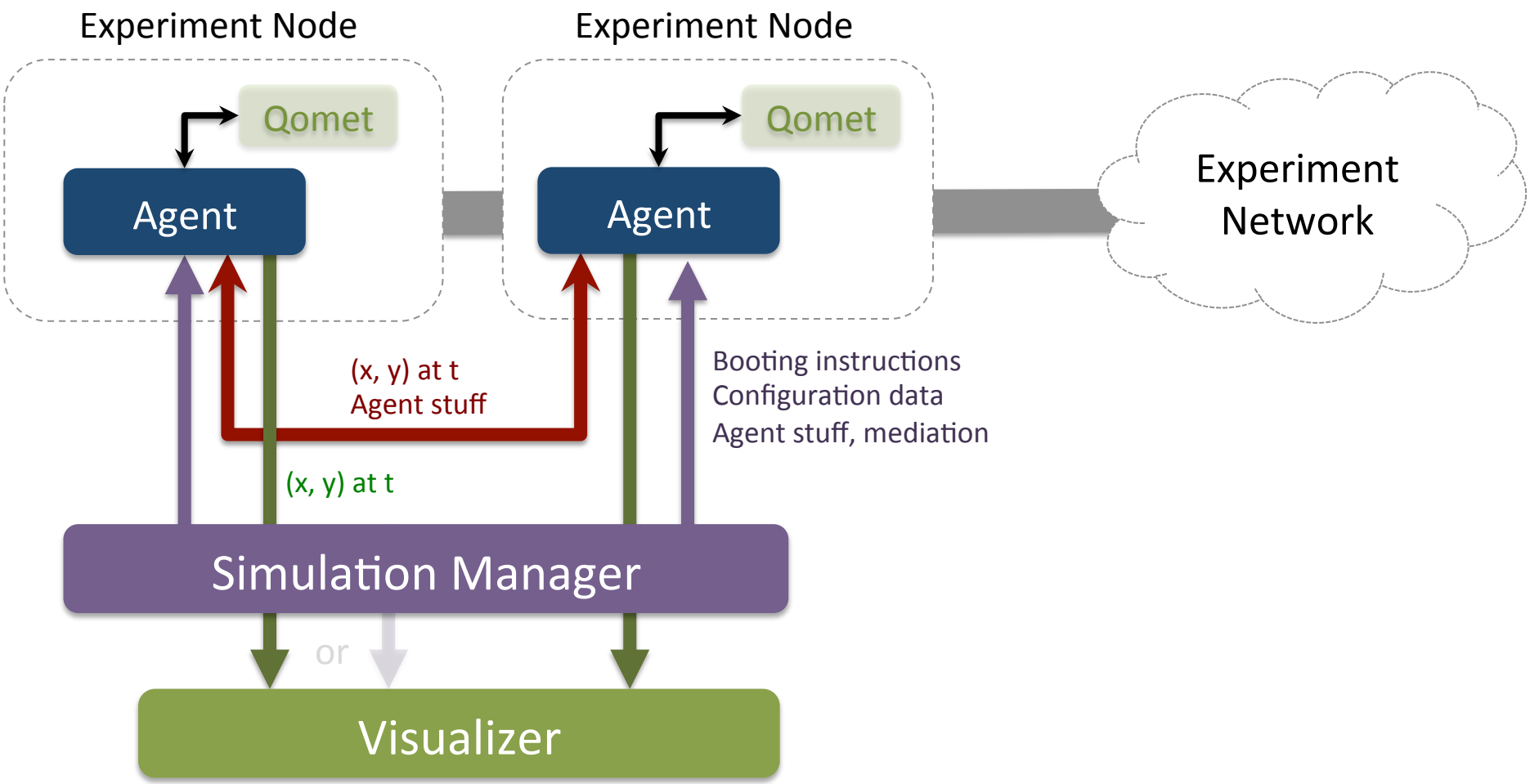
MAS on StarBED stack



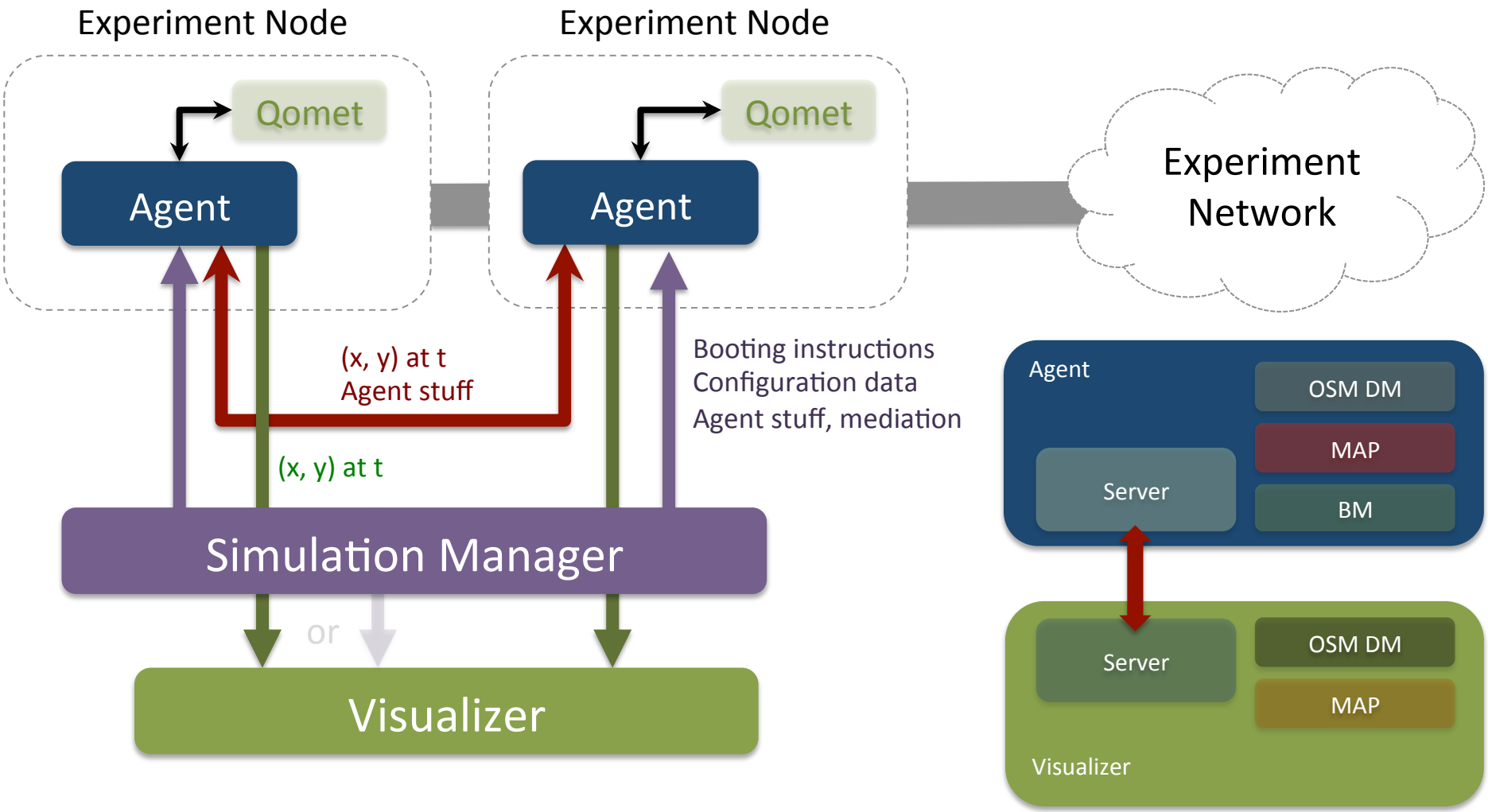
Example: Emulating a traffic coordination task in a disaster scenario (congested) involving car-embedded & mobile app agents representing risk seeking, uncertainty averse drivers (only α% are cooperative). The coordination mechanism uses a sensor network



MAS on StarBED stack



MAS on StarBED stack



Traffic simulation

- Agents will have to emulate drivers behavioral models

- Dynamic and interactive mobility generation based on 3 forces
  - Speed *v*
  - Acceleration *a*
  - Breaking *b*

- Interaction with external forces
  - Driver actions, through the steering wheel
  - Environment: traffic lights, routes
  - Interaction with other vehicles

Default speed, or “desired” speed

Current speed, resulting from the interaction with the environment

Speed modulation

Speed difference: $\delta\vec{v} = obj\vec{v} - \vec{v}$

Example of rules:

$\delta\vec{v} < -3$

$\vec{a} = 0$

$\vec{b} = \delta\vec{v} * urgency$

$\delta\vec{v} > 3$

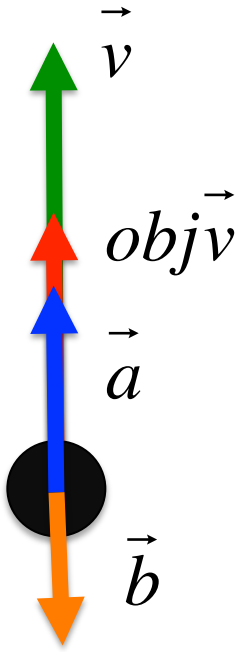
$\vec{a} = \delta\vec{v} * urgency$

$\vec{b} = 0$

$obj\vec{v} = 0$

$\vec{a} = 0$

$\vec{b} = -\vec{v}$



Default speed, or “desired” speed

Current speed, resulting from the interaction with the environment

Speed modulation

Speed difference: $\delta\vec{v} = obj\vec{v} - \vec{v}$

Example of rules:

$\delta\vec{v} < -3$

$\vec{a} = 0$

$\vec{b} = \delta\vec{v} * urgency$

$\delta\vec{v} > 3$

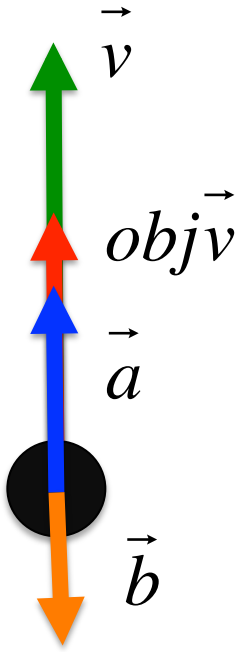
$\vec{a} = \delta\vec{v} * urgency$

$\vec{b} = 0$

$obj\vec{v} = 0$

$\vec{a} = 0$

$\vec{b} = -\vec{v}$



Angle modulation

is based on the steering wheel position and how it changes given any change in the overall direction of the vehicle and its velocity angle (θ)

Default direction

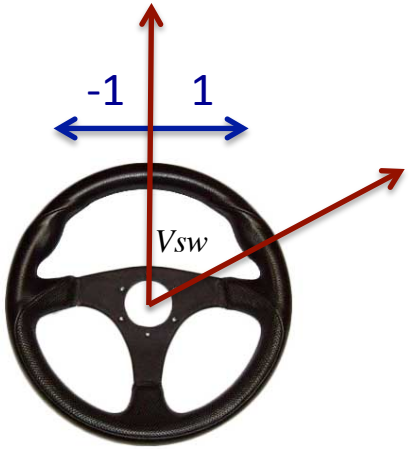
Steering wheel value at t-1

$\delta\theta = obj\vec{d} - \theta$

$Vsw = Vsw_t - Vsw_{t-1}$

Update rule

$Vsw = Vsw_{t-1} + \delta\theta$



Traffic simulation

- Rule-based directives or programmable goal-oriented agents

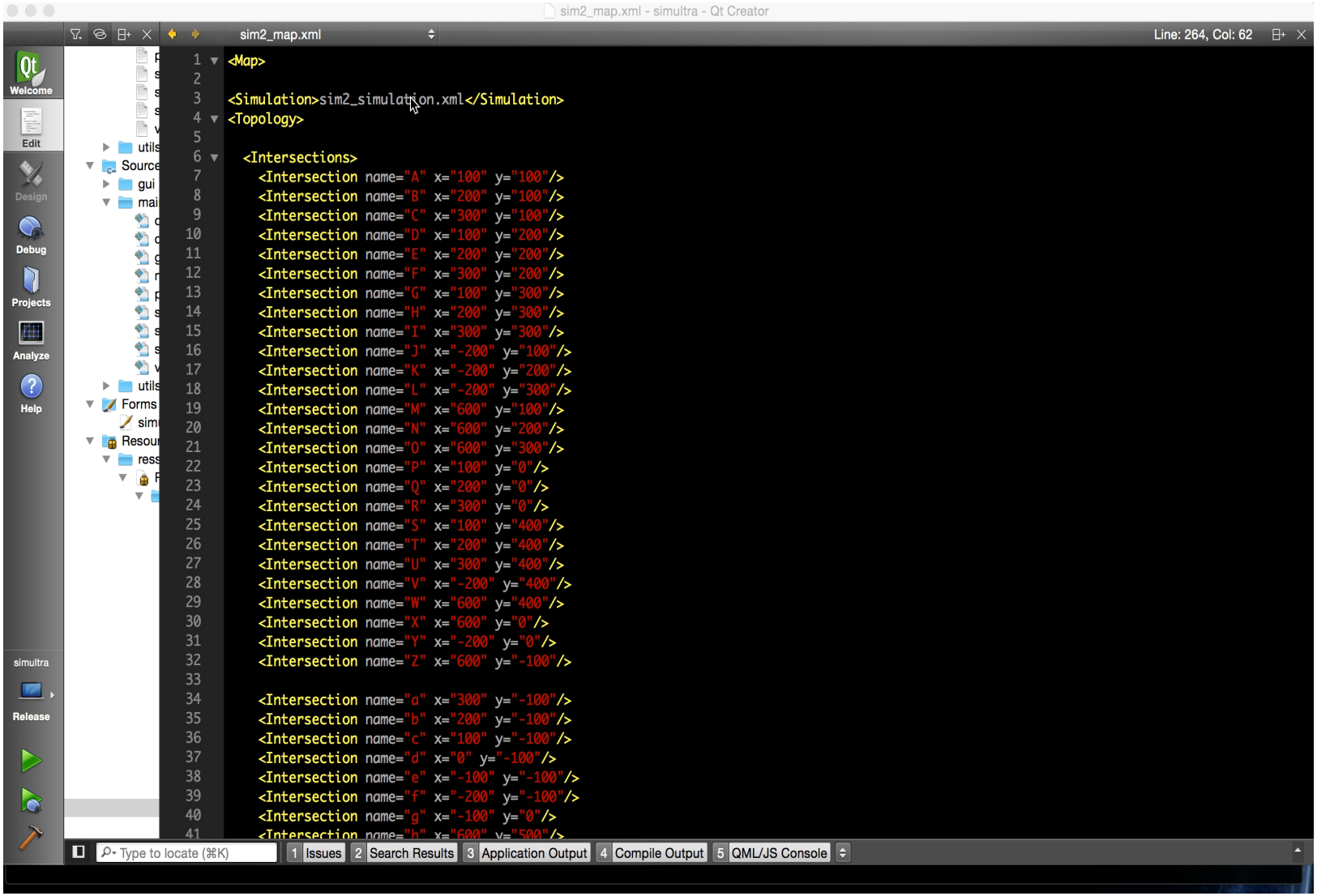
```
  </Driver>
</Vehicle>
->
<Vehicle type="Ferrari">
  <Location>0 300</Location>
  <Direction>0</Direction>
  <Driver type="Normal">
    <Lane>
    <Intersections>G H I F N</Intersections>
    </Lane>
  </Driver>
</Vehicle>

<Vehicle type="Ferrari">
  <Location>0 200</Location>
  <Direction>0</Direction>
  <Driver type="Speedy">
    <Lane>
      <Intersections>D A B C F I H G S</Intersections>
    </Lane>
  </Driver>
</Vehicle>

<Vehicle type="Twingo">
  <Location>0 100</Location>
  <Direction>0</Direction>
  <Driver type="Normal">
    <Lane>
    <Intersections>A B C F I O</Intersections>
    </Lane>
  </Driver>
</Vehicle>

<Vehicle type="Ferrari">
```

driver.cpp — Driver::drive(bool): void

```
void Driver::drive(bool urgent)
{
  if (m_Vehicle)
  {
    float speed = m_Vehicle->speed(true),
          AccelerationValue = 0.0,
          BreakingValue = 0.0,
          diffSpeed = 0.0f;
    diffSpeed = m_ObjSpeed - speed;

    if (diffSpeed < -3.0f)  {
      AccelerationValue = 0.0;
      BreakingValue = qAbs(diffSpeed * (urgent ? 100 : 3) / 100.0f);
    }
    else if (diffSpeed > 3.0f)  {
      AccelerationValue = qAbs(diffSpeed * (urgent ? 100 : 3) / 100.0f);
      BreakingValue = 0.0;
    }
    else if (m_ObjSpeed==0.0)  {
      AccelerationValue = 0.0;
      BreakingValue = 1.0;
    }

    float diffAngle = m_ObjectifDirection - m_Vehicle->direction().angle
    float valSteeringWheel = 0.0f;

    diffAngle = Point::reinit0and180(diffAngle);
    valSteeringWheel = 2 * (diffAngle) / 180.0;
    qreal diffSTRW=0;
```

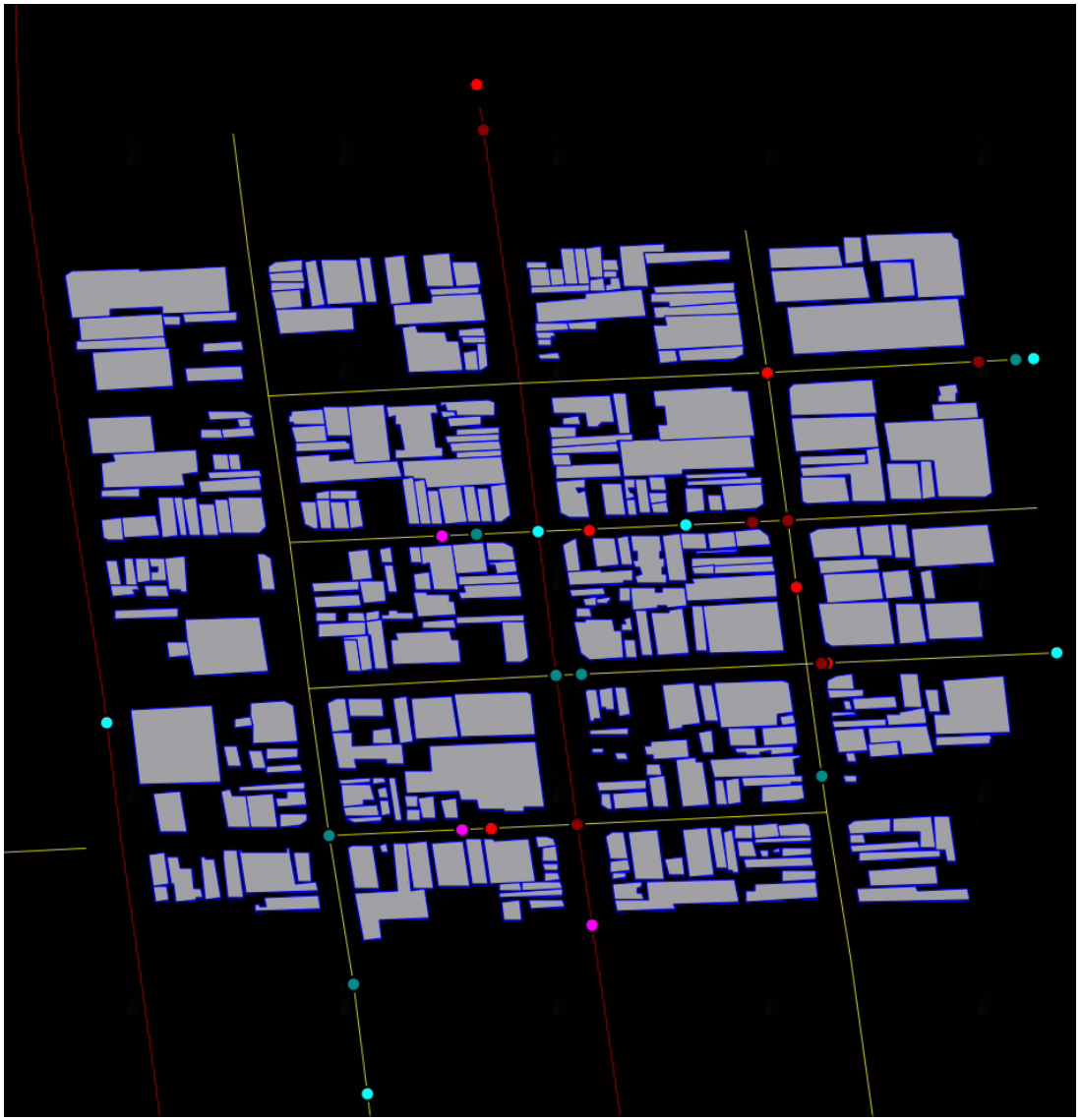
Traffic simulation

Agent-based, microscopic prototype (before distribution)



Traffic simulation

Agent-based, distributed, macroscopic prototype



Thank you