Romain Alexandre
Cécile Camillieri
Fabien Foerster
Jérôme Rancati

7 février 2014

Sensor simulation framework

Supervisor : Sébastien Mosser

Université Nice SOPHIA ANTIPOLIS

POLYTECH NICE-SOPHIA

Imagine ...

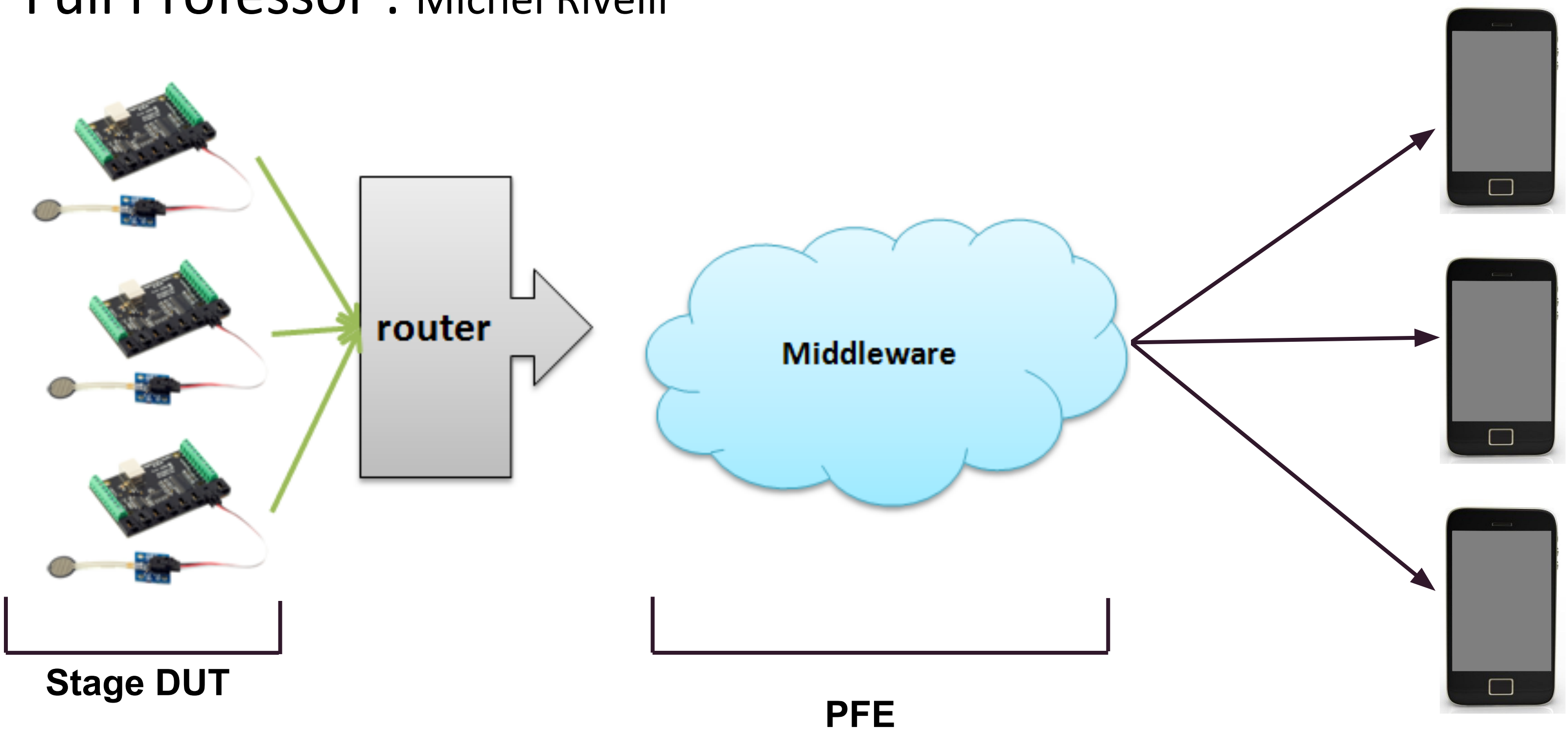
SmartCampus

SI5 : Cyril Cecchinel, Thomas Di'Meco, Matthieu Jimenez and Laura Martellotto

DUT : Jean Oudot

Associate Professor : Sébastien Mosser

Full Professor : Michel Riveill



Problem

NO SENSORS

Objectives

Framework

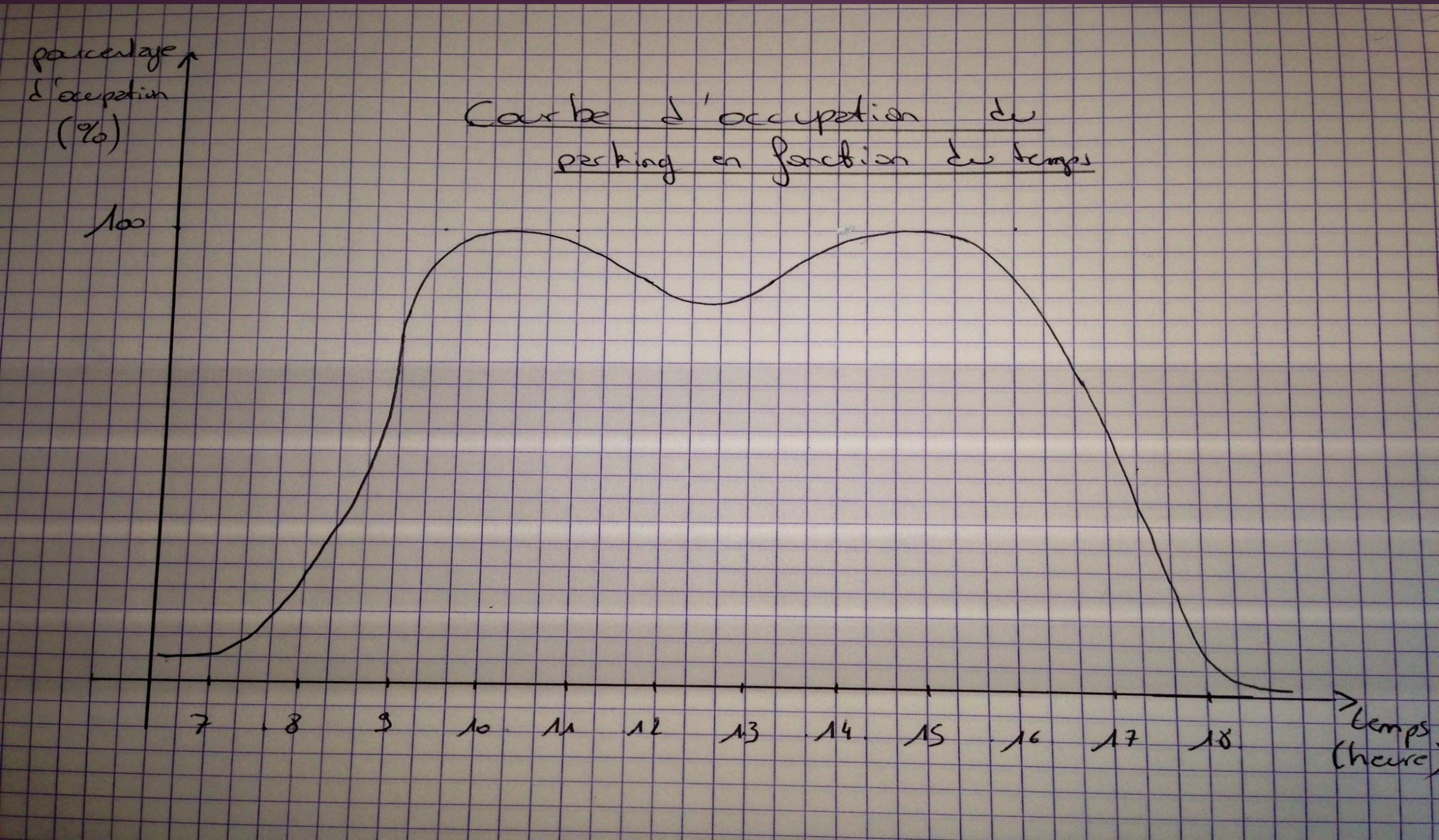
- Data simulation
- Simulation running

Standard Library

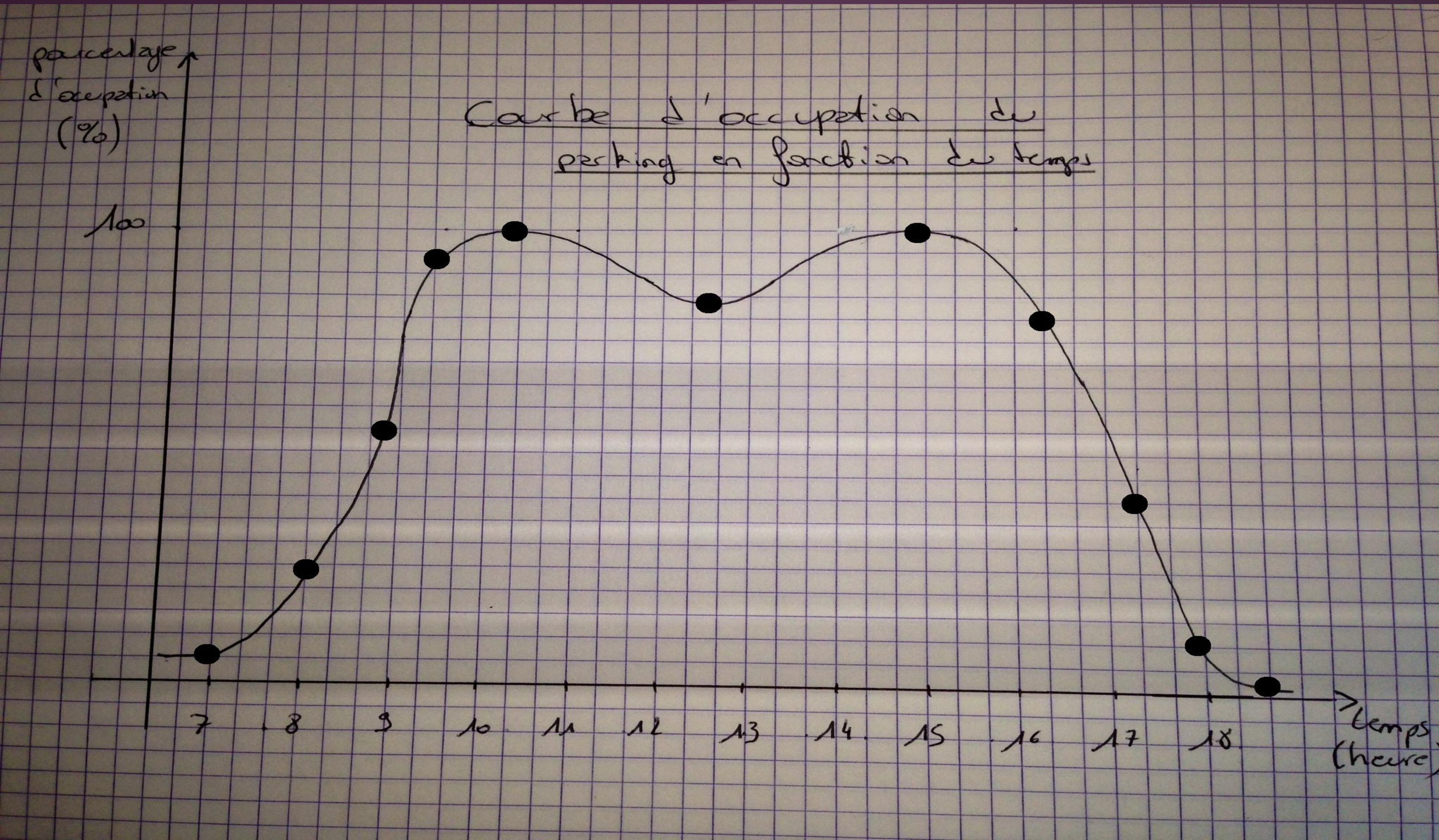
SmartCampus application

Challenges

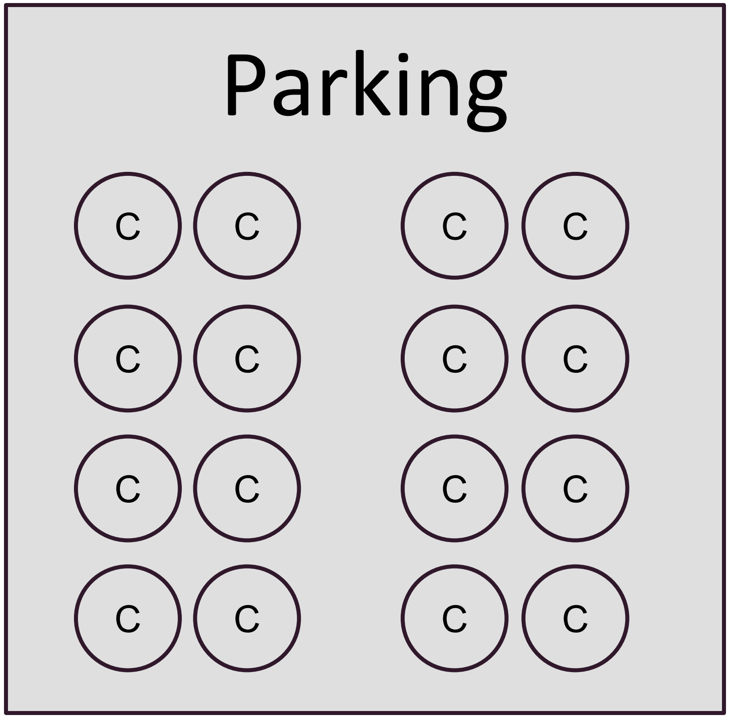
Data simulation



Data simulation



Concurrency



Advanced data simulation

send their information to a central database, or they distribute them via wireless communications. According to their web page, their sensors are close to being brought into market.

In [2] we concentrate on the dissemination of parking place occupancy information via multi-hop vehicular ad hoc communication, i. e., the second of the three issues mentioned in the introduction. We have used the dissemination protocol presented in that paper for the evaluation of our prediction model.

III. The Algorithm

In this section, we introduce our algorithm for the prediction of future parking place occupancy, and we show how it can be implemented in a vehicular ad hoc network. Our approach is based on results from queueing theory and applied stochastics. We model a parking lot as a queue and use a Markov chain to describe it. Since vehicles can park or leave at arbitrary times, we use a continuous-time model [7], [8].

A. *Dissemination of Parking Lot Information*

In our approach, five values for each parking lot are distributed in the network, namely timestamp, total capacity of the parking lot, number of parking places that are currently occupied, and finally, two rates: *the arrival rate* of vehicles, and the *parking rate*. The parking rate is the inverse of the average time for which a vehicle stays on its parking lot before

In the theory of continuous-time Markov chains, the concept of the Q-matrix is used in order to be able to calculate the transition probabilities for all $t \in \mathbb{R}_0^+$. The transition rate q_{ij} from state i to state j is defined as the right-hand derivative of $p_{ij}(t)$ at $t = 0$:

$$\forall i \neq j : q_{ij} := \lim_{t \searrow 0} \frac{p_{ij}(t)}{t}.$$

By conservation of probability, the probability of staying in a certain state i decreases with rate

$$q_{ii} = -\sum_{j \neq i} q_{ij}.$$

The Q-matrix is then defined by $Q = (q_{ij})$; its dimension is $(m+1) \times (m+1)$.

If the parameter of the arrival Poisson process is denoted by λ, and μ is the parking rate, this results in a Q-matrix with the following tri-diagonal pattern:

$$Q = \begin{pmatrix} -\lambda & \lambda & & & \\ \mu & -(\lambda+\mu) & \lambda & & \\ & 2\mu & -(\lambda+2\mu) & \lambda & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & (m-1)\mu & -(\lambda+(m-1)\mu) & \lambda \\ & & & m\mu & -m\mu \end{pmatrix}$$

The corresponding Markov chain is depicted in Figure 2.

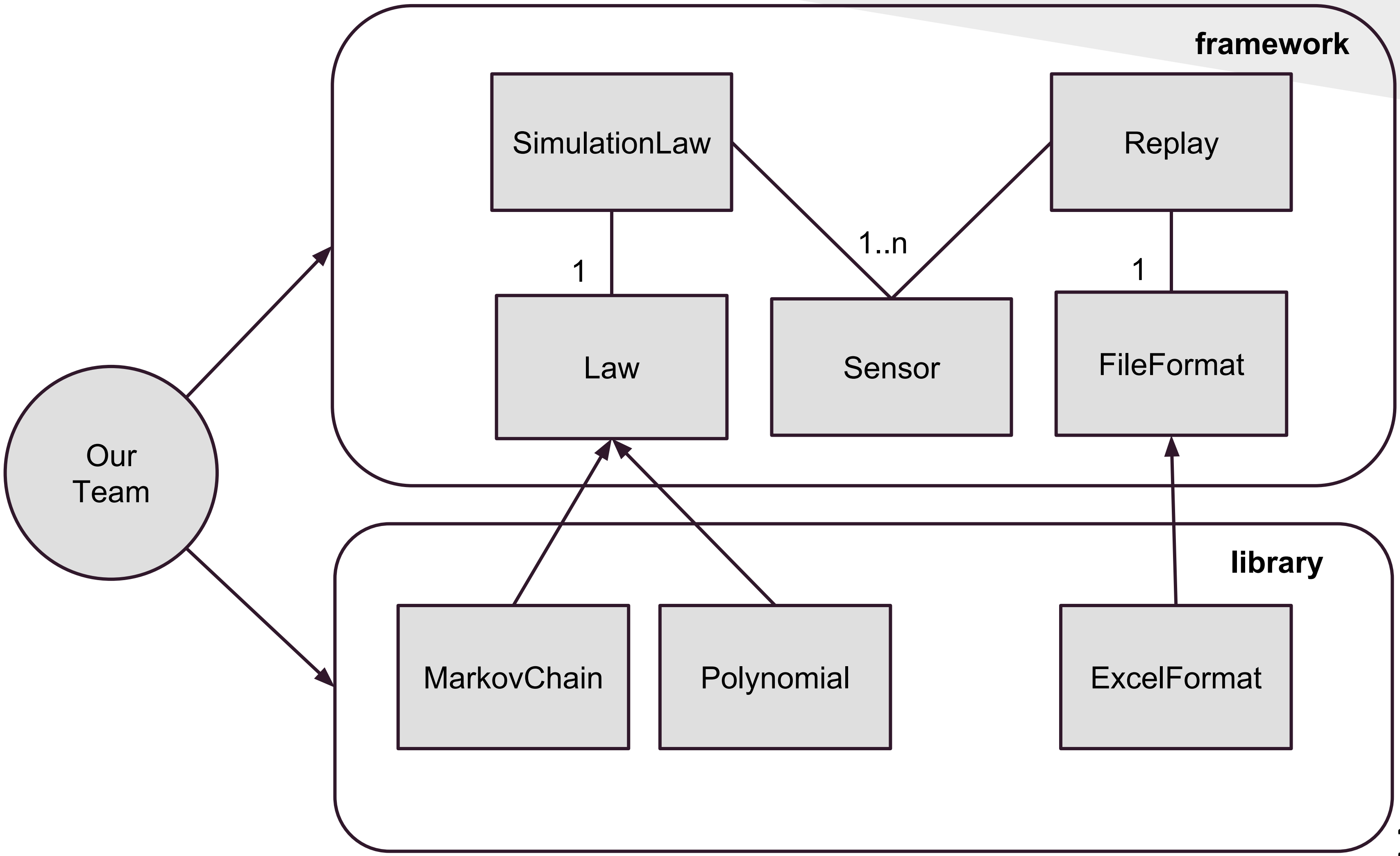
from : "Predicting Parking Lot Occupancy in Vehicular Ad Hoc Networks" by Murat Caliskan, Andreas Barthels, Bjorn Scheuermann and Martin Mauve 65th IEEE Vehicular Technology Conference, Dublin, Ireland, April 2007.

Replaying data

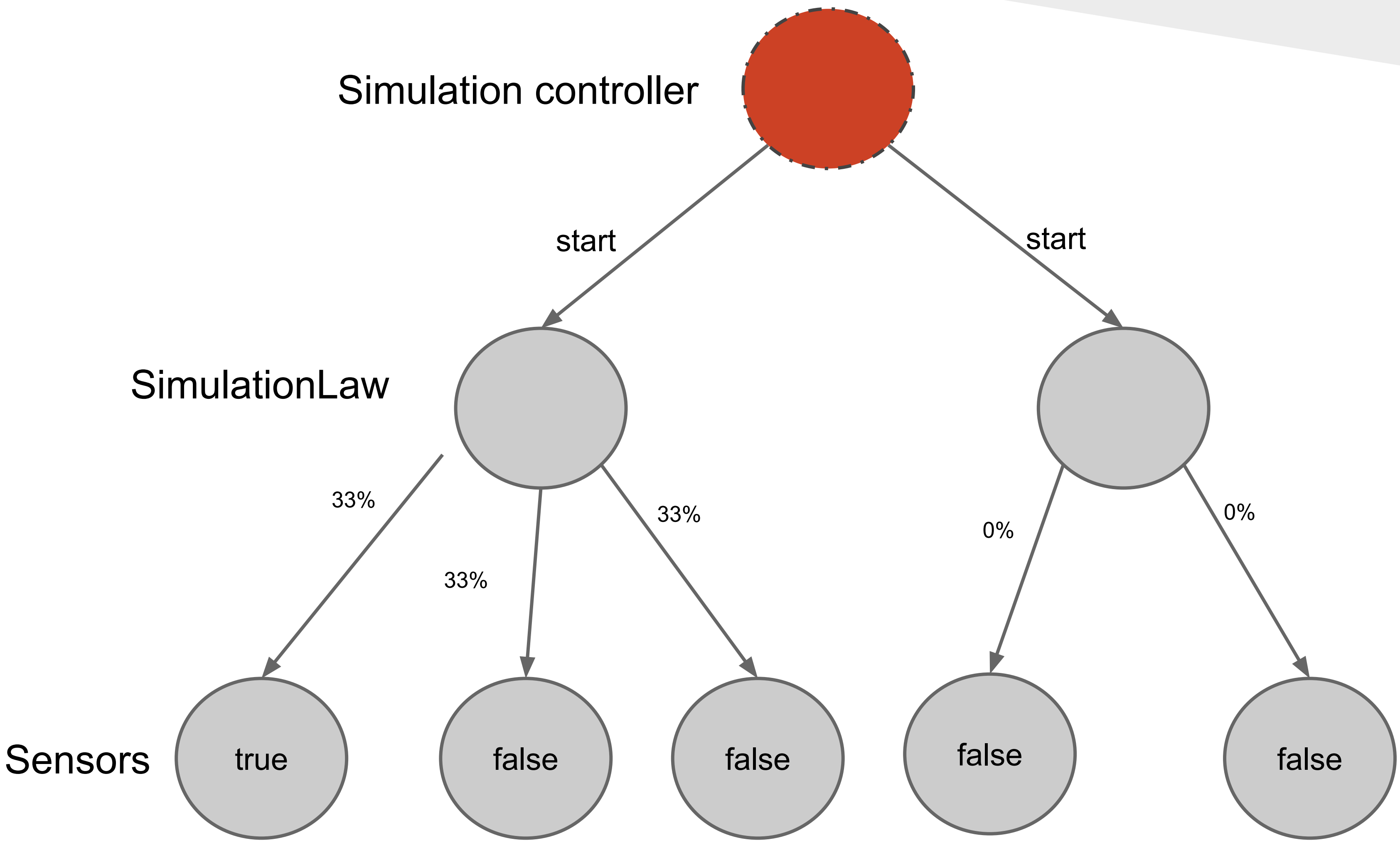
	A	B	C	D	E	F	G	H
1	date	timestamp	conductivite	debit	niveau	NH4	O2	pH
2	01/17/2011	1	237,759995	351	1,778	4	11,88	7,9548
3	01/17/2011	2	267,679993	351,4	1,696	3	11,34	8,4126
4	01/17/2011	5	98,839996	351,4	4,872	2	11,718	7,308
5	01/17/2011	6					11,88	7,9548
6	01/17/2011	7					11,34	8,4126
7	01/17/2011	8						7,308
8	01/17/2011	11						7,9548
9	01/17/2011	12					11,34	8,4126
10	01/17/2011	13					11,718	
11	01/17/2011	14					11,88	
12	01/17/2011	17					11,34	8,4126
13	01/17/2011	23					11,718	7,308

Contributions

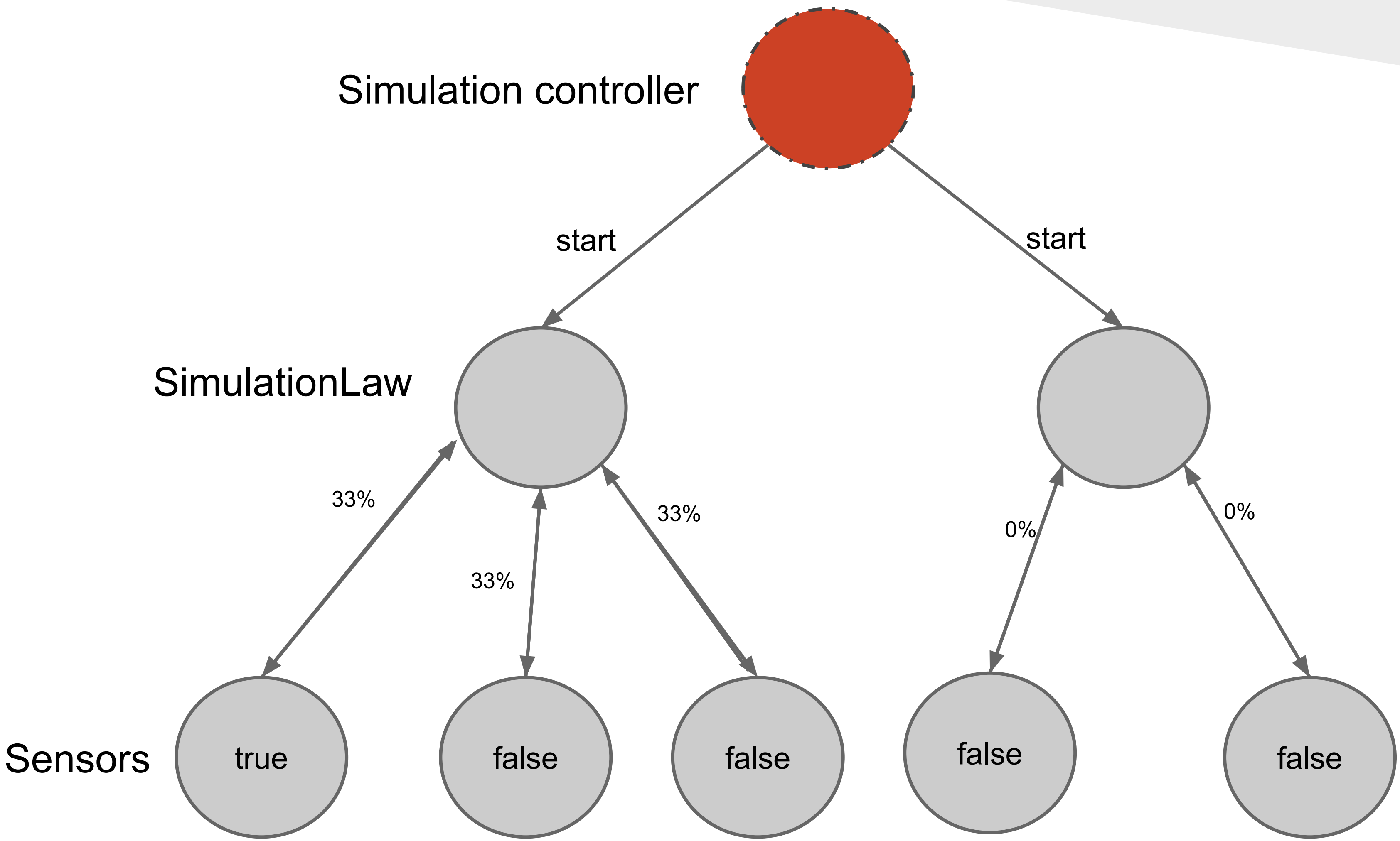
What we did


framework
SimulationLaw
Replay
1
1..n
1
Law
Sensor
FileFormat
Our
Team
library
MarkovChain
Polynomial
ExcelFormat

Concurrency : Akka

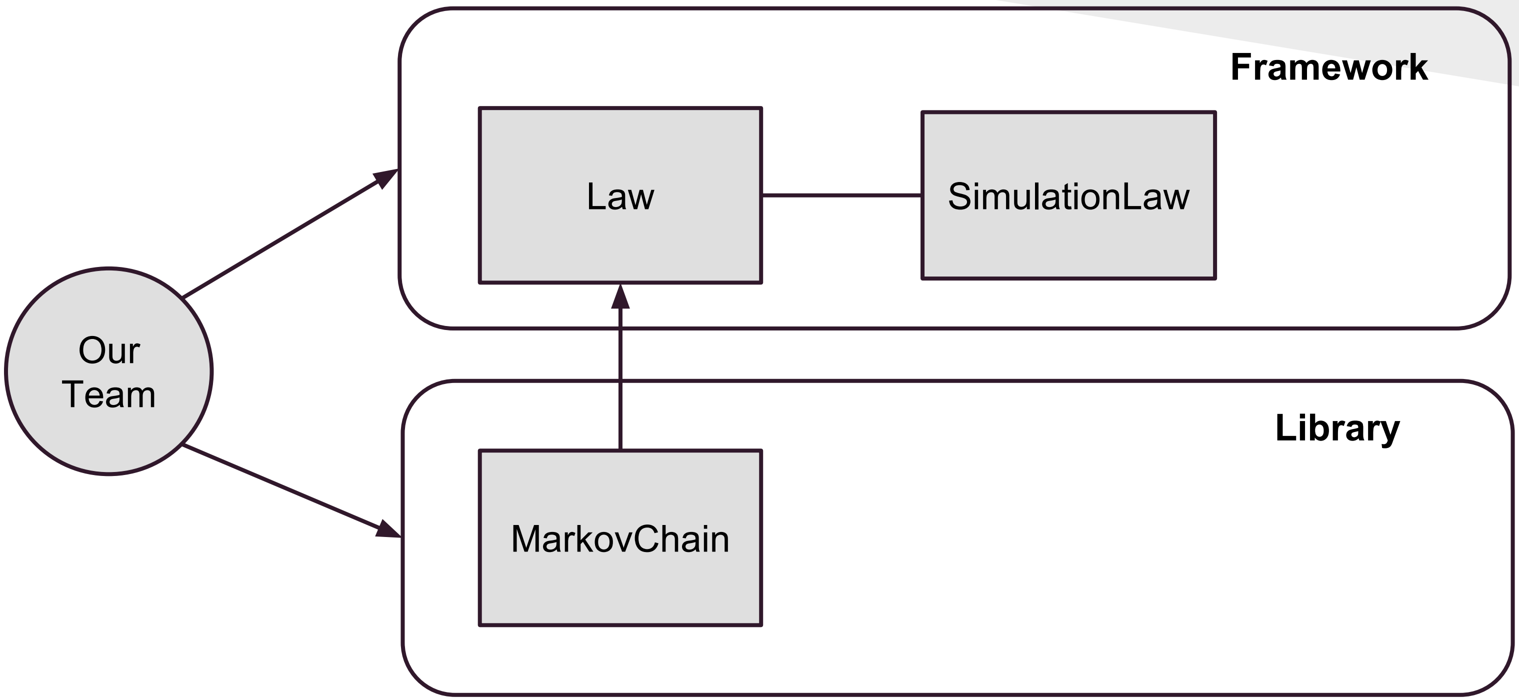


Concurrency : Akka

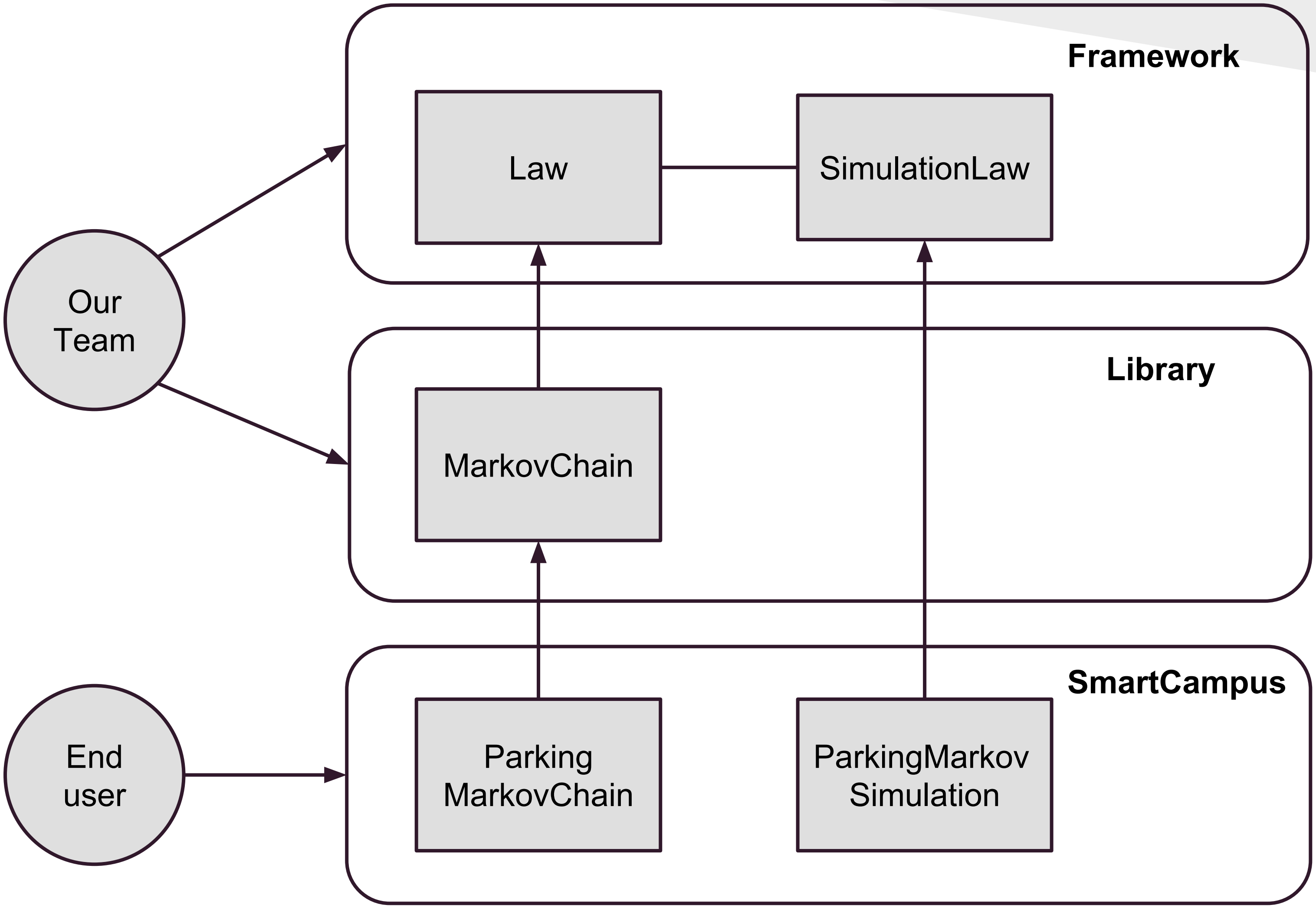


How to use it ?

Implementing our parking case



Implementing our parking case



Implementing our parking case

Setup

git clone https://login@atlas.polytech.unice.fr/stash/scm/psiqje/private
cd private
mvn install

$$\begin{pmatrix} -\lambda & \lambda & & & \\ \mu & -(\lambda+\mu) & \lambda & & \\ & 2\mu & -(\lambda+2\mu) & \lambda & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & (m-1)\mu & -(\lambda+(m-1)\mu) & \lambda \\ & & & m\mu & -m\mu \end{pmatrix}$$

Law definition

```
public class ParkingMarkovLaw extends MarkovChain {
	public Double evaluate(Integer[] t) {... return p; }
}
```

SimulationLaw definition

```
public class ParkingMarkovSimulation extends SimulationLaw<Integer, Double, Boolean> {
	protected Integer[] computeValue() {... return t; }
}
```

Is used to evaluate the law.

Running a simulation

Create a parking with 100 sensors following our Markov law

<u>Simulator creation</u>

```
Start sim = new StartImpl();

sim     .createSimulation("Parking1", ParkingMarkovSimulation.class)
        .withSensors(100, new RateToBooleanChangeSensorTransformation())
        .withLaw(new ParkingMarkovLaw(...))
        .createSimulation("Parking2", ParkingPolynomialSimulation.class)
        .withSensors(50, new RateToBooleanSensorTransformation())
        .withLaw(new PolynomialLaw(...))
```

Another parking with 50 sensors following a different law

Running a simulation

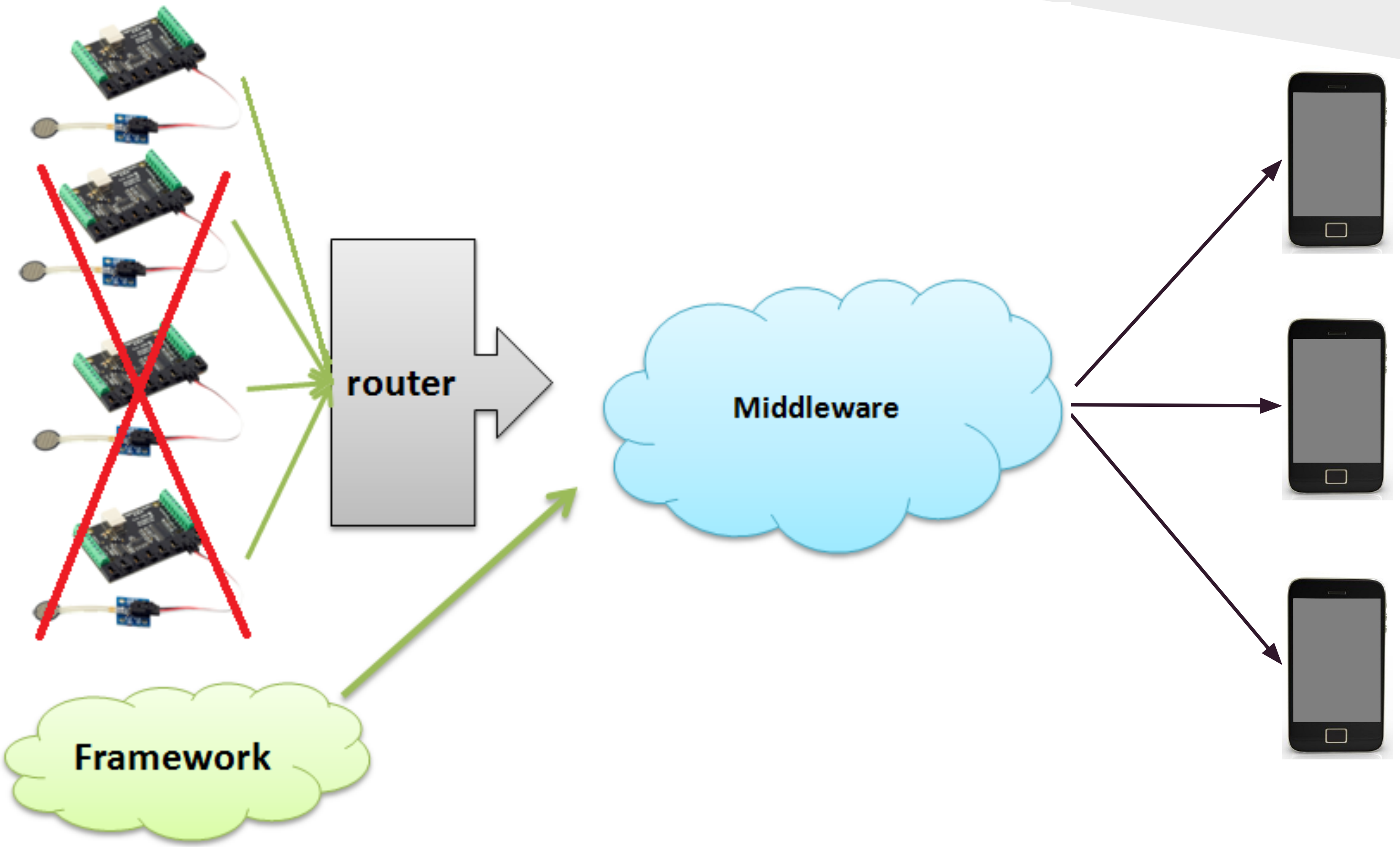
Set the url we will send the values to.

Set parameters and launch simulation

```
sim     .setOutput("http://localhost:8080/collector/value")
        .startAt("2014-02-07 11:25:00")
        .duration(Duration.create(2, TimeUnit.MINUTES))
        .frequency(Duration.create(5, TimeUnit.SECONDS))
        .simulateReal();
```

The simulation will last 2 minutes, with the sensors sending values every 5 seconds.

Demonstration



Conclusions

Results

Framework

Simulation modeling ☑

$\longrightarrow$ mathematical modeling ☑

Simulation running ☑

Standard Library ☑

SmartCampus application ☐

Results

Framework

- Simulation modeling
  - → mathematical modeling
  - → replay
- Simulation running

Standard Library

SmartCampus application


The Future
NEXT EXIT

Towards a smart campus

MinD

And more…

Questions ?

References

Pictures :

kentnguyen.com

ajence.com

visitluxembourg.com

Research documents :

"Predicting Parking Lot Occupancy in Vehicular Ad Hoc Networks" by Murat Caliskan, Andreas Barthels, Bjorn Scheuermann and Martin Mauve,

65th IEEE Vehicular Technology Conference, Dublin, Ireland, April 2007.

ParkingMarkovLaw

```
public class ParkingMarkvLaw extends MarkovChain {
  public ParkingMarkovLaw(final int nbPlaces, final double arrivalFreq,
      final double averageParkingTime) {
    super(nbPlaces + 1);
    for (int i = 0; i < this.size; i++) {
      for (int j = 0; j < this.size; j++) {
        if (i == j) {
          this.transition[i][i] = -((i * averageParkingTime) + arrivalFreq);
        }
        else if (j == (i + 1)) {
          this.transition[i][j] = arrivalFreq;
        }
        else if (j == (i - 1)) {
          this.transition[i][j] = i * averageParkingTime;
        }
      }
    }
    this.transition[nbPlaces][nbPlaces] = -(nbPlaces * averageParkingTime);
  }
  protected Double evaluate(final Integer... x) throws Exception {
    return super.evaluate(x);
  }
```

ParkingMarkovSimulation

```
public class ParkingMarkovSimulation extends SimulationLaw<Integer, Double, Boolean> {
  @Override
  protected Integer[] computeValue() {
    int i = 0;
    for (boolean b : this.values) {
      if (b) {
        i++;
      }
    }
    Integer[] t = { i, i };
    return t;
  }

  @Override
  protected void onComplete() {
    int nbPlacesOccupied = 0;
    for (Boolean b : this.values) {
      if (b) {
        nbPlacesOccupied++;
      }
    }
    this.sendValue("occupation", ((nbPlacesOccupied * 100) / this.values.size()) + "%");
  }
}
```

SuezExcelFormator

```
public class SuezExcelFormator extends ExcelFormator{
    private final SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");
    public SuezExcelFormator(){
        super(3,new String[]{"A","B"},2);
    }

    @Override
    protected long transform(String[] columns)  throws ParseException{
        long timestamp = 0 ;
            timestamp = sdf.parse(columns[0]).getTime();
            int hoursToMilli = Integer.valueOf(columns[1])*1000;
            timestamp += hoursToMilli;
        return timestamp;
    }
}
```