Balancing Active Objects on a P2P infrastructure

Javier Bustos Jimenez

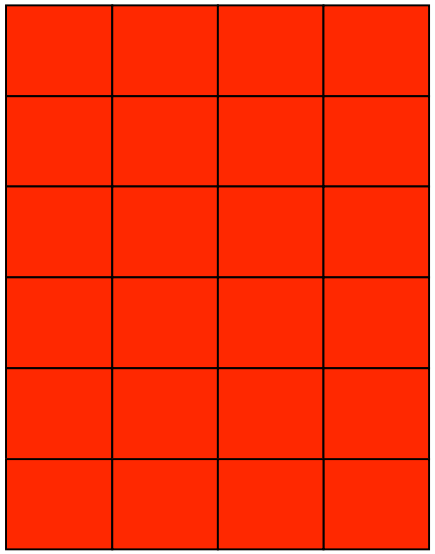
jbustos@dcc.uchile.cl

Agenda

- **Load Balancing**
- Balancing Active Objects
- Balancing in practice

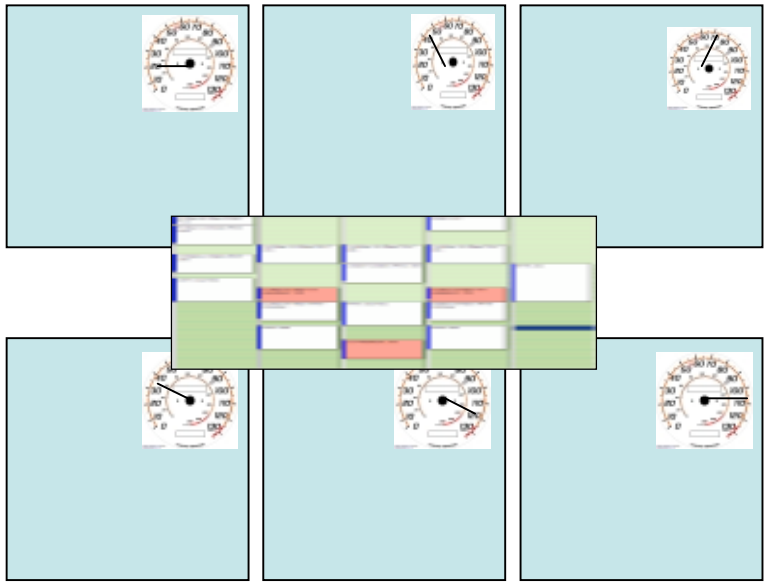
Load Balancing

The problem



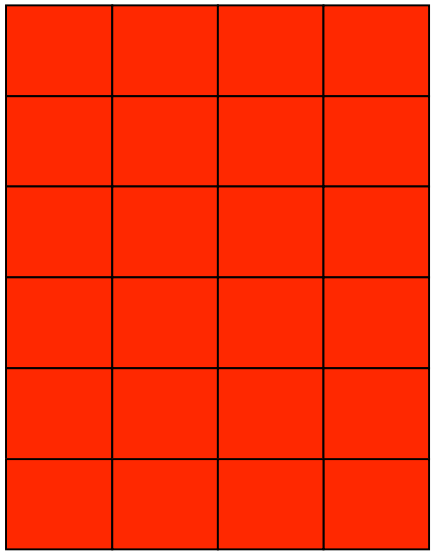
STATIC LOAD BALANCING

The machines



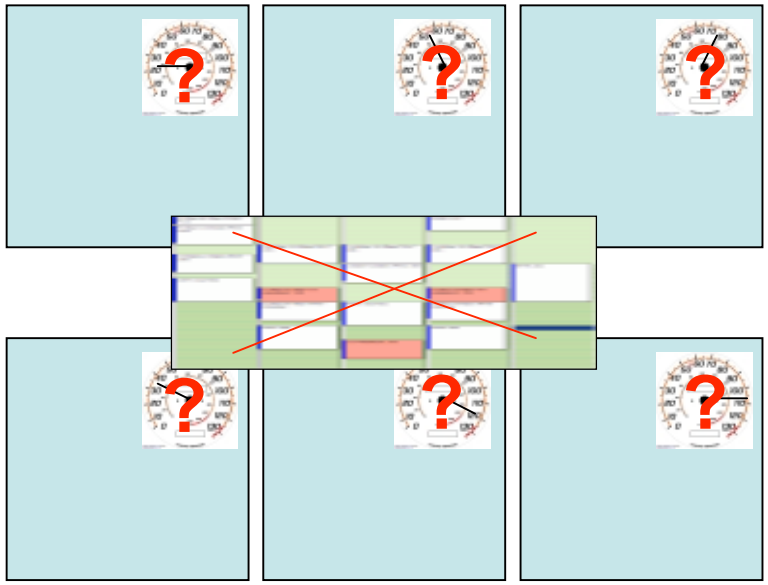
Load Balancing

The problem

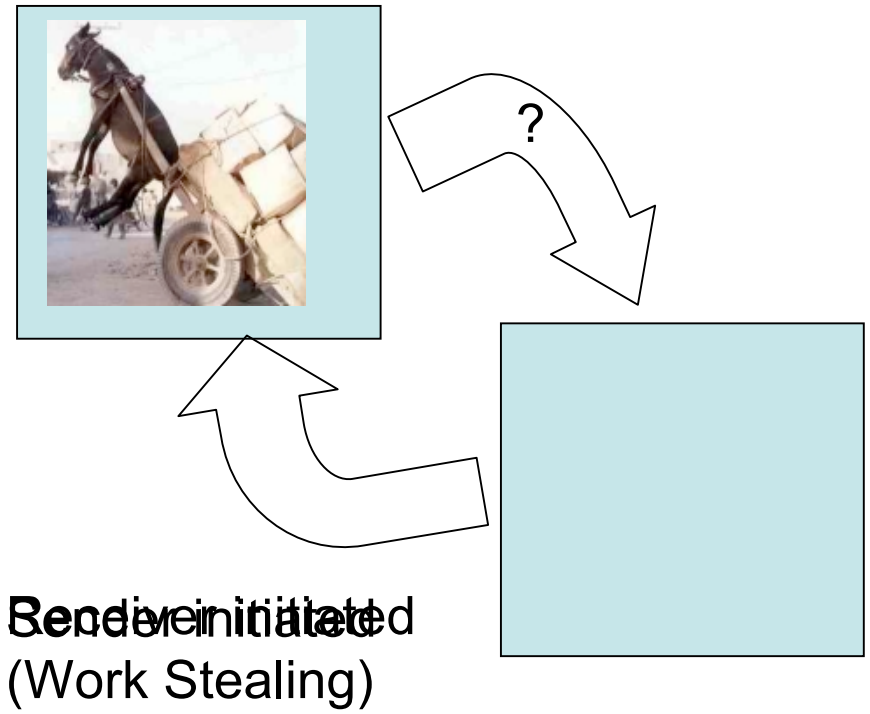


DYNAMIC LOAD BALANCING

The machines



Who will start the balance process?



Agenda

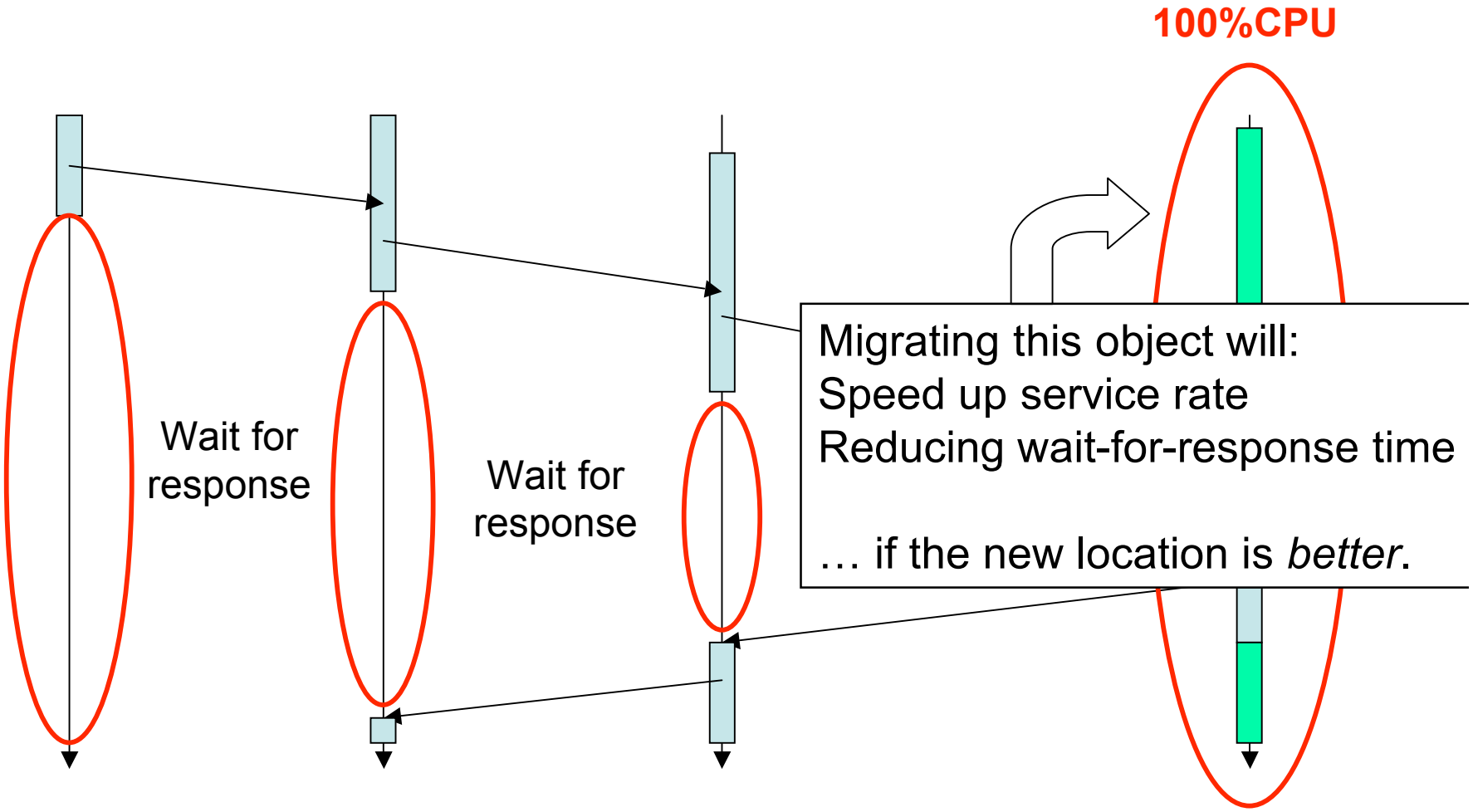
- Load Balancing
- **Balancing Active Objects**
- Balancing in practice

Balancing Constraints

- To speed up application performance
- To maximize the resource (CPU) usage.
- To reduce the bandwidth usage of Load Balance algorithm
- Fast reaction against load imbalances

P2P

Active Objects and CPU time



Migration Constraints

- What is *a better* machine?
  - Less loaded machine (idle machine?), **and**
  - Faster (or equivalent) machine

- Supposing: no active object's service will use more than 50% of CPU time

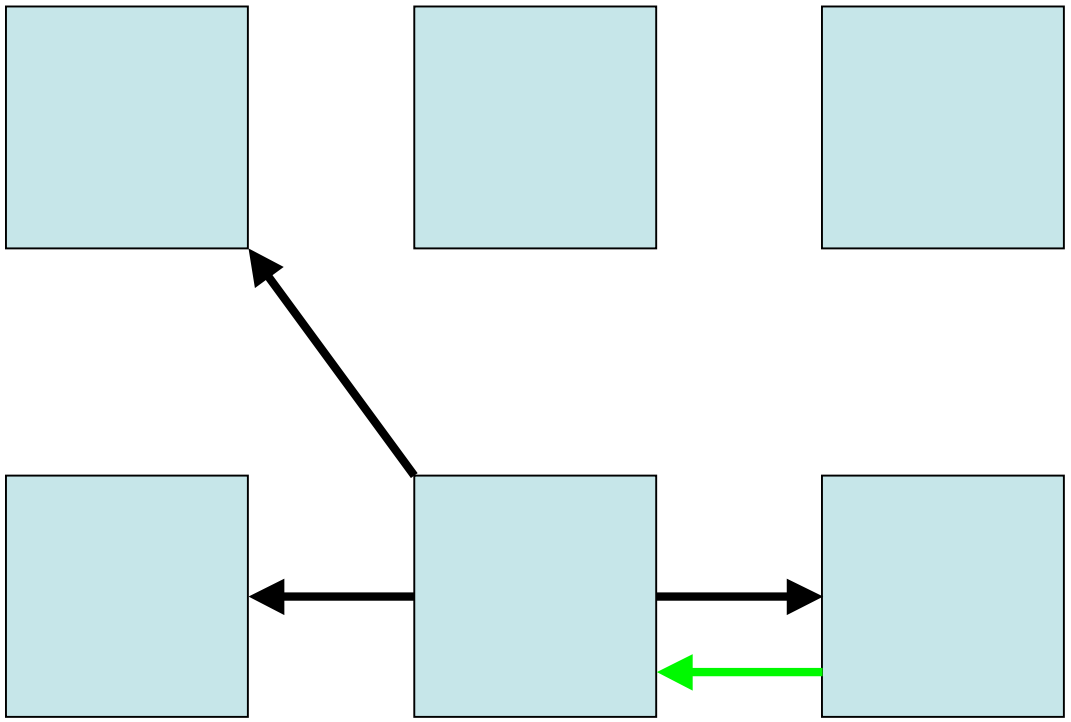
Migration to a *better* machine?

- Using a total order relation (*Rank*) among processors:
  - If P1 is overloaded, it will demand for balance to its neighbors, providing Rank(P1)
  - Let OT, UT = Overloaded (Underloaded) Threshold
  - If load(P2) < OT:

    If load(P2) < UT*Rank(P2)/Rank(P1):

    P2 will reply to P1 to start migration

Load Balancing on P2P



Messages =

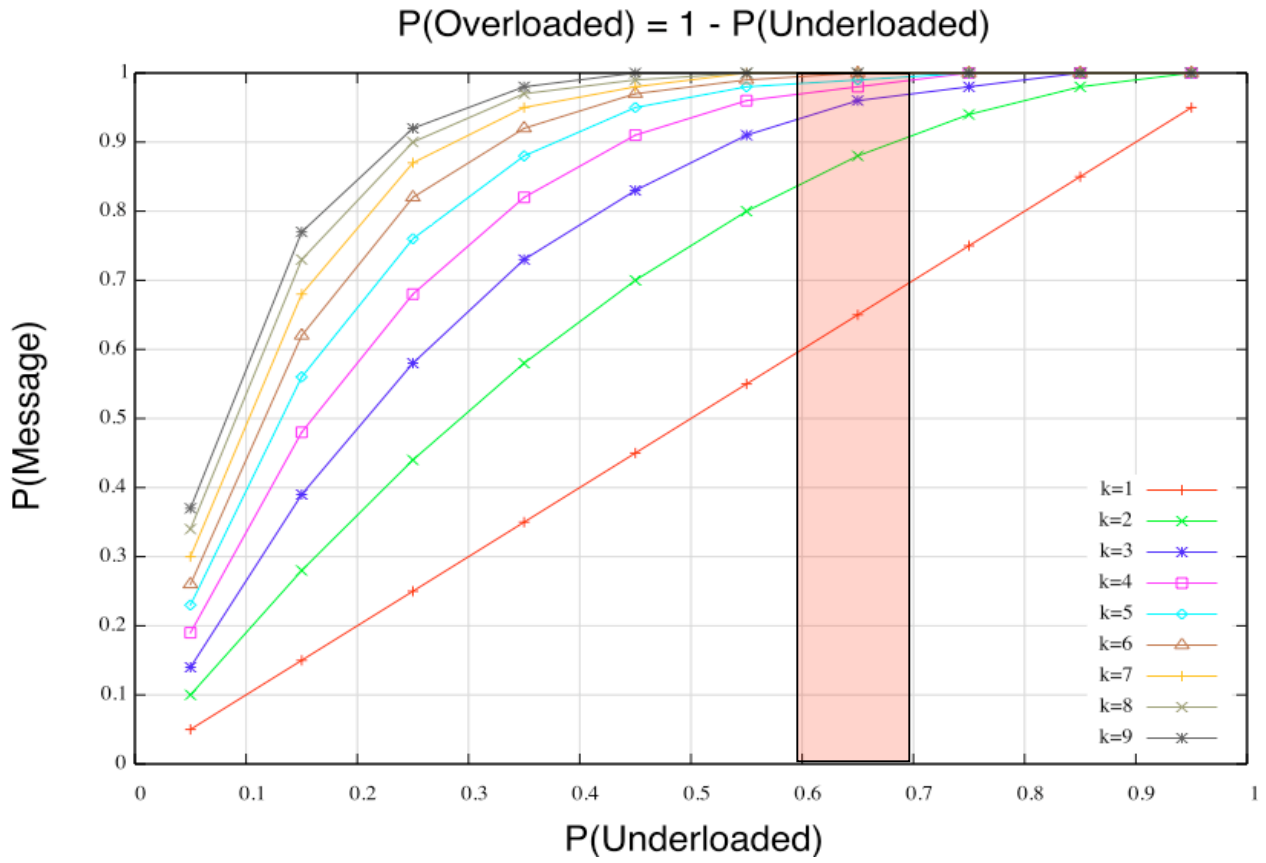
N° of Overloaded Nodes

X

K (1 + P(underloaded))

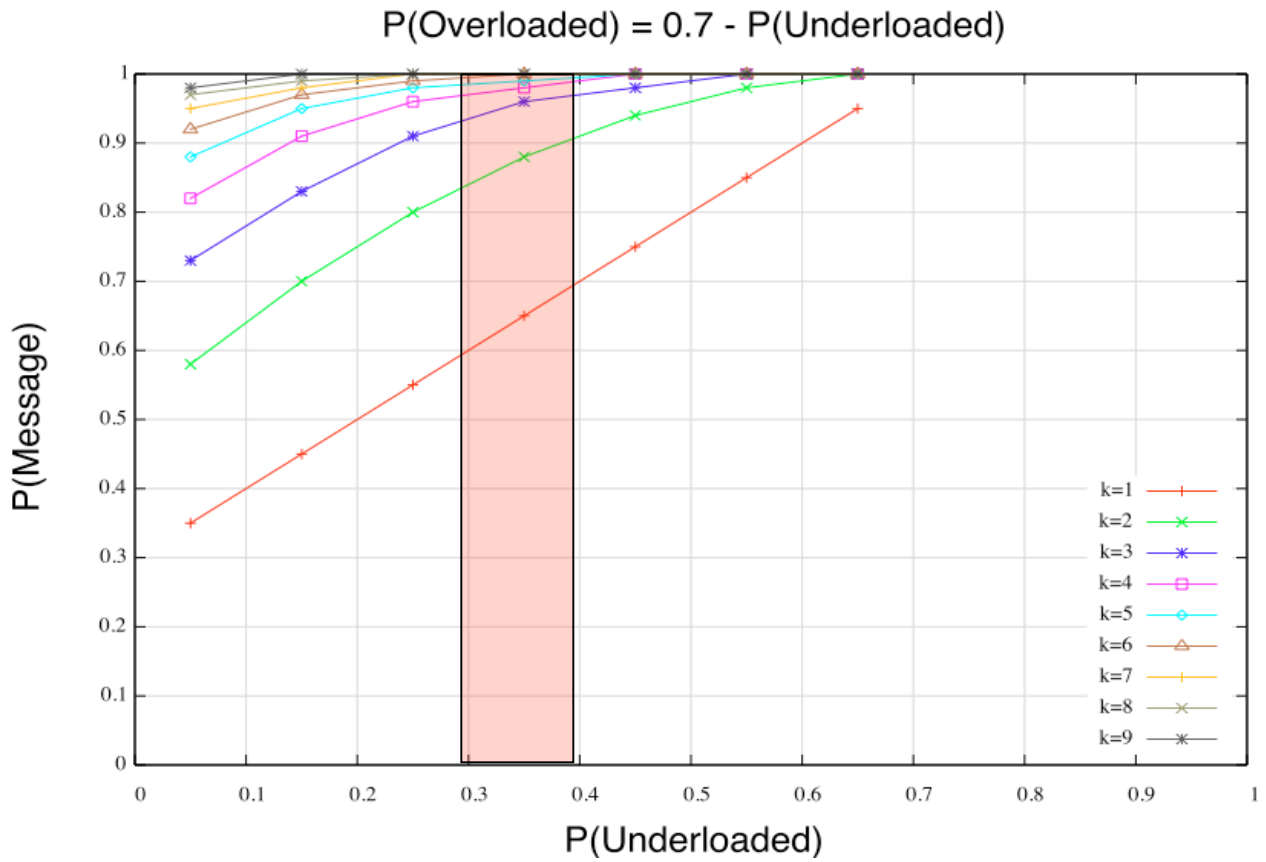
Which value of K?

- Probability of have at least one reply: $P_R = 1 - P_O^k$



Which value of K?

- Probability of have at least one reply: $P_R = 1 - P_O^k$



Agenda

- Load Balancing
- Balancing Active Objects
- **Balancing in practice**

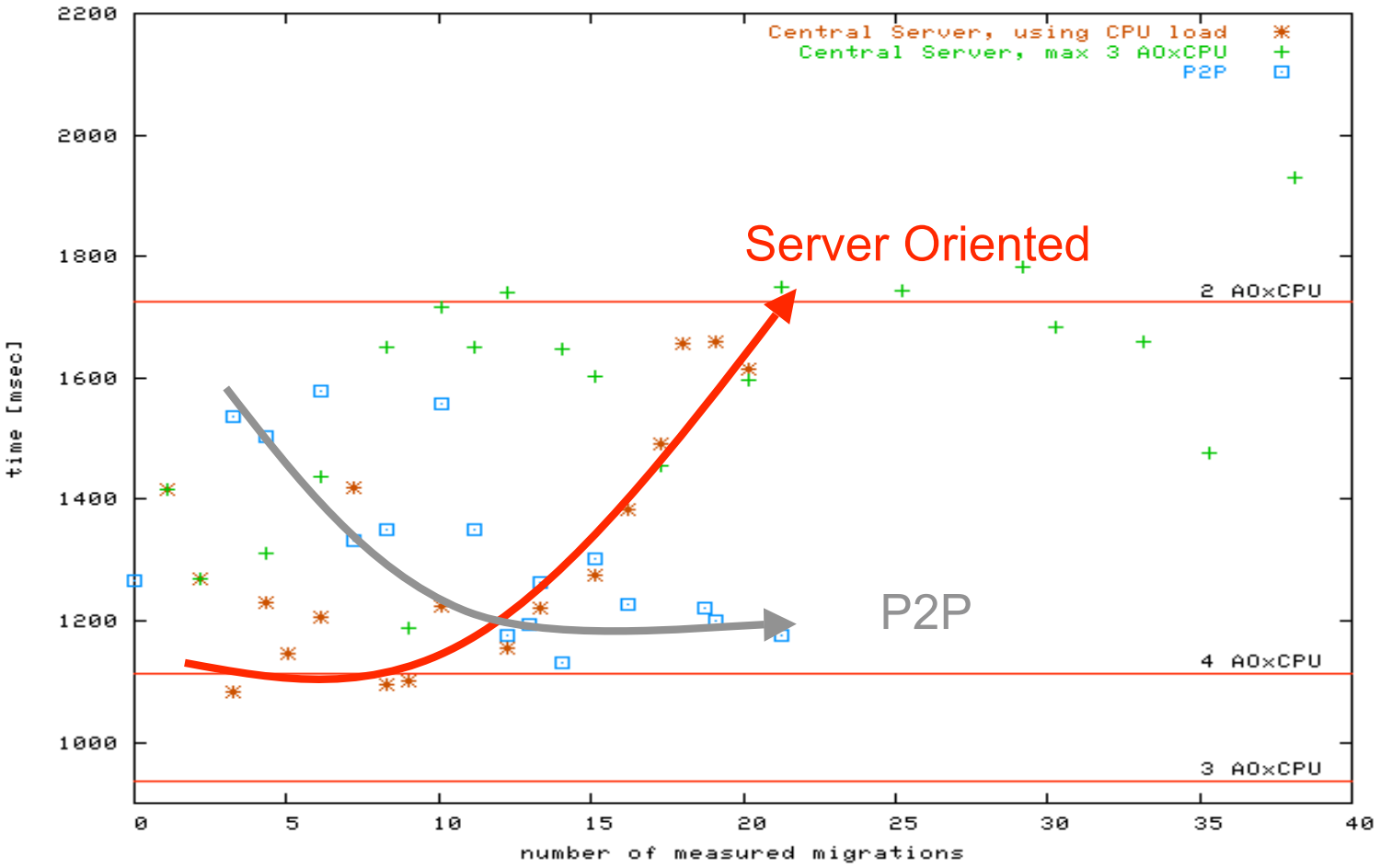
Jacobi's Iteration

- To solve linear system of equations $Ax = b$.
- Having $x_{k} = D^{-1}(L+U)x_{k-1}+D^{-1}b$, where the matrices D, -L and –U represent the diagonal, strictly lower triangular and strictly upper triangular parts of A respectively.
- Stopping when $|| x_k – x_{k-1} || < \varepsilon$

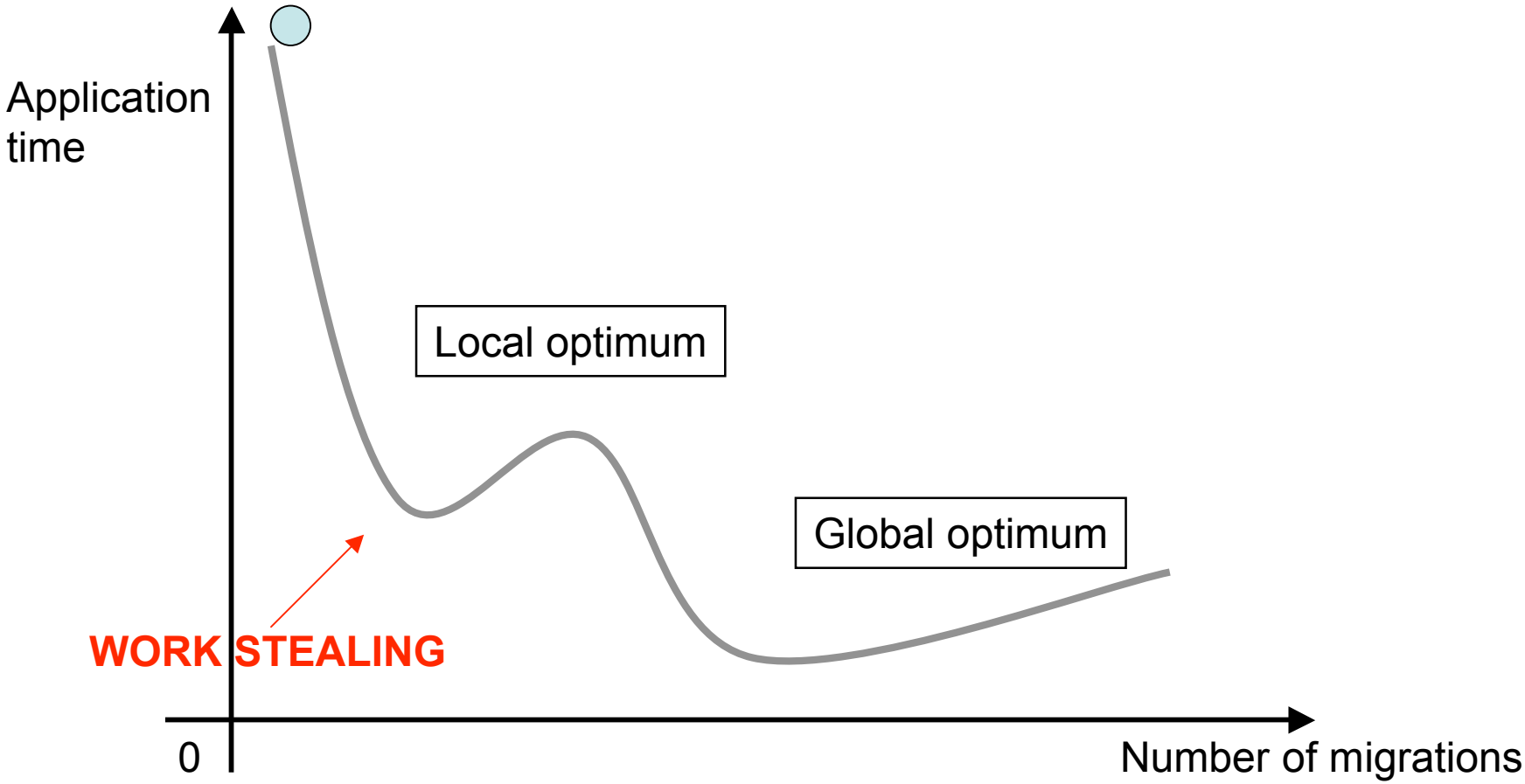
Test of P2P-LB Algorithm

- Load = [0,1] (% used CPU)
- Underloaded Threshold = 0.3
- Overloaded Threshold = 0.8
- Number of neighbors to ask = 3
- Update time = 5 + 30 *t* (1 – load) [sec]
  - *t* follows an uniform distribution
- *Rank = CPU's speed*
- 25 Machines (from 0.5 to 3.4 GHz)
- 36 Active Objects

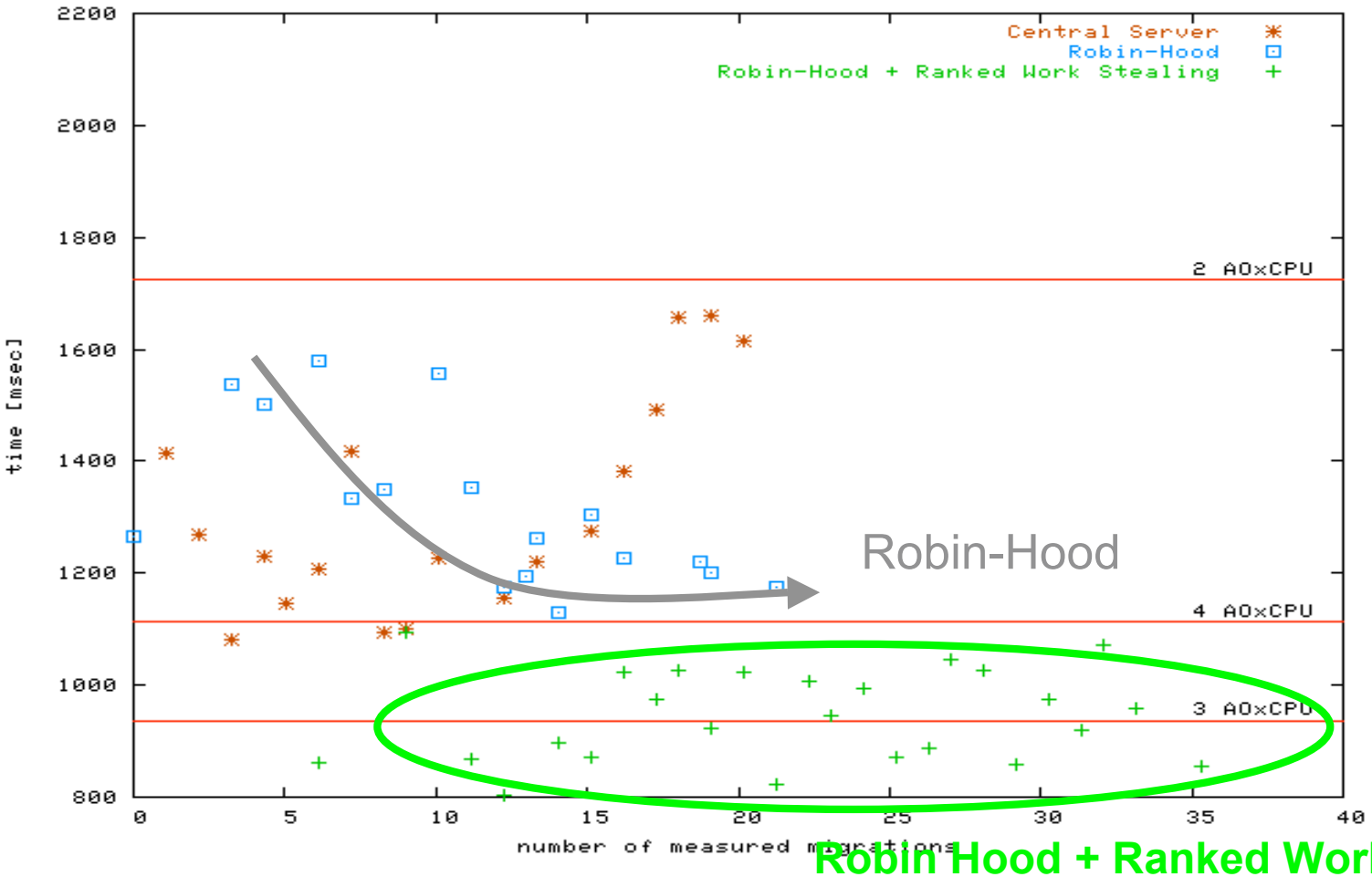
Load Balancing Benchmarks



Local to Global optimum



Ranked Work Stealing (*)



Robin Hood + Ranked Work Stealing

(*) I will steal only if I am a better machine than my target

Conclusions

- New Load Balancing for P2P Architecture was developed
- Algorithm exploits the P2P infrastructure to speed up migration time, so application time
- Preliminary results seem to be promising
- More research is needed

Questions?