POPWIN
Parallel Object Remote Programming for Wireless Network over IPv6

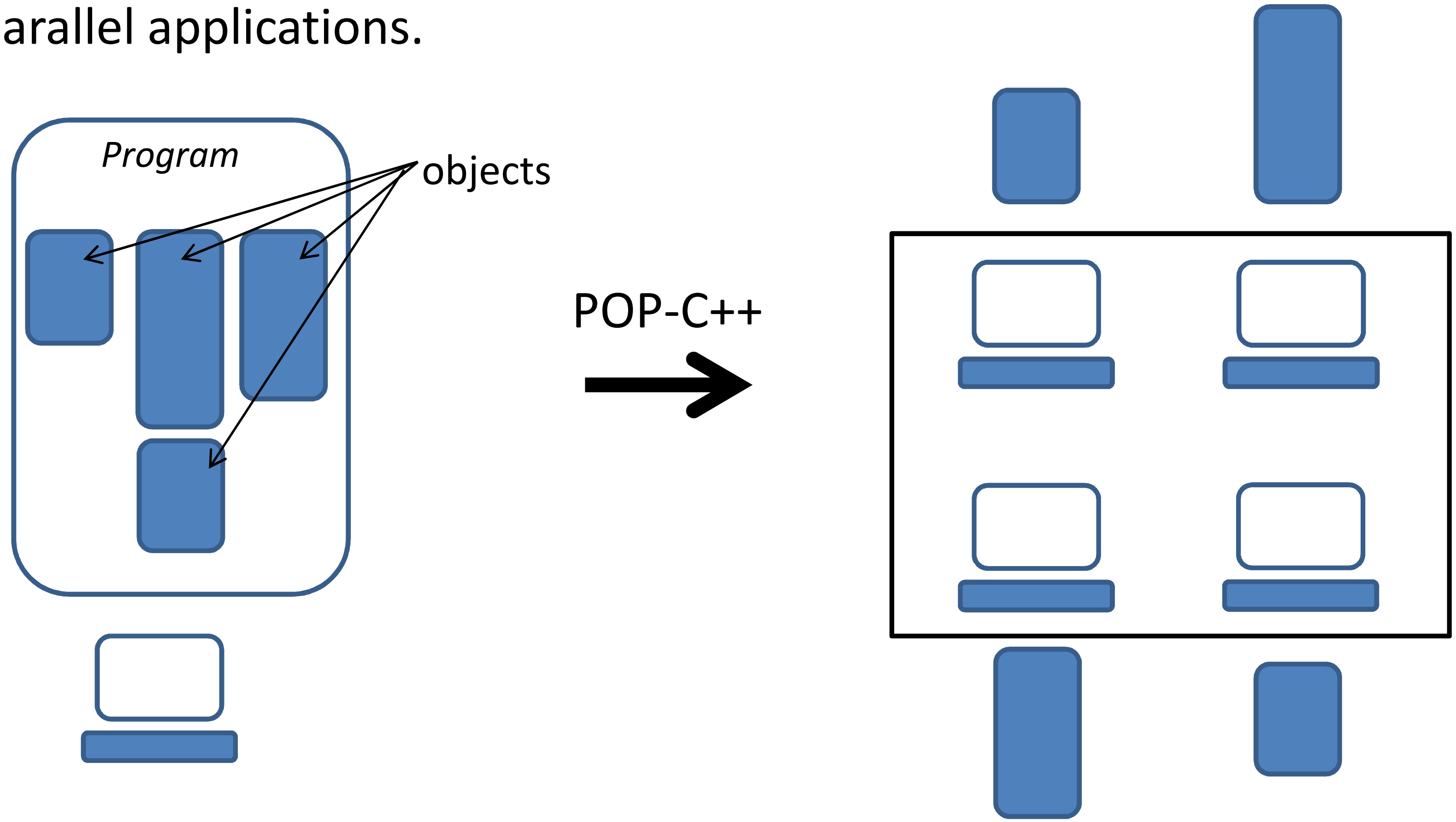
Pierre Leone

University of Geneva

Project in collbaration with Pierre Kuonen, EIA-FR/Fribourg

Project goal

POP-C++ (POP-Java) is an **object-oriented** system for programing parallel applications.



Project goal

The goal of the project is to develop POPWIN an **object-oriented system to program wireless sensor network**.

EIA-FR/Fribourg	University of Geneva
Pierre Kuonen Yao Lu	Pierre Leone Cristina Muñoz
Model of programming Implementation of POPWIN Application	Network primitives Energy balance mechanisms IPv6 – 6LowPan

Network primitives

Executing an object requires some resources

- Computing resources
- Sensory resources – temperature, pressure, etc.
- …

Classical resources might be extended resources particular to sensor network

- Location
- Geographical constraints

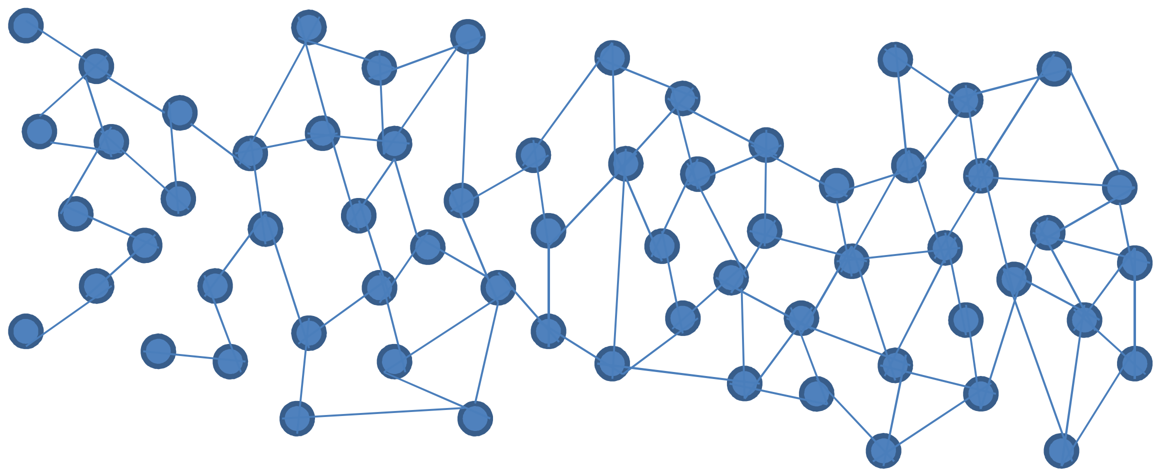
Network primitives

First problem: Given a set of sensors that communicate wirelessly, find the appropriate resources in the network.

General approach:

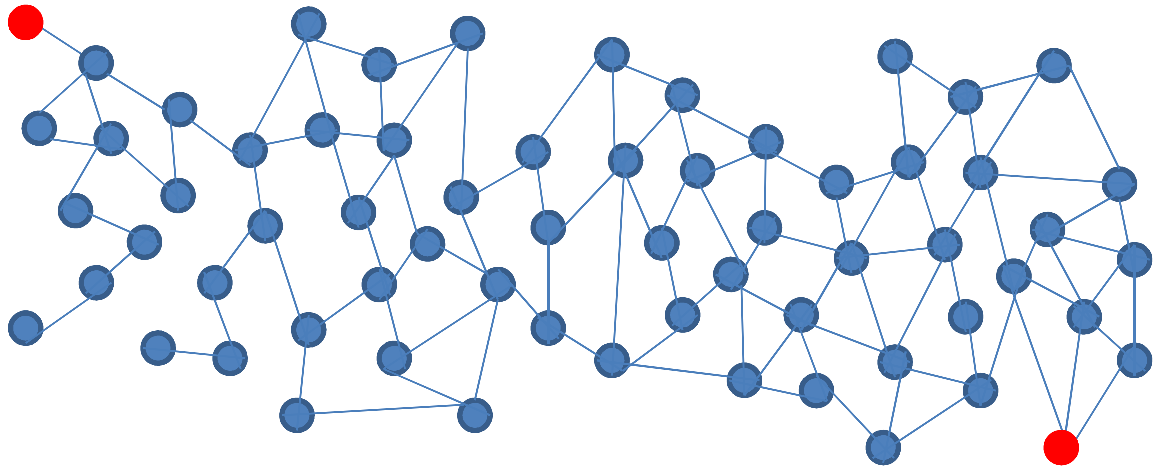
- Use the context of Publish/Subscribe system:
  - Publishers advertise about the resources they offer
  - Subscriber search for resources
- Cooperative system
  - Publishers and subscribers participate to the process of matching demands
  - All the nodes in the network participate

Matching Pub/Sub

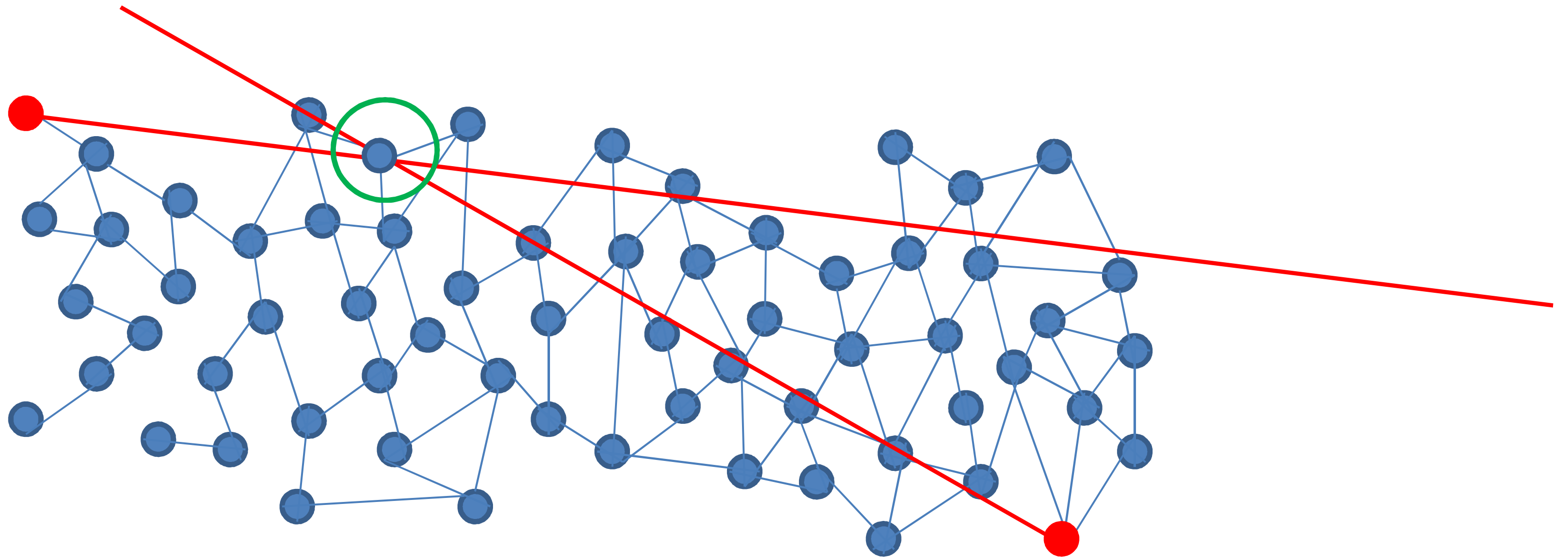


The sensors are located in a plane

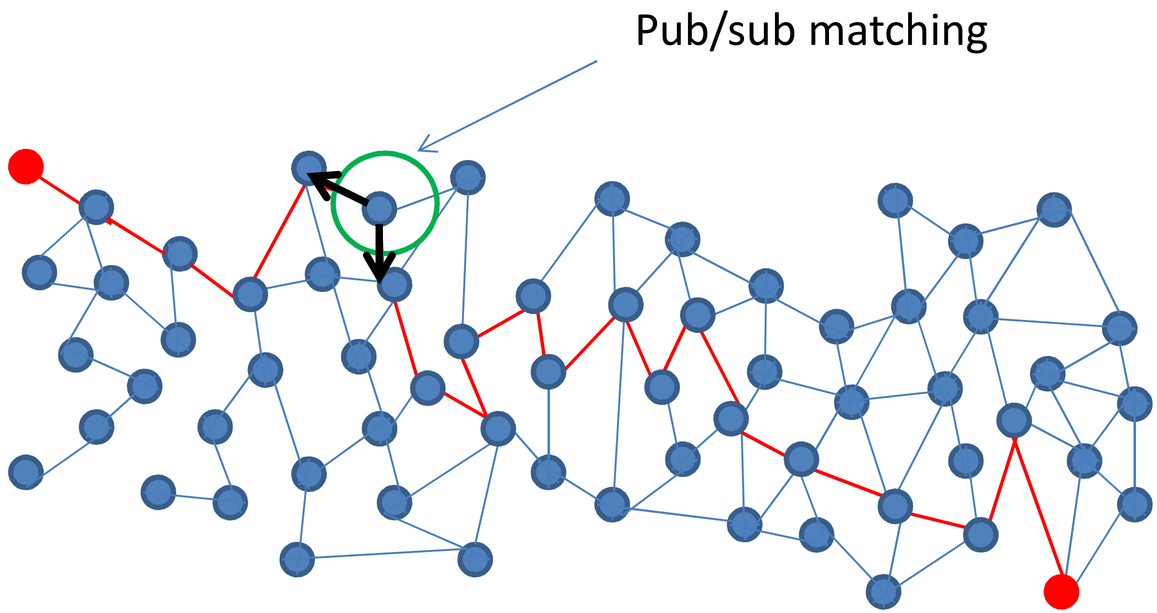
Matching Pub/Sub - Heuristic



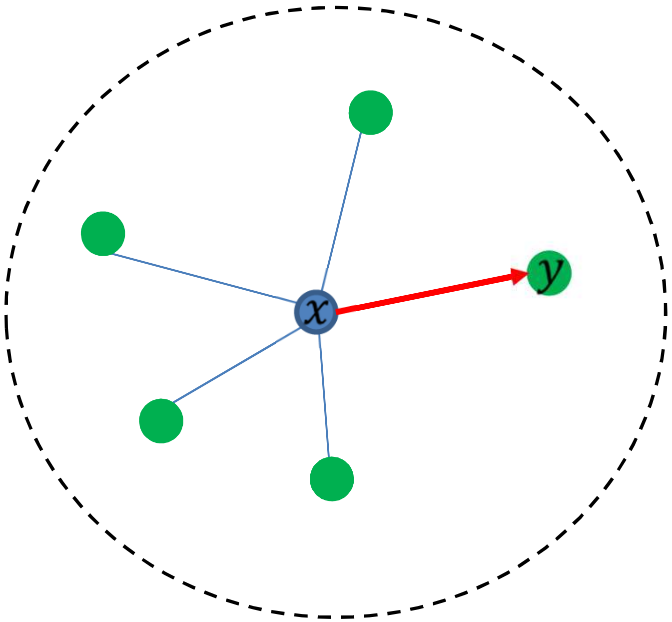
Matching Pub/Sub - Heuristic



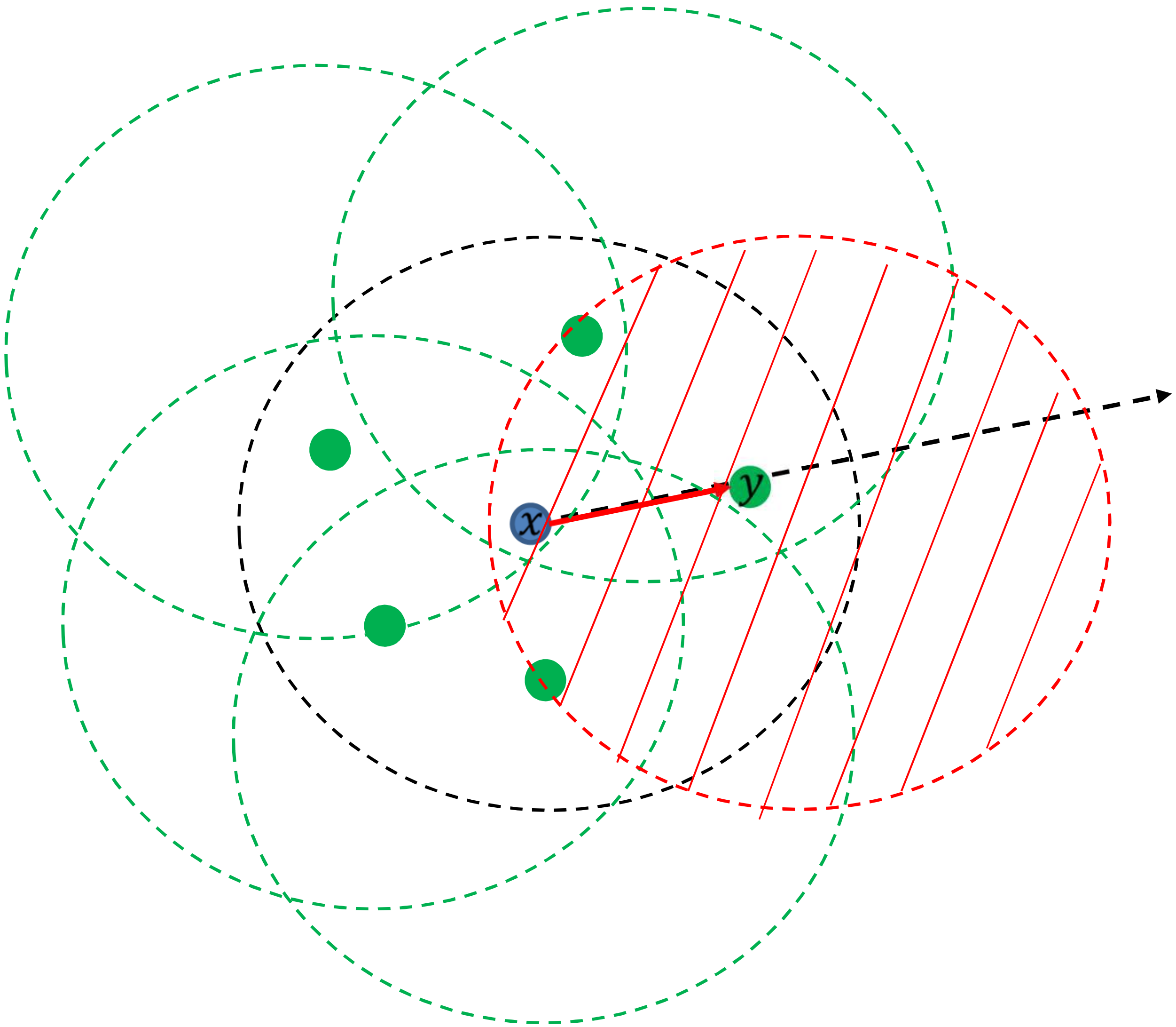
Matching Pub/Sub - Heuristic



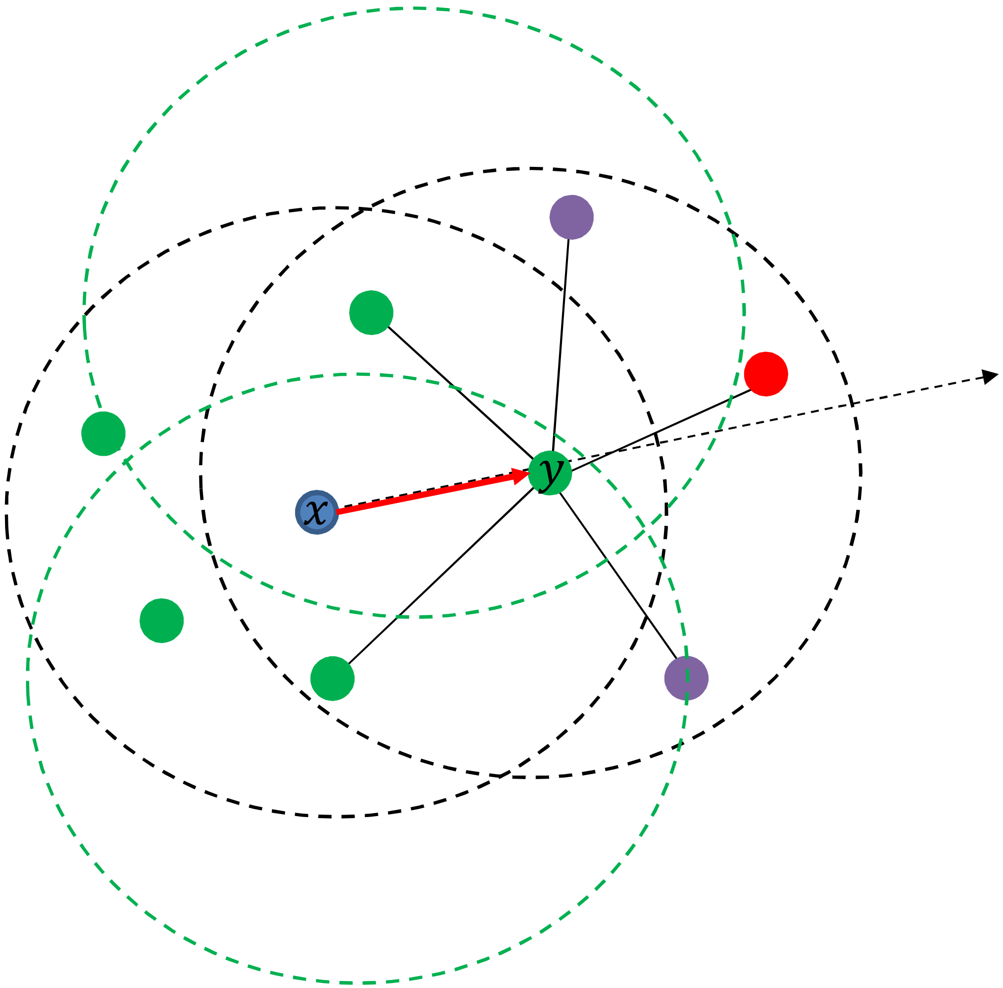
Directional Random Walk



Directional Random Walk



Directional Random Walk



1. Select the node $v \in N_y$ Such that the number of 2-hops paths from x to v is minimal, i.e.

$$\boxed{argmin_{v \in N_y} \mid N_v \cap N_x \mid}$$

2. Introduce a penalty to the nodes that are in N_x

3. Once a node is chosen add a random penalty

Directional Random Walk

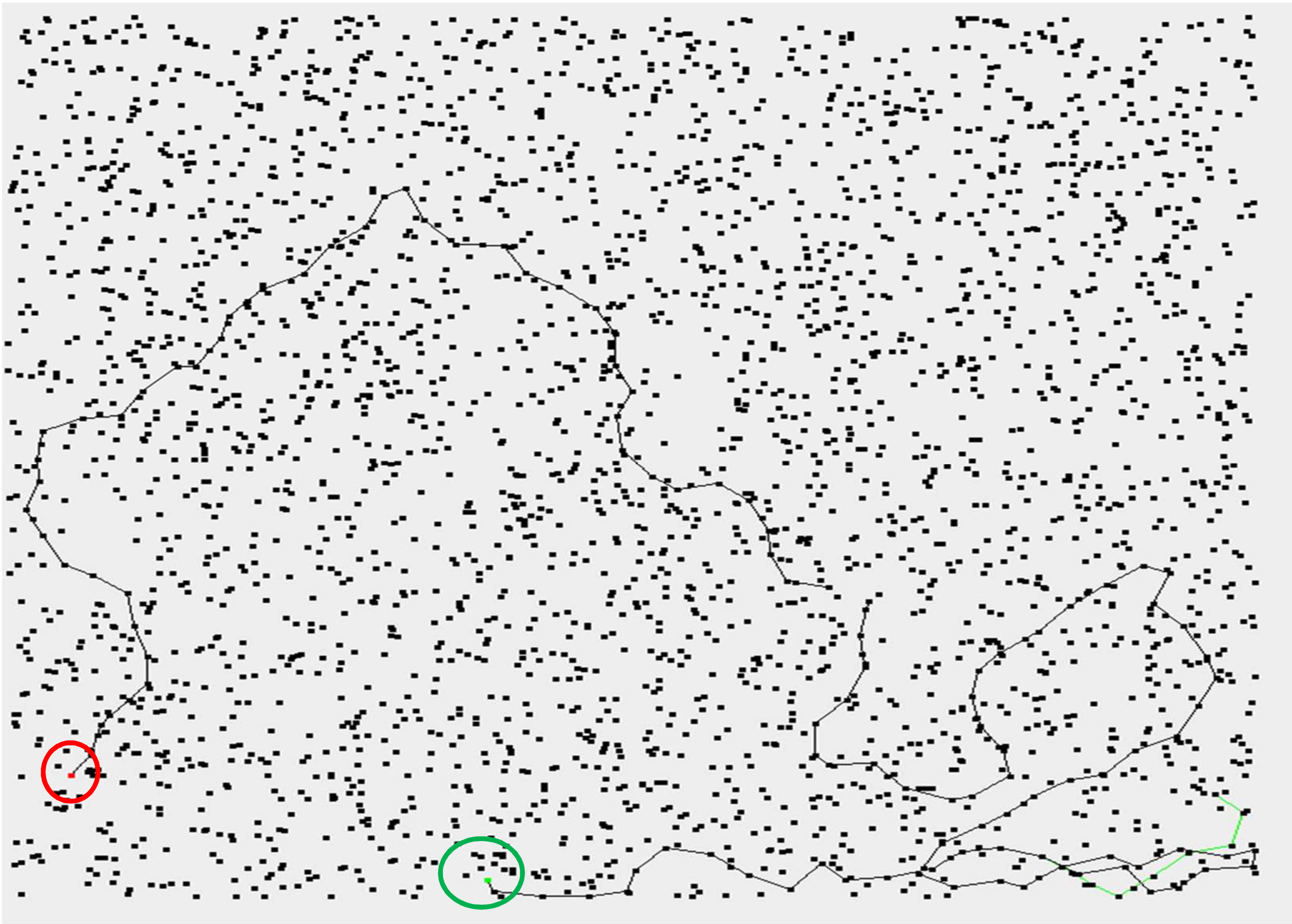
The first time the node y receives the notification it :

- Memorizes the trace of the path
- Memorizes the node x from which it (first) receives the notification

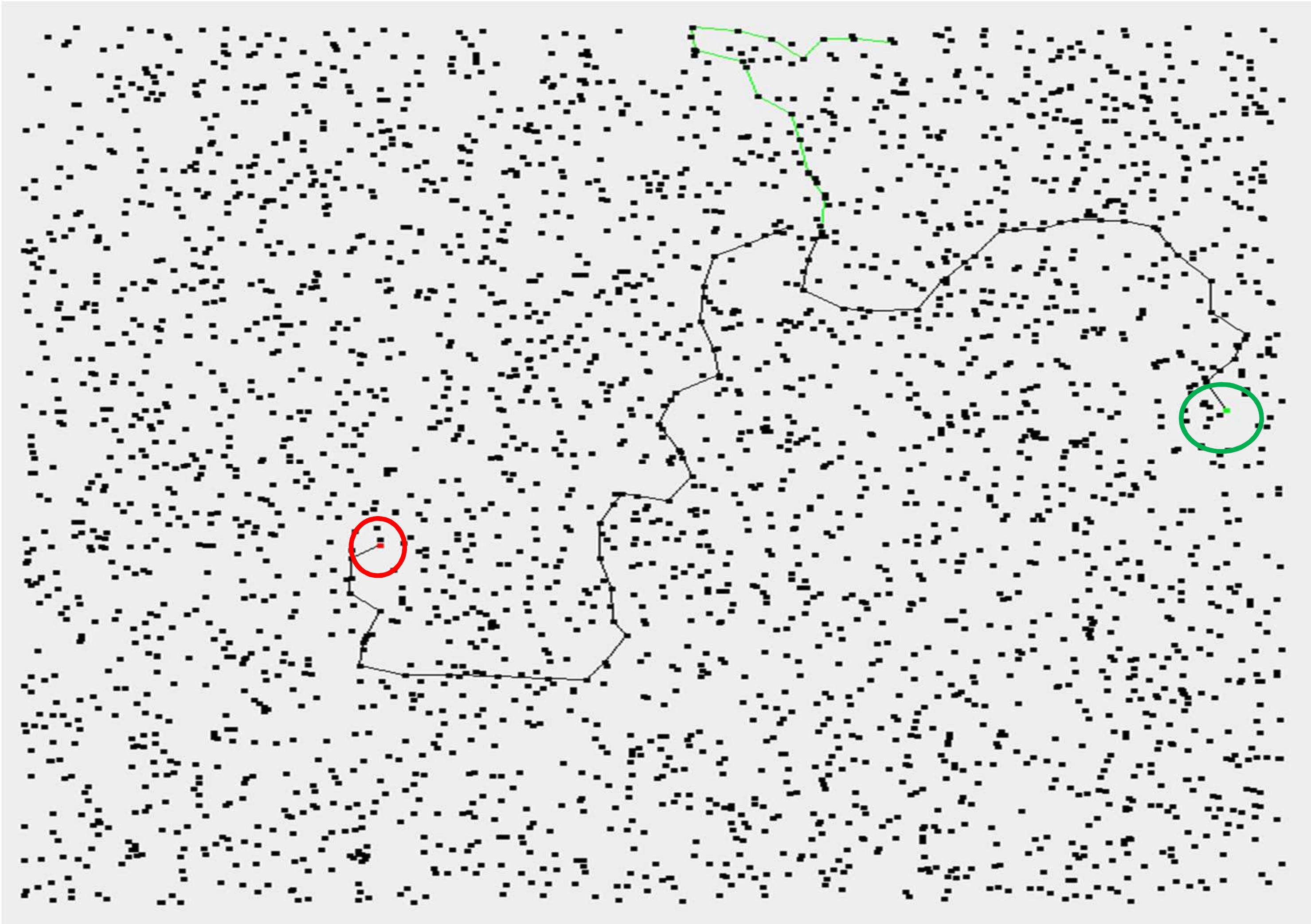
Properties :

- Covers the entire graph because of the random penalty
- The backward path is loop-erased

3000 nodes, mean number of neighbors 15



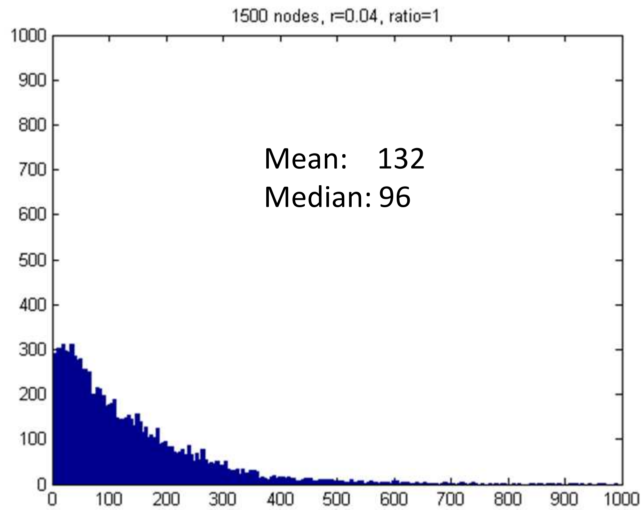
3000 nodes, mean number of neighbors 15

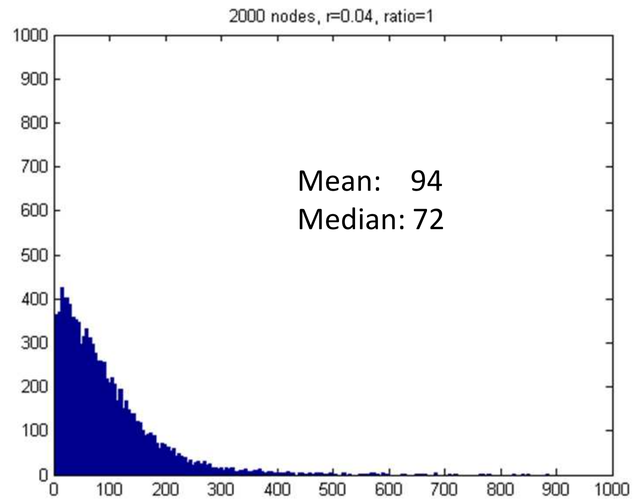


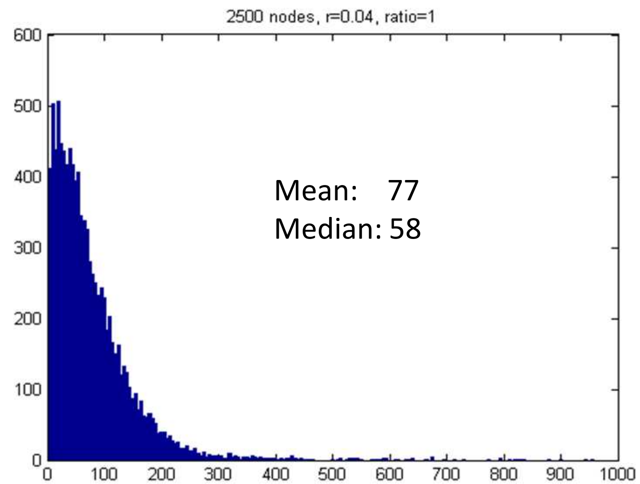
Time to intersection

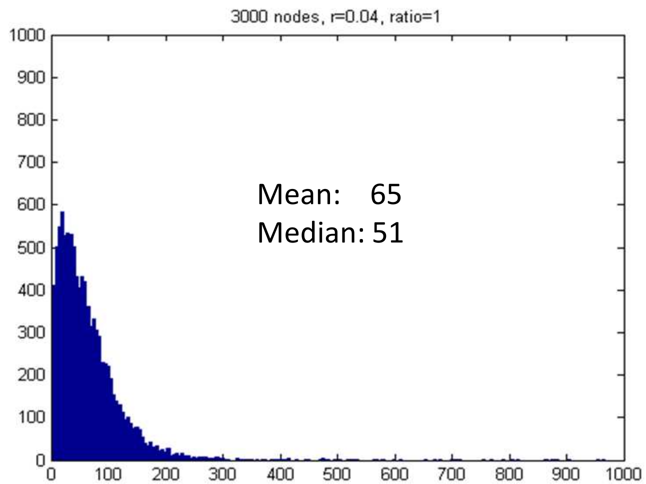
Measure: Two nodes start the process and synchronously process until intersection.

Time to intersection





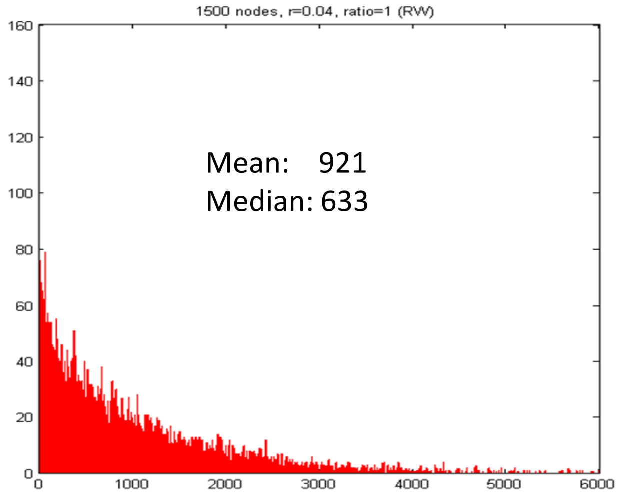


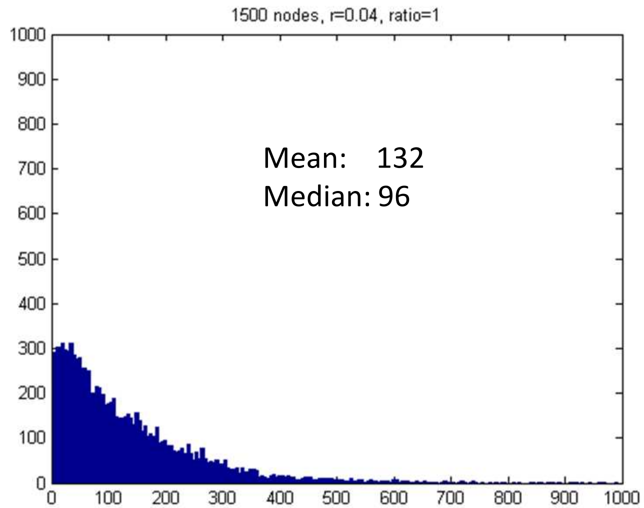


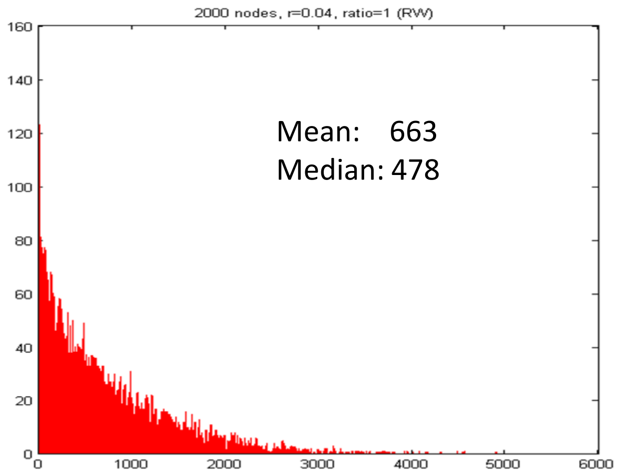
Time to intersection Comparison with RW

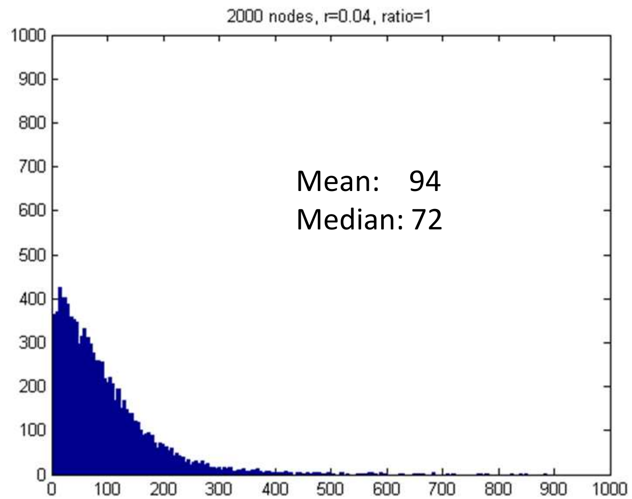
Measure: Comparison of the time to intersection against a pure Random Walk strategy

Time to intersection - RW

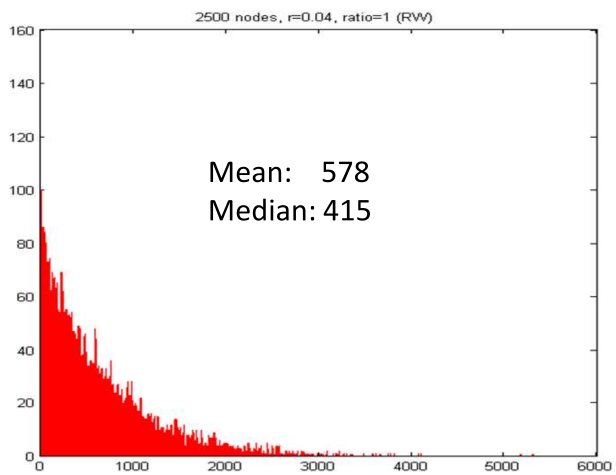


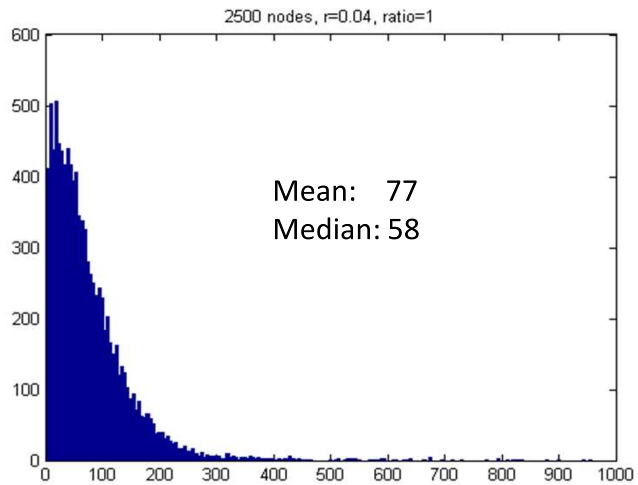




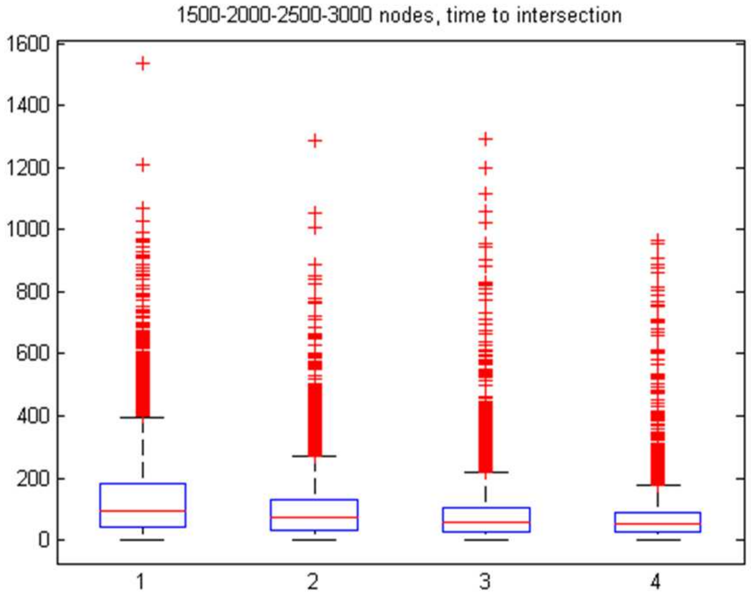


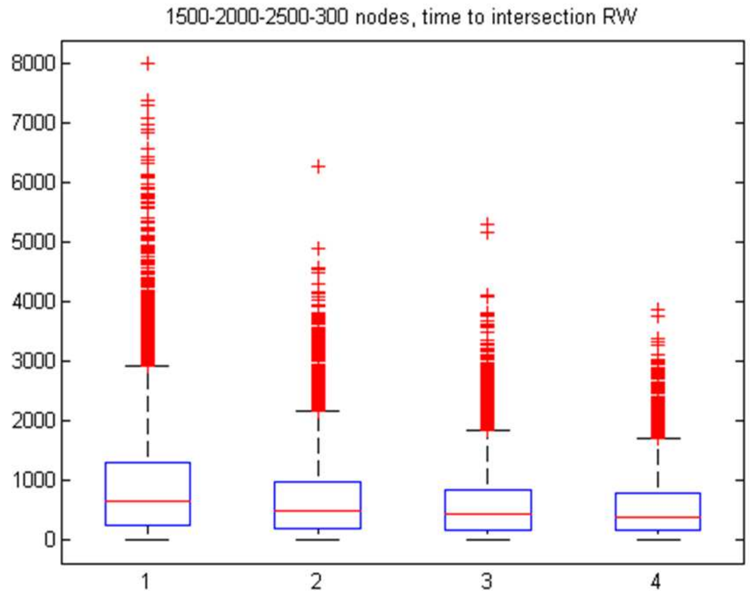
Time to intersection - RW





Time to intersection



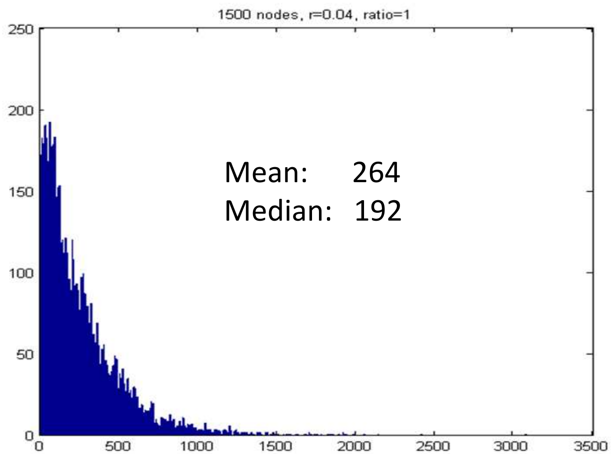


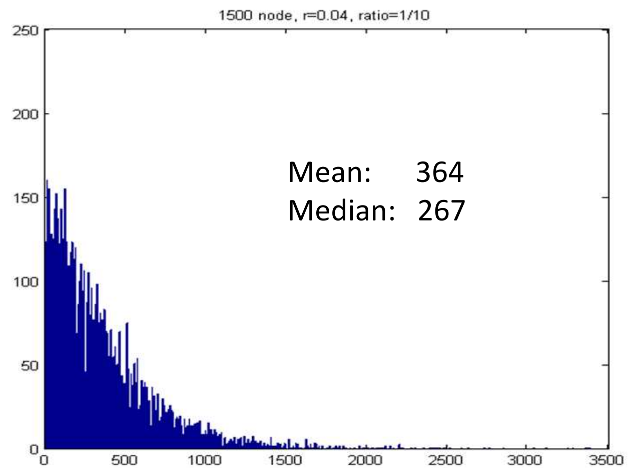
Conclusion 1: Directionality shorten the time to intersection

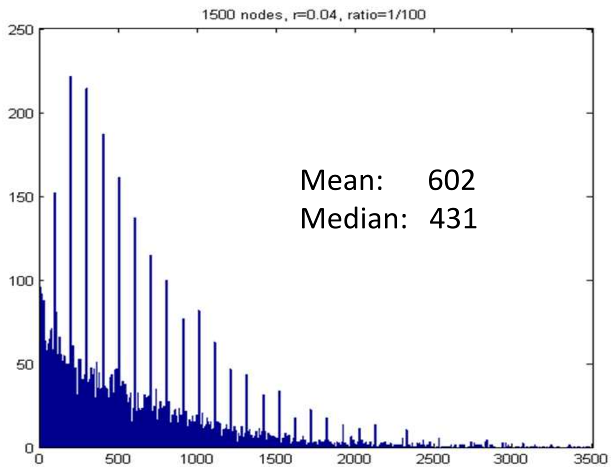
Time to intersection Asynchrony

Measure : The publisher and subscriber are not synchronous, i.e. they are **working at different speeds**.

Impact of asynchrony on the intersection time



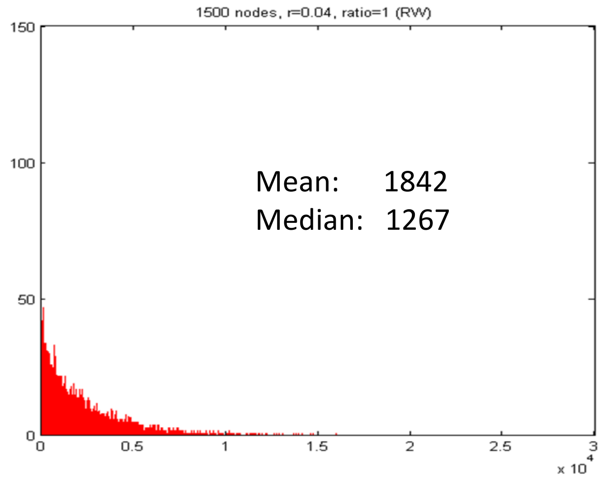


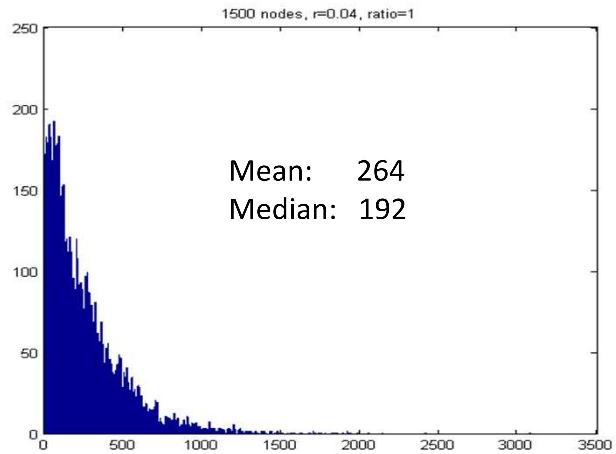


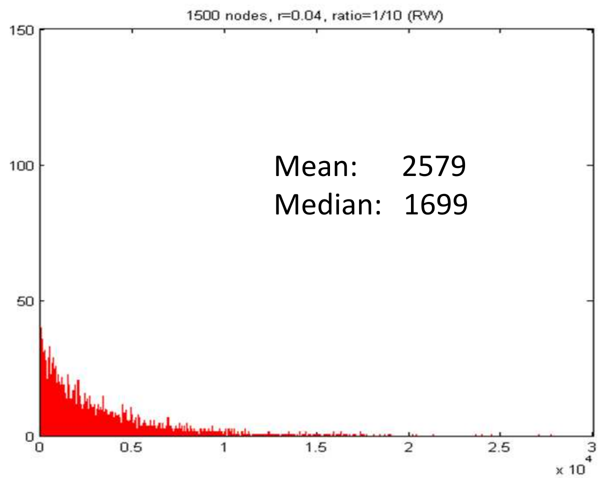
Impact of asynchrony on the intersection time

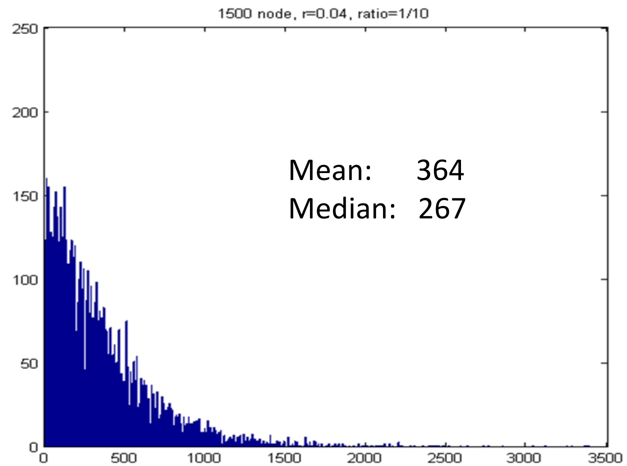
Comparison with RW

Impact of asynchrony on the intersection time

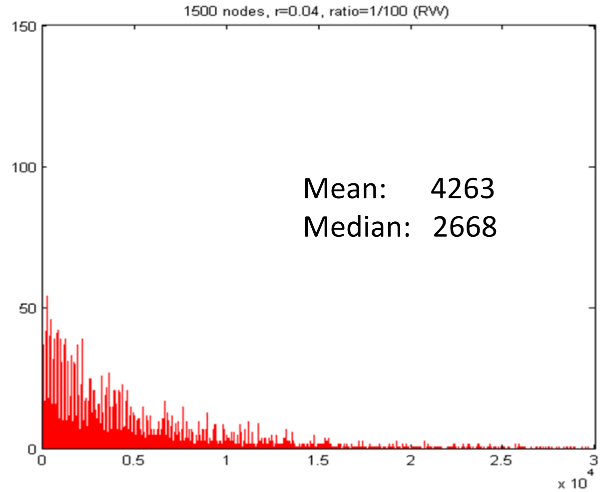


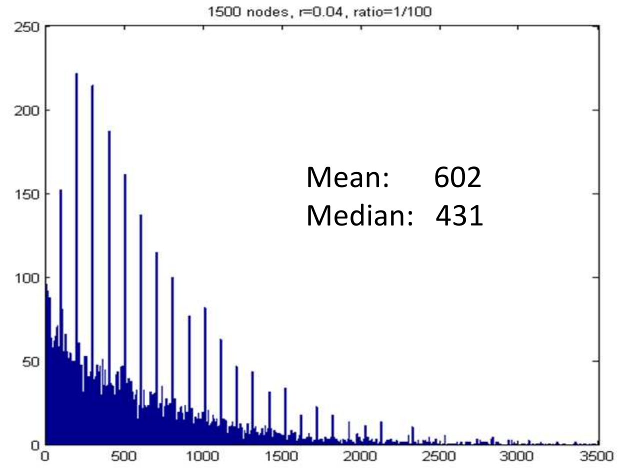






Impact of asynchrony on the intersection time





Conclusion 2: Cooperation shorten the time to intersection

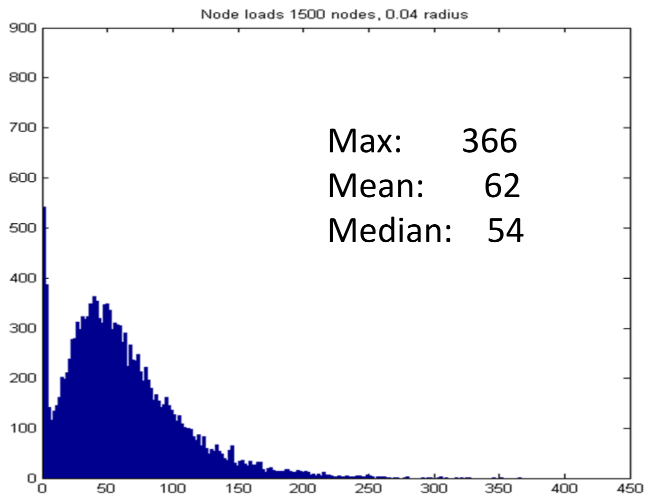
Routing efficiency

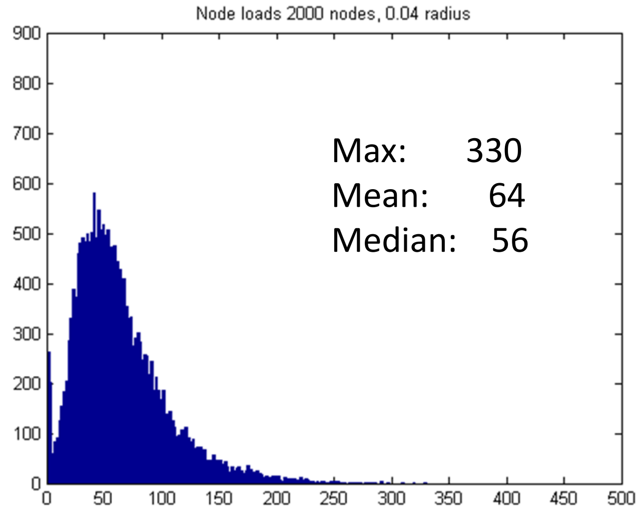
Measure: We consider a **permutation** of the nodes and for each permutation we execute the process and compute the number of path that pass through each nodes (the load).

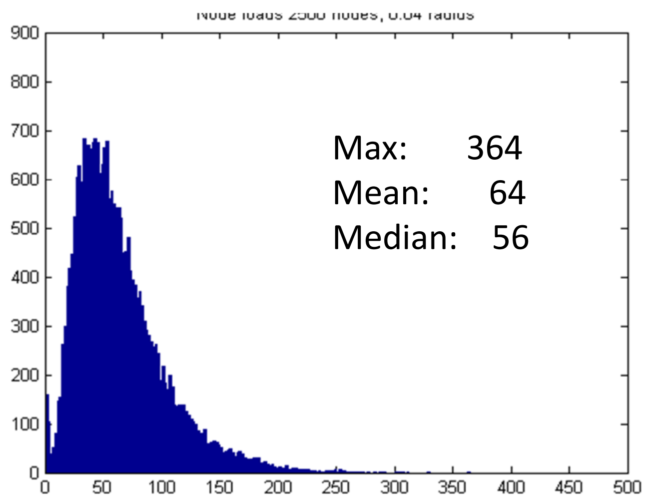
Node number:	0	1	2	3	…	n
	↓	↓	↓	↓		↓
Permutation:	perm[0]	perm[1]	perm[2]	perm[3]	…	perm[n]

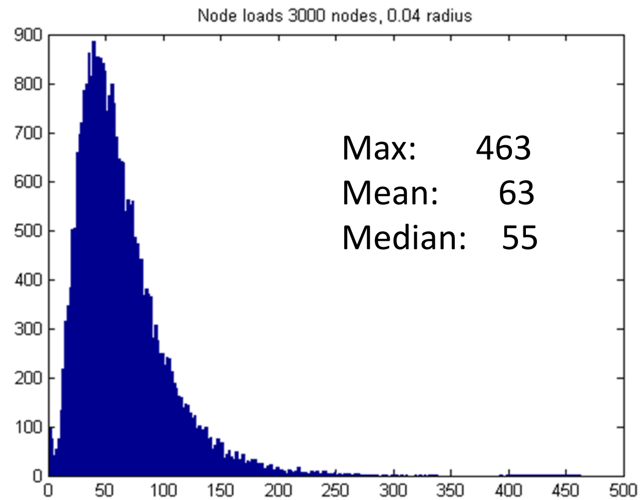
Nodes load

Nodes load


Node loads 1500 nodes, 0.04 radius
Max: 366
Mean: 62
Median: 54

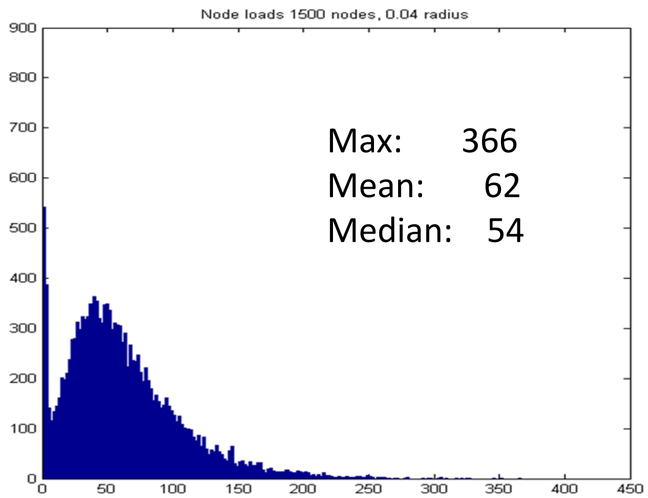

Node loads 2000 nodes, 0.04 radius
Max: 330
Mean: 64
Median: 56

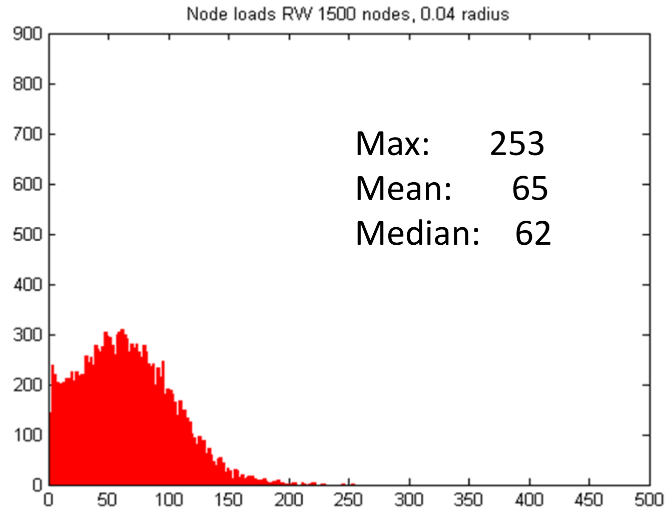

Max: 364
Mean: 64
Median: 56

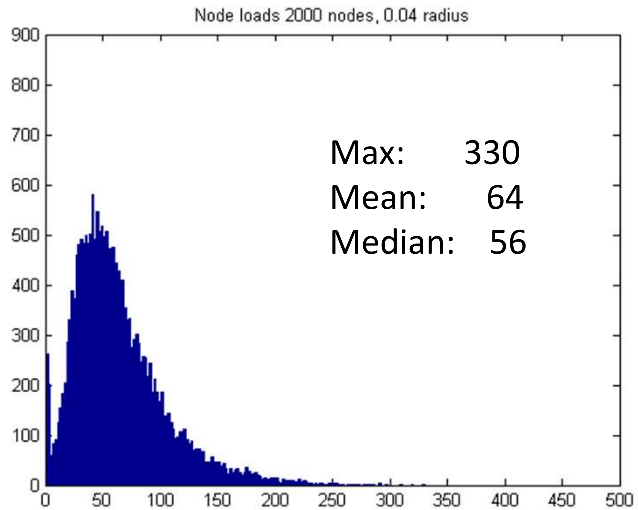

Node loads 3000 nodes, 0.04 radius
Max: 463
Mean: 63
Median: 55

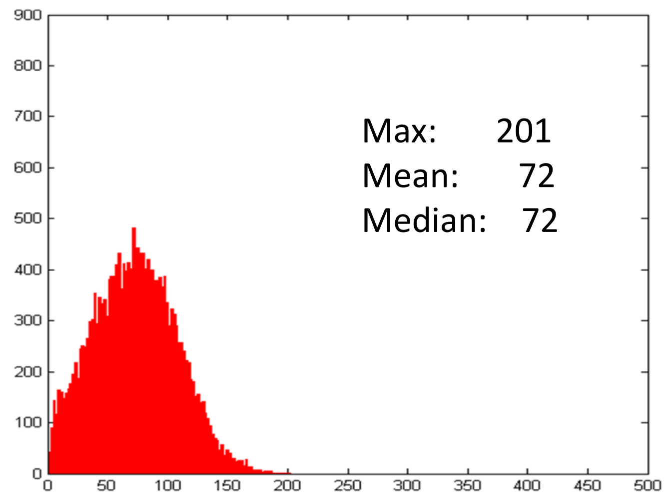
Nodes load
Comparison with RW

Nodes load - RW

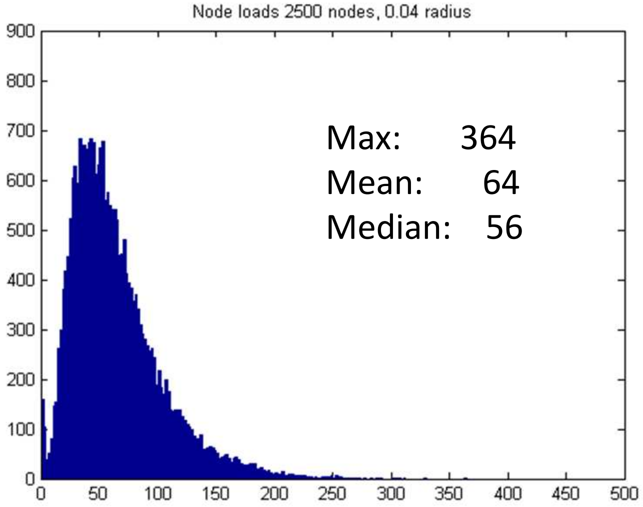


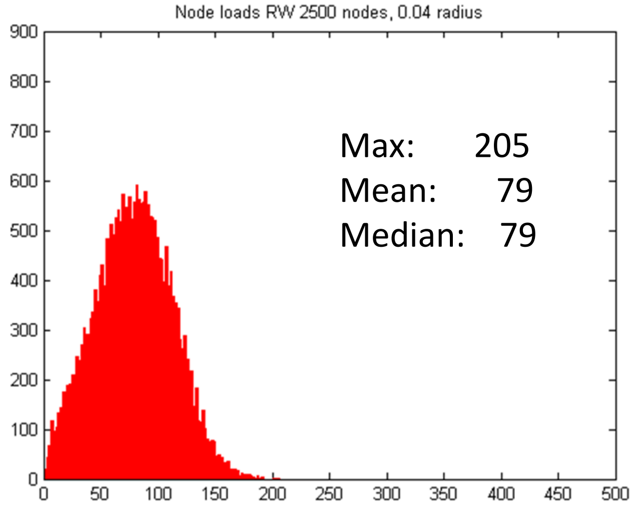


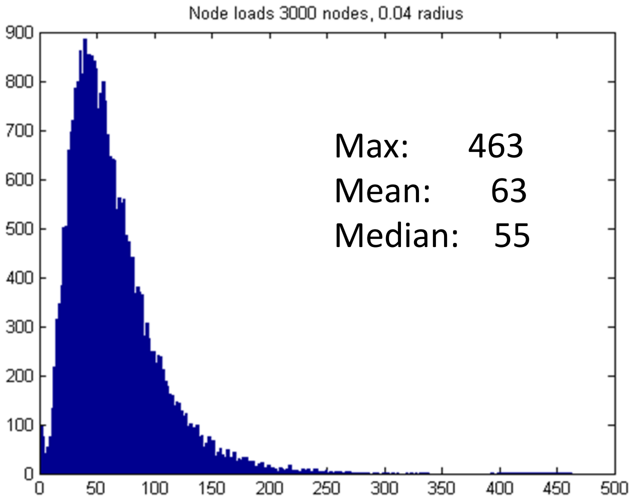


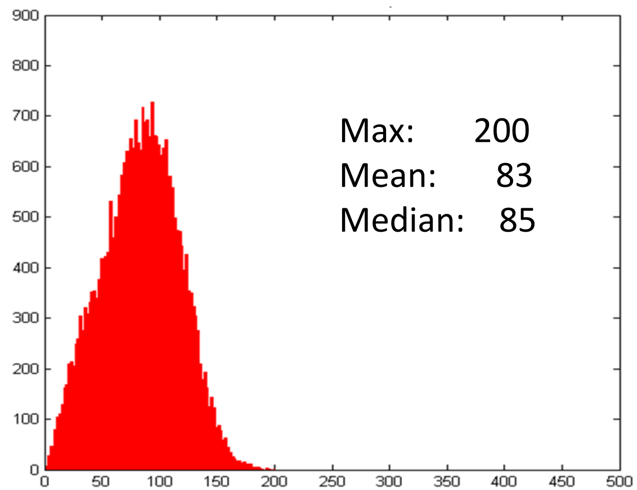


Nodes load - RW



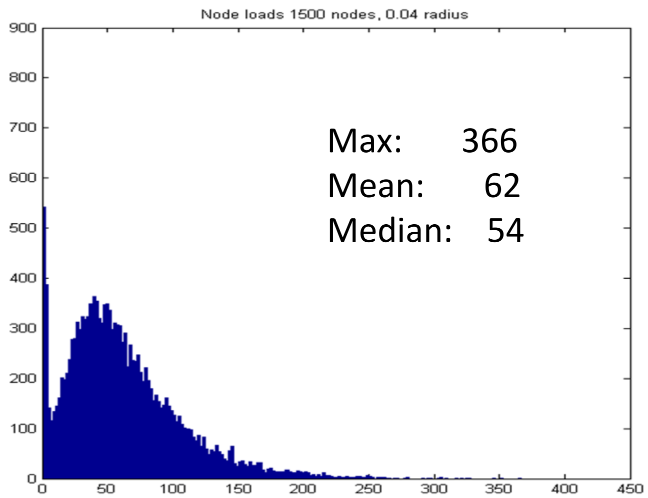


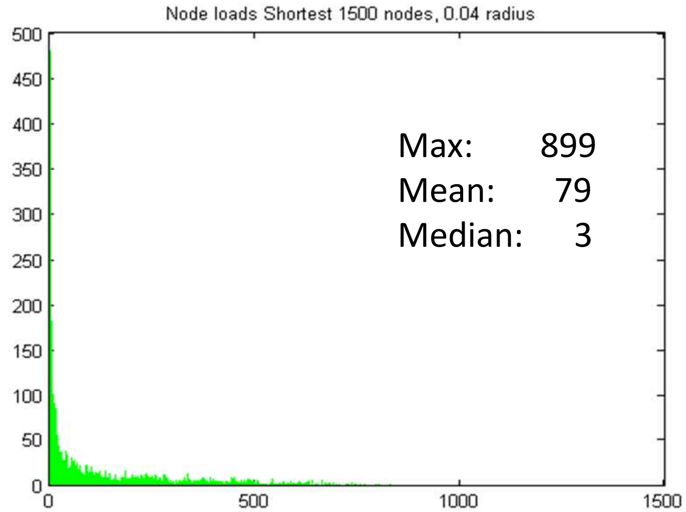


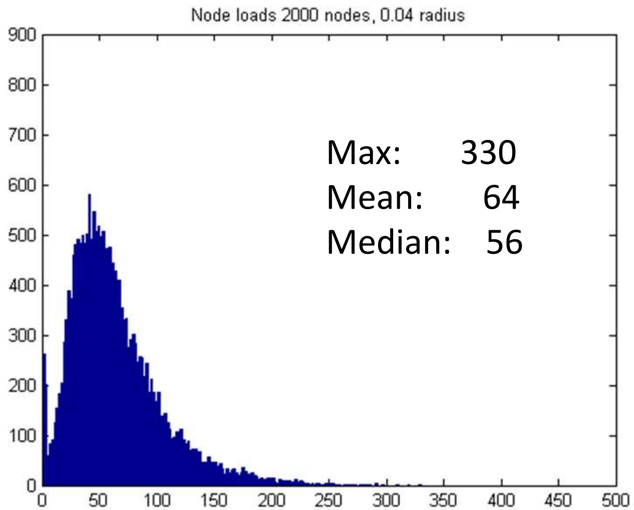


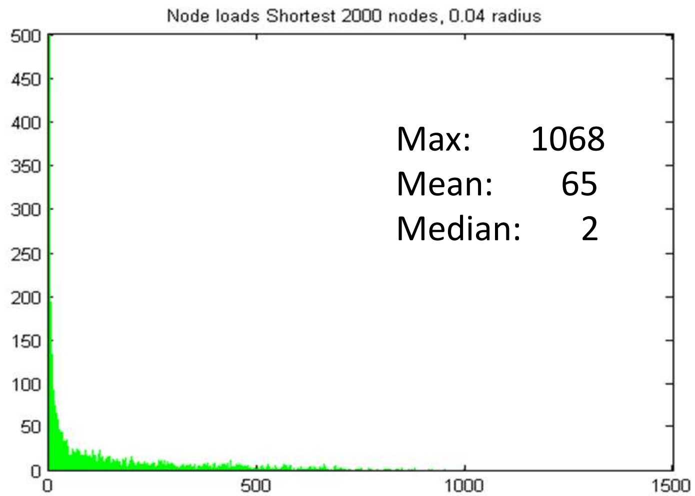
Nodes load
Comparison with Shortest path

Nodes load - RW

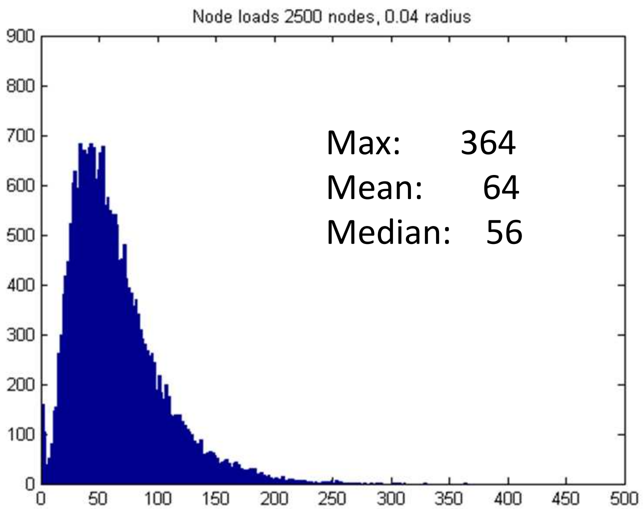


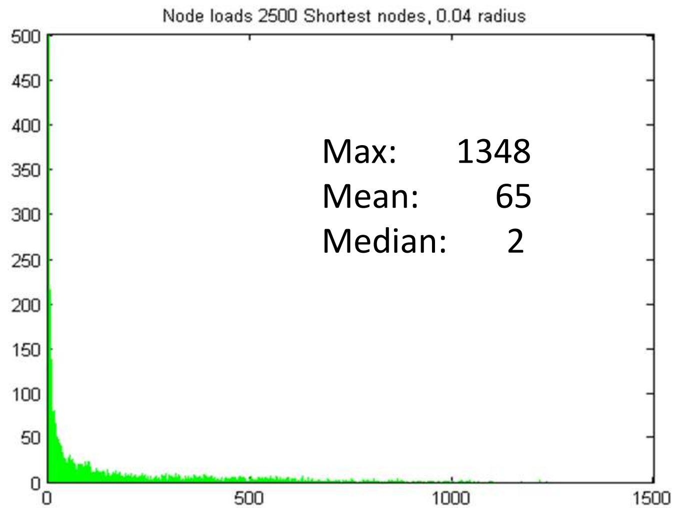


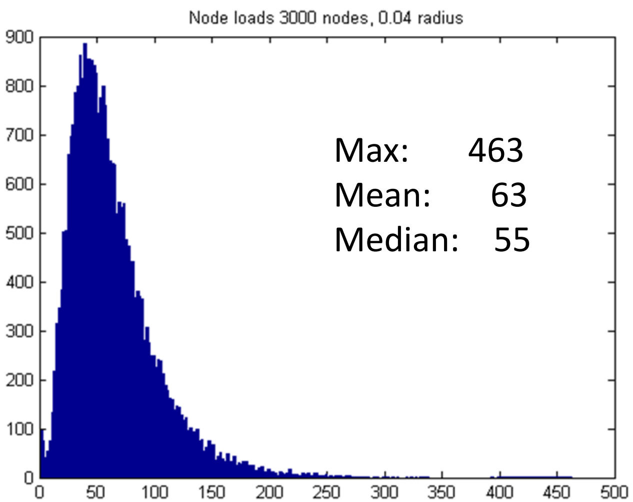


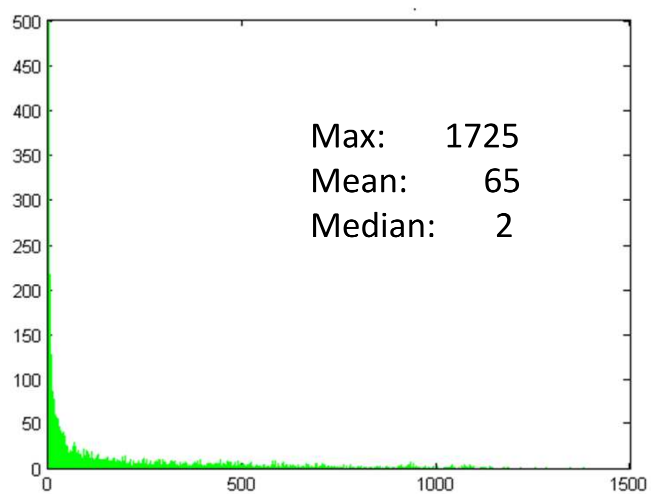


Nodes load - RW









Nodes load

Conclusion 3: The mechanism balances the number of path that pass through the nodes as efficiently as a Random Walk.

Remarks :

- In the classical setting routing a permutation is a tool to estimate lower/upper bound on the performance of routing algorithms.

- In our setting routing the permutation makes possible to match publication/subscription. A subscription from a node x follows the path x -> perm[x] -> perm[perm[x]] -> … and, check at all intermediate nodes if the path to the publication is known.

Further short term research directions

- Investigate the performance where there are one publisher many subscribers.
- We need to find efficient heuristic to stop/restart exploration of the graph.
- Alternatively we plan to investigate the routing of a permutation with a local algorithm. One path starts and stops at the ‘right time’.
- Consider different graph structures, not necessarily modeling wireless networks.

POPWIN
Parallel Object Remote Programming for Wireless Network over IPv6

Pierre Leone

University of Geneva

Project in collbaration with Pierre Kuonen, EIA-FR/Fribourg

Comparison with previous works

- Directional Rumor Routing in Wireless Sensor Networks, the nodes are localized
- A Directional Gossip Protocol for Path Discovery in MANETs., estimation of the critical probability.
- Directed diffusion, …,
- Directional Gossip: Gossip in a wide Area Network
- Lightweight tracking algorithm
- Techniques based on Rendez-vous
- Directional work, gossip and broadcast
- Centrifugal random walk, node Sampling, uses a spanning tree