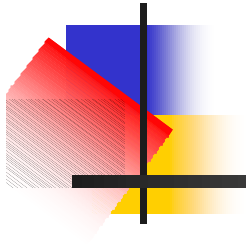
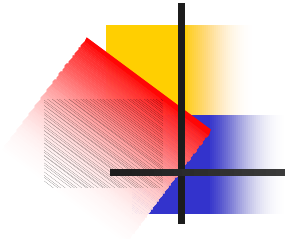


A Context-Aware Node Location Service for Ad Hoc Networks

Giovanni Turi
Istituto d'Informatica e Telematica - CNR Pisa
giovanni.turi@iit.cnr.it

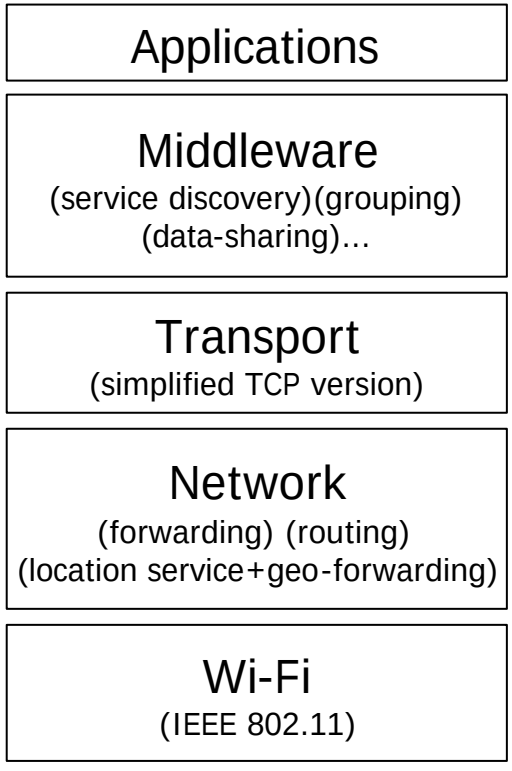
Joint work with *Marco Conti, Enrico Gregori*



Introduction

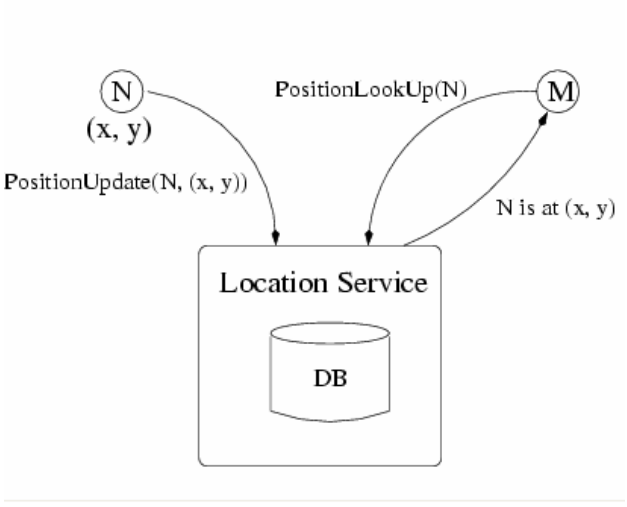
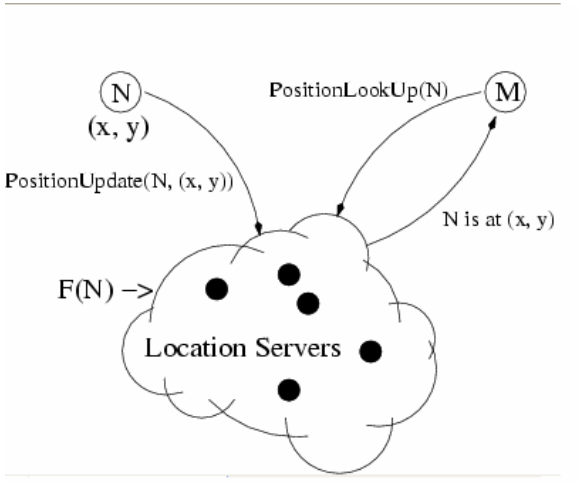
- Scenario:
  - large, i.e. metropolitan scale, mobile networks
- Problem:
  - find the actual (or approximate) position of a far away node to efficiently route data to it
- Working examples are GSM networks, see HLR/VLR
- In general, it’s a network layer issue…

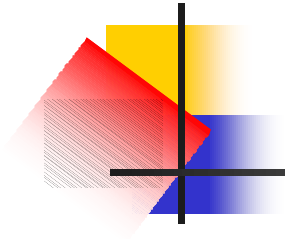
Ad Hoc networks



- Made of wireless and mobile nodes (pure model…)
- No infrastructure:
  - No access points
  - No centralized and managed network services
- Topology and available services are DYNAMIC
- A metropolitan-scale Ad Hoc network is supposed to span several KM^2 and be composed by thousands of nodes

Main issues

Classic Mobile Network	Ad Hoc Network
	
▪ Centralized position DB ▪ The network infrastructure maintains the DB	▪ No infrastructure implies that the position DB has to be distributed among the nodes ▪ Have a way to identify the location servers of a given node N (e.g., *F(N)*)

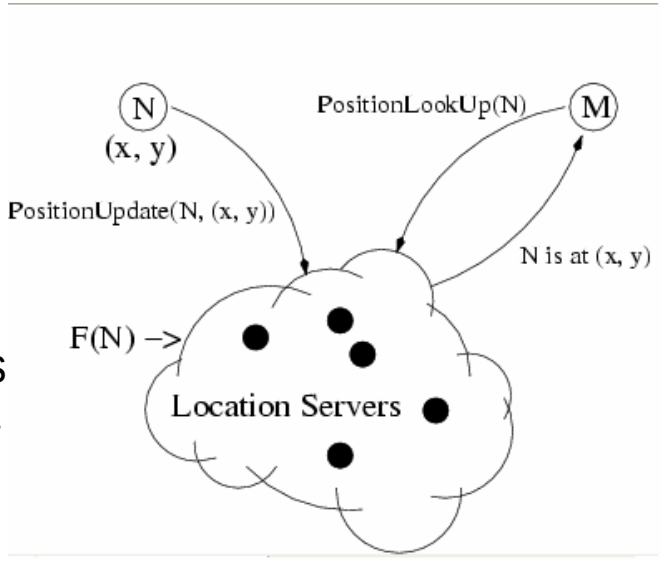


Main requirements

- Keep the overhead low
  - *Position DB distribution*: let each node in the network keep only a part of the position database
  - *Communication*: minimize flooding operations
- Face propely the dynamic of Ad Hoc networks
  - Cope with nodes mobility, variable network densities.....be *context-aware*

Proposed *all-for-some* approaches

- F(N) has often been specified as a static math rule working on node identifiers. Examples:
  - F(N) returns the coordinates of the area where location servers are located [1]
  - F(N) returns the address of a node that has to be the location server in a given subarea [2]

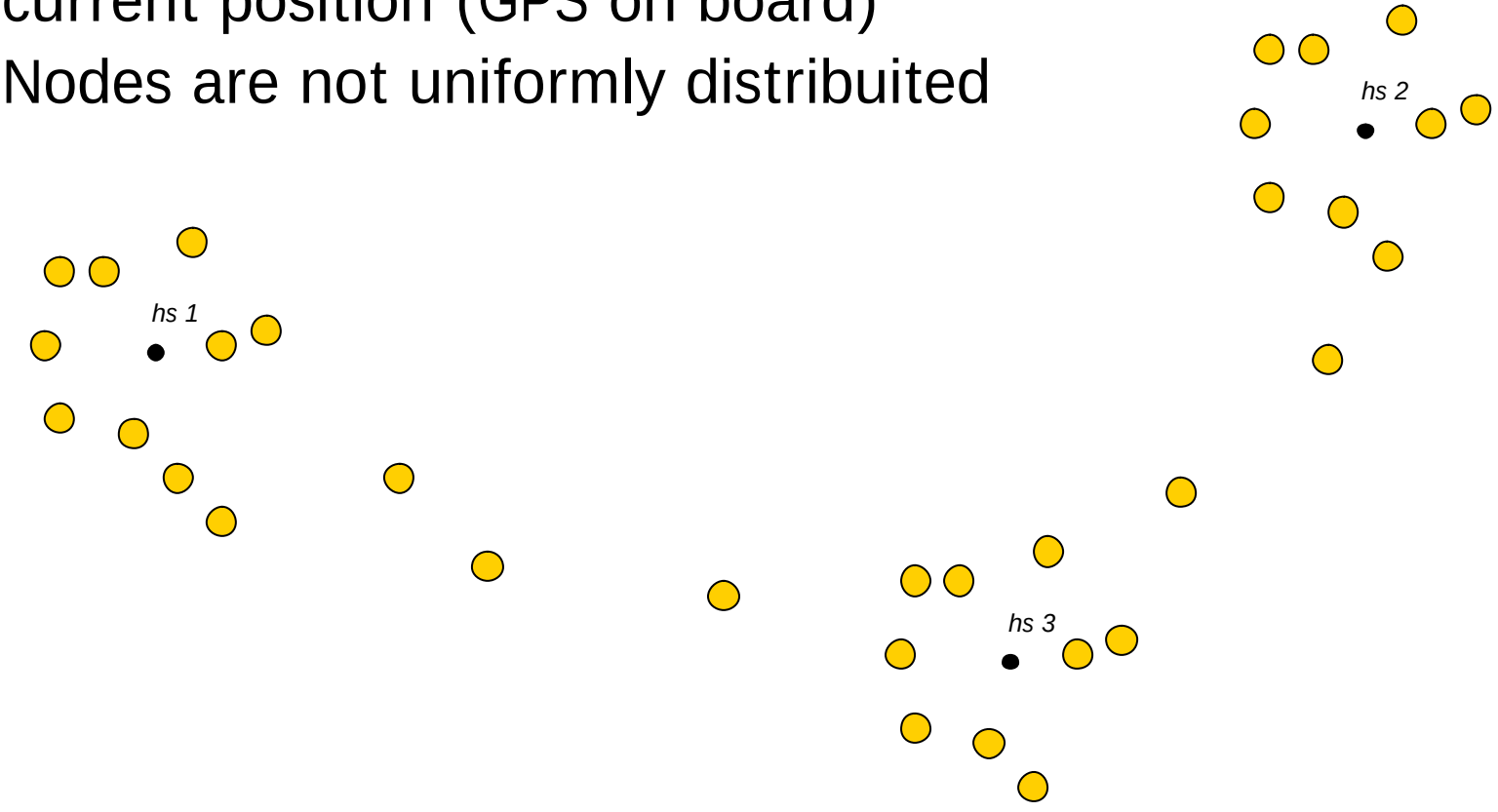


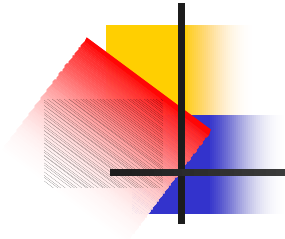
- Distribution of location servers:
  - All located in a same subarea [1]
  - Distributed all over the network, using a static grid partition [2]
- Assumption on a uniform distribuition of nodes over the network area

[1] “The Virtual Home Region”, EPFL
[2] “GLS: Grid Location Service”, MIT

Assumptions and nodes scenario

- Nodes have a way to retrieve their current position (GPS on board)
- Nodes are not uniformly distribuited





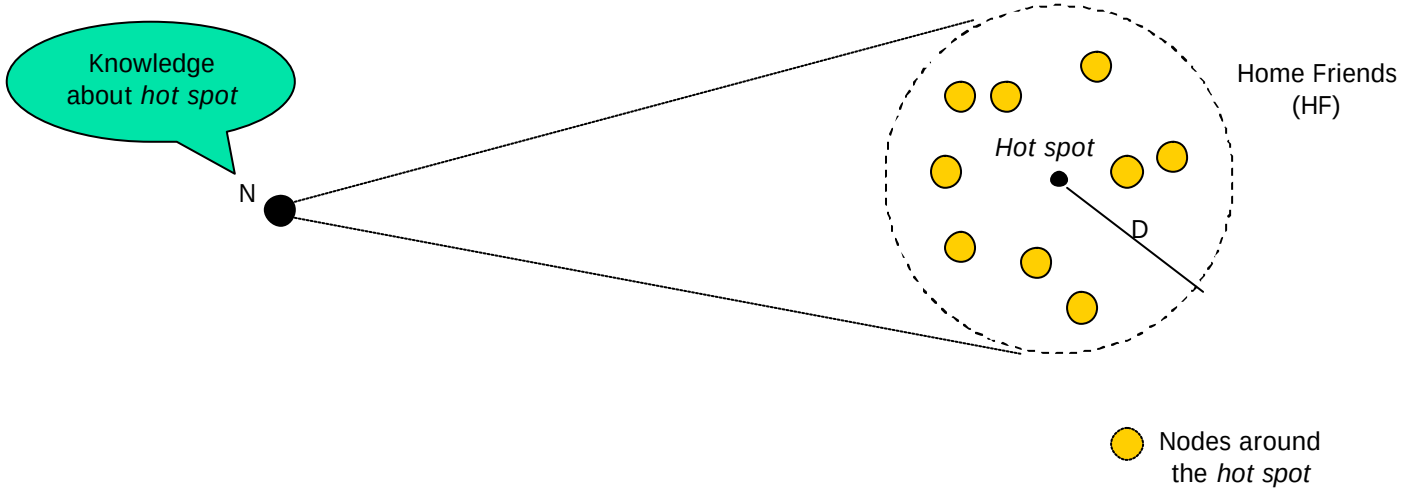
Our approach

- Use context information to identify location servers
  - Context is knowledge on *hot-spots*, points where nodes usually assemble (on a long term observation)
- This information could come:
  - Directly from users knowledge
  - From density statistics made by software running nodes themselves
- This works on large, metropolitan areas, where people moves daily along the same paths and towards the same points: offices, shopping malls etc.

The potential location servers

- Assume node N knows the coordinates of a *hot-spot*
- Identify the potential location servers of N as those nodes located around the *hot-spot*

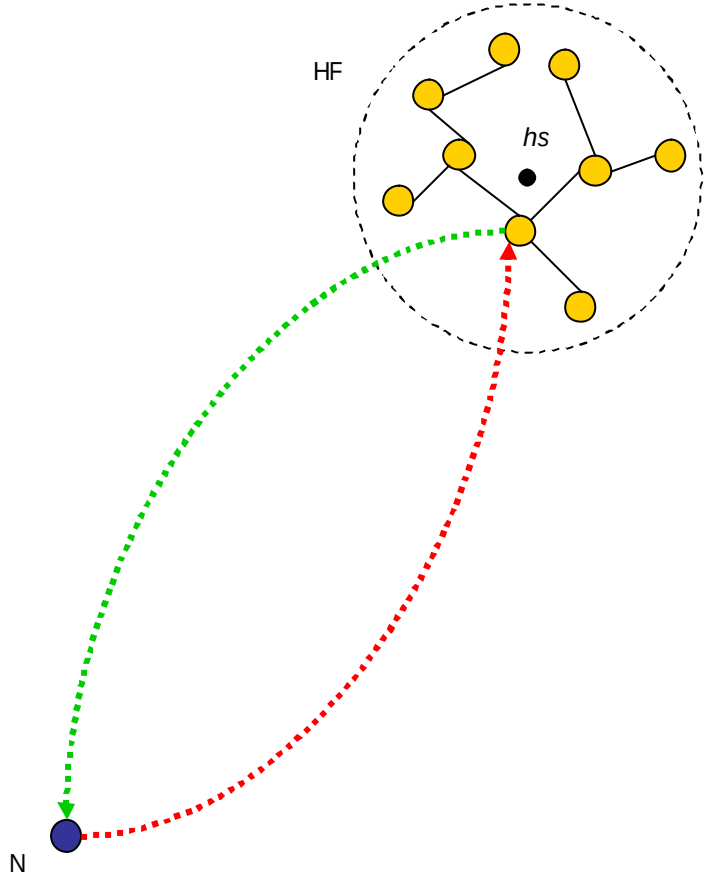
$$HF=[k \mid distance(k, Hot\text{-}Spot) < D]$$



Position update

- *N* periodically updates its current position:

1. *N geo-forwards a position update packet towards hot spot coordinates*
2. *The node in HF selected as target from point 1, starts a flooding limited to the HF area*
3. *The leader of the flooding procedure sends back ack to N, contating information about caching nodes*

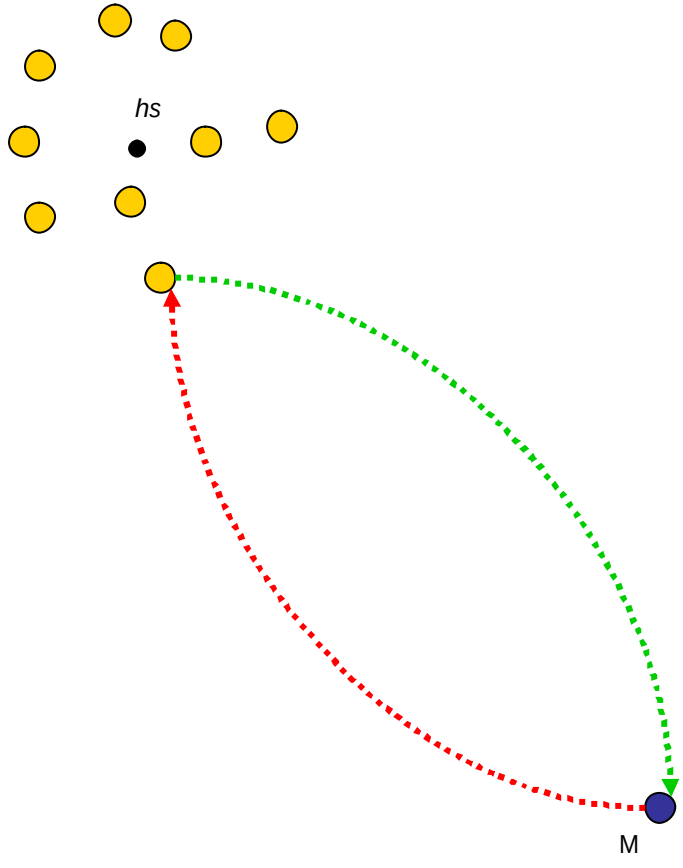


Position lookup

- *M* needs to know the position of *N*:

1. *M geo-forwards a position lookup packet towards hot spot coordinates*
2. *The first node along the path that caches the position, forwards back the answer*

- Note: no need to know the size of the *Home Friends* set



Sharing context information

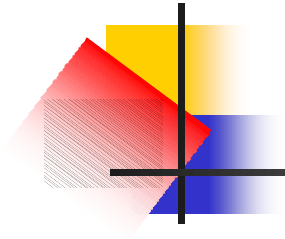
- Node *M* (the seeker) has to know:
  - The address of *N*
  - The *hot spot* coordinates
- Node N could deliver an address similar to:

$$N@(X_{hs}, Y_{hs})$$

- The discovery of *hot spots* and their distribution takes an initial time (a study of that is on going work)

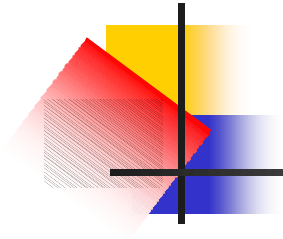
What about dynamics?

- Mobility of the updating node:
  - Variate the position update frequency depending on the node's linear speed
- Mobility of the *Home Friends*:
  - Variate the size of the *Home Friends* set according to the network density around the hot spot



Preliminary evaluation

- Create a suitable node mobility model:
  - Classic Random Waypoint does not provide a realistic scenario for this work
- Carry on mobility simulations on urban scales:
  - Areas of several square km
  - Thousands of mobile nodes moving around assembling primarily around predefined hot spots



Classic Random Waypoint

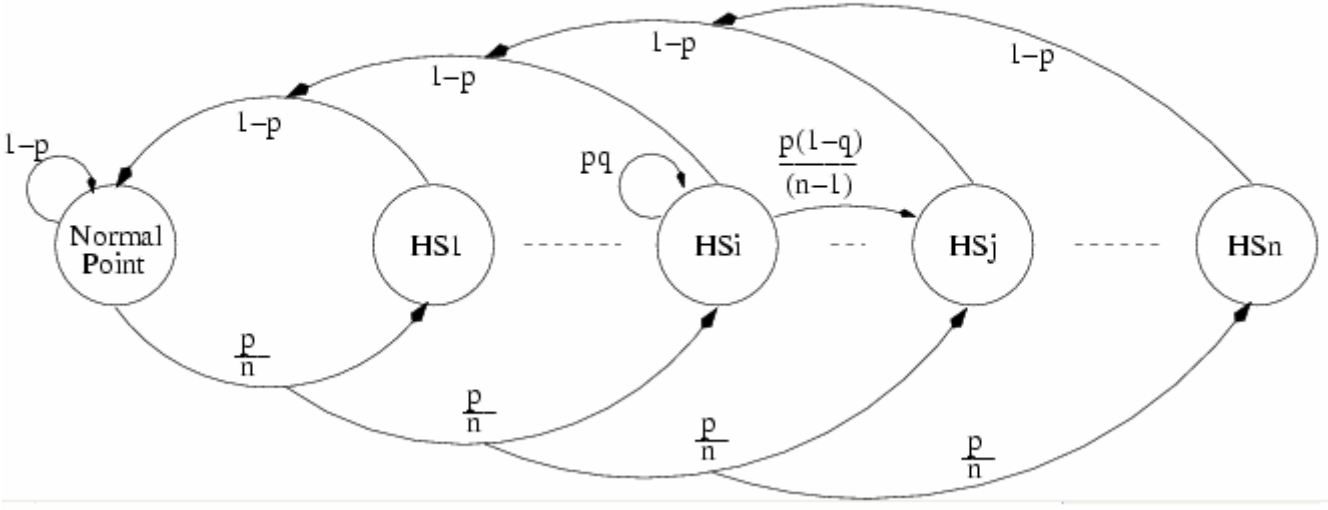
- Starting from Random Waypoint:
  1. Choose the next target position
  2. Move towards it at a non-null speed
  3. Pause on the target and go to 1
- Target position, speed and pause time are choosen randomly using uniform (stateless) distributions

New node mobility model

- Target positions and pause times choosen with different probabilities (use a Markov chain):
  - choose more likely hot spots as next target positions
  - performs longer pauses around hot spots

p= probability to choose a hot spot as the next target

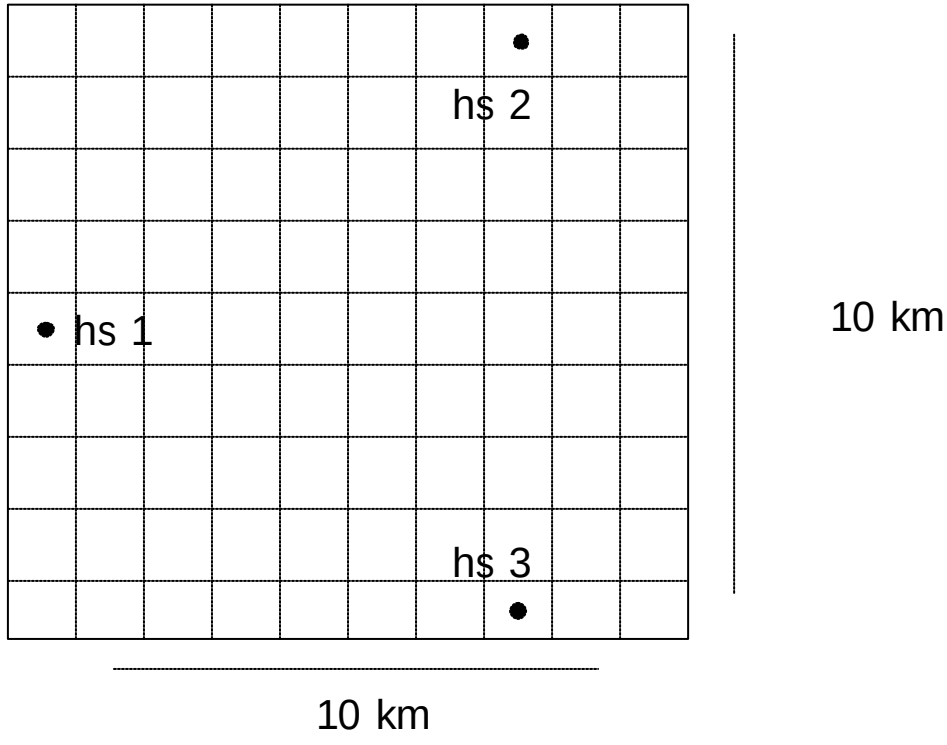
q= probabilty to choose hot spot i as the next target, being on hot spot i



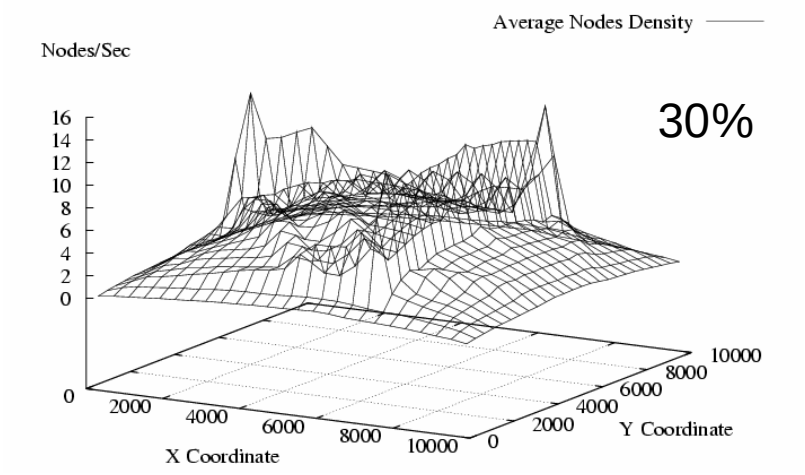
Simulation of the mobility model

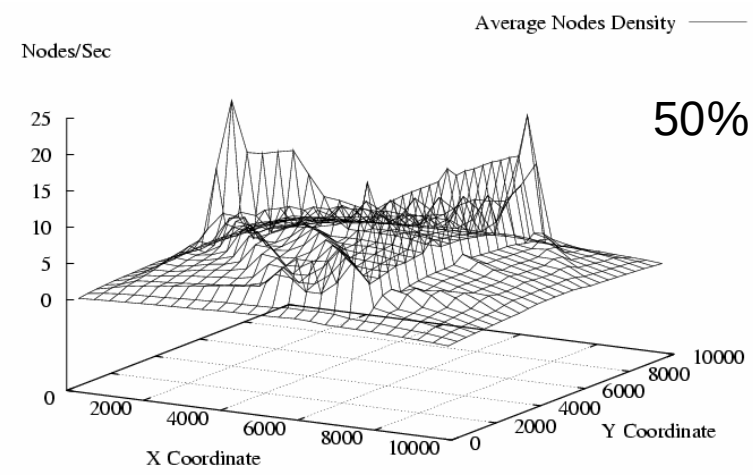
- We developed a tool to simulate thousands of nodes moving in urban areas with predefined hot spots inside
- The tool is able to mix nodes with *hot-spot* oriented behaviors and nodes following classic random waypoint
- Initial scenario :

2500 moving nodes and 3 hot spots inside



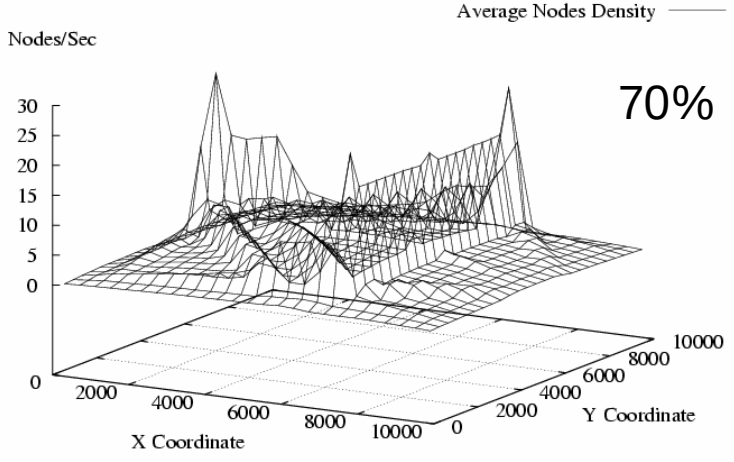
Results

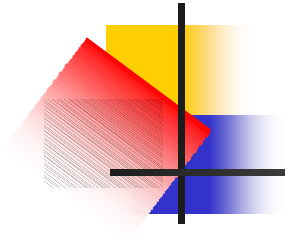




Created a mix of hot-spot oriented and normal random waypoint node behaviors

➔ Node density grows at hot spot locations, increasing the percentage of hot-spot oriented nodes



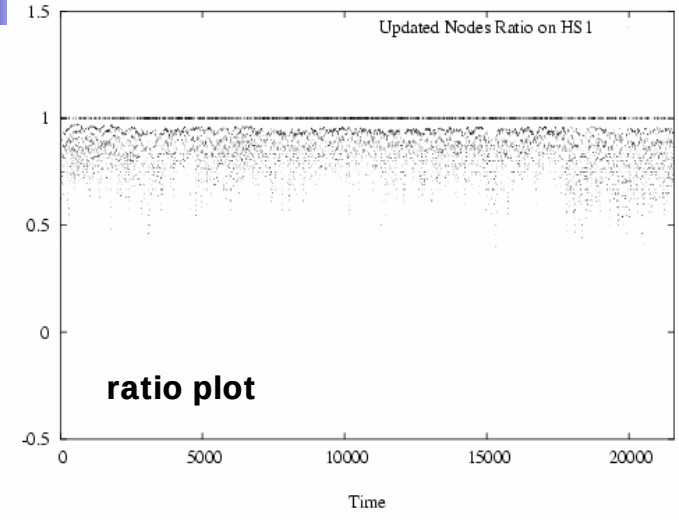


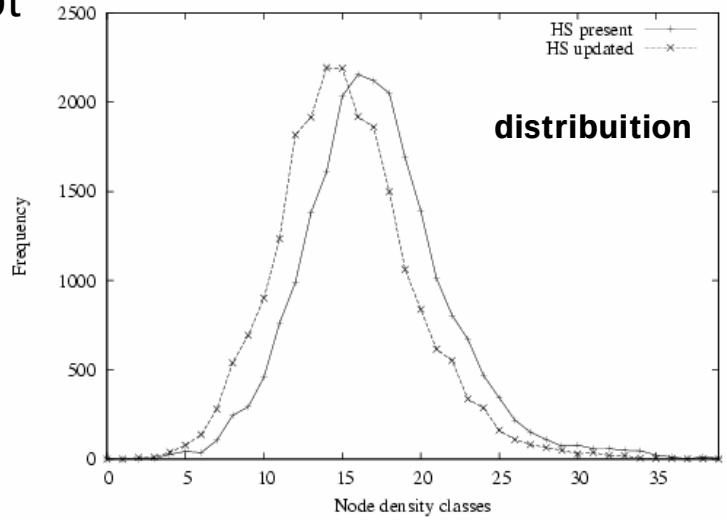
Distribution of updated nodes

- On top of the shown mobility model:
  - We took a reference node perfoming position updates on both a **hot-spot** point and a **normal point**
  - We tracked the total number of nodes in these two observed locations at each simulation tick, and the number of local updated nodes
- The updated nodes are those that are located around the point and that received the last position update

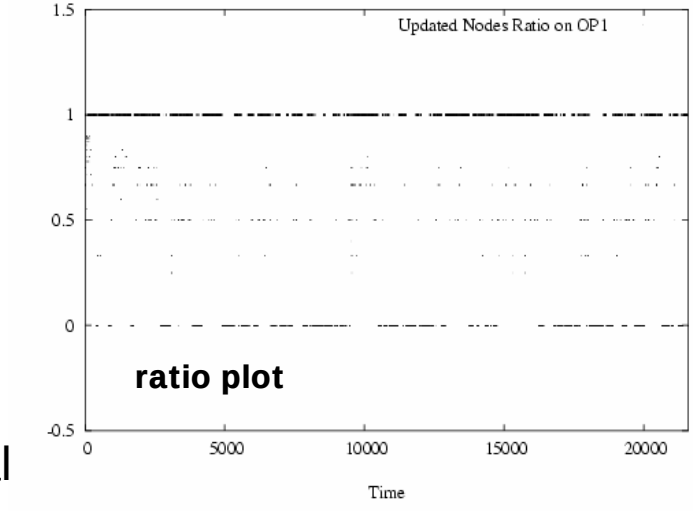
Updated/Present nodes ratio

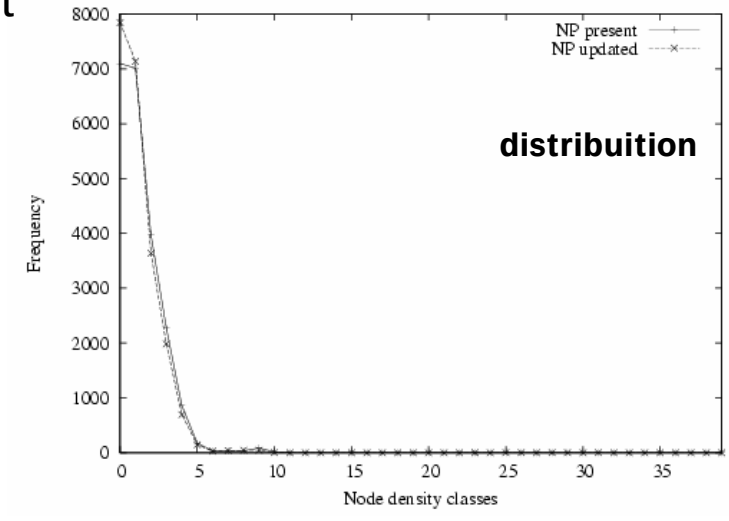
Hot Spot

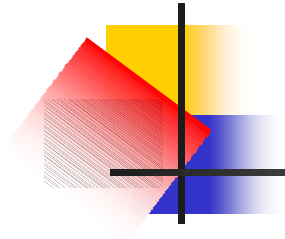




Normal Point







Future work

- Implement policies to discover hot spot automatically on the nodes

➔go towards a *self-organized* system

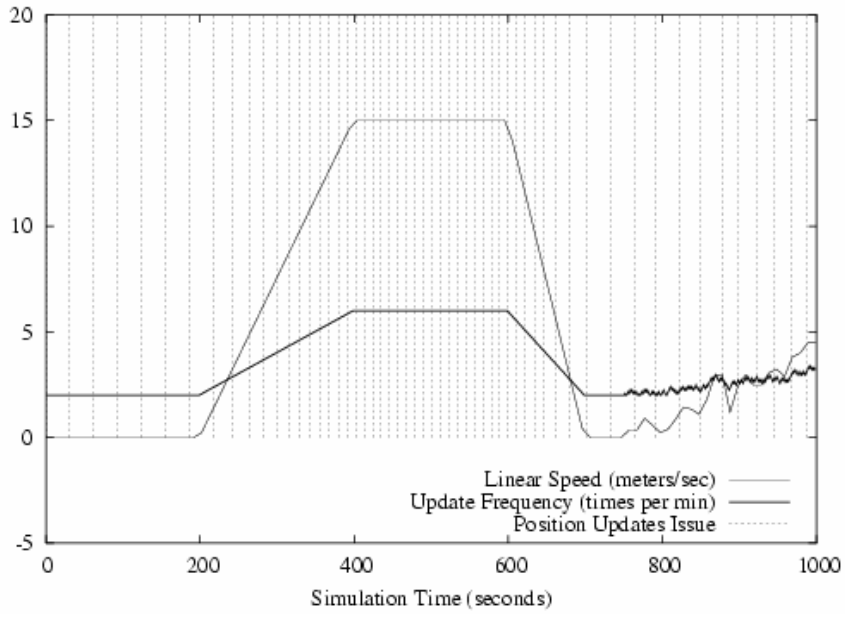
Thank you for your attention!

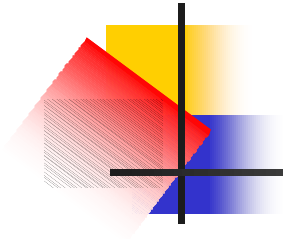
Any question?

Position update frequency

- Required to be directly proporional to the node linear displacement (linear speed) and to be scaled over the position sampling frequency (GPS goes once per second...)
- With $f_0 = f_b$, $S_0 = 0$, S_{MAX} as the maximum achieveable linear displacement (max speed), f_n ranges inside $[f_b, K * f_b]$
- Imposing f_b to be a fraction of the GPS frequency we obtain the requirements above
- Simulation shows behavior for S_{MAX}=15 m/s, K=3, f_b=2/minute

$$f_n = f_{n-1} + \frac{S_n - S_{n-1}}{S_{MAX}} * (K-1) * f_b$$





Home Friends set with variable size

- Objective:
  - Have enough nodes caching position updates getting in the HF area, so that even if some location servers leave the area between subsequent position updates issues, incoming position lookups will be satisfied with high probability
- A model for that has been developed that works with:
  - Statistics about caching nodes density in the HF (coming back with position update acks)
  - An approximation of the caching location server distribution in the HF area (uniform distribuition)