A Distributed Particle Filter Implementation for Tracking in a Wireless Sensor Network

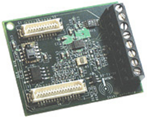
Jesse Read, Katrin Achutegui, Joaquín Míguez

Universidad Carlos III de Madrid.

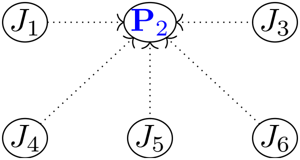
July 2, 2013

Initially: A Particle Filter in a WSN ?

The iMote2 ($\approx$ 13 MHz) . . .

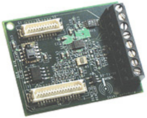


- **sensing nodes** J takes sensor measurements
- **central processing node P**



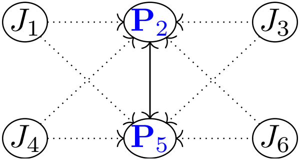
not enough processing power!

Previously: A Distributed Particle Filter in a WSN



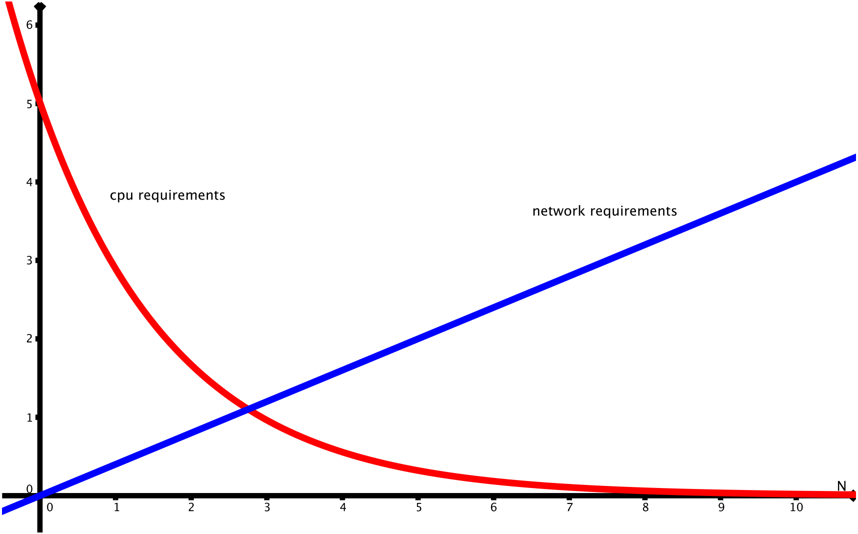
The IMOTE2 ($\approx 13N$ MHz) ...

- **sensing node** J takes sensor measurements
- N **processing nodes P**



- It worked! Tracking (a person) in an indoor scenario, accuracy of 0.5m, using light sensor readings; Extremely efficient, but ...

For N processing nodes . . .



. . . hardware is limited. We cannot do anything much more sophisticated / bigger.

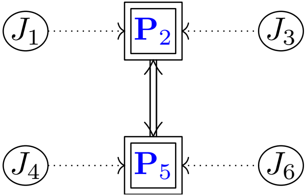
Initial Conclusions

Further development is limited by

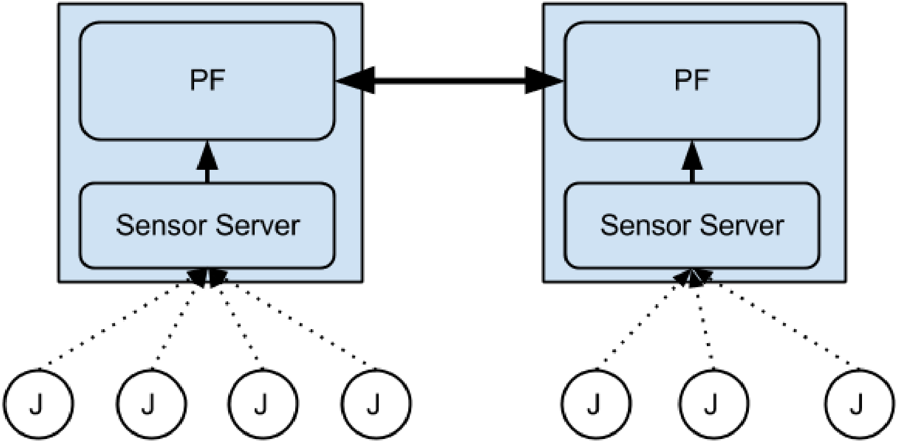
- CPU / network / battery limitations; and
- difficulty involved with working with TinyOS.

- Dedicated **sensing nodes** (J) – TELOSB, low spec
  - basic TinyOS sensor-program, just send sensor readings

- Dedicated **processing nodes** (**P**) – PANDABOARD, 1.2 GHz, Linux, standard WIFI connection
  - ample processing, and high-bandwidth
  - development in any language (e.g., Python)



New Framework: Distributed Particle Filter



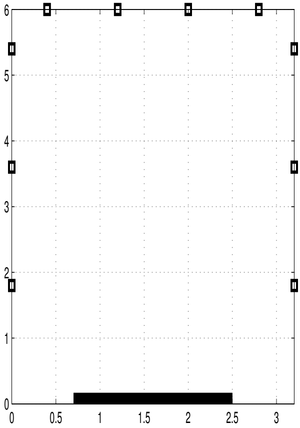
- Each PF shares sensor observations / particles with other nodes to form a DPF

Deploying a DPF for Tracking

1. Define layout
2. Define target function
3. Define observation function

Then connect **P**s to each other in a loop; algorithm begins.

1. Define Layout: `layout.py`



For example ...

- 3.6 × 7.2 metres, indoors
- single light source (a window)
- 10 TELOSB motes

2. Define Target Function: `target.py`

How the target/particles move.

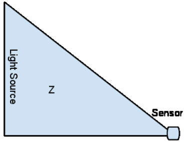
$$\mathbf{x}_t \sim p(\mathbf{x}|\mathbf{x}_{t-1})$$

- $\mathbf{x} \in \mathbb{R}^4$ (2D position and velocity)
- $2.0 m/s$ max speed, $0.1 m/s$ min speed
- change angle randomly (and at the scenario boundary)

Weight each particle according to observations.

$$w = p(\mathbf{y}_t|\mathbf{x}_t)$$

- binary observations $y_{j,t} \in \{0, 1\}$ for each sensor j at time t
- detection zone Z_j: area between sensor j and light source(s)



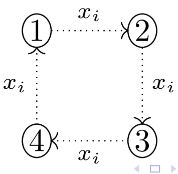
$$p(y_{j,t}|\mathbf{x}_t) = \begin{cases} 1 - \text{FPR} & \text{if } \mathbf{x}_t \in Z_j \text{ and } y_{j,t} = 1 \\ \text{FPR} & \text{if } \mathbf{x}_t \notin Z_j \text{ and } y_{j,t} = 1 \\ 1 - \text{FNR} & \text{if } \mathbf{x}_t \notin Z_j \text{ and } y_{j,t} = 0 \\ \text{FNR} & \text{if } \mathbf{x}_t \in Z_j \text{ and } y_{j,t} = 0 \end{cases}$$

FPR / FNR: false positive / negative rate

A Distributed Particle Filter

N Processing Elements (PE), running in parallel, each with M particles;
At each timestep t, each PE $n = 1, \ldots, N$:

1. receive a particle $\mathbf{x}_{i,t} \leftarrow \mathbf{x}_{i,t}^{(n-1)}$ from PE $n-1$
2. read observation $\mathbf{y}_t$ from sensors
3. for all its particles $m = 1, \ldots, M$:
    1. $\mathbf{x}_{m,t} \sim p(\mathbf{x}|\mathbf{x}_{m,t-1})$ move
    2. $w_{m,t} \leftarrow p(\mathbf{y}_t|\mathbf{x}_{m,t})$ weight
4. resample particles
5. $i \leftarrow$ index of best particle (highest weight)
6. send particle $\mathbf{x}_{i,t}$ to PE $n+1$

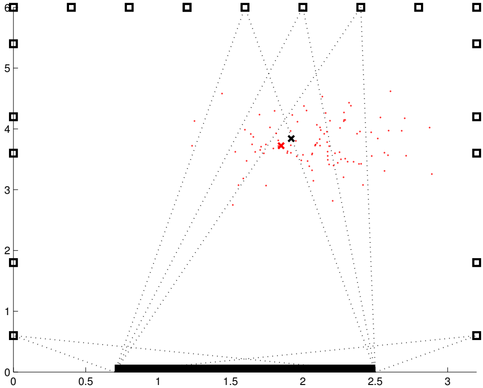


Experimental Results

VIDEO: deploying, configuring and running a WSN for tracking

Simulation Results

Adding more sensors ...



- Now: $\lesssim$ 0.2m accuracy; $\approx$ 0.05 seconds per timestep.
- Before: $\approx$ 0.5m accuracy; 1.00 seconds per timestep

Lessons Learned / Conclusions

Best ways to work with TinyOS:

1. As little as possible

It seems that ...

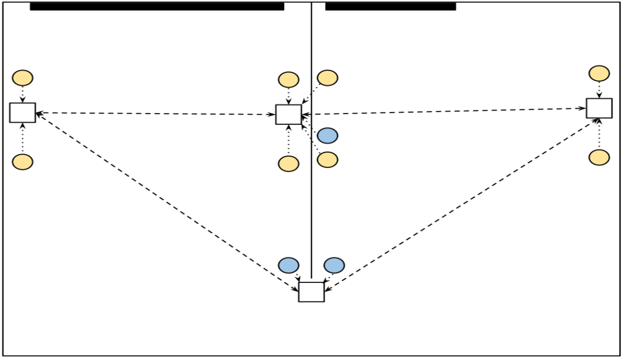
1. Motes are not likely to get much more powerful; but
2. 'PandaBoard'-type ARM boards getting popular (also, e.g., RaspberryPI, BeagleBoard).

So, can have the best of both worlds:

- benefits from distributed network
- sufficient processing power
- development (almost) as usual

Future Work

- Fusion of observations from different kinds of sensors (light, acoustic intensity, RSSI)
- Multi-target tracking
- Deployment in different/larger scenarios, with a larger network



A Distributed Particle Filter Implementation for Tracking in a Wireless Sensor Network

Jesse Read, Katrin Achutegui, Joaquín Míguez

Universidad Carlos III de Madrid.

July 2, 2013