Robust distributed node localization with error management

MobiHoc 2006

Presented by Wei-Shun Lee

Outline

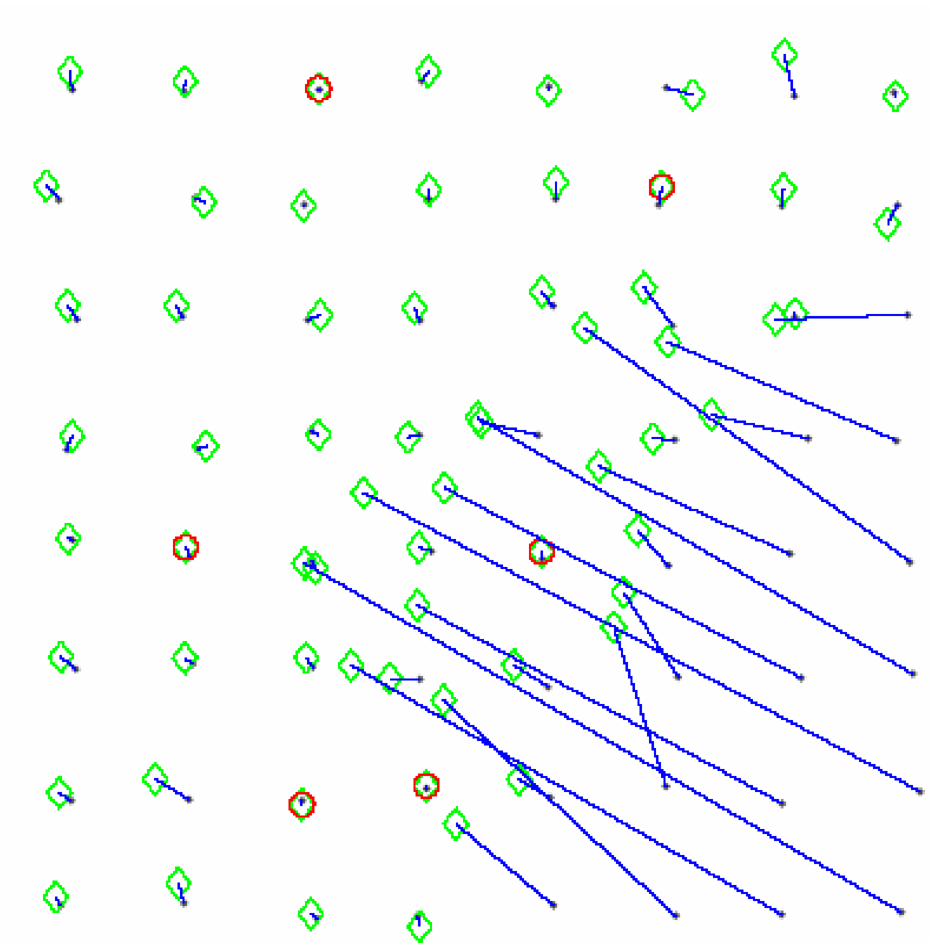
- Introduction
- Interactive localization
- Basic localization algorithm
- Error control
- Simulation and testbed experiments
- Discussion and conclusion
- Appendix

Introduction

- All sensor network tasks such as geo-routing, data-centric storage…etc, rely on precise localization technology. The localization methods includes three types
  - GPS
  - Anchor nodes
  - Local coordinate assignment
- The anchor nodes method would meet error propagation problem in which location information progressively propagates from anchor nodes to free nodes.

Introduction

- Using anchors and local computation to iteratively localize free nodes suffers from
  - Low success rate in network with low anchors
  - Being prone to error propagation
- The paper proposes an iterative localization with error control to achieve better localization quality in networks with low anchor density.



Interactive localization

- DCG (distance constraint graph)
  - Vertices – sensor nodes
  - Edges – distance between pairs
- For a node t given *N* neighbors in DCG with known locations. Two localization error metrics are:
  - Vertex errors *{* e^{v} *}*
  - Edge error *{* e^{e} *}(see Appendix A)*
- *the error of the node t is* $e_t = g\left(\left\{e_i^v\right\}_{i\in N}, \left\{e_{(t,i)}^e\right\}_{i\in N}\right)$

Interactive localization

- Initial phase
  - In a low anchor density sensor networks. Assume each node can directly or indirectly communicate to each other.
  - Anchor nodes broadcast their location information to the DCG.
  - Free nodes compute a shortest path to each of the nearby anchors(each node needs 3~5 anchors to obtain initial estimate)

Interactive localization

- 畫個圖來表示

Basic localization algorithm

ITERATIVE LOCALIZATION

Each node *i* holds the tuple $(x, e^v)_i$, where

x is the node location (or estimate);

e^v is the vertex error.

Each edge *j* corresponds to a tuple $(z, e^e)_j$, where

z is the distance measurement;

e^e is the edge error

Initialization step (optional)

computing shortest path to anchors

Basic localization algorithm

do *{*

for each free node *t*

examine local neighborhood *N*;

select neighbors based on vertex and edge errors

$\{e^e\}$ and $\{e^v\}$

compute location estimate $\hat{x}_t$;

estimate error $\hat{e}_t$;

decide whether to update *t*'s registry

with the new tuple $\left(\hat{x}_t, \hat{e}_t\right)$

} **while** termination condition is not met.

Robust least-squares formulation

- Basic least-squares multilateration

$$\| x - x_i \| = f(z_i)$$

to square both size

$$\| x \|^2 + \| x_i \|^2 - 2x_i^T x = f^2(z_i), i = 0,1,\ldots$$

To simplify the above as

$$a_i^T x = b_i \rightarrow Ax = b$$

The basic solution is

$$\hat{x}_t = (A^T A)^{-1} A^T b$$

Robust least-squares formulation

- Robust least-squares formulation
  - The least-squares solution gives

$$\hat{x}_t = \arg\min_x \| Ax - b \|^2$$

  - We take error into consideration

$$\hat{x}_t = \arg\min_x E\| (A + \Delta A)x - (b + \Delta b) \|^2$$

  - The RLS solution is

$$\hat{x}_t = (A^T A + C_A)^{-1} \cdot A^T b$$

C_A is the error statistics.

Error control

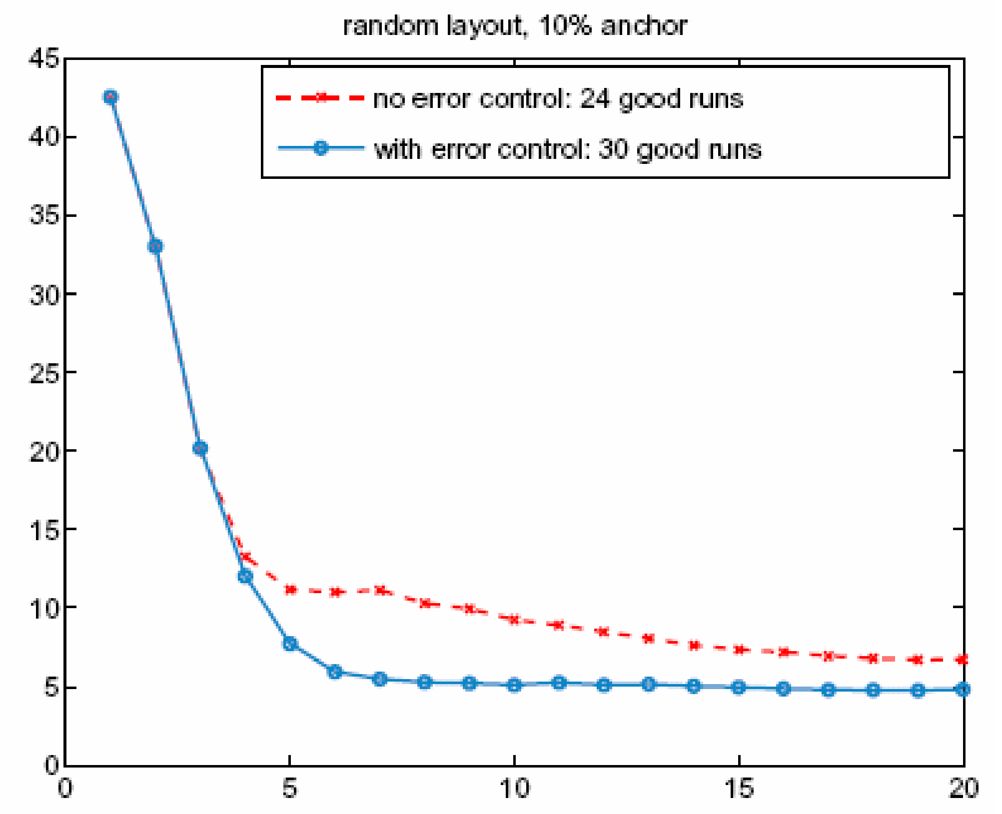
- Node registry
  - Each node maintains a registry with information sufficient to localize other nodes ,including node localization and error.
- Neighbor selection
  - Nodes with high overall errors are excluded from the neighborhood and not used to localize others.

$$e_{total}\left(i\right) = e_i^v + e_{(i,t)}^e$$

- Update criterion

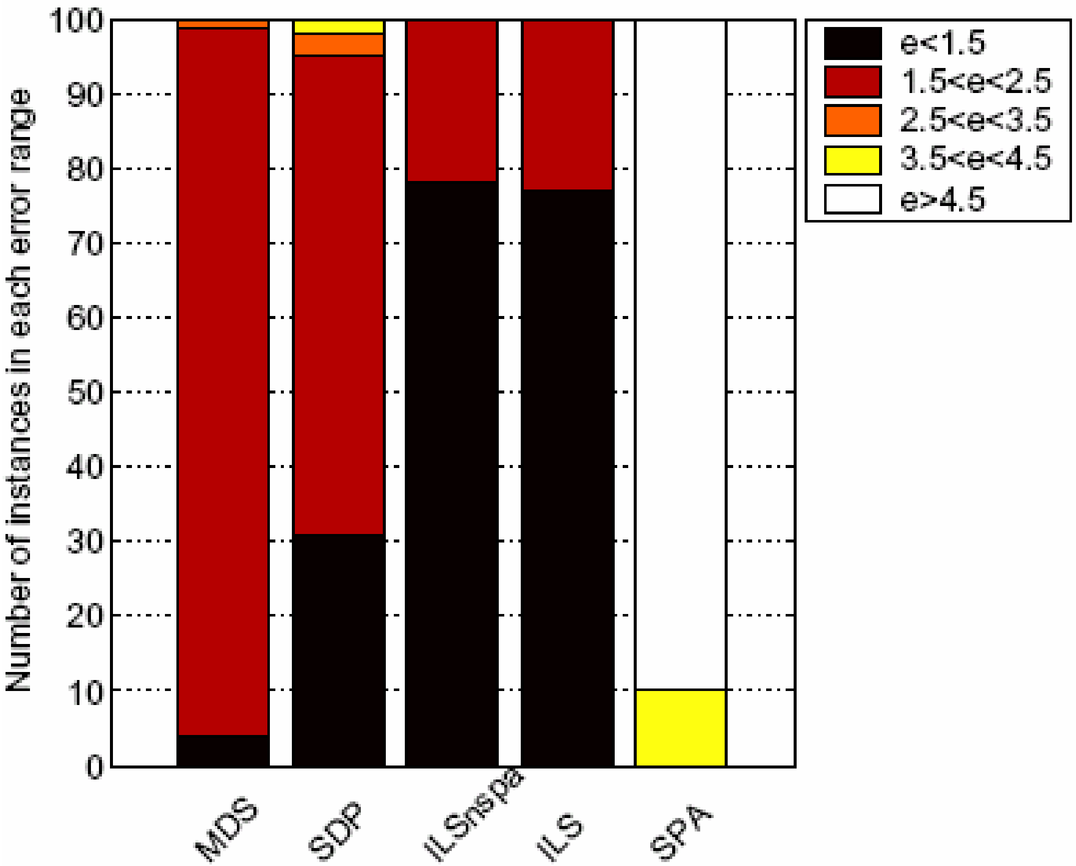
Simulations

- Networks with large numbers of anchors



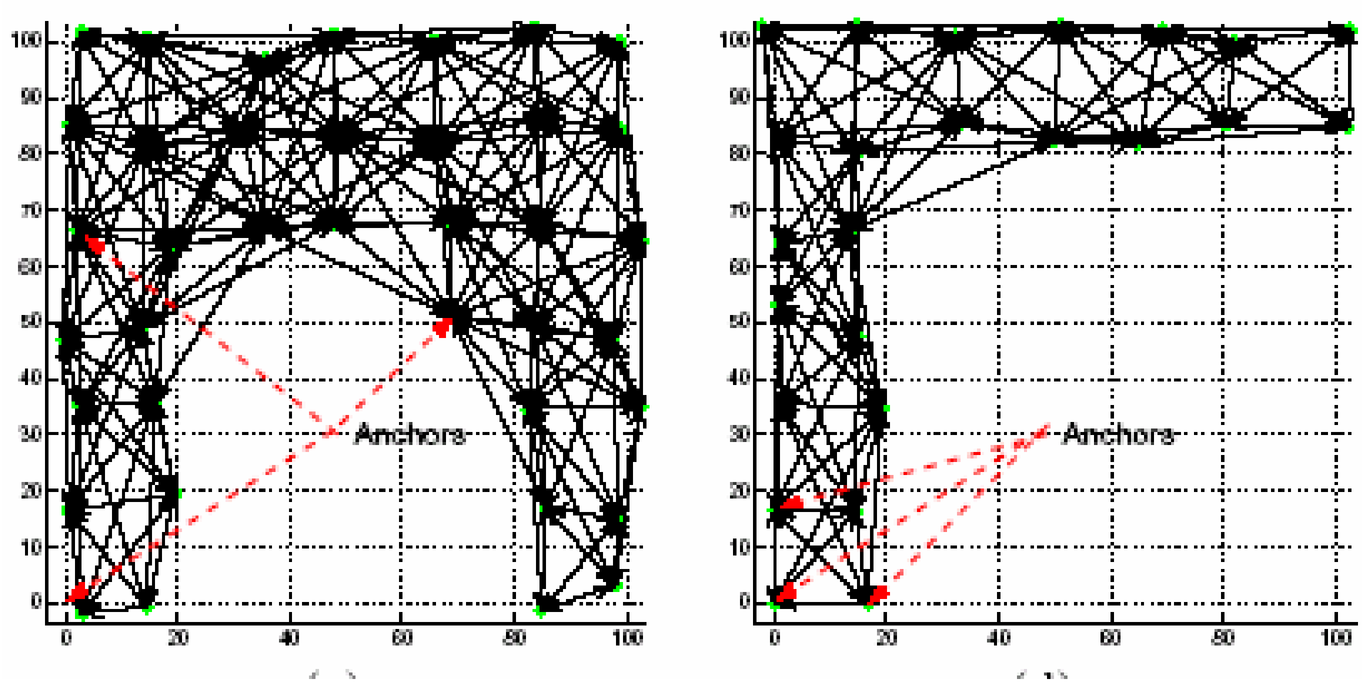
Comparisons with other algorithms

- ILS
- ILS_{nspa}
- MDS-MAP
- SDP
- SPA



Anchor perc.	MDS	SDP	ILSnspa	ILS	SPA
10%	39	0	20	**42**	0
20%	0	6	35	**64**	0

Networks with three anchor nodes only



Scenarios	MDS	SDP	ILSnec	ILS	SPA
(a)	19	0	0	86	0
(b)	24	0	6	70	0
(c)	4	3	10	84	0
(d)	0	1	56	43	0

Discussion and conclusion

- Locality in storage and computation
  - ILS is a local method that location vector and estimation error term are distributed stored on the individual nodes. No central computation or storage is required.
- Light –weight computation
  - Computing location estimate using RLS is fast and can localize themselves in paralle.