

Localized Movement Control for Fault Tolerance of Mobile Robot Networks

WSAN September 2007;
Telecommunication Systems, to appear.
Shantanu Das, Hai Liu, Amiya Nayak,
Ivan Stojmenović
www.site.uottawa.ca/~ivan

Moving and deploying to connect

- Seah, Liu, Lim, Rao, Ang: TARANTULAS, IEEE SUTC 2006
- *Mobile sensors move to fill the communication gaps* to enhance connectivity while static nodes serve as landmark nodes to help robots search the targets.
- If a mobile sensor receives largely different hop counts (toward landmarks) from sensors around it, it identifies the area as critical one, and tries to find suitable spot to bridge the gap by deploying a new sensor.
- A mobile sensor will change its heading only if its neighbor's ID is higher, and with respect to the closest higher ID sensor. The heading would be 90 degrees with respect to line joining them.

Problem Specification

Given a **connected** robot network (but not necessarily **bi-connected**), the problem is to control movement of robots, such that the *network becomes bi-connected*.
The objective is to *minimize the total distance traveled by all nodes*.

Motivation

- Faults in robot networks can be caused by hardware damage, energy depletion, harsh environment conditions, and malicious attacks.
- **Bi-connectivity** is the basic requirement for design of fault-tolerant networks.
Bi-connected:
two disjoint routes exist between any two robots
- No **localized** movement control algorithm to establish bi-connectivity from connectivity is available.
Localized: *robot makes decision based only on local knowledge (position of itself and its neighbors)*

Related Work

A **centralized** movement control algorithm [BR04]

Find blocks (=bi-connected components) and move the smallest one (all its nodes in parallel) toward a neighboring block to merge.

[BR04] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks", *IEEE Network*, 18(4): 36-44, 2004.

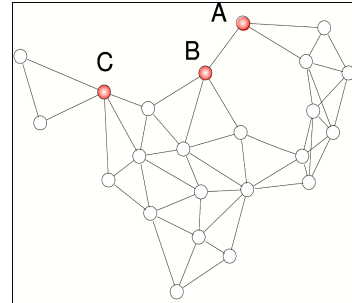
Assumptions

- Network is connected but not bi-connected.
- All nodes have common communication range r .
- Each node has a unique ID and information on position of itself and its p -hop neighbors.
- **p -hop sub-graph** of a node is the graph which contains all nodes that are within p -hop from that node and all corresponding links.

Critical node

- A node is p -hop **critical node** iff its p -hop sub-graph is disconnected without the node.
- Jorgic, Stojmenovic, Hauspie, Simplot-Ryl 2004

Example



C critical for any p , B and A are critical nodes for $p < 3$.

Movement of (Non) Critical Nodes

Observation used in new algorithm:

- Movement of a critical node may cause disconnection of the network.
- Movement of one non-critical node will never cause disconnection of the rest of the network.

Basic Idea



Move non-critical nodes while keeping critical nodes static

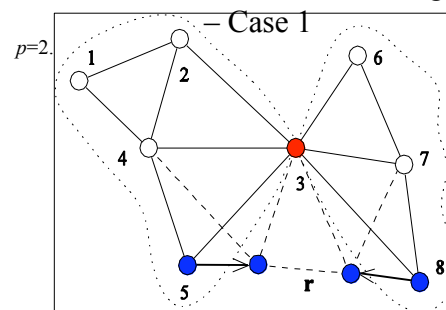
Status may change in the next round
 Network without critical nodes is bi-connected
 (since globally critical node for connectivity is also locally critical, global problem can be locally identified)

Overview of Our Solution

Three cases of movement control are considered:

- Critical node **without** critical neighbors
- Critical node with **one** critical neighbor
- Critical node with **several** critical neighbors

Critical Node without Critical Neighbors

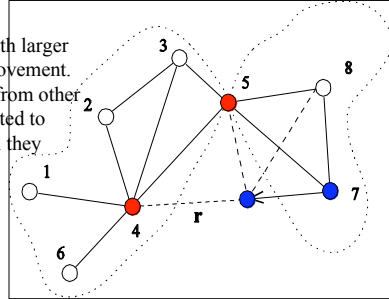


Node 3 selects neighbors 5 and 8 from two components and tells them to move half the distance until they connect

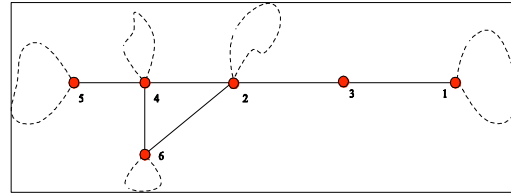
Critical Node with One Critical Neighbor – Case 2

$p=2$.

The critical node with larger ID, node 5, leads movement. The closest node 7 from other components is directed to move toward 4 until they connect



Critical Node with Several Critical Neighbors – Case 3



All red nodes are critical nodes.

The sub-graph of a node is represented by a dashed circle

Definitions

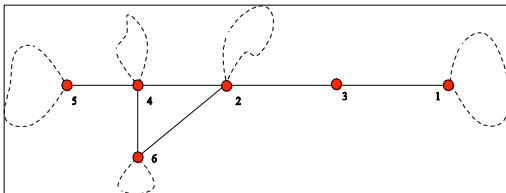
A critical node is **available** if it has non-critical neighbors, and is **non-available** otherwise.

A critical node is a **critical head** iff it is available and its ID is larger than the IDs of any available critical neighbor, or it does not have any available critical neighbors.

Basic Idea

- Use the **pairwise** merging strategy.
- Each critical head dominates a pair of critical nodes to merge.
- The algorithm for case 2 is applied in each pair.

Example



Node 3 is non-available and others are available.

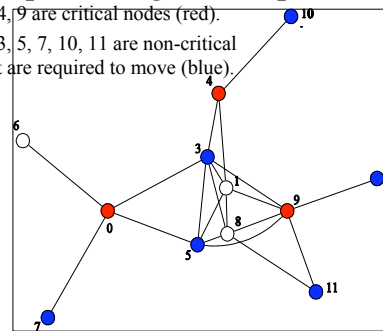
Nodes 1, 5, 6 are critical heads.

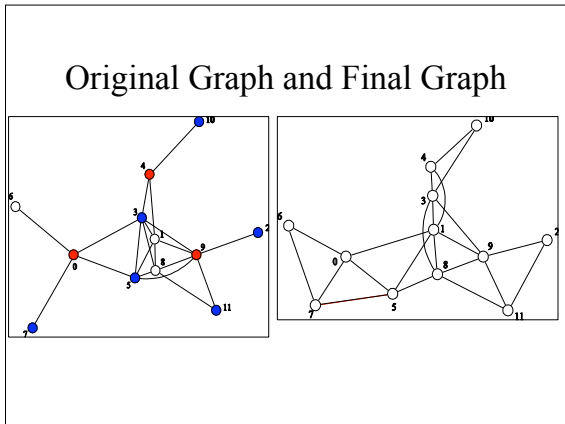
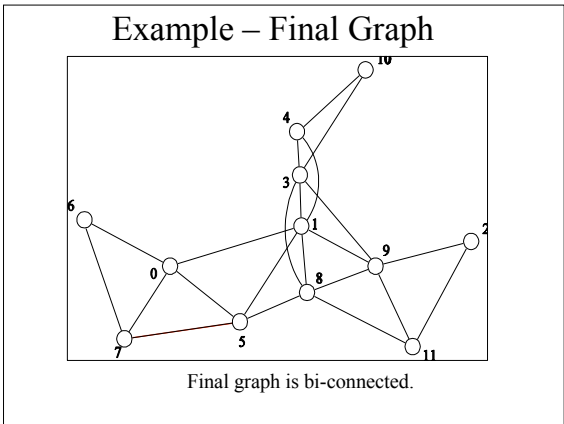
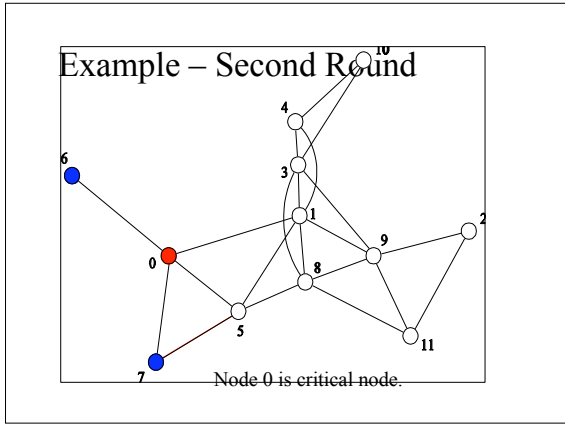
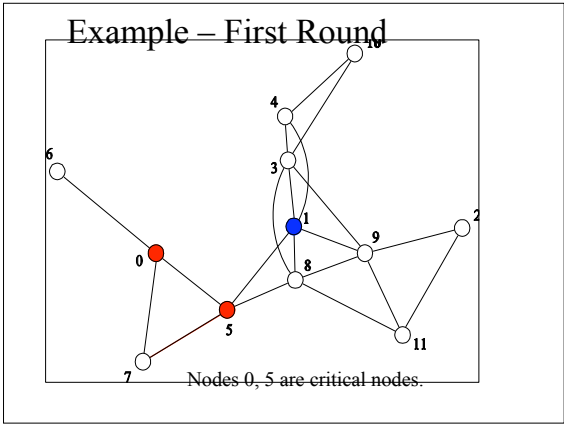
Nodes 1, 5, 6 dominate pairs (1,3), (5,4), (6,4), respectively.

Example – Original Graph

Nodes 0, 4, 9 are critical nodes (red).

Nodes 2, 3, 5, 7, 10, 11 are non-critical nodes that are required to move (blue).





Question

Does there always exist critical heads if the network is connected but not bi-connected?

NO, example: all nodes critical, none is available.

Future Work

- Connected but not bi-connected network may consist of only critical nodes
- So our algorithm may be blocked before bi-connecting nodes
- Will it always terminate? (centralized algorithm has loop problem)
- Network may be partitioned by presented algorithm
- (perhaps no localized algorithm can avoid it – need proof).
- Move to connect, then to bi-connect ?
- Move to also preserve good functionality,
- e.g. area coverage, or satisfactory service to sensor network?