Tutorial
Data communication and coordination in wireless sensor and sensor-actuator networks

SENSORCOMM

Valencia, Spain

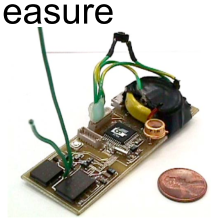
October 15, 2007, 15:45-18:30

Ivan Stojmenovic
UNIVERSITY OF BIRMINGHAM, UK
www.site.uottawa.ca/~ivan

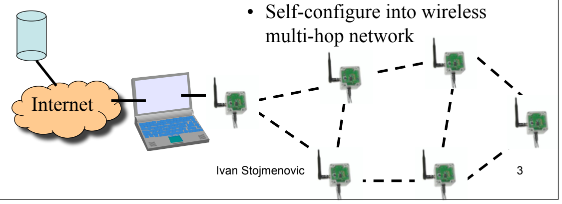
Content

Sensor may measure

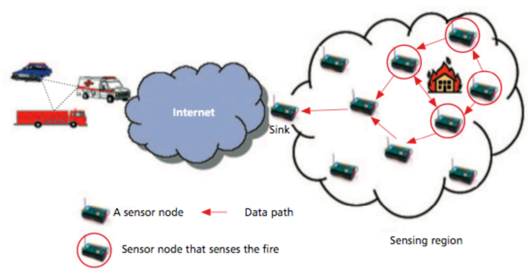
- Distance, Direction, Speed
- Humidity, Soil makeup
- Temperature, Chemicals
- Light, Vibrations, Motion
- Seismic data, Acoustic data
- strain, torque, load, pressure



- Self-configure into wireless multi-hop network

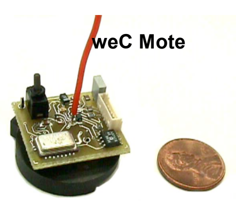


Traditional wireless sensor networks



Sensors: Physical layer

- Sensing hardware
- Small processor
- Small memory
- (low) Power supply
- Transceiver, Receivers
- Low-cost, Miniature ≈cm
- Multi-functional
- Hundreds/thousands sensors spread
- Wireless communication
- Physical risk - defective, lost, damaged, compromised…
- Position information ??

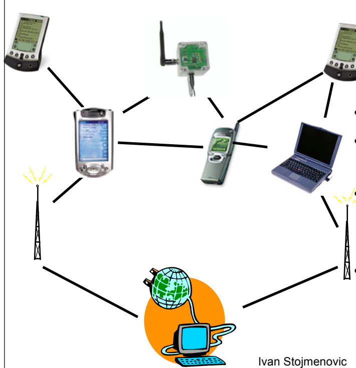


Wireless networks: a taxonomy

- ***Single hop networks***
- Cellular networks
- Satellite networks

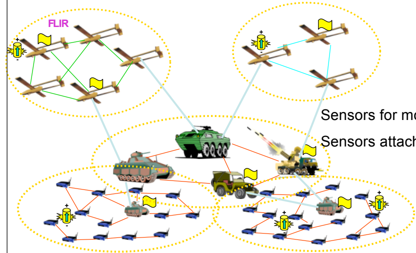
- ***Multi-hop self-organized networks***
- Conference, battlefield, rescue
- Peer to peer networks
- ***<u>Ad hoc networks</u>***

Hybrid ad hoc wireless networks



- Sensor networks
- Cellular multi-hop networks
- Mesh/rooftop networks: wireless fast Internet access
- Vehicles on highway

Merging sensor and ad hoc networks



Detecting mine fields, firing locations

Targeting, target tracking

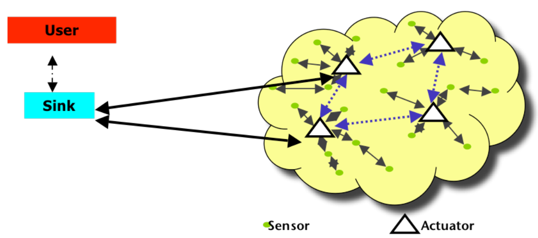
Chemical and biological attack detection

Actuators/actors=

Ad hoc nodes that may act on sensors and environment

Wireless sensor and actor networks

Actors: active nodes, higher energy and computation, action possible, may be mobile



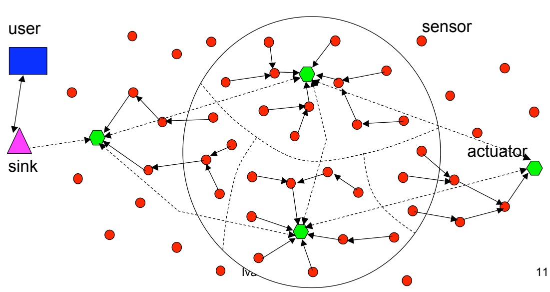
Wireless Sensor-Actuator Networks: SANET

Sensor nodes:
- small size nodes in large numbers,
- low communication and computation capability

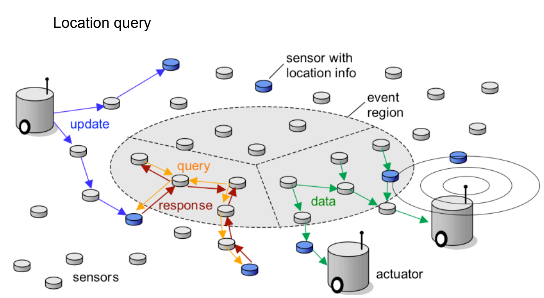
- Actuator nodes:
  - more capable mobile nodes (humans, robots)
  - collect sensor readings, relocate sensors
  - Act on environment (sprinklers..)

Coordinated actuator movement

- move/place sensors to improve area coverage
- move to help sensors determine positions
- move to create fault tolerant network



Communicating data, queries, responses and updates among sensors and actuators



Mobility?

- Sensors are assumed normally to be static
- Actors are normally assumed to be mobile (robots, human, vehicles..)
- Some actors however could be static (e.g. sprinklers)
- And sensors could be mobile !
- There is similarity between mobile sensors and actuator functionality in some aspects, e.g. sensor deployment to cover an area

Mobile sensors and actuators?

- Relocation: passive sensors are moved to a location of failed sensor
- Actuator can move it
- Or it could be mobile sensor that moves itself..
- Is it then mobile sensor also an actuator?
- Similarities and differences in solutions

What are actuators ?

- Wark et all, ACM IPSN 2007
- Prevent bulls from fighting in a farm
- Bulls are nodes in network, carrying collars with sensing and actuation capabilities
- Actuation: stimuli when two bulls come near each other.

People as actuators ?

Daniel Steingart, Wireless Industrial Technologies,USA
Sensors measure temperature in aluminum production
(one-hop communication to sink)
Human adjust energy supply to keep temperature stable

Equipment as actuators:
Light and sound signals, augmented reality
(firefighting applications)

One-hop wireless links only ?

- Korber, Wattar, School, IEEE Trans. Industrial Informatics May 2007
- Star topology
- Base station (BS) is master, several nodes (SAM = sensor actuator modules) are each directly linked to BS, on separate channels
- Argues that this topology is needed for reliable industrial applications

Which sensors can be served by actuators?

- 1) Any actuator can serve any sensor, directly or indirectly (by sending message to another actuator or a device)
- 2) Shah, Bozyigit, Aksoy 2007:
  Actuator serves only sensors which are at distance <=R.
- It broadcasts its presence to sensors up to that distance based on its mobility

Sensors, actuators, and actuator devices

- Ozaki, Hayashibara, Enokido, Takizawa IEEE ARES07
- Semi-passive coordination in multi-actuator multi-sensor model
- Sensors report values to multiple actuators in the area
- Backup actuators broadcast these values to primary actuator
- Updates are sent from primary to backup actuators
- Backup actuators acknowledge them to primary actuator
- Decision are sent by primary actuator to backup ones
- Action is sent to actuation device
- Fault tolerance
- No specific protocols for flooding tasks involved

Network layer issues

- **Generating sensor and actuator networks**
  Revisit random unit disk graph model and current simulation practices
- **Coordinated movement for bi-connectivity**
  - Robots in a connected network move to establish biconnected network
- **Movement for energy optimal routing**
  - A sensor reports continuously to a sink (e.g. video monitoring). Establish initial route with mobile actors or sensors as interim nodes and move to optimize energy
- **Anycasting:** send report from sensor to any actor
- **Multicasting:** from sensor to fixed set of actors

Network layer continuing

- **Sensor relocation:** mobile actors/sensors move to replace failed monitoring sensors
- **Moving to collect sensor readings**
  - Design routes for actors to optimize energy/mobility and collect reports periodically
- **Restricted search for best actor to respond**
  - Sensors reports to one actor. How to efficiently find best actor to respond, without flooding all actors?
- **Coordination for location service**
  - How sensors maintain position information for nearest (at least) actor, and how actors help sensors in providing position information

Load balancing for actors

- Ngai, Liu, Ryu, An Adaptive Delay-Minimized Route Design for Wireless Sensor-Actuator Networks, IEEE MASS 2007.
- Cluster large network, allocate actors to clusters
- *A priori routes by constructing TSP (Traveling Salesman Problem) for each actor*
- for weighted sensors: apply probabilistic visiting
- Actors with variable speeds
- Distributed implementation

Robot deploys sensors

- Batalin, Sukhatme 2005
- Already deployed sensor node records movement directions by robot when it is nearby
- Sensor advices robot on the direction to take:
- *Least recently taken* direction
- If there is no signal from any sensor then robot deploys new sensor
- Robot also notifies sensor when it is nearby
- Finite number of directions

Actor-Actor coordination

- Melodia, Pompili, Gungor, Akyildiz, Mobihoc 2005
- Centralized integer non-linear program
- Localized ‘**auction protocol’**:
- Each actor reports back to originating actor the offer to provide service and the cost of it
- Stojmenovic 2007: ‘**auction aggregation’** protocol
- Collecting actor may have low cost
- Tree expansion and tree contraction phases
- Cost added to flooded message to other actors
- Actors accumulating high cost do not respond
- Responses aggregated and only best offer proceeds

Routing link metric

- Gungot, Sastry, Song, Integlia ICC 2007
- *Routing via both sensors and actors on routes*
- Link Quality Indicator correlates with packet reception probability
- Formula includes:
- Ratio of initial and current node energy for transmitter and receiver
- Energy consumption for transmission and reception
- Cost applied on sensor nodes while actor nodes have zero costs
- Routing otherwise uses so defined link costs

Coverage based clustering

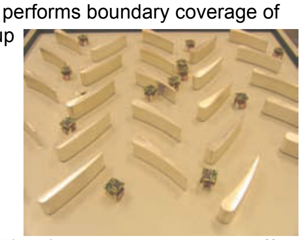
- McLaughan, Akkaya IEEE IPCS 2007
- K-hop independent dominating sets
- Weight to prefer nodes with more *k*-hop neighbors and are farther from borders of other clusters
- Limited flooding to win territory
- *Actors are placed at end at cluster-head positions*
- Thus a variant of known *k*-hop clustering schemes

Networked robots/actuators

Boundary coverage

- Correll, Cianci, Raemy, Martinoli EPFL 2007:
- 'Self-Organized embedded sensor/actuator networks for 'smart' turbines'
- A swarm on miniature robots performs boundary coverage of blades in a jet turbine mock-up
- Depth first search of spanning tree of blades

Division of search boundary among robots not explained



Dynamic task assignment

- McLurkin, Yamins, 2006
- p_i <u>*robots*</u> to be assigned to *i*-th task
- **Random choice** (inbalance for small teams)
- **Extreme-Com**: flood received info until all robots learned; assign in same sorted order
- **Card-dealer**: wave propagation to learn leader in each round, assign task, repeat
- **Tree-recolor**: wave propagation to learn one leader only, creating spanning tree rooted at leader, who decides roles of each robot and communicates them

Robot dispersion

- McLurkin, Smith 2004
- Robots move opposite to vector some of forces toward neighbors
- Frontier robots move forward
- To prevent disconnections and oscillations:
- Preserve connectivity with two children
- Leaves preserve also coverage of initial area
- and keep near robots stationary while frontier moves

How to generate sensor-actor graphs ?

Atay, Stojmenovic IEEE WoWMom 2007

Generating Random Graphs for the Simulation of Wireless Ad Hoc, Actuator, Sensor, and Internet Networks

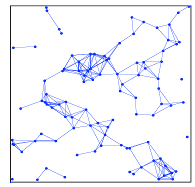
---

Existing generation of ad hoc, sensor, actuator/actor graphs

Standard Algorithm *N = 100 nodes, density d = 5*

choose x and y coordinates of each node at random

CRUG: Connected Random Unit disk Graph



Problems with CRUG

- Did **not** look like **evenly spread** over area
- Would students seat like that in a classroom?
- **Slow** to generate sparse connected networks
- So we want **fast** generation of sparse connected networks, and we want these networks to look more natural, like robots moving as **collaborative team** and fairly dividing area to control

---

CAG: Connected Actuator Graphs Generation Algorithms (Common Properties)

- Candidate graph:
  - Calculate an approximate transmission range such that *(expected node degree) = d* $r = (Ad/((N-1)\pi))^{1/2}$
  - place *N* nodes sequentially, in *N* rounds.
  - Place the *i*-th node based on the positions of the *(i-1)* previous nodes
- Proximity constraint:
  - Proximity constraint is satisfied if node-*i* is not isolated from the previous nodes based on the approximate range r and it is no closer than d_{min} to any of the previous nodes.

CAG Generation Algorithms

- Furuzan Atay, Ivan Stojmenovic IEEE WoWMoM 2007
- Center node based Algorithms:
  - Eligible Proximity Algorithm (**EPA**)
  - Weighted Proximity Algorithm (**WPA**)
  - Minimum Degree Proximity Algorithm (**MIN-DPA**)
- Maximum Degree Proximity Algorithm (**MAX-DPA**)

Center node based algorithms

- Distribute degree more uniformly while maintaining connectivity
- Place the first node randomly in *A*
- In round-*i*, choose a center node and place node-*i* within the transmission range (*r*) of the center node
- After all the nodes are placed, check for connectivity

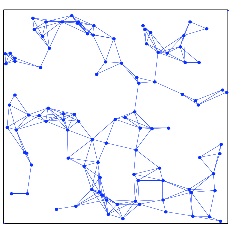
Center node based algorithms

- Center node selection:
  - Calculate the approximate degrees (d_i) of all the nodes already placed based on r
  - **MIN-DPA**: Choose the node with the minimum d_i
  - **EPA**: All the nodes with $d_i < d$ are eligible to accept more neighbors. Choose one of them at random
  - **WPA**: Assign weights to nodes proportional to $(d_i - d)$; Choose one at random according to these weights

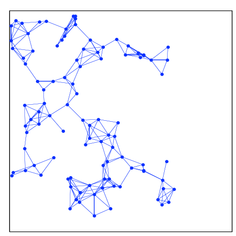
MAX-DPA

- In round-i, choose a random position for node-i.
- Calculate the approximate degrees (d_i) of all the nodes already placed based on r
- Accept this position if
  - it satisfies proximity test
  - it does not result in $d_i > d_{max}$ where $d_{max} = d+n$ and n is a parameter of the algorithm
- After all the nodes are placed, check for connectivity

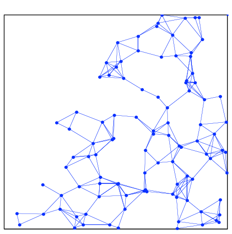
$N = 100$, $d = 5$ **Standard Algorithm**



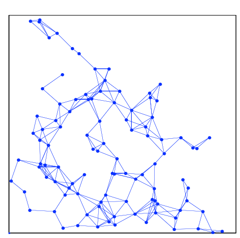
$N = 100$, $d = 5$ **MIN-DPA**



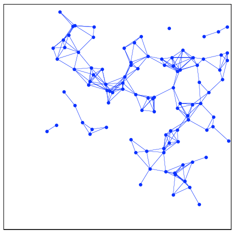
Sample candidate graphs
$N = 100$, $d = 5$
MAX-DPA$_2$ ($d_{max} = 7$)



$N = 100$, $d = 5$ **MAX-DPA$_4$** ($d_{max} = 9$)



N = 100, d = 5 **MAX-DPA$_8$** *(d_{max} = 13)*



Generating sensor networks?

- Add more proximity constraints, e.g.:
- Do not place new sensor if its covering circle is covered already by other sensors
- How to generate realistic sensor and actor networks?
- Wireless Internet networks: generate gateways, then new nodes must be connected to one of gateways.

More on graph generation ?

- Faster generation with smaller degree deviations
- Average size of the largest connected component increased
- How well new algorithms model realistic actuator networks?
- Connectivity analysis by formal methods?
- Theoretical differences from random unit graphs?

Localized Movement Control for Fault Tolerance of Mobile Robot Networks

WSAN, Albacete September 2007

Shantanu Das, Hai Liu, Ajith Kamath, Amiya Nayak,
Ivan Stojmenović
www.site.uottawa.ca/~ivan

Moving and deploying to connect

- Seah, Liu, Lim, Rao, Ang: TARANTULAS, IEEE SUTC 2006
- Robots 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 closest higher ID sensor. The heading would be 90 degrees with respect to line joining them.

Problem Specification

Given a **connected**, but not necessarily **bi-connected**, robot network, 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.

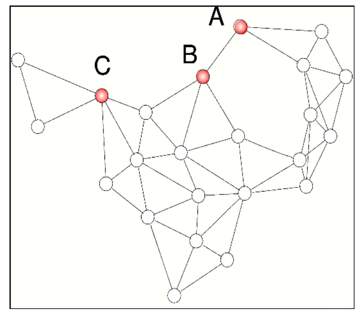
Assumptions

- 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

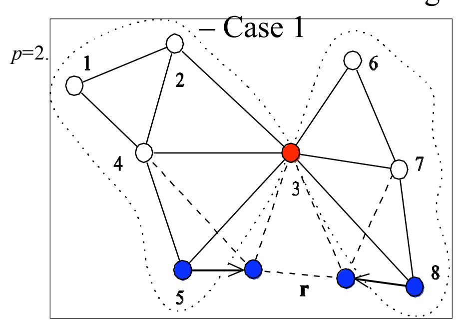
Network without critical nodes is bi-connected

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

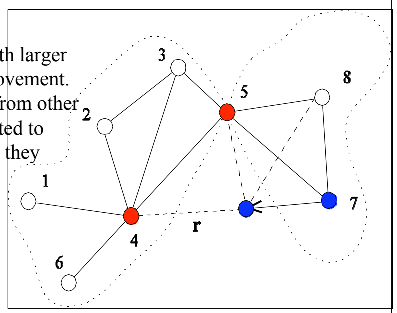
Critical Node without Critical Neighbors – Case 1



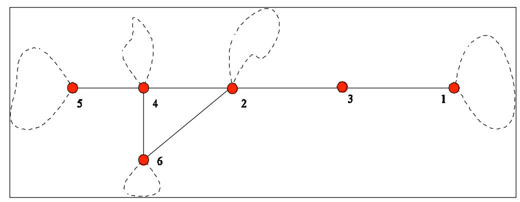
Node 3 selects neighbors 5 and 8 from two components and tells them to move half the distance until they connect

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

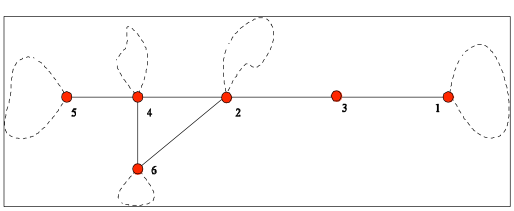
A critical node is **available** if it has non-critical neighbors, and is **non-available** otherwise.

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 has no 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.

Question

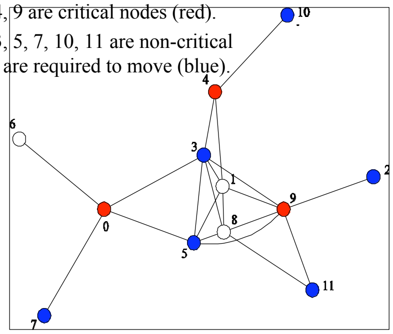
Does there always exist critical heads if the network is connected but not bi-connected?

Theorem 1. If the network is connected but not bi-connected then it has a critical node without critical neighbor or a critical head.

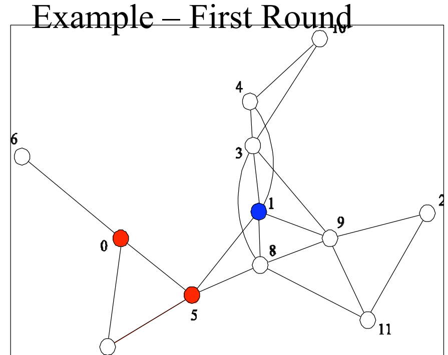
Example – Original Graph

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).

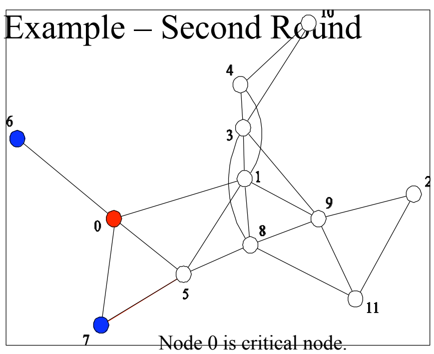


Example – First Round



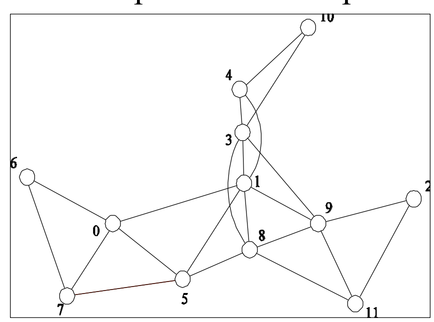
Nodes 0, 5 are critical nodes.

Example – Second Round



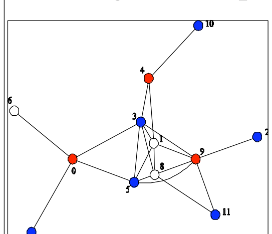
Node 0 is critical node.

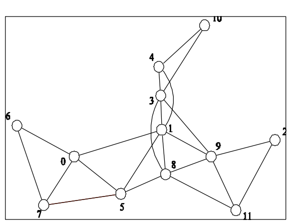
Example – Final Graph



Final graph is bi-connected.

Original Graph and Final Graph





Theoretical Support and Future Work

- Any connected network has non-critical nodes
- Any connected but not bi-connected network has critical nodes without critical neighbor, or critical head
- So problem means action!
- Will it always terminate? (centralized algorithm has loop problem)
- Network may be partitioned (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?

Localized Movement Control for Fault Tolerance of Mobile Robot Networks

WSAN, Albacete September 2007

Shantanu Das, Hai Liu, Ajith Kamath, Amiya Nayak,
Ivan Stojmenović
www.site.uottawa.ca/~ivan

Moving and deploying to connect

- Seah, Liu, Lim, Rao, Ang: TARANTULAS, IEEE SUTC 2006
- Robots 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 closest higher ID sensor. The heading would be 90 degrees with respect to line joining them.

Problem Specification

Given a **connected**, but not necessarily **bi-connected**, robot network, 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.

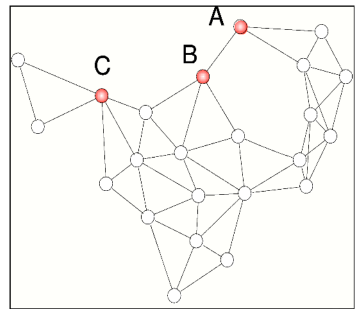
Assumptions

- 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

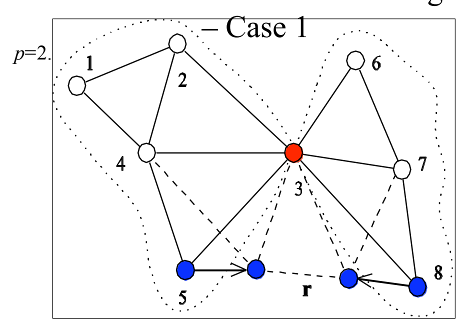
Network without critical nodes is bi-connected

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

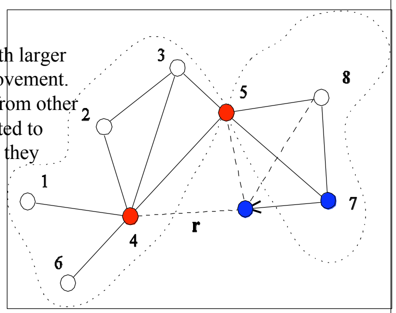
Critical Node without Critical Neighbors – Case 1



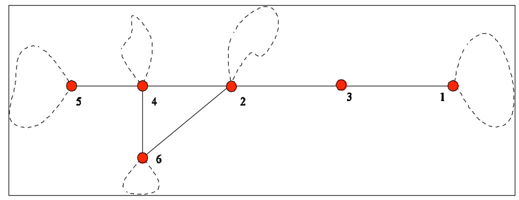
Node 3 selects neighbors 5 and 8 from two components and tells them to move half the distance until they connect

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

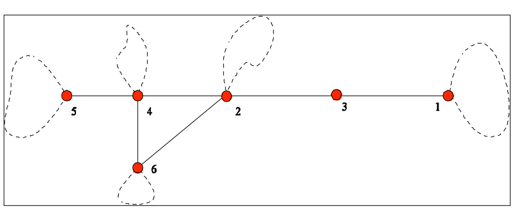
A critical node is **available** if it has non-critical neighbors, and is **non-available** otherwise.

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 has no 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.

Question

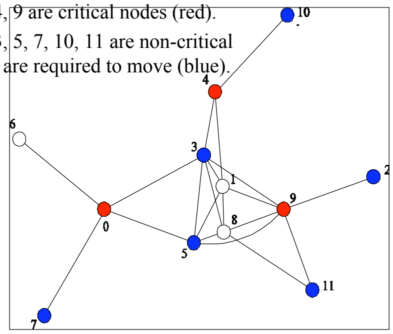
Does there always exist critical heads if the network is connected but not bi-connected?

Theorem 1. If the network is connected but not bi-connected then it has a critical node without critical neighbor or a critical head.

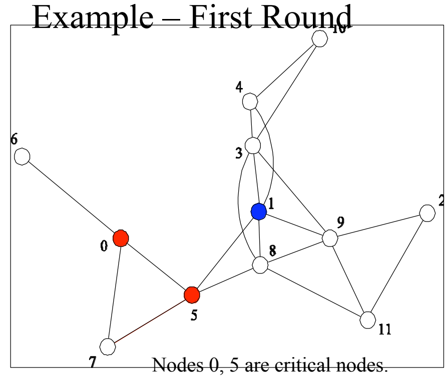
Example – Original Graph

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).

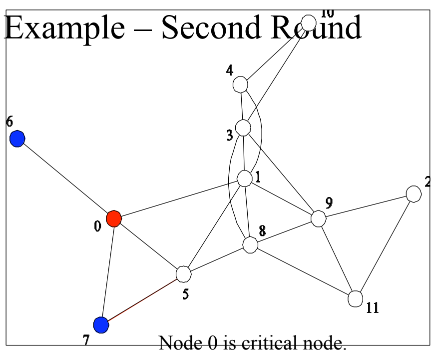


Example – First Round



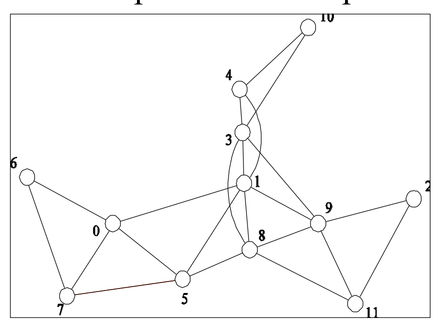
Nodes 0, 5 are critical nodes.

Example – Second Round



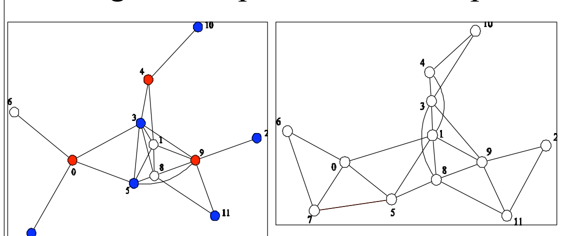
Node 0 is critical node.

Example – Final Graph



Final graph is bi-connected.

Original Graph and Final Graph



Theoretical Support and Future Work

- Any connected network has non-critical nodes
- Any connected but not bi-connected network has critical nodes without critical neighbor, or critical head
- So problem means action!
- Will it always terminate? (centralized algorithm has loop problem)
- Network may be partitioned (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?

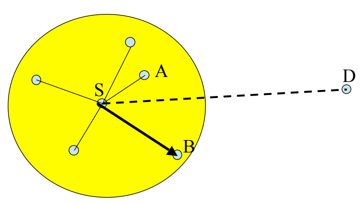
Routing, anycasting, multicasting for sensor-actuator networks

Ivan Stojmenovic

Routing without position information

- **Proactive**: Bellman-Ford, Shortest path (OLSR)
- Can be applied toward nearest actuator
- **Reactive**: Flooding to discover route to an actor, like AODV/DSR
- Overhead at sensors ?
- **Tree creation and maintenance** ?
- Flooding from each actor to establish routes, modify links near moving actors,
- Sensor maintain hop counts or cost toward actors

Greedy position based localized routing

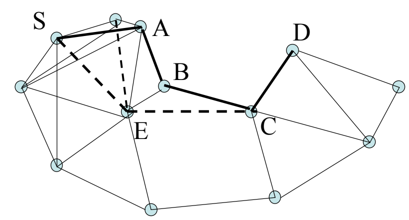


Localized protocol: S knows only position of itself, its neighbors and destination D

S forwards to neighbor B closest to D

Finn 1987

Greedy: SABCD vs shortest path SECD



Localized vs. globalized protocol

SP Overhead: messages to maintain global information at each node following mobility and/or sleep/active periods changes

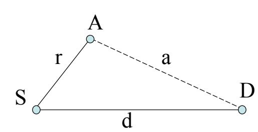
Is hop count the best metric ?

- Power consumption
- Reluctance (avoiding nodes with low energy)
- Power_reluctance
- Delay
- Expected hop count (realistic physical layer)
- COST - selected metric

Cost to progress ratio framework

- Progress: measures advance toward destination
- Progress = |SD|-|AD|=d-a
- Select neighbor A that minimizes cost(SA)/progress(A)
- Hop count: cost=1
- →Maximize advance

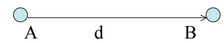
Stojmenovic IEEE Network 2006

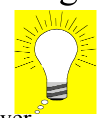


Parameterless behavior

- Cost-to-progress ratio framework has no added parameters such as thresholds
- Threshold based approach: eliminate ‘bad’ links, drop packet if there is no ‘good’ neighbor
- What if a solid path has just one weak ‘bridge’?
- Experiments so far indicate that threshold based approaches are inferior for all threshold values - either high failure rate or suboptimal since there is no notion of ‘best’ neighbor

Power saving localized routing





Constant power → minimize hop count

power $=u(d)= d^{\alpha} + c$ → minimize total power

$2 \le \alpha \le 6$ model by Rodoplu, Meng 1999

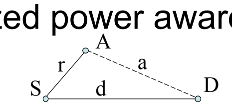
Many articles assume $c=0$; in practice $c>0$ since power is needed to run hardware at each node, and correct reception requires minimal transmission power (no energy free transmission at zero distance)

reluctance $f(A)$ to forward packets =

$=1/g(A)$ $g(A)$ in [0,1] lifetime → minimize total cost

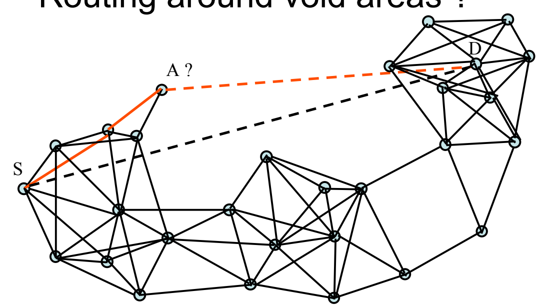
Power_reluctance $= f(A)u(d)$

Localized power aware routing



- Kuruvila, Nayak, Stojmenovic 2004
- *Power progress:* minimize $(r^{\alpha}+c)/(d-a)$
- *Iterative power progress:* select *B* if *power(SB)+power(BA) < power(SA)*
- (Iterative) Projection power progress
- Shortest weighted path toward selected neighbor (Ruiz, Sanchez, 2007)

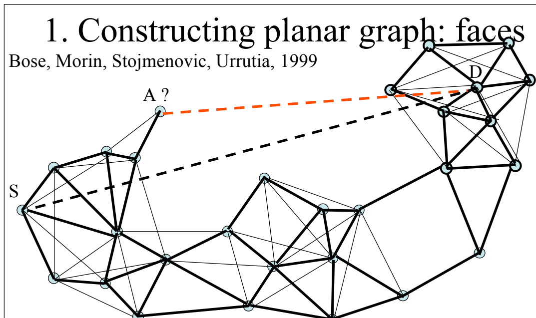
Routing around void areas ?



Recovery, perimeter, face mode

1. Constructing planar graph: faces

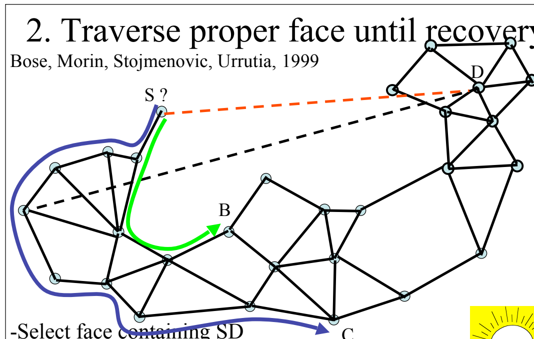
Bose, Morin, Stojmenovic, Urrutia, 1999



Some planar graphs (Gabriel graph) can be constructed without message exchange!

2. Traverse proper face until recovery

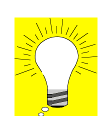
Bose, Morin, Stojmenovic, Urrutia, 1999



-Select face containing SD

- Follow that face by **left** hand or **right** hand rule

until recovery (= closer node reached)

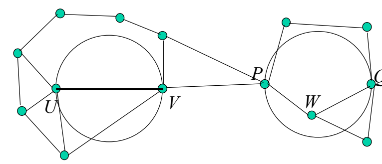


GFG= Greedy-FACE-Greedy

- run Greedy until delivery or a failure node A, |AD|=d,
- run FACE until delivery or B reached, |BD|<d,
- run Greedy …
- paths close to SP for higher degrees,
- <3.5 times longer than SP for low degrees
- No traffic memorization, localized, close to SP → ***scalable*** !! Bose, Morin, Stojmenovic, Urrutia, 1999
- Karp and Kung MOBICOM 2000 duplicated (with citation) GPSR= GFG (added MAC, mobile nodes)

Gabriel graph

Gabriel, Sokal 1984



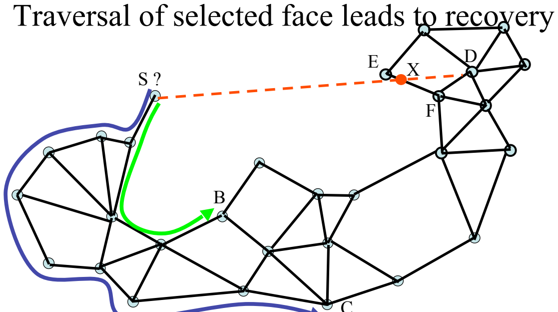
Gabriel graph GG(S) contains an edge (U,V) iff the disk with diameter (U,V) contains no other point from S

= distance from other points to center of UV is > |UV|/2

= Acute angles for all joint neighbors → in GG

GG(S) is planar and connected (contains MST)

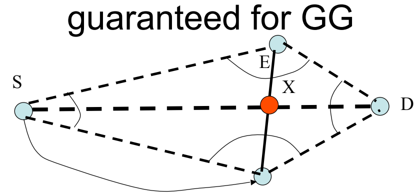
Traversal of selected face leads to recovery



-Line SD intersects the face in X on an edge EF

- E or F is closer to D than A (if nothing else found before)

Getting closer on the face is guaranteed for GG

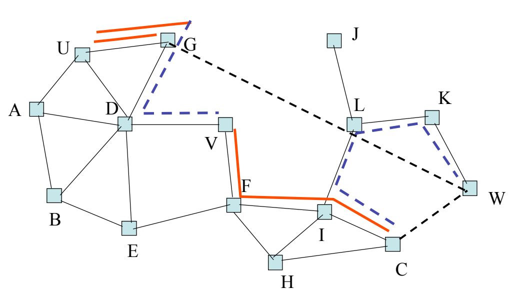


$\angle S < \pi/2$, $\angle D < \pi/2$ since EF is in GG → $\angle E > \pi/2$ or $\angle F > \pi/2$

$\angle F > \pi/2$ → |SD| > |FD| → F is closer to D than S

Frey, Stojmenovic MOBICOM 2006

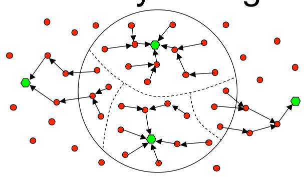
Greedy, GFG (greedy-face-greedy)



Robustness of GFG

- GFG requires unit graph = equal transmission radius, no obstacles, nodes in plane
- Extension for fuzzy unit graphs = connected if distance < r, nor connected if distance >R, may or may not be connected otherwise, R/r < 1.41
  Barriere, Fraigniaud, Narajanan, and Opatrny 2001
- Loop-free for static nodes; loops can be created by mobile nodes but exit can be found by adding timestamp of the last intersection with imaginary line SD and ignoring links created afterwards

Anycasting



Routing from a sensor to one of sinks/actuators

Position of sinks/actuators known

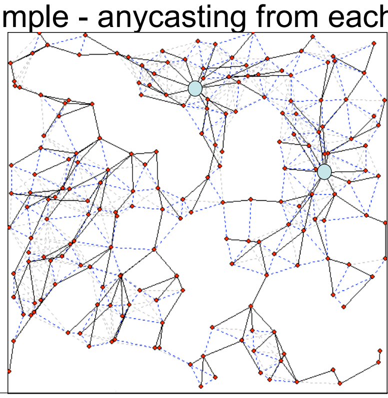
Anycasting may advance toward one sink but could eventually reach a different one

Kalosha, Stojmenovic 2007 (in progress)

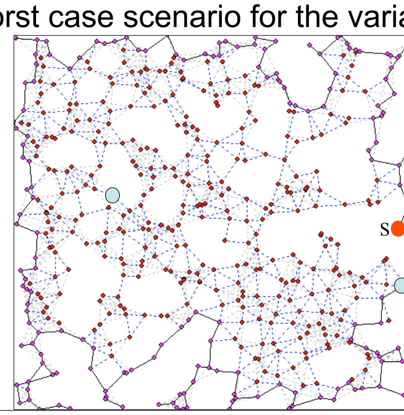
Algorithm - one variant

- Modified GFG approach
- In greedy mode, select neighbor providing best cost/progress ratio toward any actuator
- to preserve a single path, select only the closest sink node D for face routing toward it.
- The distance to D is recorded and forwarded with the message. Recovery mode stops when a node has a neighbor that is at shorter distance to one of sinks (not necessarily D) than recorded distance.
- Other variants: work in progress

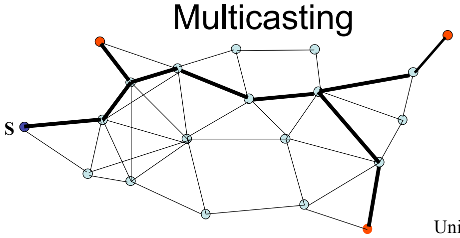
Example - anycasting from each sensor



A worst case scenario for the variant



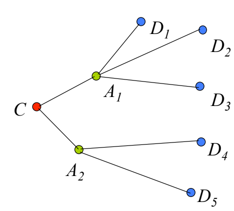
Multicasting



Unit graphs radius

- source → several destinations
  Position information
- Sanchez, Ruiz, Liu, Stojmenovic 2006
- Stojmenovic IEEE Network Jan. 2006

Evaluating the candidate forwarding from C to $A1$ and $A2$



The current total distance is $T1=|CD_1|+|CD_2|+|CD_3|+|CD_4|+|CD_5|$.

new total distance is $T_2=|A_1D_1|+|A_1D_2|+|A_1D_3|+|A_2D_4|+|A_2D_5|$,

Progress made is T1-T2, cost is 2 transmissions

forwarding set $\{A1, A2\}$ is evaluated as $2/(T_1-T_2)$.

GMR: Multicasting algorithm

- Greedy advance toward each group of destinations, with or without splitting
- If no greedy advance toward any destination, follow face routing toward it
- Several destinations could be followed by same faces for a while
- Continue greedy advance after recovery
- Power instead of hop count as a metric ?

Multicasting to many destinations

- Das, Pucha, Hu 2006
- Destinations are locally grouped
- Group leaders report to source
- Source constructs Minimal Spanning Tree of group leaders, and
- Initiates greedy routing between edges in MST (face routing added to recover)
- MST can be replaced by cost-to-progress ratio framework (in progress, Stojmenovic et all)

HGMR: Hierarchical multicasting

- Hierarchical Geographic Multicast Routing for Wireless Sensor Networks
- Dimitrios Koutsonikolas, Saumitra Das, Y. Charlie Hu, and Ivan Stojmenovic 2007
- starts with a hierarchical decomposition of a multicast group into subgroups of manageable size using HRPM's key concept *mobile geographic hashing*.
- Within each subgroup, HGMR uses GMR's *local multicast scheme* to forward a data packet along multiple branches of the multicast tree in one transmission.

Localized Mobility Control Routing in Robotic Sensor and Actuator Wireless Networks

2007

Hai Liu
Amiya Nayak
Ivan Stojmenović
www.site.uottawa.ca/~ivan

Problem Specification

Fixed source and destination, long term traffic

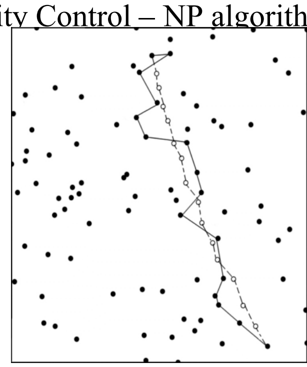
Mobile sensors, robots, actuators, human, vehicles … as intermediate nodes:

find a route and move each node on the route, such that

total transmission power is minimized,

total movement distance is minimized.

Routing Paths Before and After Mobility Control – NP algorithm



Assumptions

- All nodes have the common communication radius *r*.
- Energy cost model is $d^{\alpha} + c$ where *d* is distance
- Each node knows locations of its neighbors and its own location
- Energy to move is proportional to distance moved

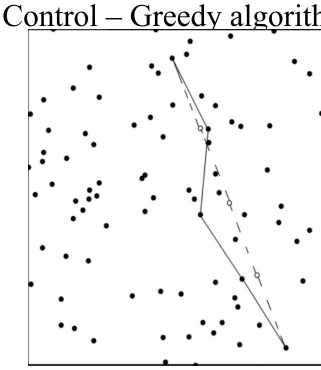
Existing Solutions

apply some routing (Greedy, NP) to establish an initial route; iteratively, each node (except for source and destination) moves to the midpoint of its upstream node and downstream node on the route.

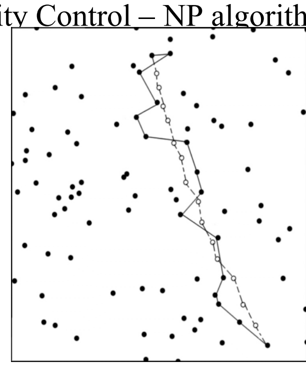
Greedy (forward to neighbor closest to destination), or
NP (forward to nearest neighbors with progress)

[GLM] D.K. Goldenberg, J. Lin, and A.S. Morse, "Towards Mobility as a Network Control Primitive," ***Mobihoc'04***, pp. 163-174, Japan, 2004.

Routing Paths Before and After Mobility Control – Greedy algorithm



Routing Paths Before and After Mobility Control – NP algorithm



Drawbacks and Motivation

- Initial route is not energy optimized
- Too many or too little forwarders
- Route after node movement may be far from energy optimal
- Iterative movement of nodes in rounds requires messages for synchronization and causes unnecessary zig-zig movement
- Large delay and possible communication failures

Contributions

- Study the optimal number of hops and optimal distance of adjacent nodes on the route.
- Propose OHCR algorithm which is based on the optimal **number of hops** on the route.
- Propose MPoPR algorithm which minimizes transmission **power** over progress.
- Study both strategies of **move in rounds** and **move directly**.

Overview of Our Solutions

two steps:

- compute **optimal number of hops** and **optimal distance of adjacent nodes** on the route.
- a routing algorithm that is based on the **optimal number of hops**, and
  a greedy algorithm that minimizes transmission **power over progress** in selecting a forwarding neighbor.

Optimal Number of Hops and Distance

Theorem 1. to minimize total transmission power of route from s to t, the optimal number of hops on the straight line route is integer k, minimizing

$$|k - d(s,t)\times((\alpha-1)/c)^{1/\alpha}|$$

and the optimal distance of adjacent nodes is $d(s,t)/k$, with energy cost model $d^{\alpha}+c$.

Optimal Hop Count Routing (OHCR)

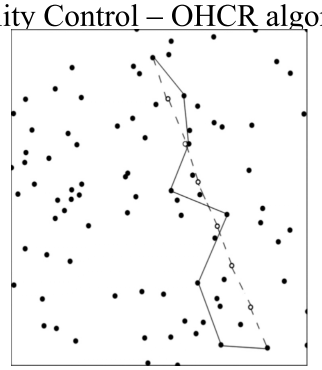
round $d(s,t)\times((\alpha-1)/c)^{1/\alpha}$ to the nearest integer k;

compute optimal distance of adjacent nodes $d(s,t)/k$;

if $k\leq 0$ and $d(s,t)\leq r$ s transmits directly to t;

current node u selects neighbor v such that $|d(u,v)-d(s,t)/k|$ is minimized;

Routing Paths Before and After Mobility Control – OHCR algorithm



Minimum Power over Progress Routing (MPoPR)

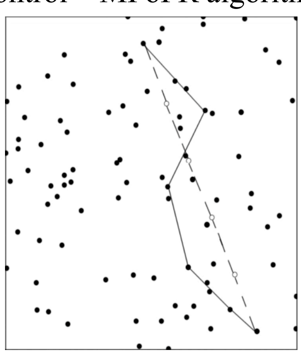
minimize transmission power of unit progress in selecting a forwarding neighbor.

u selects neighbor v such that

$$(d(u,v)^{\alpha}+c)/(d(u,t)-d(v,t))$$

is minimized;

Routing Paths Before and After Mobility Control – MPoPR algorithm



Move Directly Strategy

- Destination learns actual number of hops
- Which is routed backward to all nodes on route
- That then learn actual target location to move
- Moves directly to decided position, no zig-zag

Conclusions and Future Work

- MPoPR is a good solution for move in rounds strategy while OHCR is good for move directly strategy.
- Move directly strategy costs less total energy than move in rounds strategy.
- Use mobility control to improve network performance on other aspects, e.g., network capacity.
- Incorporate face routing into our algorithms to adapt sparse networks.