

1 Introduction

In this scribe note, we review two leader election algorithms for mobile ad hoc networks in the paper *Leader election algorithms for mobile ad hoc networks* by Navneet Malpani etc. Mobile ad hoc networks are different from wired networks, where links change infrequently. They have the following properties:

- No static infrastructure
- Wireless Interfaces
- Highly mobile nodes
- Each node acts as processor / router

In addition, they have the following difficulties.

- Node mobility : Dynamic topology
- Link failures and formation
- Partitions / Merging

Leader election has applications on key distribution, routing coordination, token distribution, etc. The algorithms proposed in this paper ensure that eventually each connected component of the topology graph has exactly one leader. The algorithms are based on a routing algorithm called Temporally-Ordered Routing Algorithm (TORA Park-Corson 1997), which in turn is based on an algorithm by GB (Gafni-Bertsekas 1981).

1.1 Definition. *Wireless link failure occurs when previously communicating nodes move such that they are no longer within transmission range of each other.*

1.2 Definition. *Wireless link formation occurs when nodes that were too far separated to communicate move such that they are within transmission range of each other.*

The standard definition of the leader election problem for static networks is that

1. eventually there is a leader
2. there should never be more than one leader

This paper modified these ideas in the following ways.

1. Instead of having a single destination-oriented DAG, we ensure that each component eventually forms a leader-oriented DAG.
2. When a partition from the current leader is detected (using the TORA mechanism), a new leader is elected and its id is propagated throughout the component.
3. When two components merge, a contest takes place between the leaders so that the winner's id is propagated and wipes out the loser's id.
4. When multiple topology changes occur, additional complications arise. This is due to the fact that while a new leader's id is being propagated changes could occur in the component and the process of electing a leader may be repeated.

2 Definition - System Model and Assumptions

The system contains a set of n independent mobile nodes, communicating by message passing over a wireless network. The network is modeled as a dynamically changing, not necessarily connected, undirected graph, with nodes as vertices and edges between vertices corresponding to nodes that can communicate. Assumptions on the mobile nodes and network are:

1. The nodes have unique node identifiers.
2. Communication links are bidirectional, reliable and FIFO. Unidirectional links, if any, are not used and ignored.
3. A link-level protocol ensures that each node is aware of the set of nodes with which it can currently directly communicate by providing indications of link formations and failures.
4. For the algorithm that we present in section 4, we assume that only one change can occur at a time. The next change occurs only after the entire network has recovered from the previous change.

3 Definition - Problem Statement

Each node i in the system must have a local variable lid_i that holds the identifier of the node currently considered to be the leader of i 's component. We require that in every execution with a finite number of topology changes, eventually it holds that:

- For every connected component C of the topology graph, there is a node l in C such that $lid_i = l$ for all nodes i in C .

An additional requirement, which might be useful in some applications, and is satisfied by our algorithm, is that each edge has a direction imposed on it by the endpoints such that eventually

- Each connected component is a directed acyclic graph with the leader as the single sink (called a *leaderoriented* or *l -oriented* DAG).

4 Leader Election Algorithm for a Single Topology Change

The proposed algorithm is a modification of the TORA routing algorithm, which in turn is based on a routing algorithm by Gafni and Bertsekas (GB).

4.1 Overview of the GB Algorithm

Full Reversal Method: a node i other than the destination has no outgoing link if for every neighbor j of i we have $(\alpha_i^k, i) < (\alpha_j^k, j)$. At the k th iteration such a node i increases α_i^k to

$$\alpha_i^{k+1} = \max\{\alpha_j^k \mid j \text{ is a neighbor of } i\} + 1,$$

while all other nodes j maintain the same number, i.e., $\alpha_j^{k+1} = \alpha_j^k$.

Partial Reversal Method: a node i other than the destination for which $(\alpha_i^k, \beta_i^k, i) < (\alpha_j^k, \beta_j^k, j)$ for all neighbors j increases α_i^k to

$$\alpha_i^{k+1} = \min\{\alpha_j^k \mid j \text{ is a neighbor of } i\} + 1$$

and sets β_i^k to

$$\beta_i^{k+1} = \begin{cases} \min\{\beta_j^k \mid j \text{ is a neighbor of } i \text{ with } \alpha_i^{k+1} = \alpha_j^k\} - 1 \\ \text{if there exists a neighbor } j \text{ with } \alpha_i^{k+1} = \alpha_j^k \\ \beta_i^k \text{ otherwise.} \end{cases}$$

All other nodes j maintain the same integers α_j and β_j , i.e., $\alpha_j^{k+1} = \alpha_j^k, \beta_j^{k+1} = \beta_j^k$.

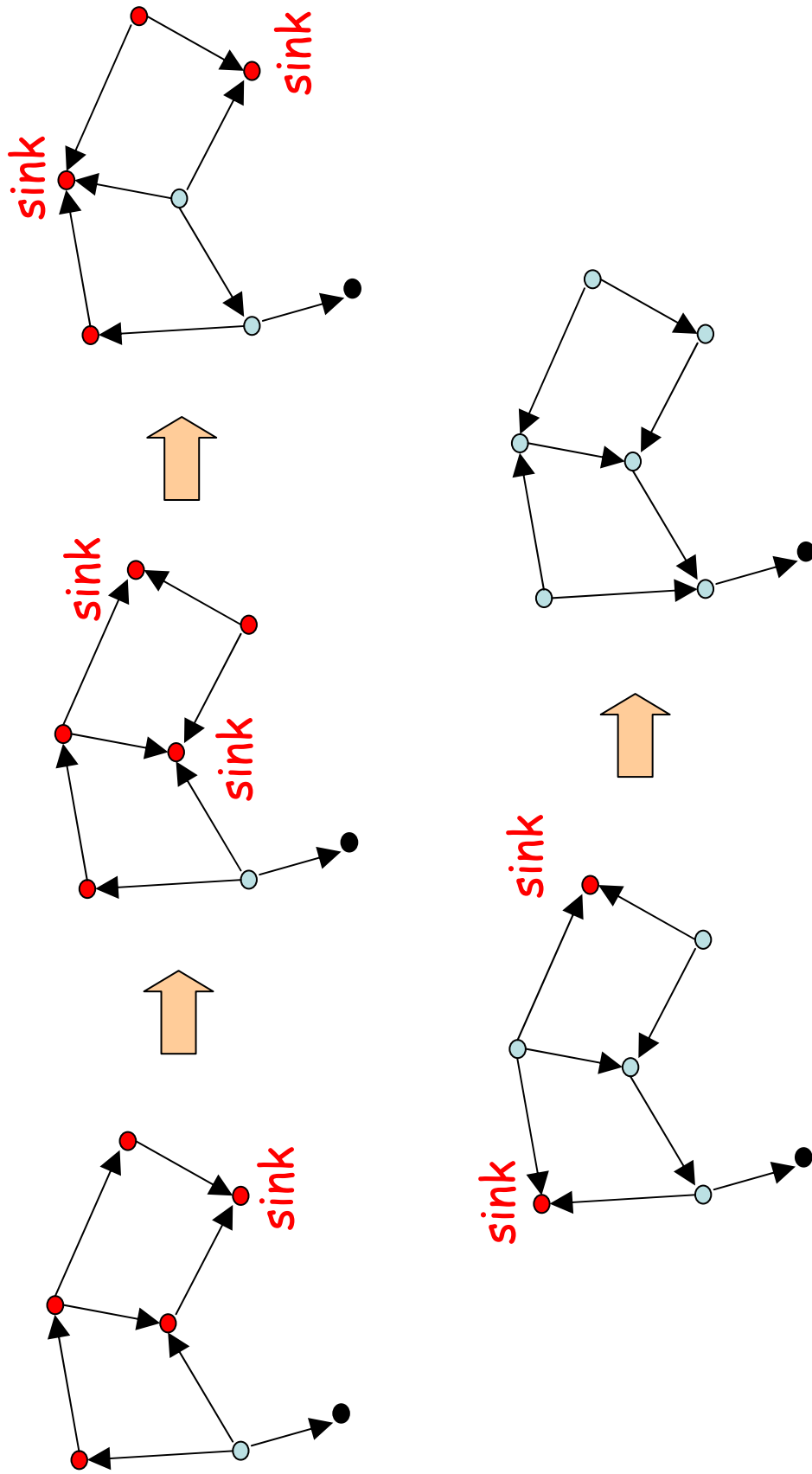


Figure 1: GB - Full Reversal Method

Sinks reverse all their links

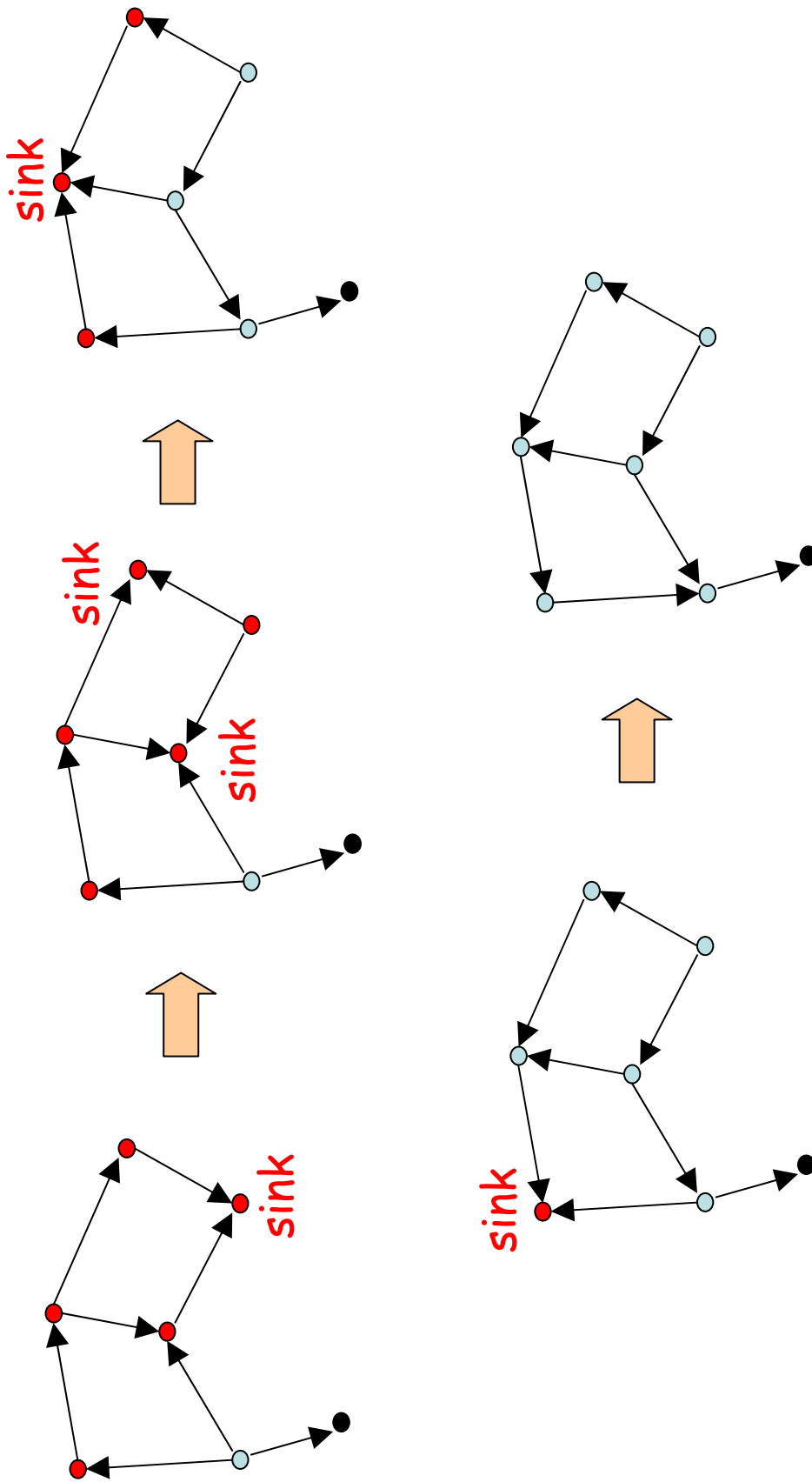


Figure 2: GB - Partial Reversal Method

Sinks reverse some of their links

4.2 Overview of TORA

Park and Corson adapted the GB algorithm for routing in mobile ad hoc networks, calling the result TORA (for Temporally Ordered Routing Algorithm). TORA provides mechanism for detecting when a piece of the network has been partitioned so that the destination is no longer reachable. GB algorithms would cause an infinite cycle of messages in this case.

In TORA, the height of node i is a 5-tuple $(\tau_i, oid_i, r_i, \delta_i, i)$. The first three components form a reference level. Once a partition has been detected, the node that first detected the partition sends out indications to the other nodes in its component so that they cease performing height changes and sending useless messages.

A node i can lose all its outgoing links due to a neighbor's height change under a number of different circumstances, which are now explained.

- If the neighbors of i do not all have the same reference level, then i sets its reference level to the largest among all its neighbors and sets its δ to one less than the minimum δ value among all neighbors with the largest reference level (a partial reversal).
- If all of i 's neighbors do have the same reference level and it is an unreflected one, then i starts a reflection of this reference level by setting its reference level to reflected version of its neighbors' (with $r_i = 1$) and its δ to 0.
- If all of i 's neighbors have the same reflected reference level with i as the originator, then i has detected a partition and takes appropriate action.
- If all of i 's neighbors have the same reflected reference level with an originator other than i , then i starts a new reference level. This situation only happens if a link fails while the system is recovering from an earlier link failure.

5 Leader Election Algorithm

Here we describe the code executed by node i . Each step is triggered either by the notification of the failure or formation of an incident link or by the receipt of a message from a neighbor. Node i stores its neighbors' ids in local variable N_i . When an incident link fails, i updates N_i . When an incident link forms, i updates N_i and sends an Update message over the link with its current height.

At the end of each step, if i 's height has changed, then it sends an Update message with the new height to all its neighbors. The pseudocode below explains how and when node i 's height is changed. Parts B through D are executed only if the leader id in the received Update message is the same as lid_i .

In part E, if the new id is smaller than yours, then adopt it. If the new id is larger than yours, then adopt it, but only if it is the case that the originator of a new reference level has detected a partition and elected itself.

A. When node i has no outgoing links due to a link failure:

1. if node i has no incoming links as well then
2. $lid_i := i$
3. $(\tau_i, oid_i, r_i) := (-1, -1, -1)$
4. $\delta_i := 0$
5. else
6. $(\tau_i, oid_i, r_i) := (t, i, 0)$ // t is the current time
7. $\delta_i := 0$

B. When node i has no outgoing links due to a link reversal following reception of an Update message and the reference levels (τ_j, oid_j, r_j) are not equal for all $j \in N_i$:

1. $(\tau_i, oid_i, r_i) := \max\{(\tau_j, oid_j, r_j) | j \in N_i\}$
2. $\delta_i := \min\{\delta_j | j \in N_i \text{ and } (\tau_j, oid_j, r_j) = (\tau_i, oid_i, r_i)\} - 1$

C. When node i has no outgoing links due to a link reversal following reception of an Update message and the reference levels (τ_j, oid_j, r_j) are equal with $r_j = 0$ for all $j \in N_i$:

1. $(\tau_i, oid_i, r_i) := (\tau_j, oid_j, 1)$ for any $j \in N_i$
2. $\delta_i := 0$

D. When node i has no outgoing links due to a link reversal following reception of an Update message and the reference levels (τ_j, oid_j, r_j) are equal with $r_j = 1$ for all $j \in N_i$ and $oid_j = i$:

1. $lid_i := i$
2. $(\tau_i, oid_i, r_i) := (-1, -1, -1)$
3. $\delta_i := 0$

E. When node i receives an Update message from neighboring node j such that $lid_j \neq lid_i$:

1. if $lid_i > lid_j$ or $(oid_i = lid_j \text{ and } r_i = 1)$ then
2. $lid_i := lid_j$
3. $(\tau_i, oid_i, r_i) := (0, 0, 0)$
4. $\delta_i := \delta_j + 1$

Figure 3: Pseudocodes of Leader Election Algorithms