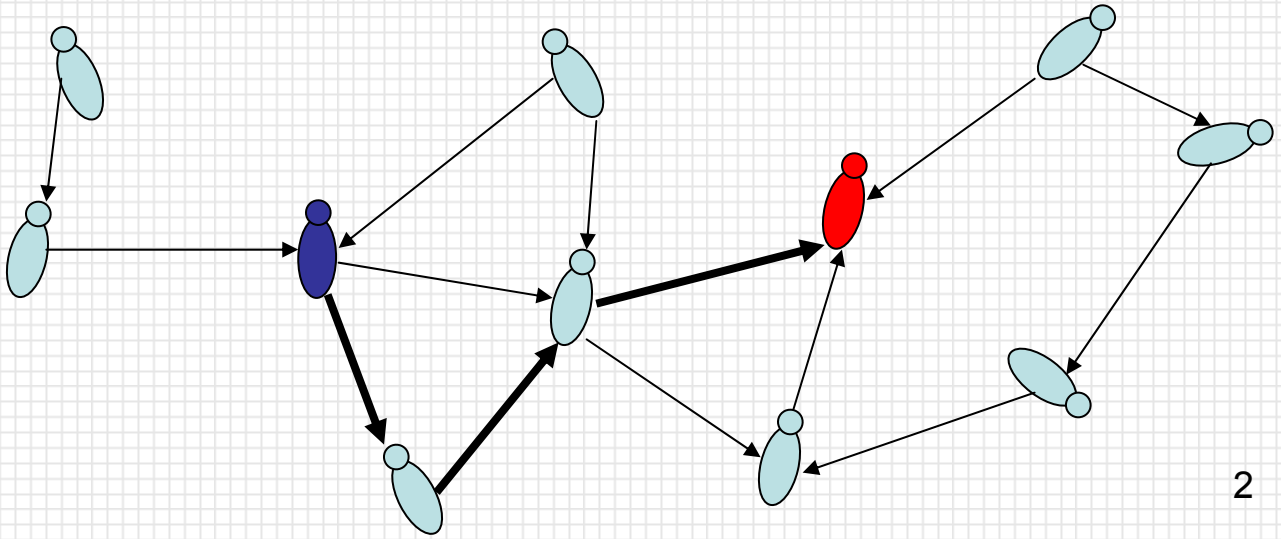
Overview

- Mobile Ad Hoc Networks
- Routing
  - Gafni-Bertsekas Link Reversal
  - Temporally Ordered Routing Algorithm (TORA)
- Leader Election

Routing Problem

- Given G=(V,E), undirected, connected
- Devise a new graph G´
  - Directed
  - Acyclic
  - Destination-oriented

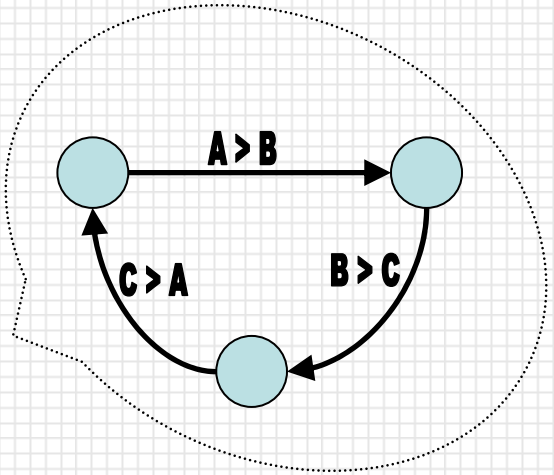


Routing Solutions

- Minimum Spanning Tree
  - O(n·log(n)) time
  - O(n·log(n) + |E|) messages
- Breadth-First-Search
  - O(*diam*) time
  - O(*diam*·|E|) messages
- Centralized
  - One node calculates central routing tables
  - Other nodes request and cache routes

Step 1: Choose Randomly

- Randomly choose a direction for each link
  - Choose a random number *n* between 1 and K
  - $(n_i, id_i) > (n_j, id_j) \Rightarrow i \rightarrow j$
- Result:
  - Directed (a<b or b<a)
  - Acyclic
  - Destination **dis-**oriented



Step 2: Orient the Graph

- Full Reversal
  - If no outgoing edges, reverse all links
- Partial Reversal
  - If no outgoing edges, reverse some links
  - Keep list of all reversed links
  - Reverse all links not in list
  - Empty list

Theorem 5: It Works

- If:
  - Given an undirected, connected graph
  - Arbitrary edges in the graph are added and removed, such that the graph is always connected
  - Arbitrary nodes are added and removed such that the graph is always connected
  - If there is eventually a period of time in which there are few enough changes
- Then
  - Eventually there will be a destination-oriented, directed, acyclic graph

Partitioned Networks

- Gafni-Bertsekas algorithm does not terminate (if not connected)!
  - Generates large network traffic
  - Creates unbounded large heights
- Goal
  - Partition Detection

TORA

- Temporally Ordered Routing Algorithm
- Use heights to maintain a destination-oriented DAG
- When no path to destination is available, detect and delete all paths

TORA Heights

- Height (T, oid, r, d, i)
  - Reference Level
    - T = time
    - oid = origin id
  - Partition detection
    - r = reflection
  - Generic (local) height
    - d = height
    - i = node id

Leader Election

- Connected Network
  - Use GB: smallest initial height → leader
- Disconnected Network
  - One leader per component
  - “Eventually” there is a leader
  - “Eventually” only one leader
  - Use TORA

Leader Election

- Leader-oriented, directed, acyclic graph
- On partition
  - First node to detect is new leader
- On merge
  - Leader with lower UID is new leader
- Height (lid, T, oid, r, d, i)

Using TORA

- Case A: Link Failure
  - If no incoming links, leader:=self
  - Else, new reference level
    - (lid, T, i, 0, 0, i)
- Case B: Link reversed, Neighbors different
  - (T, i, r) = reference level of highest neighbor
  - d = min(d_j)-1, for all neighbors j with (T, i, r)

Using TORA

- Case C: Link reversal, neighbors equal, r==0
  - (T, oid, r) = (T_j, oid_j, 1)
  - d := 0
- Case D: Link reversal, neighbors equal, r==1, oid_j == i
  - lid := i
  - (lid, -1, -1, -1, 0, i)

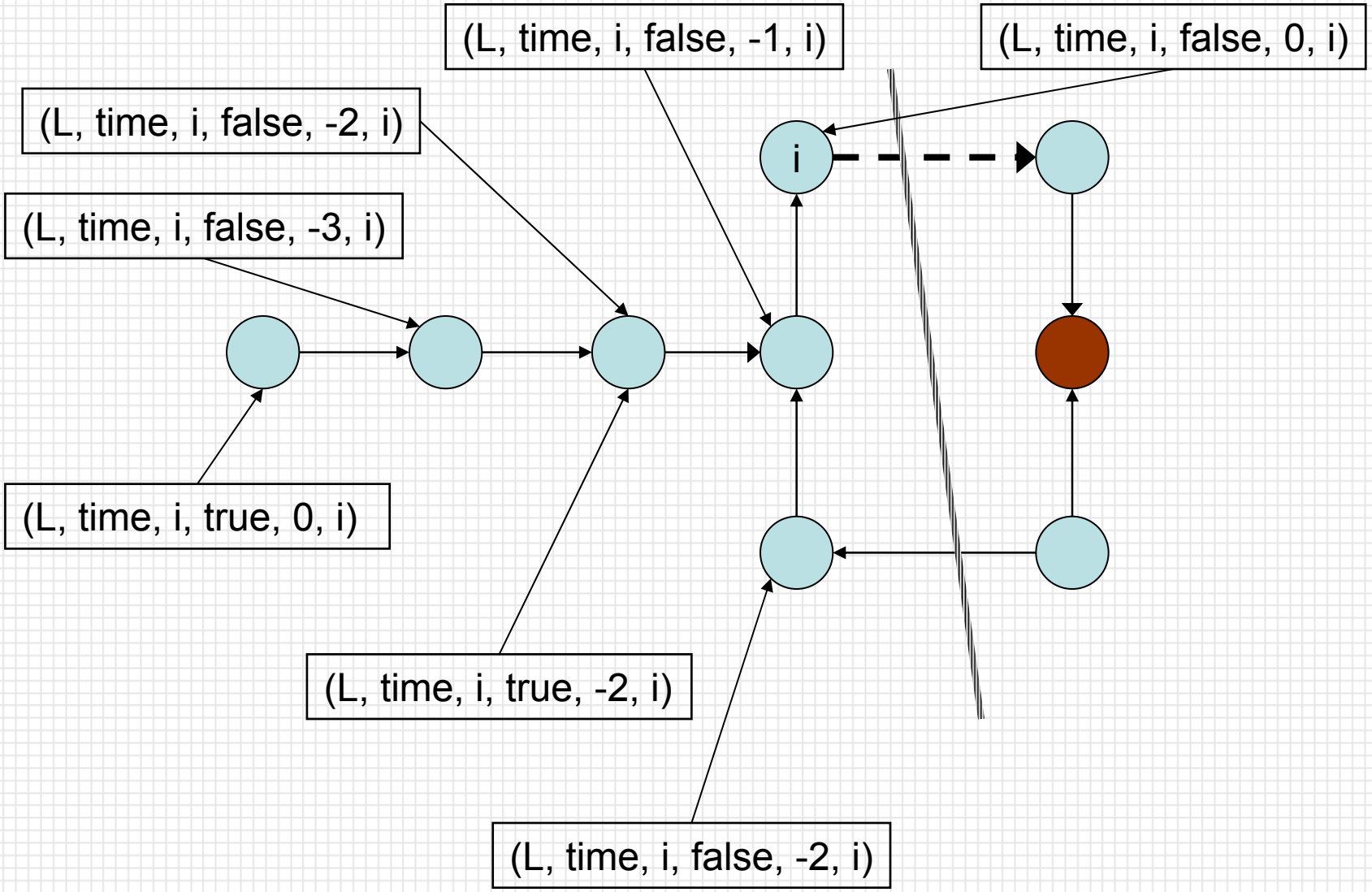
Using TORA

- Case E: $lid_j \neq lid_i$
  - if $lid_i > lid_j$ (or oid == lid_j and r == 1)
    - lid := lid_j
    - (lid, 0, 0, 0, d+1, i)

Correctness

- Assume leader-oriented DAG
- Assume only one link failure/addition
- Assume synchronous
- Theorem 7
  - Each component eventually has exactly one unique leader

Example


(L, time, i, false, -1, i)
(L, time, i, false, 0, i)
(L, time, i, false, -2, i)
i
(L, time, i, false, -3, i)
(L, time, i, true, 0, i)
(L, time, i, true, -2, i)
(L, time, i, false, -2, i)