Distributed Graph Algorithms

Distributed Graph Algorithms

- Distributed algorithms exist for many graph problems
- Given in assignments so far
  - Spanning tree (arbitrary)
  - BFS Spanning Tree
  - DFS Spanning Tree
    - An example of straightforward conversion of a sequential algorithm: DFS Spanning Tree
- You already know
  - Shortest Path (Distance Vector Routing in Networks)
- We will do
  - An example of an algorithm requiring 2-hop neighborhood info: Dominating Set
  - An example of a local algorithm that is not so obvious: Topology Control Algorithm (Wattenhofer-Zollinger)

Dominating Set

- Input: A graph G = (V, E)
- Output: A dominating set – A subset S of V such that for any node u in V, either u is in S or a neighbor of u is in S
  - Ideally, we want the minimum dominating set (a dominating set of minimum size among all dominating sets)
  - Computing Minimum Dominating Set is NP-Hard
- Many variations with interesting applications
  - Routing backbone, information spread,….

Greedy Distributed Algorithm for Dominating Set

- Model: Synchronous, reliable
- Idea:
  - Start with an empty set S
  - Iteratively add nodes to S greedily
  - Stop when S is a dominating set
- A node is said to be covered if either it is in S or it has a neighbor in S
- For any node v, define its span as the no. of uncovered nodes in $\{v\} \cup N(v)$

- Algorithm works in rounds
- Each node does 3 steps in each round
  - Compute its span
  - Sends its (span, id) to all 2-hop neighbors (nodes within 2 hop)
  - Adds itself to S if its span is greater than span of all nodes within 2-hop neighborhood (break ties with id)
- Termination when all nodes are covered
- $\ln(\Delta)$-approximation algorithm, polynomial running time (no. of rounds)

Topology Control

- Important problem in ad hoc networks
- Input: A graph $G = (V, E)$
- Output: A graph $G_c = (V, E_c)$ such that E_c is a subset of E and G_c satisfies some desired properties
  - connectivity, degree constraints, path lengths,….

XTC Algorithm (Wattenhofer-Zollinger)

- Weighted graph
- Code for node u

```
Order neighbors of u by ascending order
  of weights in a list L
Broadcast the order to all neighbors,
  and receive all neighbors' ordering of
  their neighbors
N = M = Φ
While L has unprocessed nodes
  v = first unprocessed node in L
  If there exists w in (N U M) such that
    w comes earlier than u in v's list
    then add v to M
  Else add v to N
Output N as the final neighbor set of u
```

- Guarantees
  - If u chooses v as neighbor, then v chooses u as neighbor (true for arbitrary orderings also)
  - Output graph is connected if and only if input graph is connected
  - There is no cycle of length 3 in the output graph
- Classic example of a distributed algorithm where nodes takes some local action and some global properties are satisfied
  - Not so obvious why it works, unlike the algorithms you have seen so far