Massachusetts Institute of Technology
6.895: Advanced Distributed Algorithms
Professor Nancy Lynch

Lecture 17
November 5, 2002

Applications of Sparse Covers

1 Introduction

- Awerbuch and Peleg
- Presenting two algorithms
    1. Routing (static)
    2. Tracking Mobile Users (dynamic)
- Application of Sparse Covers
- Focus on trade-off between degree and radius

2 Routing

- Problem Statement:
    - Graph: $G = (G, E)$
    - Edge weights: $w : E \to Z$
    - Deliver messages from node $u \to v$
- Topology is known
- Trivial (optimal time) solution
    - Run all-pairs-shortest path
    - Maintain next-hop routing table
    - $O(n \cdot \log(n))$ space
- This gives optimal distance solution, but takes too much space
- Trivial space-efficient solution
    - Choose a leader L
    - u sends message to L
    - L sends message to v
    - good space bound (only have to route to L)
    - ITR (Interval Routing)
- Bad time bound
- At worst $2 * diameter$

- Use hierarchical covers to find a mid-point
- Let C be a cover
  - if u, v in same cluster in C then
    - deliver messages
  - else
    - fail
  - Use ITR as above:
    - For each cluster, choose a leader
    - Route from u to leader, leader to v
- Good space bound: only route to leader
- Larger degree, larger space bound
- Must route to leader in each cluster
- Good time bound: $2 * diam(C)$
- Note: larger diameter, worse delivery time
- Can't deliver to all nodes!! (only nodes in cluster)
- Therefore: choose hierarchy of covers C(1),C(2),C(3), ...
- If $dist(u, v) < m$:
  - then u and v are in the same cluster in cover $C(\log(m))$
  - $Radius(C(k)) = 2^k$
- Algorithm
  - Start at cluster C(1)
  - If routes successfully, stop
  - Else go to cluster C(2) and repeat
- Analysis
  - Let d = dist(u, v)
  - Worst delivery time (at end of hierarchy): 4d
  - Note: delivered over log(n) smaller levels of hierarchy as well
  - Larger radius, worse delivery time
  - Good space bounds: log(n) levels of routing information, instead of n table entries of routing information
  - Larger degree, worse space bounds
- How to get such a cover?
- Can't get EXACTLY this good a cover, so results aren't as clean
  - Take one neighborhood cover of size m
  - Coarsen until sets are about the right size

  - If $dist(u,v) < m$:
  - Take cover in $N_m(u)$
  - We know v is in $N_m(u)$
  - By coarsening rules:
  - $\exists S \in C st. N_m(u) \subseteq S$

- Final results:
- For all integers $k > 0$:
- Delivery $= d * k^2$
- Space $= n^{(\frac{1}{k})} * \log(n)^2 * \log(diam)$

3 Tracking Mobile Users

- Warning:
- Slightly different presentation from the paper
- Emphasizes similarity to Routing problem
- (and slightly simpler)
- Problem Statement
  - Graph G=(V,E)
  - Weights: $w : E \to Z$
  - User, u, mobile
  - User, v, fixed
  - FIND: Send message from v to u
  - MOVE Allow u to travel
- Assume no concurrency!
- Goal
  - Time to FIND should be proportional to $dist(u,v)$
  - Time to MOVE should be proportional to $dist(u,u')$
- Assume a cover C
  - u updates every leader, for all clusters, to its new location
  - v sends a message to the leader of its cluster
  - leader sends message to u
- Note similarity to routing case!
- If u and v are in the same cluster, success!
- FIND(C) proportional to diameter
- MOVE(C) proportional to diameter*degree

- Use hierarchy of covers C(1), C(2),
- Choose covers such that if:
- $d = dist(u, v)$
- u and v share a cluster in cover $C(\log(d))$
- Simple FIND algorithm:
  - Start a smallest cover, and incrementally search larger covers until u is found
- Good FIND time!
- Update all covers??
- Expensive to update high diameter covers
- Only update high diameter covers when moving far
- New Move algorithm:
  - Remember the location known to each cover:
  - L(1), L(2), ...
  - Remember how far you have traveled since you last updated it
  - if $dist - traveled(L(i), u) > 2^i - 1$
  - update cover i (and all smaller)
  - Only update covers when you move far
- Beware stale data!
- Some covers contain bad location
- At old location $L(i + 1)$ store forwarding pointer
- New Find algorithm:
  - Start at smallest cover
  - Incrementally search larger covers until u is found
  - Follow forwarding pointers to u
- Search is fast:
  - if $d = dist(u, v)$
  - then u and v share a cluster in cover $C(\log(d))$
  - At most search $\log(d)$ covers
  - Search takes time proportional to d
  - Forwarding pointers are fast:
  - Assume, v finds u in cover $C(i)$
  - $i \ \log(d)$
  - Update rule: $dist - traveled(L(i), u) < 2^i$
  - $dist - traveled(L(i), u) < d$
- Trade-off between diameter and degree

- Need small diameter, small degree
- Concurrent solutions:
  - Atomic updates of covers
  - locking
  - Ensure that v doesn't chase u forever down a forwarding path, as u keeps moving faster than v can search

4 Conclusion

- Hierarchical use of covers provides good locality
- Interesting generalization to m-matching (locality sensitive quorums)
- No fault tolerance!
- Assume static network
- Section on dynamic network, but still much work to be done