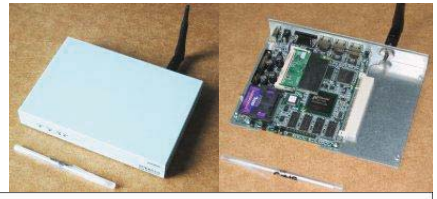
Traffic Monitoring and Routing Planning (CarTel)

- GPS equipped cars for optimal route predictions, not necessarily "shortest" or "fastest" but also "most likely to get me to target by 9am"
- Various other applications e.g. Pothole Patrol



Rating

- Area maturity

- Practical importance

- Theoretical importance

Overview

- Motivation
- (Connected) Dominating Set
- Some Algorithms
  - The "Greedy" Algorithm
  - The "Tree Growing" Algorithm
  - The "Marking" Algorithm
  - The "Complicated" Algorithm
- Connectivity Models: UDG, BIG, UBG, …
- More Algorithms
  - The "Largest ID" Algorithm
  - The "MIS" Algorithm

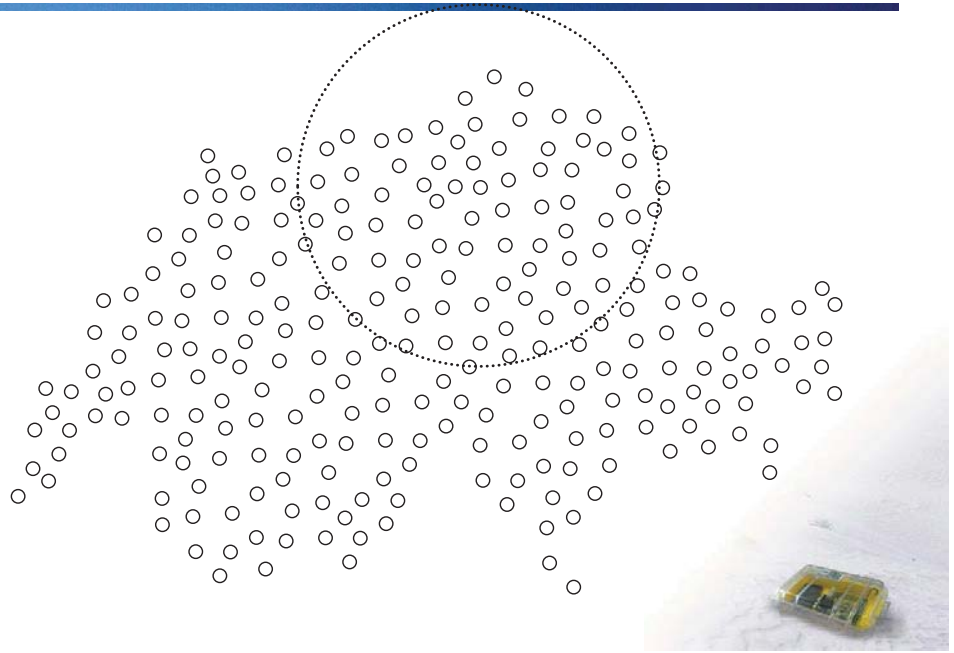
Motivation

- In theory clustering is the answer to dozens of questions in ad hoc and sensor networks. It improves almost any algorithm, e.g. in data gathering it selects cluster heads which do the work while other nodes can save energy by sleeping. Also clustering is related to other things, like coloring (which itself is related to TDMA). Here, we motivate clustering with routing:

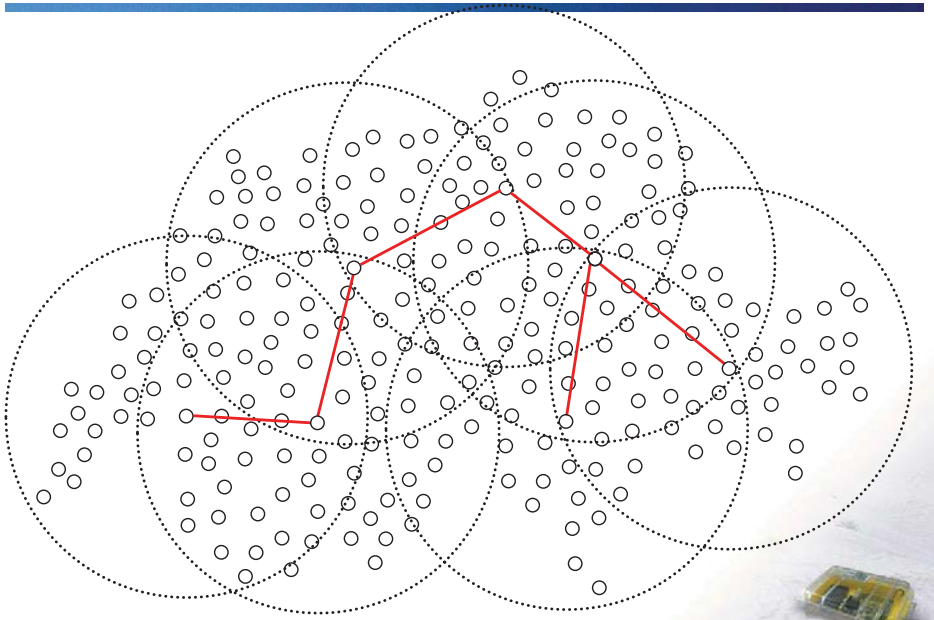
- There are thousands of routing algorithms…
- Q: How good are these routing algorithms?!? Any hard results?
- A: Almost none! Method-of-choice is simulation…

- Flooding is key component of (many) proposed algorithms, including most prominent ones (AODV, DSR)
- At least flooding should be efficient

Finding a Destination by Flooding



Finding a Destination *Efficiently*

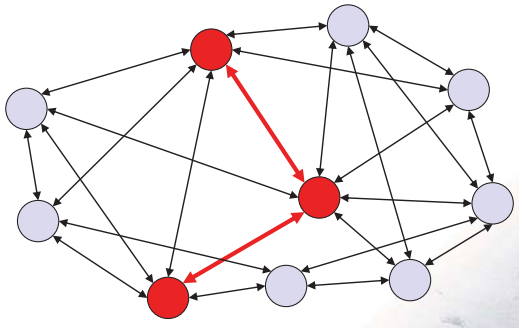


Backbone

- Idea: Some nodes become backbone nodes (gateways). Each node can access and be accessed by at least one backbone node.

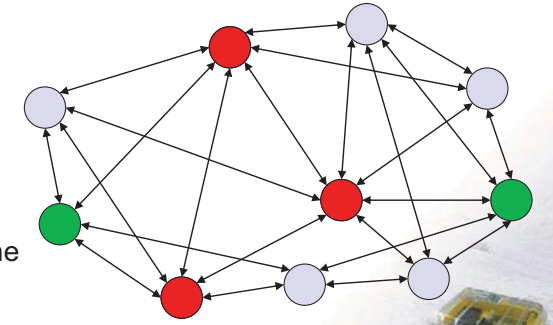
- Routing:

1. If source is not a gateway, transmit message to gateway
2. Gateway acts as proxy source and routes message on backbone to gateway of destination.
3. Transmission gateway to destination.



(Connected) Dominating Set

- A Dominating Set DS is a subset of nodes such that each node is either in DS or has a neighbor in DS.
- A Connected Dominating Set CDS is a connected DS, that is, there is a path between any two nodes in CDS that does not use nodes that are not in CDS.
- A CDS is a good choice for a backbone.
- It might be favorable to have few nodes in the CDS. This is known as the Minimum CDS problem



Formal Problem Definition: M(C)DS

- Input: We are given an (arbitrary) undirected graph.
- Output: Find a Minimum (Connected) Dominating Set, that is, a (C)DS with a minimum number of nodes.
- Problems
  - M(C)DS is NP-hard
  - Find a (C)DS that is "close" to minimum (approximation)
  - The solution must be local (global solutions are impractical for dynamic networks) – topology of graph "far away" should not influence decision which nodes belong to (C)DS

Greedy Algorithm for Dominating Sets

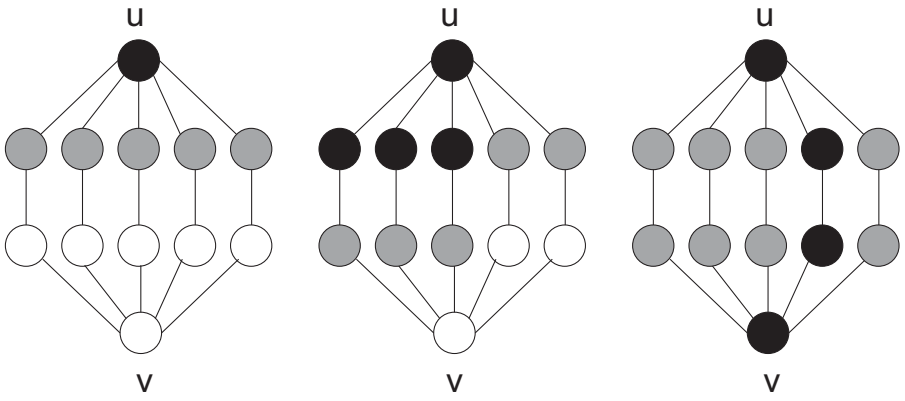
- Idea: Greedily choose "good" nodes into the dominating set.
- Black nodes are in the DS
- Grey nodes are neighbors of nodes in the DS
- White nodes are not yet dominated, initially all nodes are white.
- Algorithm: Greedily choose a node that colors most white nodes.
- One can show that this gives a log Δ approximation, if Δ is the maximum node degree of the graph.
  - The proof is similar to the "Tree Growing" proof on the following slides
  - It was shown that there is no polynomial algorithm with better performance unless P≈NP.

CDS: The "too simple tree growing" algorithm

- Idea: start with the root, and then greedily choose a neighbor of the tree that dominates as many as possible new nodes.
- Black nodes are in the CDS.
- Grey nodes are neighbors of nodes in the CDS.
- White nodes are not yet dominated, initially all nodes are white.
- Start: Choose a node with maximum degree, and make it the root of the CDS, that is, color it black (and its white neighbors grey).
- Step: Choose a grey node with a maximum number of white neighbors and color it black (and its white neighbors grey).

Example of the “too simple tree growing” algorithm

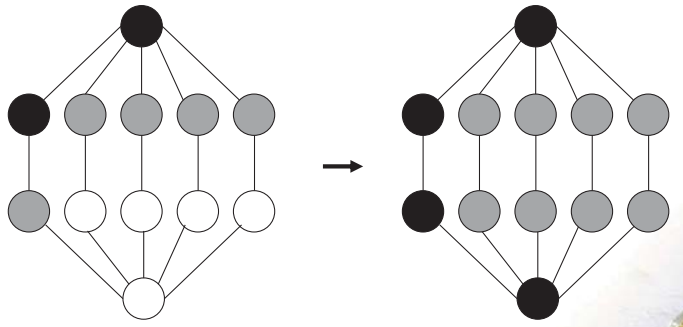
Graph with 2n+2 nodes; tree growing: |CDS|=n+2; Minimum |CDS|=4



tree growing: starting … Minimum CDS

Tree Growing Algorithm

- Idea: Don’t scan one but two nodes!
- Alternative step: Choose a grey node and its white neighbor node with a maximum sum of white neighbors and color both black (and their white neighbors grey).



Analysis of the tree growing algorithm

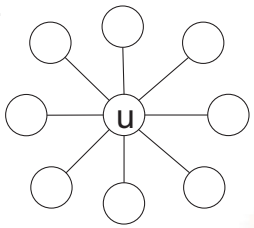
- Theorem: The tree growing algorithm finds a connected set of size $|CDS| \leq 2(1+H(\Delta)) \cdot |DS_{OPT}|$.
  - DS_{OPT} is a (not connected) minimum dominating set
  - Δ is the maximum node degree in the graph
  - H is the harmonic function with $H(n) \approx \log n + 0.7$
  - In other words, the connected dominating set of the tree growing algorithm is at most a O(log Δ) factor worse than an optimum minimum dominating set (which is NP-hard to compute).
  - With a lower bound argument (reduction to set cover) one can show that a better approximation factor is impossible, unless P≈NP.

Proof Sketch

- The proof is done with amortized analysis.
- Let S_u be the set of nodes dominated by $u \in DS_{OPT}$, or u itself. If a node is dominated by more than one node in DS_{OPT}, we put it in any one of the sets.
- Each node we color black costs 1. However, we share this cost and charge the nodes in the graph for each node we color black. In particular we charge all the newly colored grey nodes. Since we color a node grey at most once, it is charged at most once. Coloring 2 nodes black will turn γ nodes from white to grey, hence each of the γ nodes will be charged cost $2/\gamma$. We will show that the total charge on the vertices in S_u is at most $2(1+H(\Delta))$, for any u.

Charge on S_u

- Initially $|S_u| = u_0$ (in the example picture $u_0 = 9$).
- Whenever we color some nodes of S_u, we call this a step.
- The number of white nodes in S_u after step i is u_i.
- After step k there are no more white nodes in S_u.



- In the first step $u_0 - u_1$ nodes are colored (grey or black). Each vertex gets a charge of at most $2/(u_0 - u_1)$.
- After the first step, node u becomes eligible to be colored black (as part of a pair with one of the grey nodes in S_u). If u is not chosen in step i (with a potential to paint u_i nodes grey), then we have found a better (pair of) node. That is, the charge to any of the new grey nodes in step i in S_u is at most $2/u_i$.

Adding up the charges in S_u

$$C \le \frac{2}{u_0 - u_1}(u_0 - u_1) + \sum_{i=1}^{k-1} \frac{2}{u_i}(u_i - u_{i+1})$$

$$= 2 + 2\sum_{i=1}^{k-1} \frac{u_i - u_{i+1}}{u_i}$$

$$\le 2 + 2\sum_{i=1}^{k-1} \big(H(u_i) - H(u_{i+1})\big)$$

$$= 2 + 2(H(u_1) - H(u_k)) = 2(1 + H(u_1)) \le 2(1 + H(\Delta))$$

Discussion of the tree growing algorithm

- We have an extremely simple algorithm that is asymptotically optimal unless P≈NP. And even the constants are small.
- Are we happy?
- Not really. How do we implement this algorithm in a real (dynamic) network? How do we figure out where the best grey/white pair of nodes is? How slow is this algorithm in a distributed setting?
- We need a fully distributed algorithm. Nodes should only consider local information.

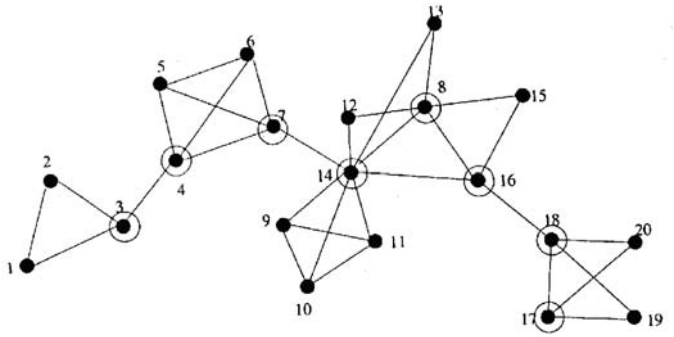
The Marking Algorithm

- Idea: The connected dominating set CDS consists of the nodes that have two neighbors that are not neighboring.

1. Each node u compiles the set of neighbors N(u)
2. Each node u transmits N(u), and receives N(v) from all its neighbors
3. If node u has two neighbors v,w and w is not in N(v) (and since the graph is undirected v is not in N(w)), then u marks itself being in the set CDS.

+ Completely local; only exchange N(u) with all neighbors
+ Each node sends only 1 message, and receives at most Δ
- Is the marking algorithm really producing a connected dominating set? How good is the set?

Example for the Marking Algorithm



[J. Wu]

Correctness of Marking Algorithm

- We assume that the input graph G is connected but not a clique.
- Note: If G was a clique then constructing a CDS would not make sense. Note that in a clique (complete graph), no node would get marked.
- We show:

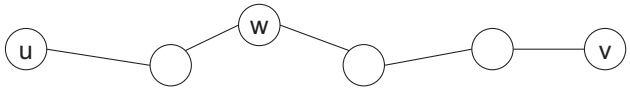
  The set of marked nodes CDS is
  a) a dominating set
  b) connected
  c) a shortest path in G between two nodes of the CDS is in CDS

Proof of a) dominating set

- Proof: Assume for the sake of contradiction that node u is a node that is not in the dominating set, and also not dominated. We study the nodes in $N^+(u) := u \cup N(u)$:
  - If a node $v \in N(u)$ has a neighbor w outside N(u), then node v would be in the dominating set (since u and w are not neighboring).
  - In other words, nodes in $N^+(u)$ only have neighbors in $N^+(u)$. If any two nodes v,w in N(u) are not neighboring, node u itself would be in the dominating set. In other words, our graph is the complete graph (clique) $N^+(u)$. We precluded this in the assumptions, therefore we have a contradiction.

Proof of b) connected, c) shortest path in CDS

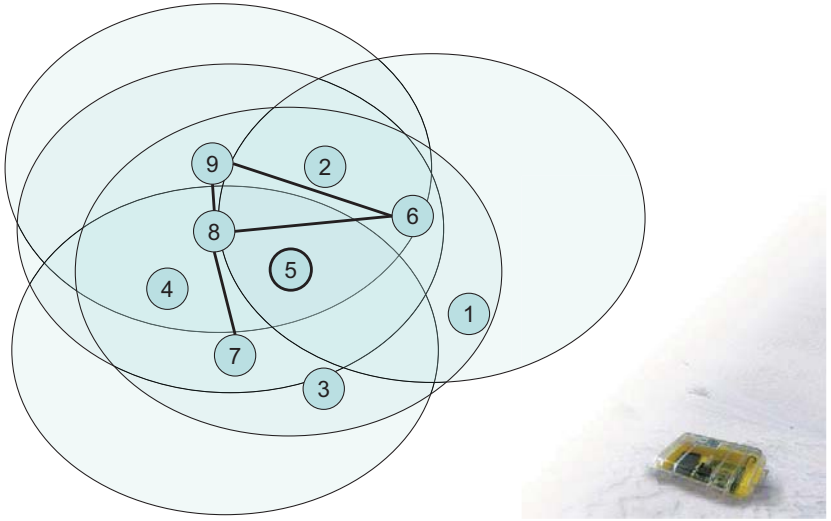
- Proof: Let p be any shortest path between the two nodes u and v, with $u,v \in$ CDS.
- Assume for the sake of contradiction that there is a node w on this shortest path that is not in the connected dominating set.



- Then the two neighbors of w must be connected, which gives us a shorter path. This is a contradiction.

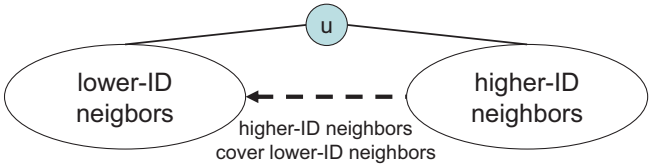
Improved Marking Algorithm

- If neighbors with larger ID are connected and cover all other neighbors, then don't join CDS, else join CDS



Correctness of Improved Marking Algorithm

- Theorem: Algorithm computes a CDS S
- Proof (by induction of node IDs):
  - assume that initially all nodes are in S
  - look at nodes u in increasing ID order and remove from S if higher-ID neighbors of u are connected
  - S remains a DS at all times: (assume that u is removed from S)



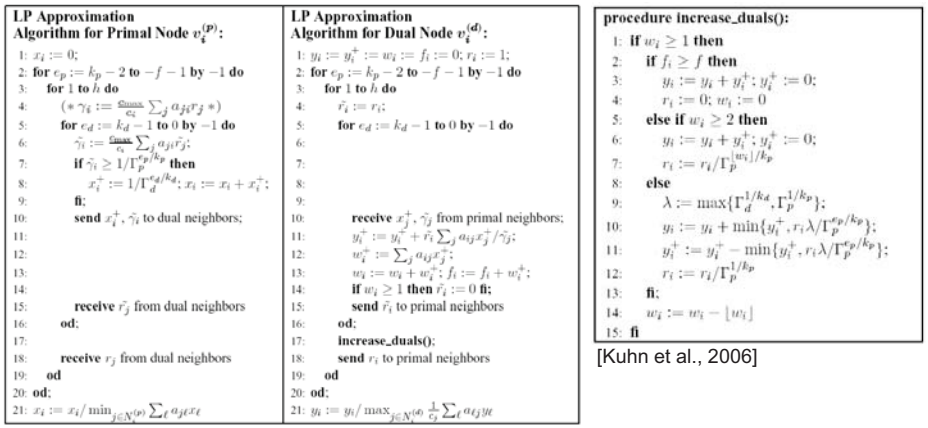
  - S remains connected:
    replace connection v-u-v' by v-n_1,…,n_k-v' (n_i: higher-ID neighbors of u)

Quality of the (Improved) Marking Algorithm

- Given an Euclidean chain of n homogeneous nodes
- The transmission range of each node is such that it is connected to the k left and right neighbors, the IDs of the nodes are ascending.

- An optimal algorithm (and also the tree growing algorithm) puts every k^{th} node into the CDS. Thus $|CDS_{OPT}| \approx n/k$; with $k = n/c$ for some positive constant c we have $|CDS_{OPT}| = O(1)$.
- The marking algorithm (also the improved version) does mark all the nodes (except the k leftmost and/or rightmost ones). Thus $|CDS_{Marking}| = n - k$; with $k = n/c$ we have $|CDS_{Marking}| = \Omega(n)$.
- This is as bad as not doing anything!
  - Is there at all a fast distributed way to compute a dominating set?

This problem is tough…



LP Approximation Algorithm for Primal Node $v_i^{(p)}$:

$x_i := 0$;
for $e_p := k_p - 2$ **to** $-f - 1$ **by** -1 **do**
  for l **to** h **do**
    $(* \gamma_i := \frac{c_{max}}{c_i} \sum_j a_{ji} r_j *)$
    for $e_d := k_d - 1$ **to** 0 **by** -1 **do**
      $\tilde{\gamma}_i := \frac{c_{max}}{c_i} \sum_j a_{ji} \tilde{r}_j$;
      if $\tilde{\gamma}_i \geq 1/\Gamma_p^{e_p/k_p}$ **then**
        $x_i^+ := 1/\Gamma_d^{e_d/k_d}$; $x_i := x_i + x_i^+$;
      fi;
      send $x_i^+, \tilde{\gamma}_i$ to dual neighbors;
      receive $\tilde{r}_j$ from dual neighbors
    od;
    receive r_j from dual neighbors
  od
od;
$x_i := x_i / \min_{j \in N_1^{(p)}} \sum_\ell a_{j\ell} x_\ell$

LP Approximation Algorithm for Dual Node $v_i^{(d)}$:

$y_i := y_i^+ := w_i := f_i := 0$; $r_i := 1$;
for $e_p := k_p - 2$ **to** $-f - 1$ **by** -1 **do**
  for l **to** h **do**
    $\tilde{r}_i := r_i$;
    for $e_d := k_d - 1$ **to** 0 **by** -1 **do**
      receive $x_j^+, \tilde{\gamma}_j$ from primal neighbors;
      $y_i^+ := y_i^+ + \tilde{r}_i \sum_j a_{ij} x_j^+ / \tilde{\gamma}_j$;
      $w_i^+ := \sum_j a_{ij} x_j^+$;
      $w_i := w_i + w_i^+$; $f_i := f_i + w_i^+$;
      if $w_i \geq 1$ **then** $\tilde{r}_i := 0$ **fi**;
      send $\tilde{r}_i$ to primal neighbors
    od;
    increase_duals();
    send r_i to primal neighbors
  od
od;
$y_i := y_i / \max_{j \in N_1^{(d)}} \frac{1}{c_j} \sum_\ell a_{ij} y_\ell$

procedure increase_duals():

if $w_i \geq 1$ **then**
  if $f_i \geq f$ **then**
    $y_i := y_i + y_i^+$; $y_i^+ := 0$;
    $r_i := 0$; $w_i := 0$
  else if $w_i \geq 2$ **then**
    $y_i := y_i + y_i^+$; $y_i^+ := 0$;
    $r_i := r_i / \Gamma_p^{\lfloor w_i \rfloor / k_p}$
  else
    $\lambda := \max\{\Gamma_d^{1/k_d}, \Gamma_p^{1/k_p}\}$;
    $y_i := y_i + \min\{y_i^+, r_i \lambda / \Gamma_p^{e_p/k_p}\}$;
    $y_i^+ := y_i^+ - \min\{y_i^+, r_i \lambda / \Gamma_p^{e_p/k_p}\}$;
    $r_i := r_i / \Gamma_p^{1/k_p}$
  fi;
  $w_i := w_i - \lfloor w_i \rfloor$
fi

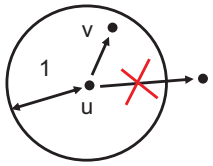
[Kuhn et al., 2006]

- … however, there are some complicated algorithms that achieve non-trivial results, e.g. in *k* rounds of communications

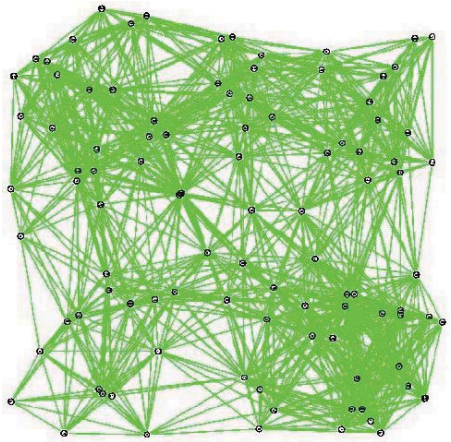
$$E[|DS|] = O\left((\Delta+1)^{c/\sqrt{k}} \log \Delta \cdot |DS_{OPT}|\right)$$

Better and faster algorithm

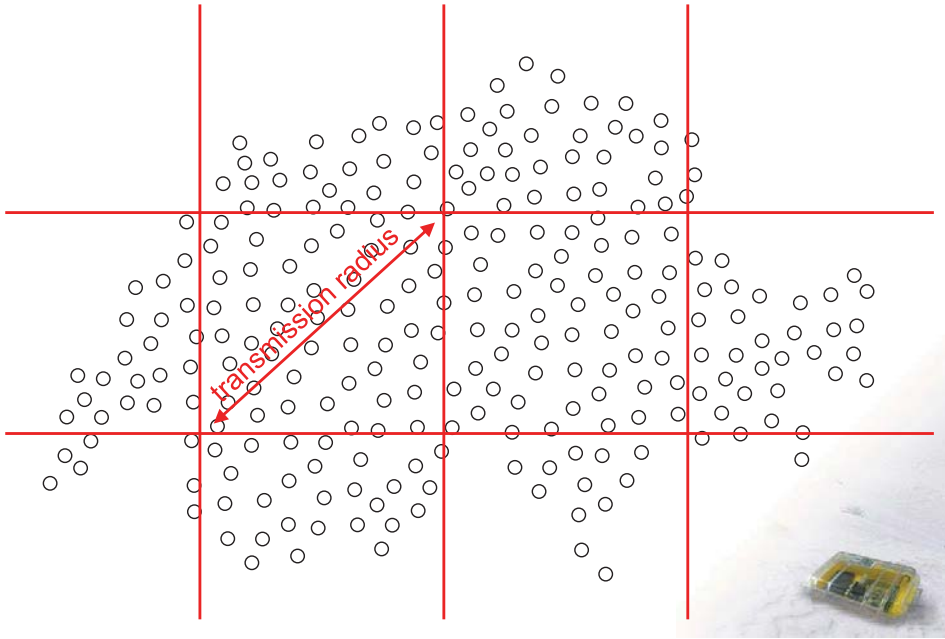
- Assume that graph is a unit disk graph (UDG)



- Assume that nodes know their positions (GPS)



Then…



Grid Algorithm

1. Beacon your position
2. If, in your virtual grid cell, you are the node closest to the center of the cell, then join the DS, else do not join.
3. That's it.

- 1 transmission per node, O(1) approximation.
- If you have mobility/dynamics, then simply "loop" through algorithm, as fast as your application/mobility wants you to.

Comparison

Complicated algorithm	Grid algorithm
• Algorithm computes DS	• Algorithm computes DS
• $k^2+O(1)$ transmissions/node • $O(\Delta^{O(1)/k} \log \Delta)$ approximation	• 1 transmission/node • O(1) approximation
• General graph • No position information	• Unit disk graph (UDG) • Position information (GPS)

The model determines the distributed complexity of clustering

Let's talk about models…

- General Graph
- Captures obstacles
- Captures directional radios
- Often too pessimistic

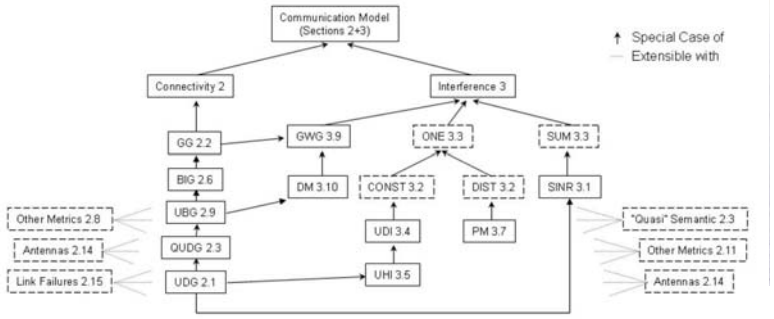
- UDG & GPS
- UDG is not realistic
- GPS not always available
  - Indoors
- 2D → 3D?
- Often too optimistic

too pessimistic ⟷ too optimistic

Let's look at models in between these extremes!

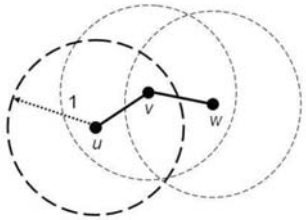
Why are models needed?

- Formal models help us understanding a problem
- Formal proofs of correctness and efficiency
- Common basis to compare results
- Unfortunately, for ad hoc and sensor networks, a myriad of models exist, most of them make sense in some way or another. On the next few slides we look at a few selected models

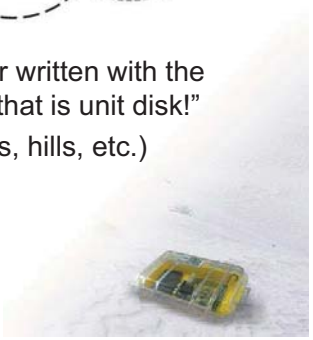


Unit Disk Graph (UDG)

- Classic computational geometry model, special case of disk graphs
- All nodes are points in the plane, two nodes are connected iff (if and only if) their distance is at most 1, that is $\{u,v\} \in E \Leftrightarrow |u,v| \leq 1$

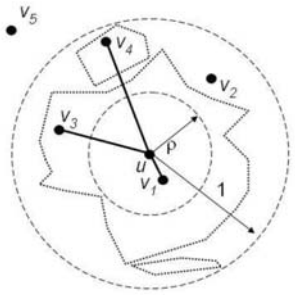


+ Very simple, allows for strong analysis
– Not realistic: "If you gave me $100 for each paper written with the unit disk assumption, I still could not buy a radio that is unit disk!"
– Particularly bad in obstructed environments (walls, hills, etc.)
- Natural extension: 3D UDG

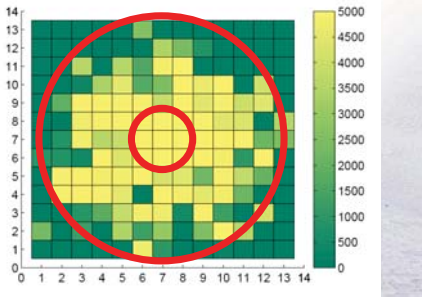


Quasi Unit Disk Graph (QUDG)

- Two radii, 1 and ρ, with $\rho \leq 1$
  - $|u,v| \leq \rho \Leftrightarrow \{u,v\} \in E$
  - $1 < |u,v| \Leftrightarrow \{u,v\} \notin E$
  - $\rho < |u,v| \leq 1 \Leftrightarrow$ it depends!
    - … on an adversary
    - … on probabilistic model
    - …

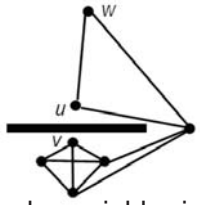


+ Simple, analyzable
+ More realistic than UDG
– Still bad in obstructed environments (walls, hills, etc.)
- Natural extension: 3D QUDG



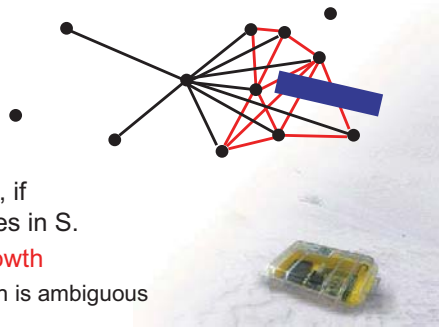
Bounded Independence Graph (BIG)

- How realistic is a QUDG?
  - u and v can be close but not adjacent
  - model requires very small ρ in obstructed environments (walls)



- However: in practice, neighbors are often also neighboring

- Solution: BIG Model
  - Bounded independence graph
  - Size of any independent set grows polynomially with hop distance *r*
  - e.g., $f(r) = O(r^2)$ or $O(r^3)$
  - A set S of nodes is an independent set, if there is no edge between any two nodes in S.
  - BIG model also known as bounded-growth
    - Unfortunately, the term bounded-growth is ambiguous



Unit Ball Graph (UBG)

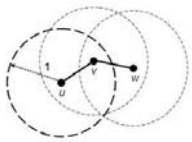
- ∃ metric (V,d) with constant doubling dimension.

- Metric: Each edge has a distance d, with
  1. $d(u,v) \geq 0$ (non-negativity)
  2. $d(u,v) = 0$ iff $u = v$ (identity of indiscernibles)
  3. $d(u,v) = d(v,u)$ (symmetry)
  4. $d(u,w) \leq d(u,v) + d(v,w)$ (triangle inequality)

- Doubling dimension: log(#balls of radius r/2 to cover ball of radius r)
  - Constant: you only need a constant number of balls of half the radius

- Connectivity graph is same as UDG:
  such that: $d(u,v) \leq 1 : (u,v) \in E$
  $d(u,v) > 1 : (u,v) \notin E$

Connectivity Models: Overview



General Graph

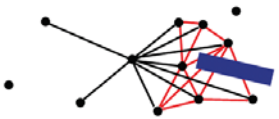
UDG

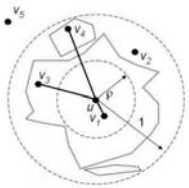
too pessimistic — too optimistic

Bounded Independence

Unit Ball Graph

Quasi UDG





Models are related

- BIG is special case of general graph, BIG ⊆ GG

- UBG ⊆ BIG because the size of the independent sets of any UBG is polynomially bounded

- QUDG(constant ρ) ⊆ UBG

- QUDG(ρ=1) = UDG

GG

BIG

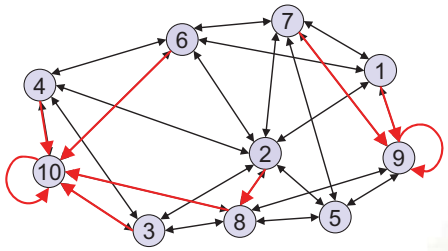
UBG

QUDG

UDG

The "Largest-ID" Algorithm

- All nodes have unique IDs, chosen at random.
- Algorithm for each node:
  1. Send ID to all neighbors
  2. Tell node with largest ID in neighborhood that it has to join the DS
- Algorithm computes a DS in 2 rounds (very local!)

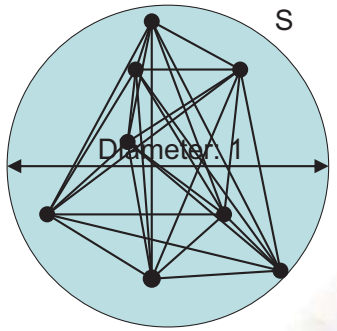


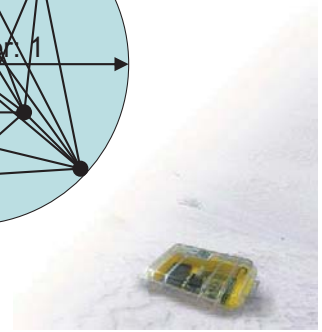
"Largest ID" Algorithm, Analysis 1

- To simplify analysis: assume graph is UDG
  (same analysis works for UBG based on doubling metric)
- We look at a disk S of diameter 1:

Nodes inside S have distance at most 1.
$\rightarrow$ they form a clique

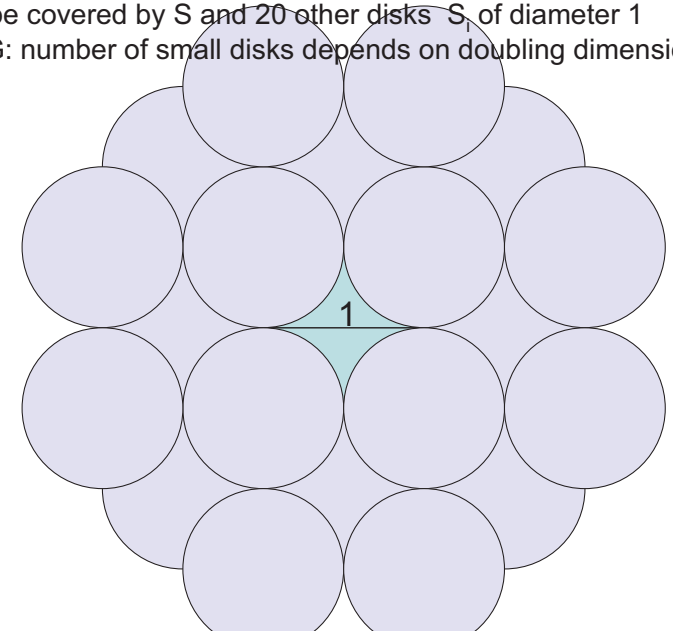
How many nodes in S are selected for the DS?





"Largest ID" Algorithm, Analysis 2

- Nodes which select nodes in S are in disk of radius 3/2 which can be covered by S and 20 other disks S_i of diameter 1
  (UBG: number of small disks depends on doubling dimension)



"Largest ID" Algorithm: Analysis 3

- How many nodes in S are chosen by nodes in a disk S_i?
- A node $u \in S$ is only chosen by a node in S_i if $\mathrm{ID}(u) > \max_{v \in S_i}\{\mathrm{ID}(v)\}$ (all nodes in S_i see each other).
- The probability for this is: $\frac{1}{1+|S_i|}$
- Therefore, the expected number of nodes in S chosen by nodes in S_i is at most:

$$\min\left\{|S_i|, \frac{|S|}{1+|S_i|}\right\}$$

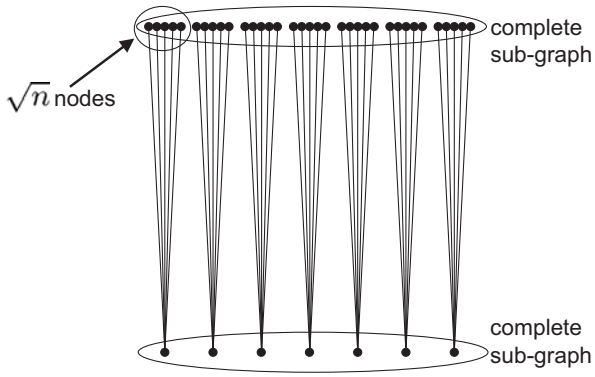
Because at most $|S_i|$ nodes in S_i can choose nodes in S and because of linearity of expectation.

"Largest ID" Algorithm, Analysis 4

- From $|S| \leq n$ and $|S_i| \leq n$, it follows that $\min\left\{|S_i|, \frac{|S|}{1+|S_i|}\right\} \leq \sqrt{n}$
- Hence, in expectation the DS contains at most $20\sqrt{n}$ nodes per disk with diameter 1.
- An optimal algorithm needs to choose at least 1 node in the disk with radius 1 around any node.
- This disk can be covered by a constant (9) number of disks of diameter 1.
- The algorithm chooses at most $O(\sqrt{n})$ times more disks than an optimal one

"Largest ID" Algorithm, Remarks

- For typical settings, the "Largest ID" algorithm produces very good dominating sets (also for non-UDGs)
- There are UDGs where the "Largest ID" algorithm computes an $\Theta(\sqrt{n})$-approximation (analysis is tight).



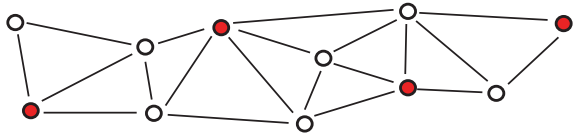
Optimal DS: size 2

"Largest ID" algorithm:

- bottom nodes choose top nodes with probability≈1/2
- 1 node every 2nd group $\Theta(\sqrt{n})$ nodes

Maximal Independent Set (MIS)

- A Maximal Independent Set (MIS) is a non-extendable set of pair-wise non-adjacent nodes:

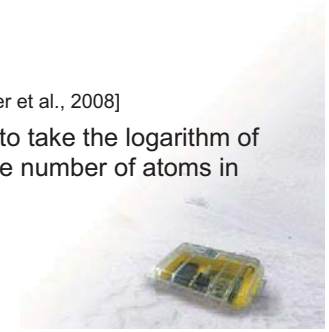


- An MIS is also a dominating set:
  - assume that there is a node v which is not dominated
  - $v \notin MIS$, $(u,v) \in E \rightarrow u \notin MIS$
  - add v to MIS
- In contrast: A Maximum Independent Set (MaxIS) is an independent set of maximum cardinality.

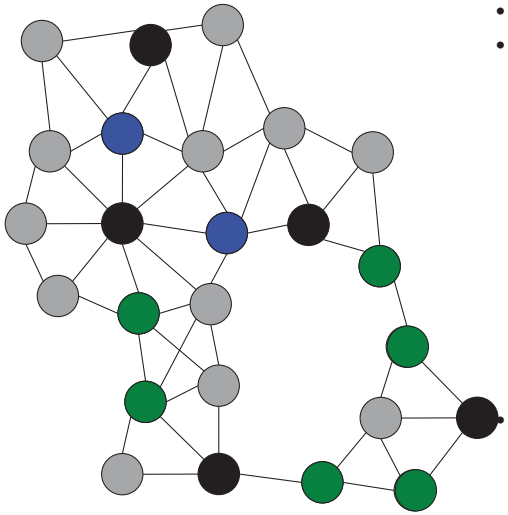
Computing a MIS

- Lemma: On BIG: $\mathbf{|MIS| \leq O(1) \cdot |DS_{OPT}|}$
- Proof:
  1. Assign every MIS node to an adjacent node of DS_{OPT}
  2. $u \in DS_{OPT}$ has at most f(1) neighbors $v \in MIS$
  3. At most f(1) MIS nodes assigned to every node of DS_{OPT}

  $\rightarrow |MIS| \leq f(1) \cdot |DS_{OPT}|$
- Time to compute MIS on BIGs: $O(\log^* n)$ [Schneider et al., 2008]
  - The function „log-star" says how often you need to take the logarithm of a value to end up with 1 or less. Even if *n* was the number of atoms in the universe, we have $\log^* n = 5$.



MIS (DS) → CDS



- MIS gives a dominating set.
- But it is not connected.
  - Connect any two MIS nodes which can be connected by one additional node.
  - Connect unconnected MIS nodes which can be connected by two additional nodes.
  - This gives a CDS!
  - #2-hop connectors $\leq$ f(2)·|MIS|
  - #3-hop connectors $\leq$ 2f(3)·|MIS|
  - |CDS| = O(|MIS|)

Similarly, one can compute other structures, e.g. coloring, very fast!

Open problem

- This chapter got a lot of attention from the research community in the last few years, and it made remarkable progress. Many problems open just a few years ago are solved now.
- However, some problems are still open. The classic open problem in this area is MIS for general graphs. A randomized algorithm [Luby 1985, and others] constructs a MIS in time O(log *n*). It is unknown whether this can be improved, or matched by a deterministic algorithm.
- Another nice open question is what can be achieved in constant time? For instance, even though we know that an MIS (or CDS or Δ-coloring) can be computed in O(log**n*) time on a UDG [Schneider et al., 2008], it is unclear what can be done in constant time!