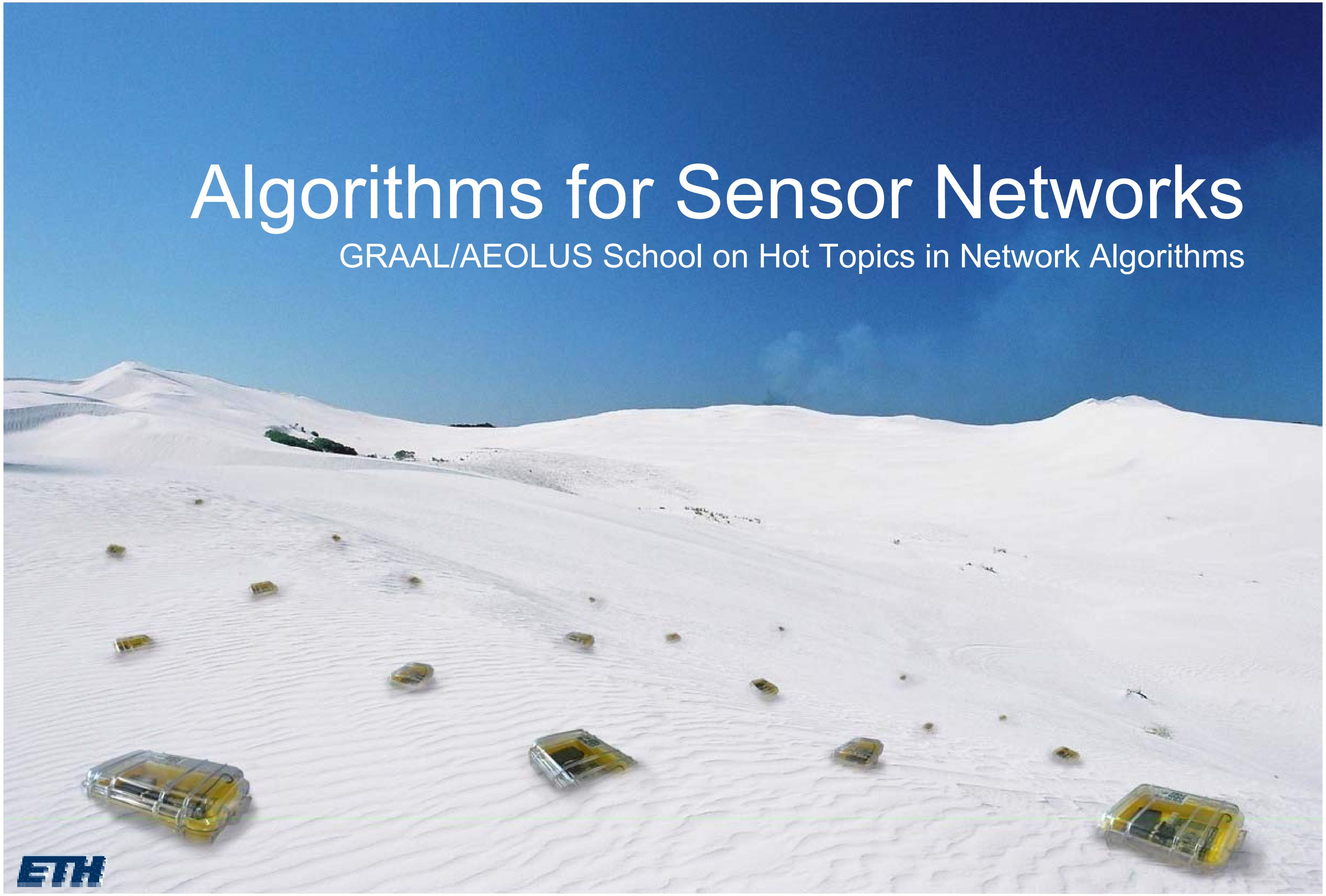


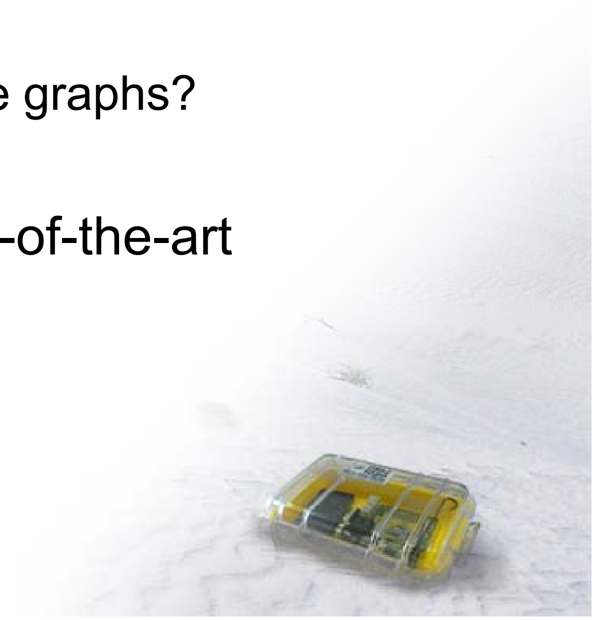
Algorithms for Sensor Networks

GRAAL/AEOLUS School on Hot Topics in Network Algorithms

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Sensor Networks = Distributed Algorithms?

- Reloaded
  - Distributed (message passing) algorithms
  - Message complexity → Support for energy efficiency
  - Time complexity → Support for dynamics
- Revolutions
  - Wireless → Interference issues → Not standard message passing, but new types of distributed algorithms
  - Wireless → New types of connectivity/interference graphs?
- Finally an application that can't live without state-of-the-art distributed graph algorithms?!

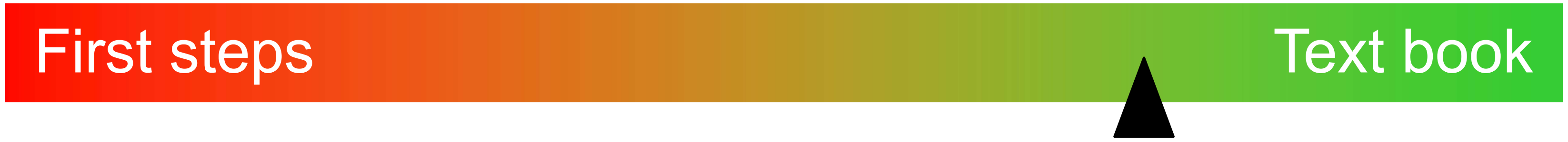


Clustering etc.

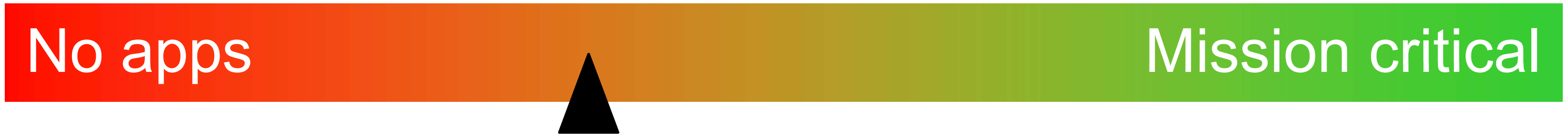
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Rating

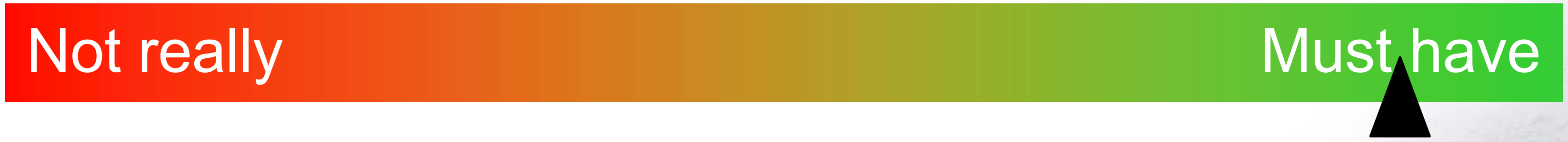
- Area maturity



- Practical importance



- Theoretical importance



Overview

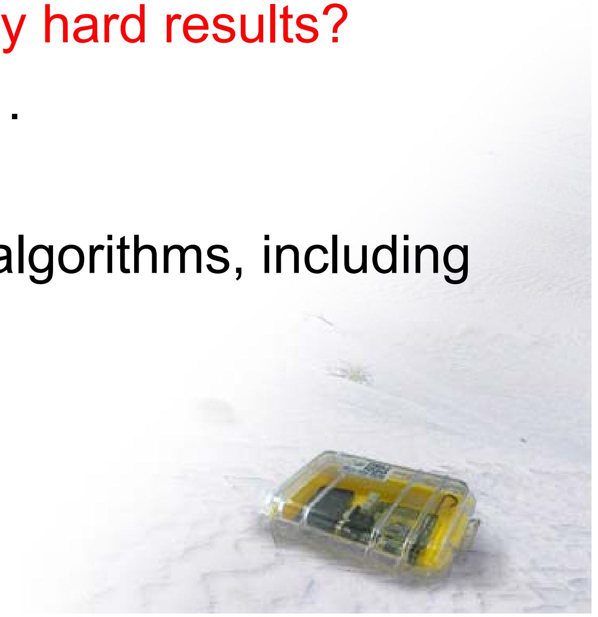
- Motivation
- Dominating Set
- Some algorithms
- Model discussion
- Some more algorithms
- Models reloaded

Motivation

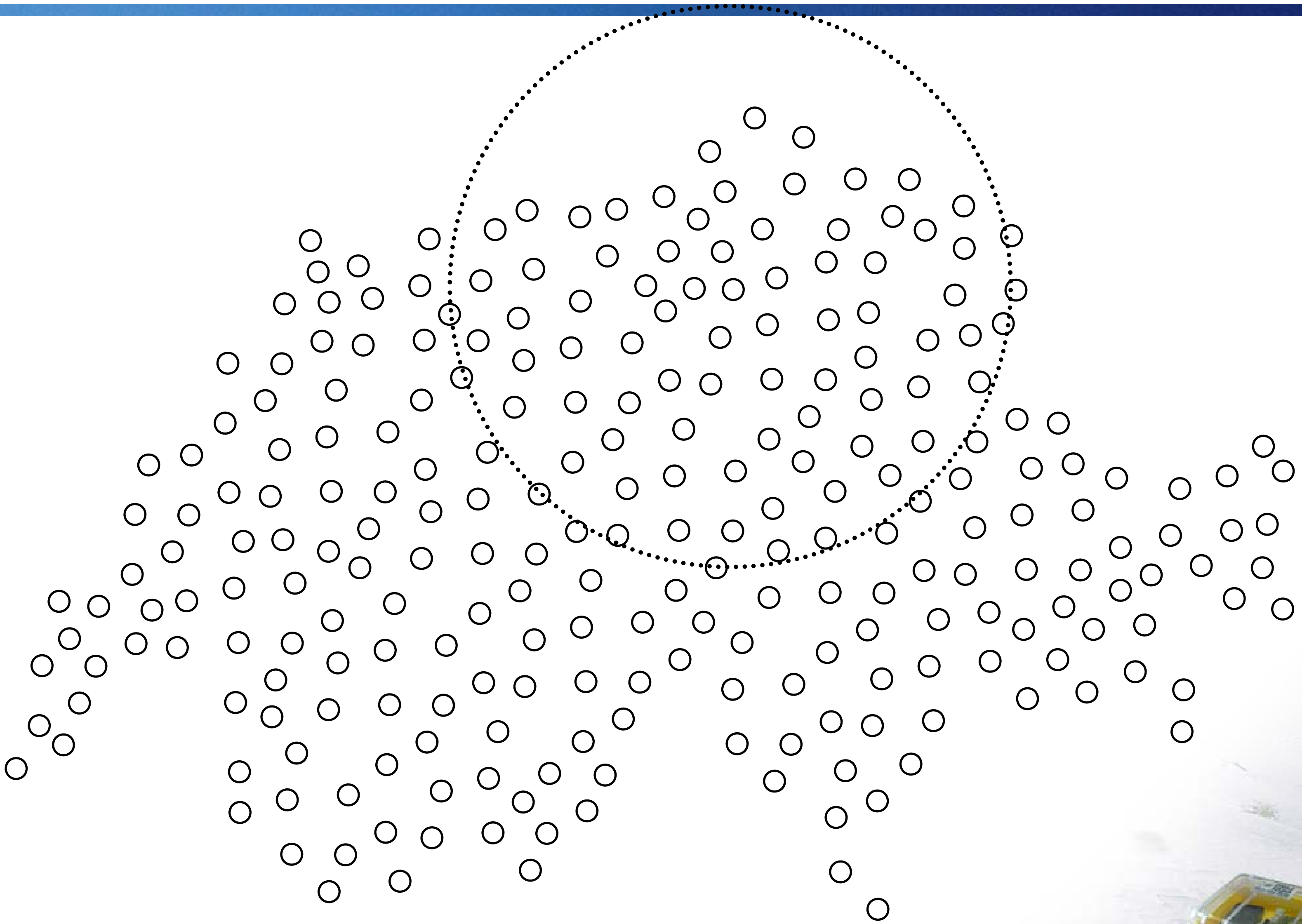
- In theory clustering is the solution to almost any problem in ad hoc and sensor networks. It improves almost any algorithm, e.g. in data gathering it selects cluster heads which do the work while other nodes can save energy by sleeping. Here, however, we motivate clustering with routing:

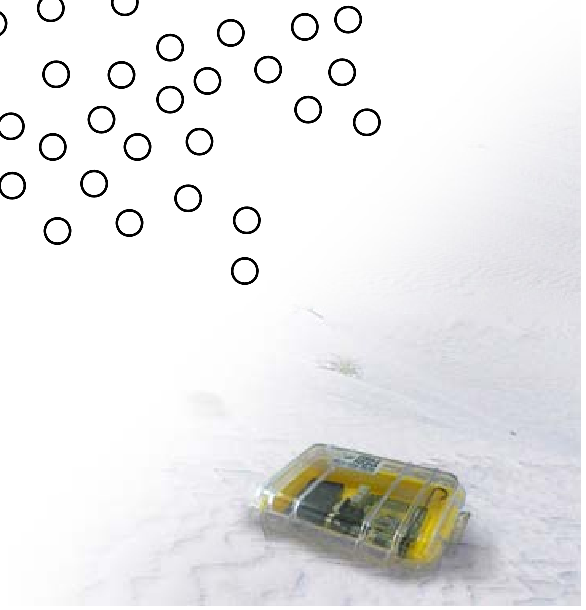
- There are thousands of routing algorithms…
- Q: How good are these routing algorithms?!? Any hard results?
- A: Almost none! Method-of-choice is simulation…

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

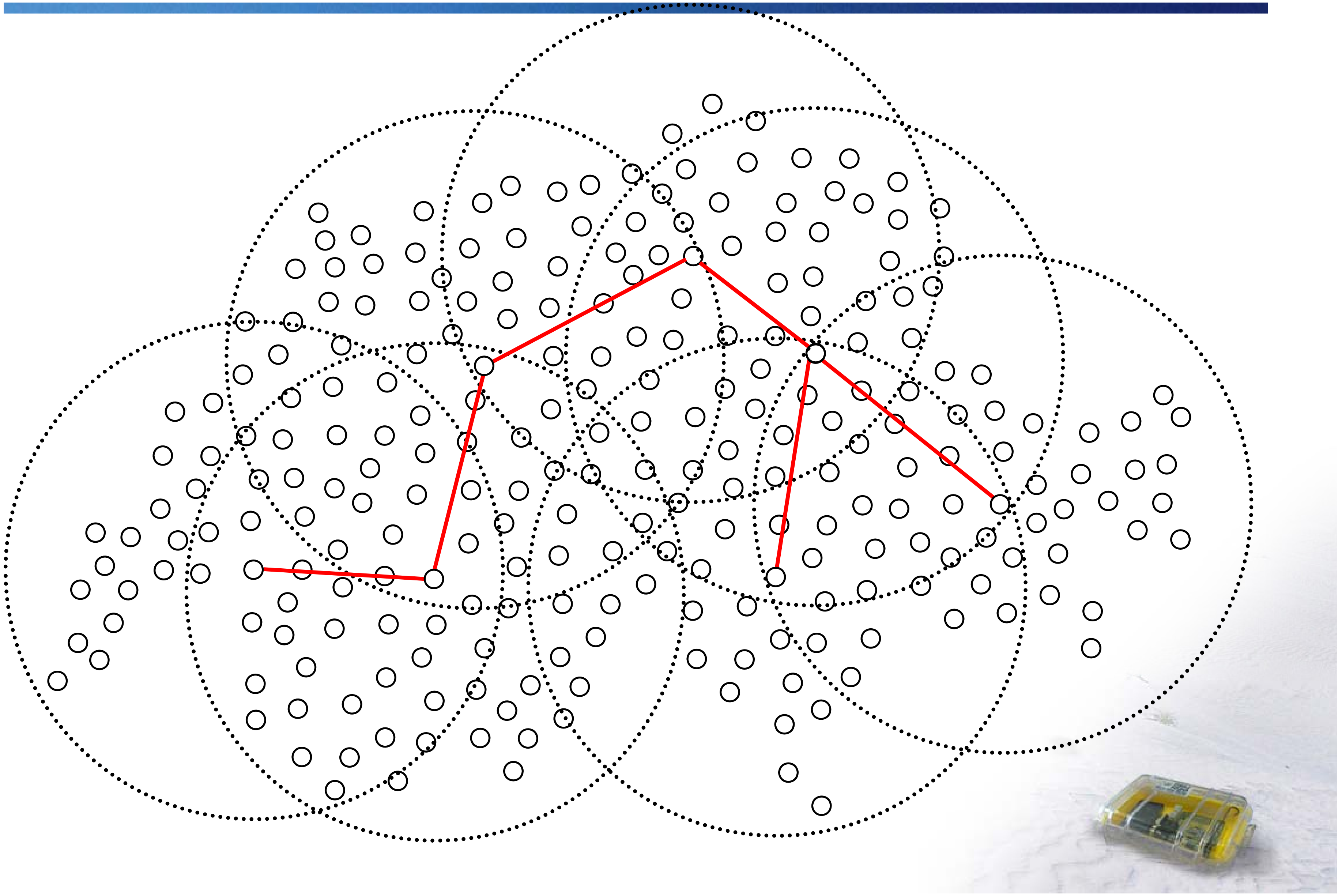


Finding a Destination by Flooding





Finding a Destination *Efficiently*

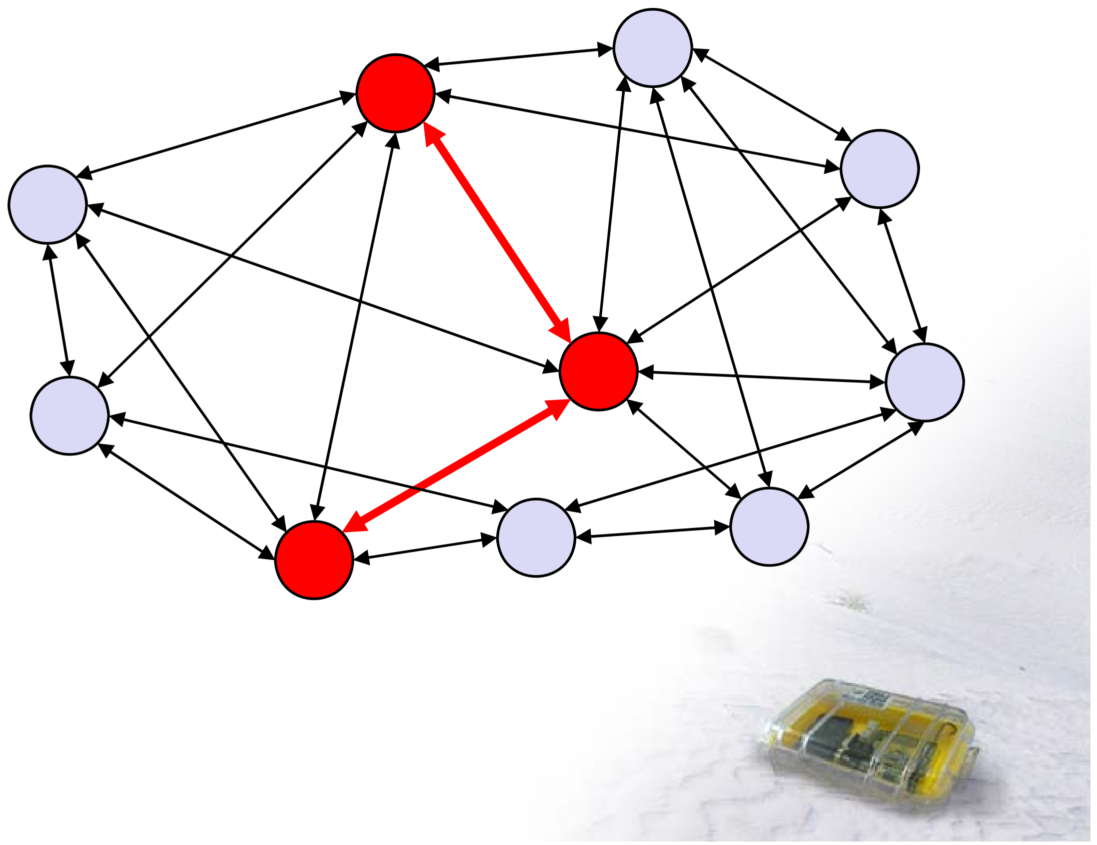


Backbone

- 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.



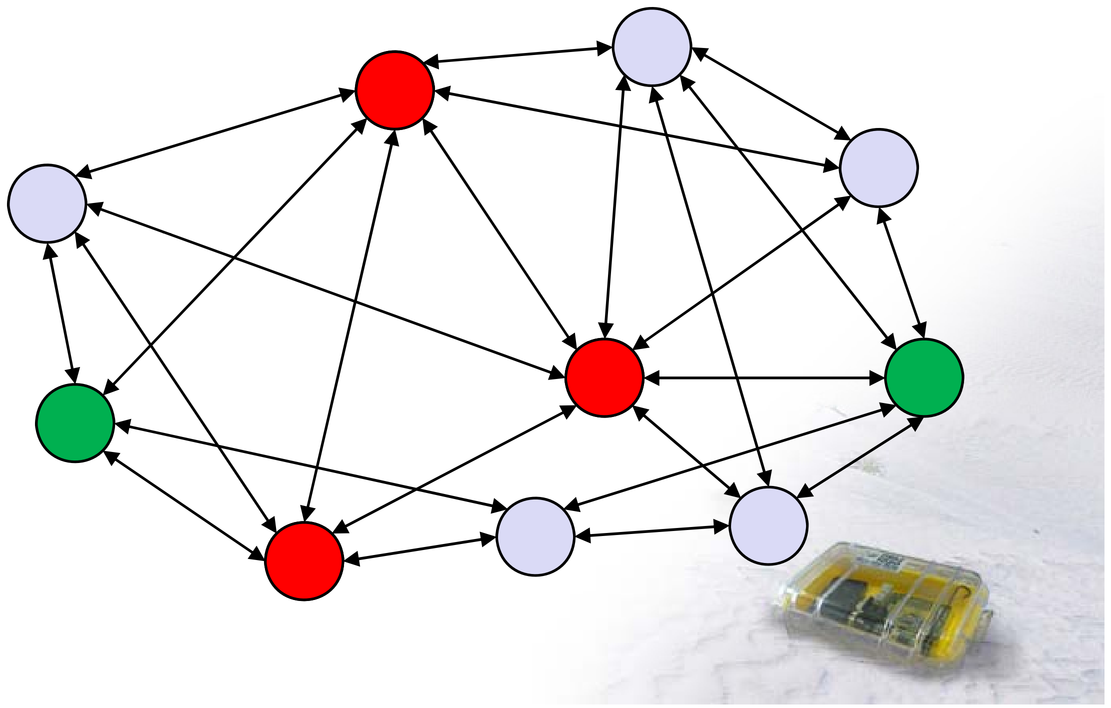
(Connected) Dominating Set

- 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.

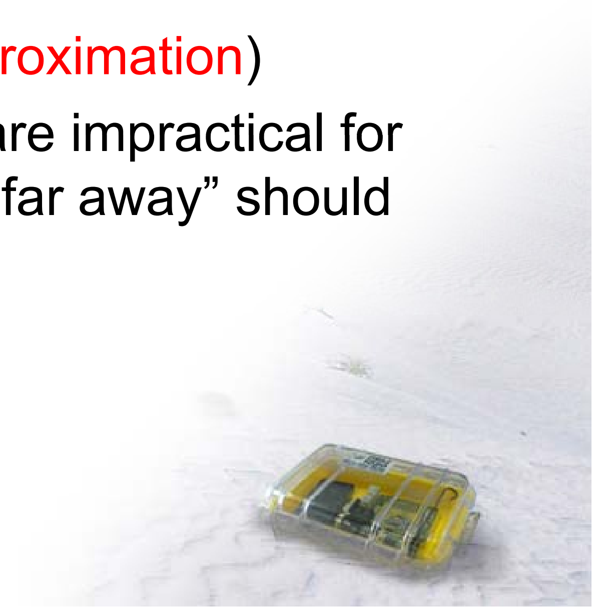
- A CDS is a good choice for a backbone.

- 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 mobile ad-hoc network) – topology of graph “far away” should not influence decision who belongs to (C)DS



Other useful structures

- Maximal Independent Set (MIS)
- Maximum Independent Set (MaxIS)
- (Δ+1) or O(Δ) Coloring
- (Connected) Domatic Partition

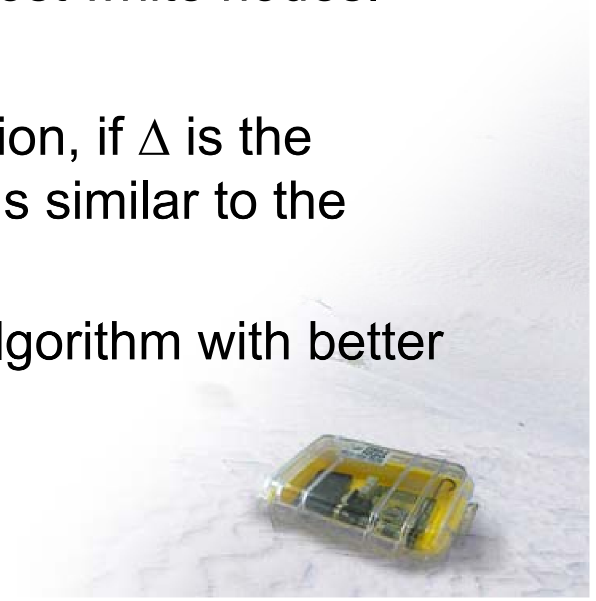
Greedy Algorithm for Dominating Sets

- 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.)
- One can also show that there is no polynomial algorithm with better performance unless $P \approx NP$.



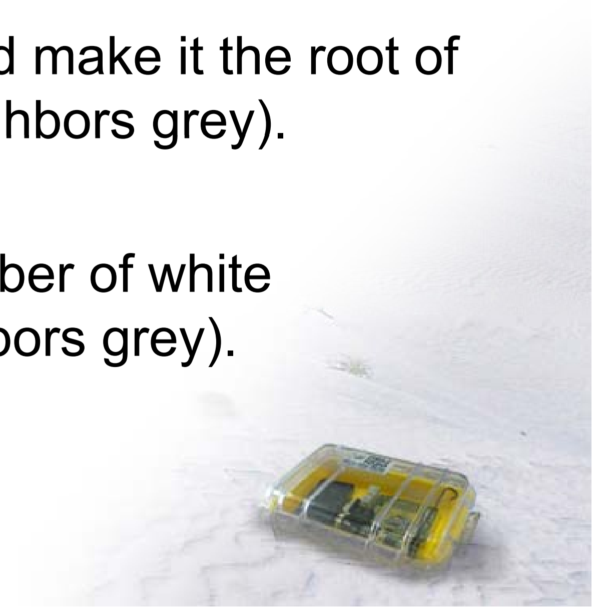
CDS: The “too simple tree growing” algorithm

- 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.

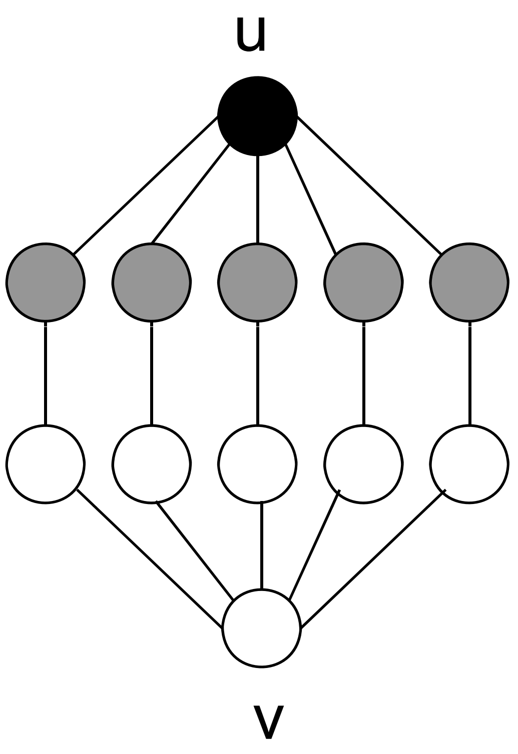
- Start: Choose a node with maximum degree, and make it the root of the CDS, that is, color it black (and its white neighbors grey).

- Step: Choose a grey node with a maximum number of white neighbors and color it black (and its white neighbors grey).

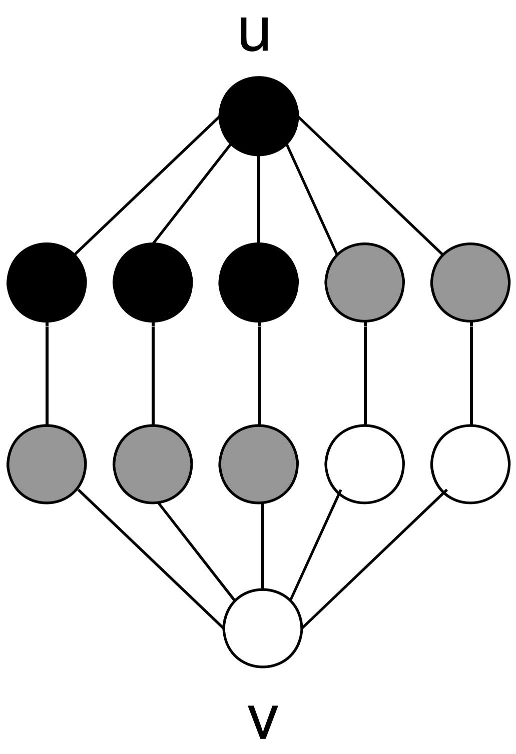


Example of the "too simple tree growing" algorithm

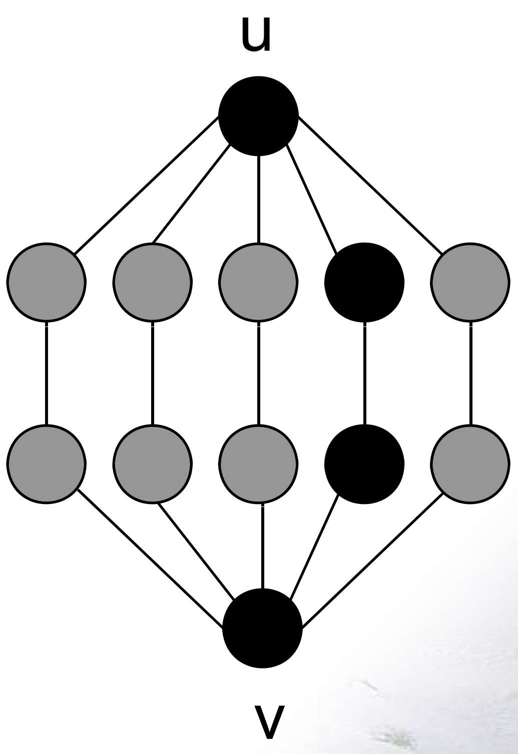
Graph with 2n+2 nodes; tree growing: |CDS|=n+2; Minimum |CDS|=4



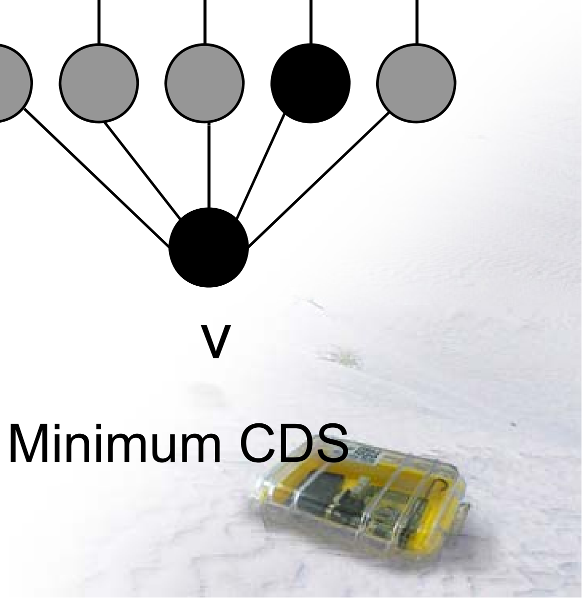
tree growing: start



…

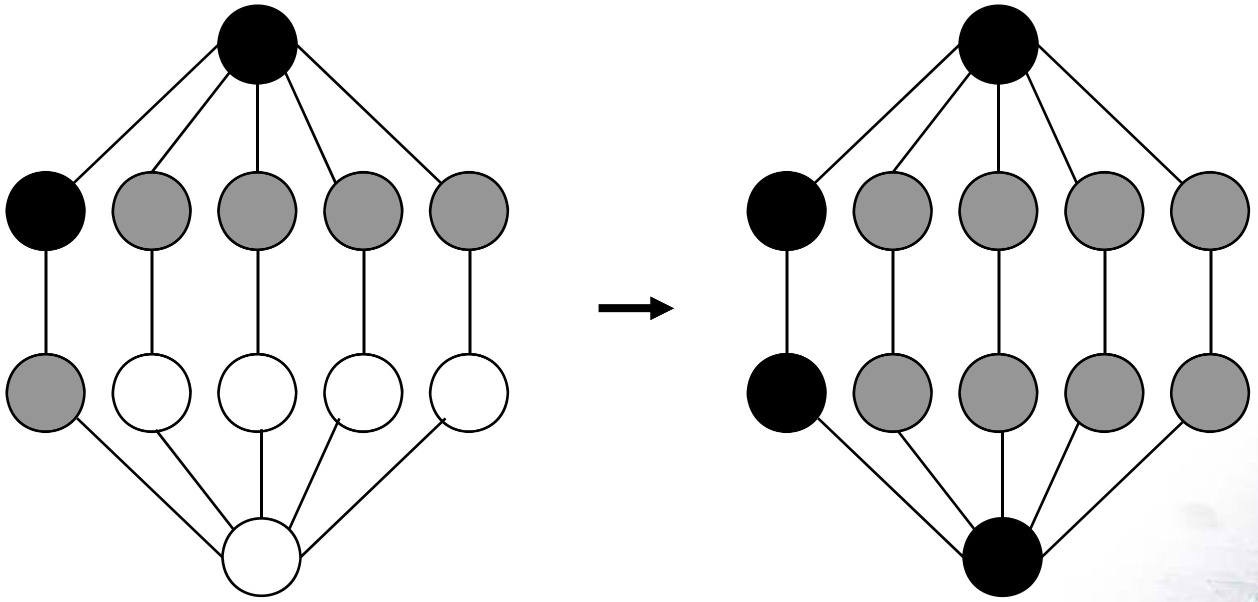


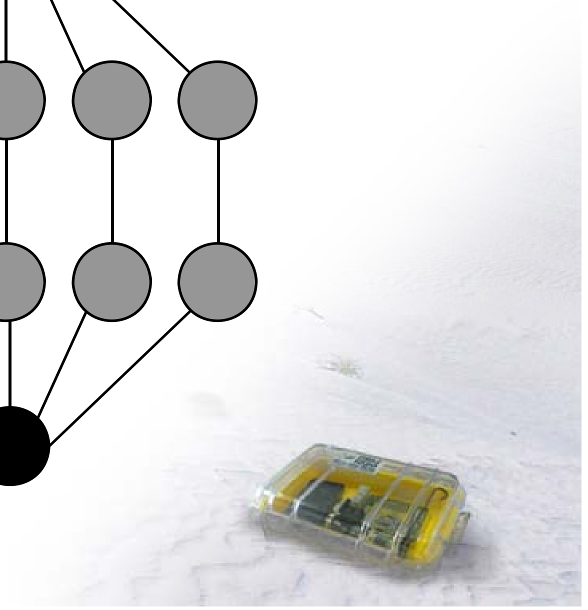
Minimum CDS



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).





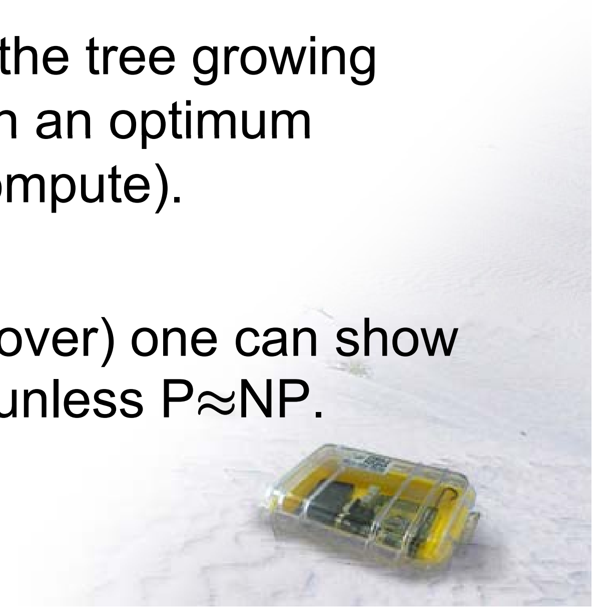
Analysis of the tree growing algorithm

- 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 $H(n) \approx \log(n)+0.7$

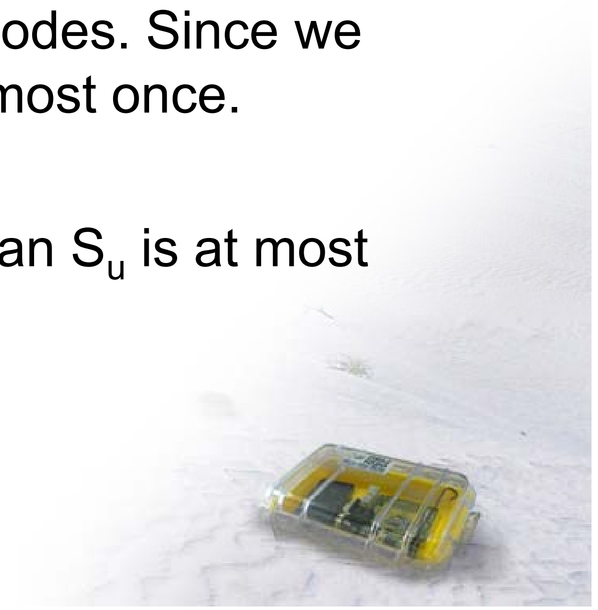
- In other words, the connected dominating set of the tree growing algorithm is at most a $O(\log(\Delta))$ factor worse than an optimum minimum dominating set (which is NP-hard to compute).

- With a lower bound argument (reduction to set cover) one can show that a better approximation factor is impossible, unless $P \approx NP$.



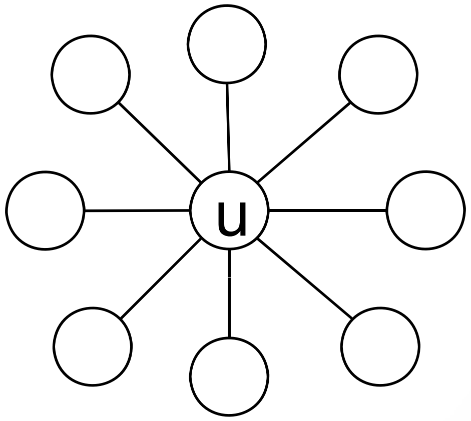
Proof Sketch

- 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, we put it in one of the sets.
- We 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.
- We show that the total charge on the vertices in an S_u is at most $2(1+H(\Delta))$, for any u.



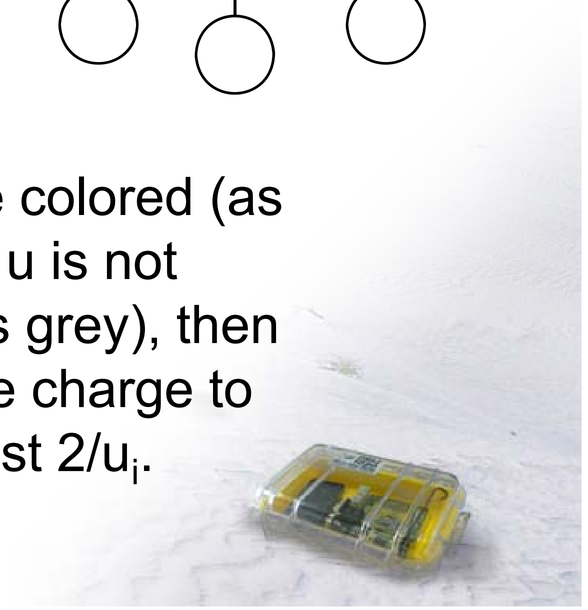
Charge on S_u

- Initially $|S_u| = u_0$.
- 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 (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 $2/u_i$.



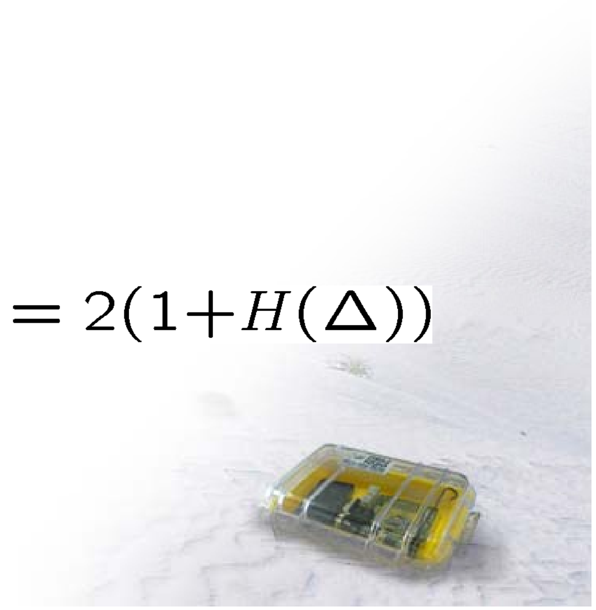
Adding up the charges in S$_u$

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

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

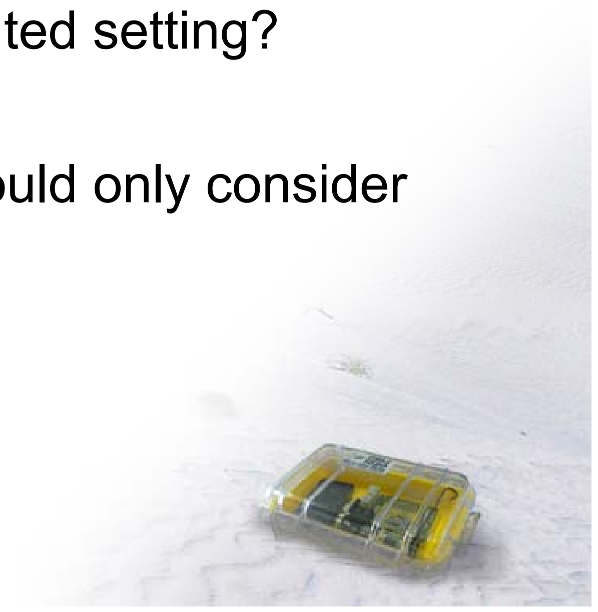
$$\le 2 + 2\sum_{i=1}^{k-1} \Big(H(u_i) - H(u_{i+1})\Big)$$

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



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.

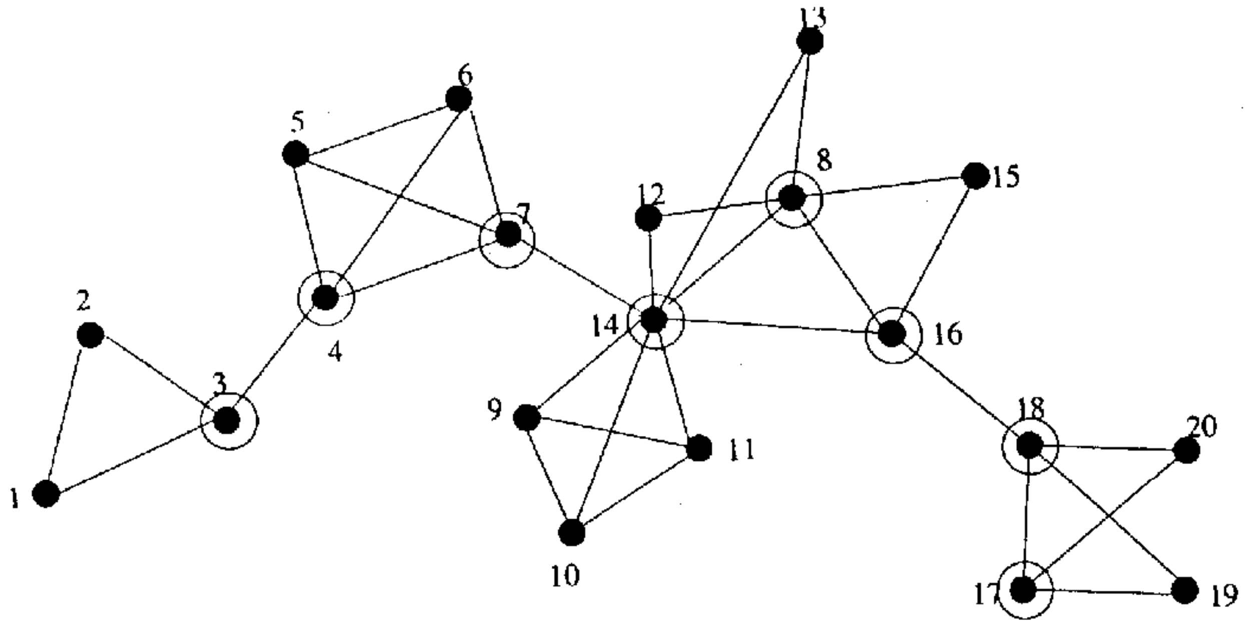
+ Completely local; only exchange N(u) with all neighbors

+ Each node sends only 1 message, and receives at most Δ

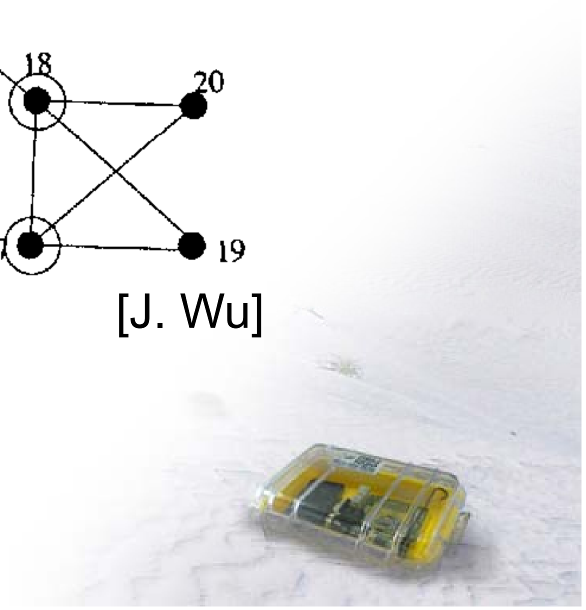
+ Messages have size $O(\Delta)$

- 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 complete.
- Note: If G was complete then constructing a CDS would not make sense. Note that in a complete graph, no node would be marked.
- We show:

  The set of marked nodes CDS is

  a) a dominating set

  b) connected

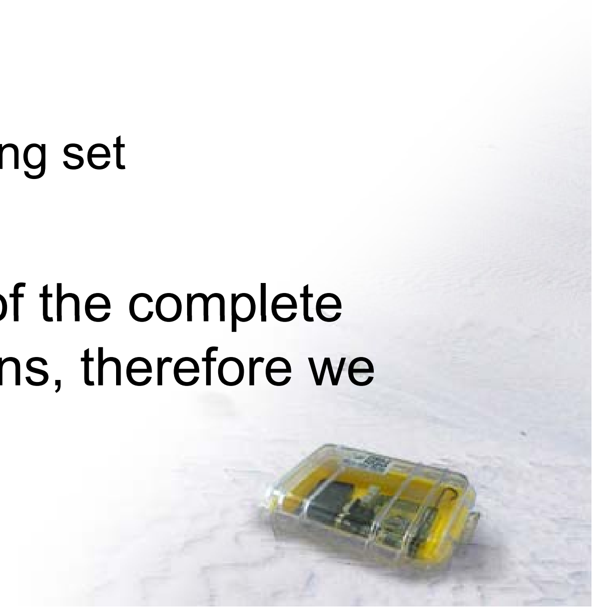
  c) a shortest path in G between two nodes of the CDS is in CDS

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. Since no neighbor of u is in the dominating set, the nodes $N^+(u) := u \cup N(u)$ form:

- a complete graph
  - if there are two nodes in N(u) that are not connected, u must be in the dominating set by definition
- no node $v \in N(u)$ has a neighbor outside N(u)
  - or, also by definition, the node v is in the dominating set

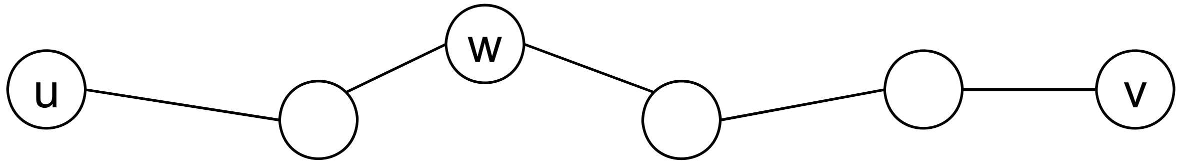
- Since the graph G is connected it only consists of the complete graph $N^+(u)$. We precluded this in the assumptions, therefore we have a contradiction



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.

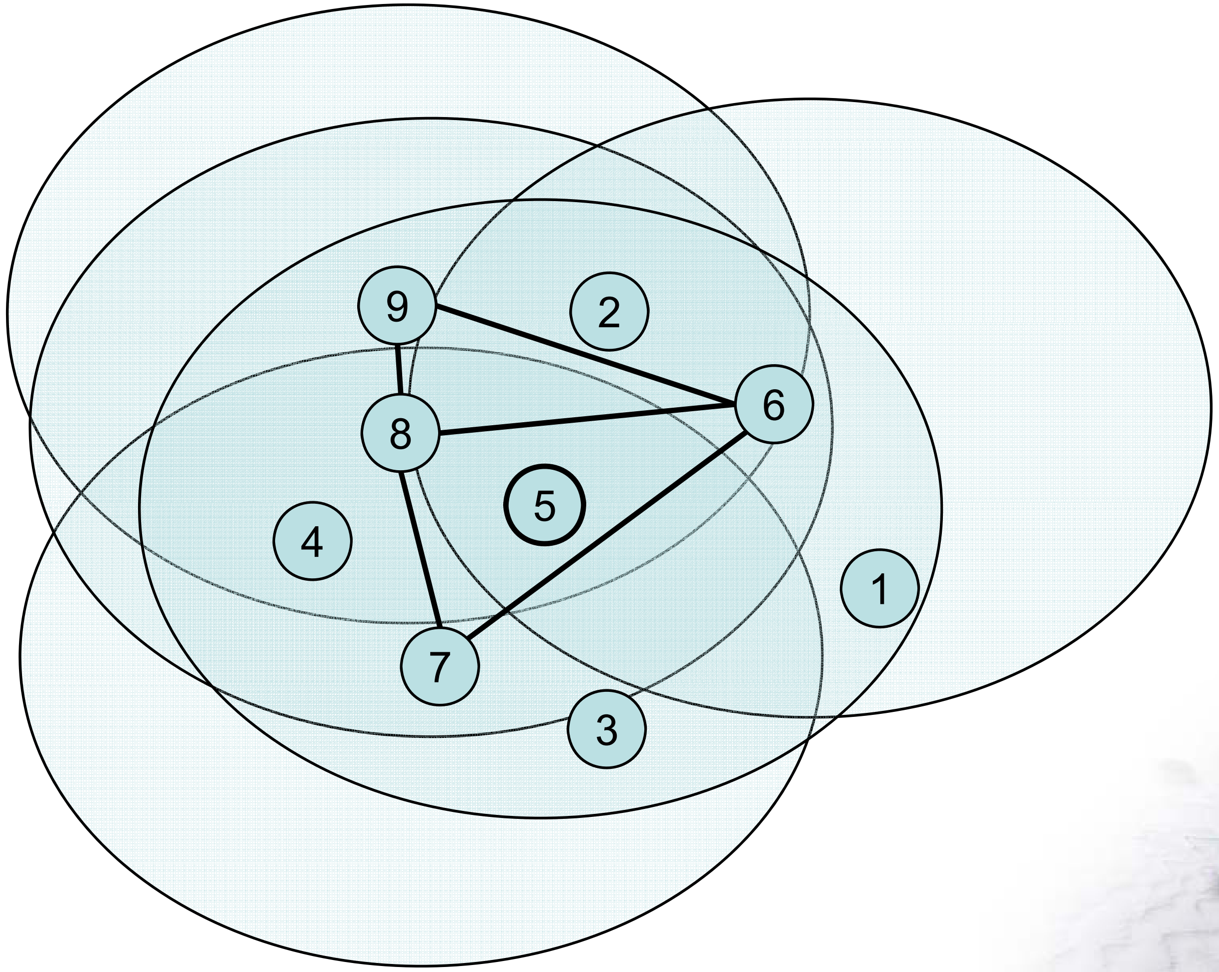
- Assume for the sake of contradiction that there is a node w on this shortest path that is not in the connected dominating set.

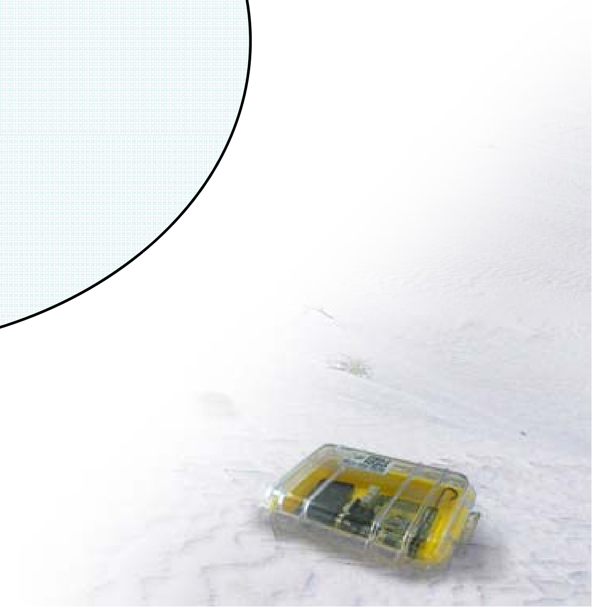


- Then the two neighbors of w must be connected, which gives us a shorter path. This is a contradiction.

Improved Marking Algorithm

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

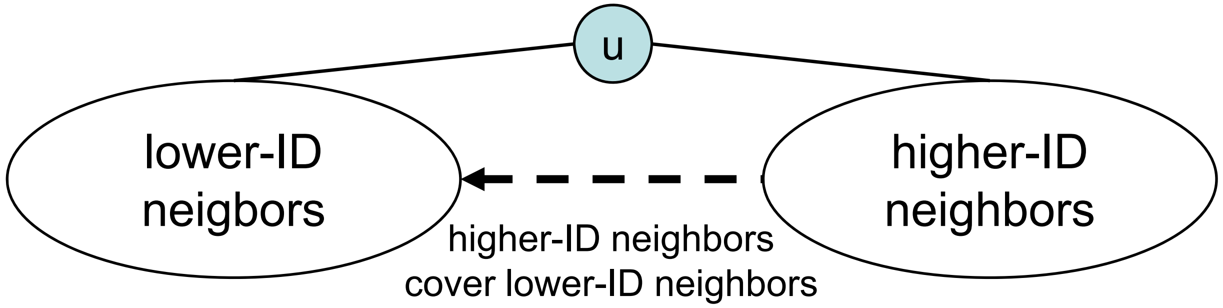




Correctness of Improved Marking Algorithm

- 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)

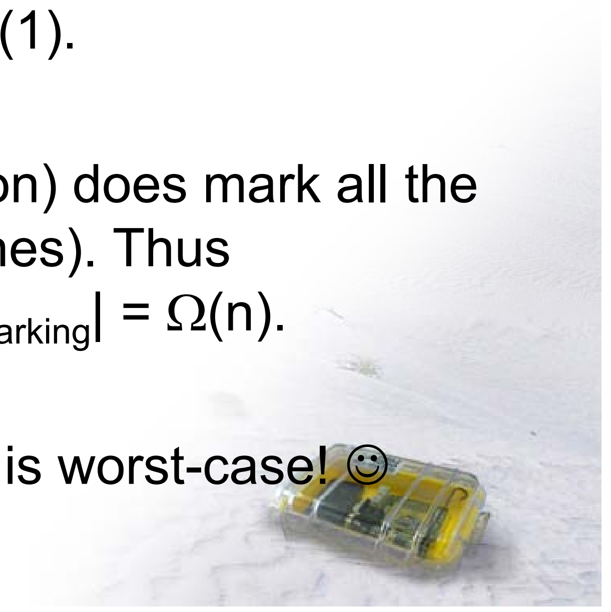
Quality of the (Improved) Marking Algorithm

- 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 id's of the nodes are ascending.

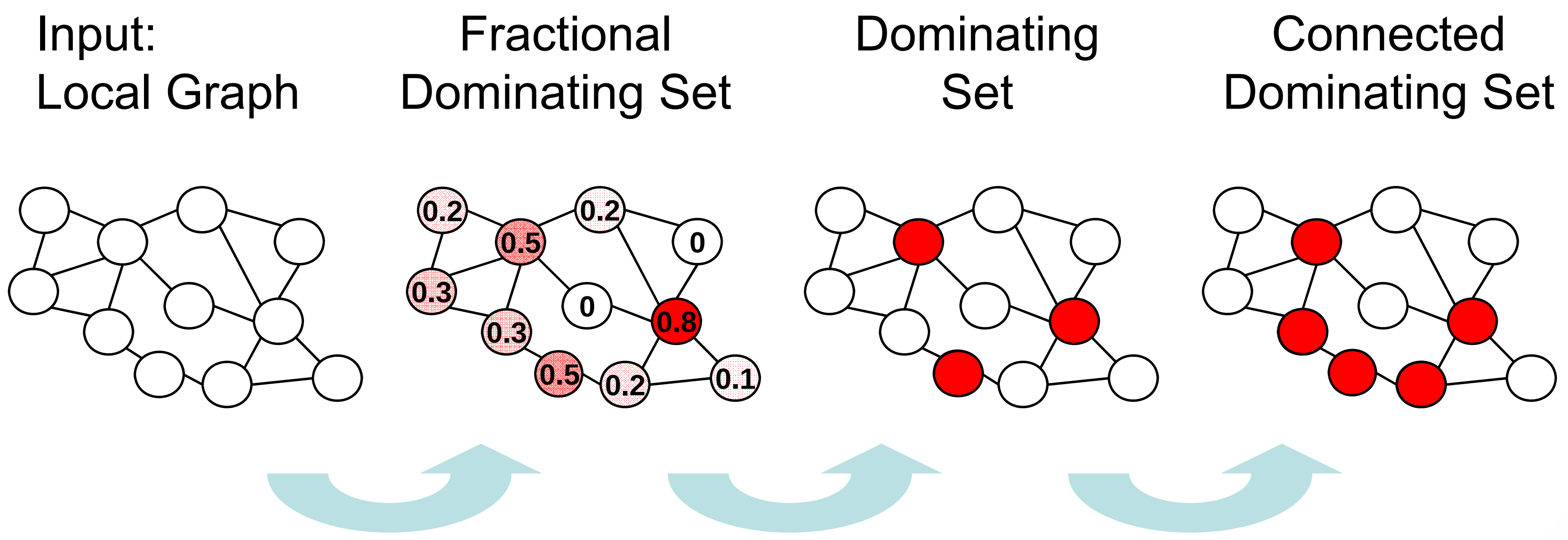
- An optimal algorithm (and also the tree growing algorithm) puts every k'th node into the CDS. Thus $|CDS_{OPT}| \approx n/k$; with $k = n/c$ for some positive constant c we have $|CDS_{OPT}| = O(1)$.

- 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 $|CDS_{Marking}| = \Omega(n)$.

- The worst-case quality of the marking algorithm is worst-case! ☺



“k-local” Algorithm: Overview



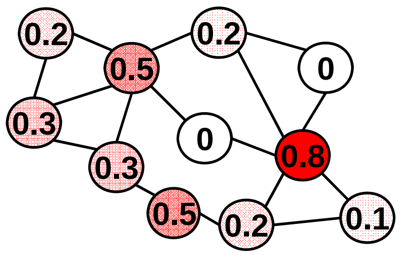
Phase A:
Distributed
linear program
rel. high degree
gives high value

Phase B:
Probabilistic
algorithm

Phase C:
Connect DS
by “tree” of
“bridges”

Phase A is a Distributed Linear Program

- Nodes 1, …, *n*: Each node *u* has variable x_u with $x_u \geq 0$
- Sum of *x*-values in each neighborhood at least 1 (local)
- Minimize sum of all *x*-values (global)



$0.5+0.3+0.3+0.2+0.2+0 = 1.5 \geq 1$

Linear Program

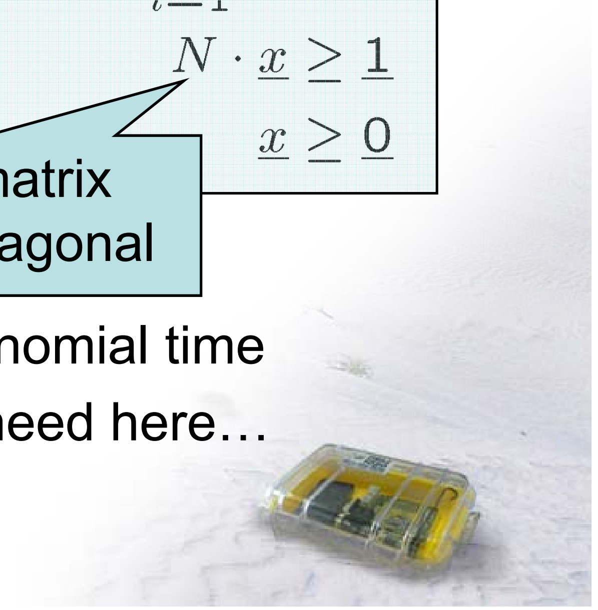
$$\min \sum_{i=1}^{n} x_i$$

$$\text{subject to} \quad \underline{N} \cdot \underline{x} \geq \underline{1}$$

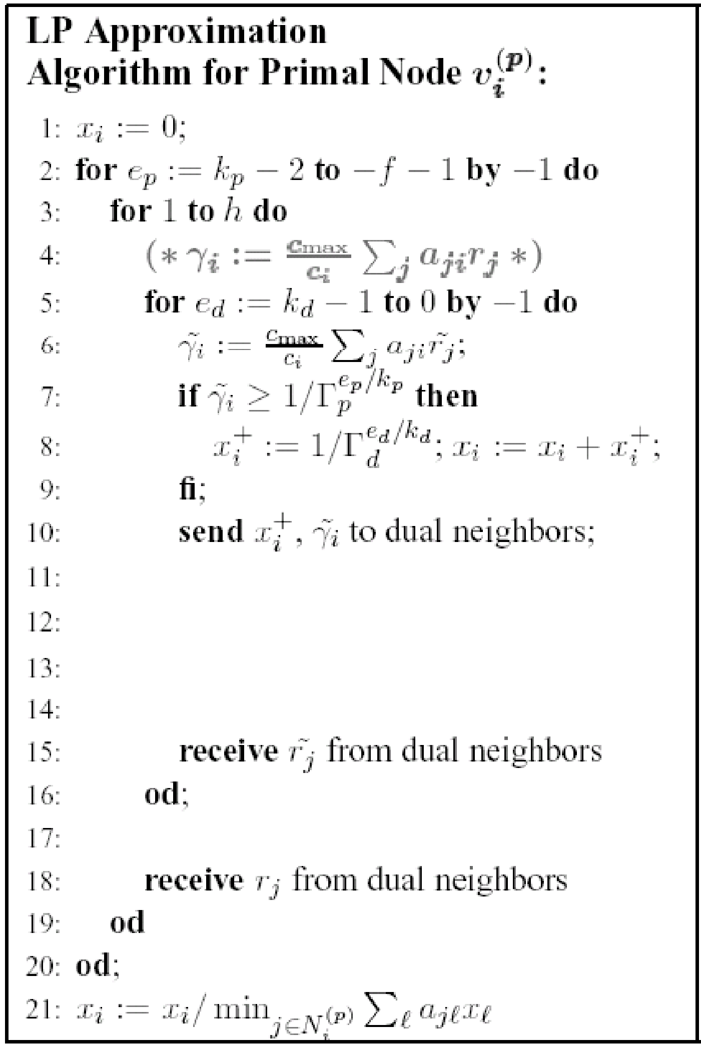
$$\underline{x} \geq \underline{0}$$

Adjacency matrix with 1's in diagonal

- Linear Programs can be solved optimally in polynomial time
- But not in a distributed fashion! That's what we need here…

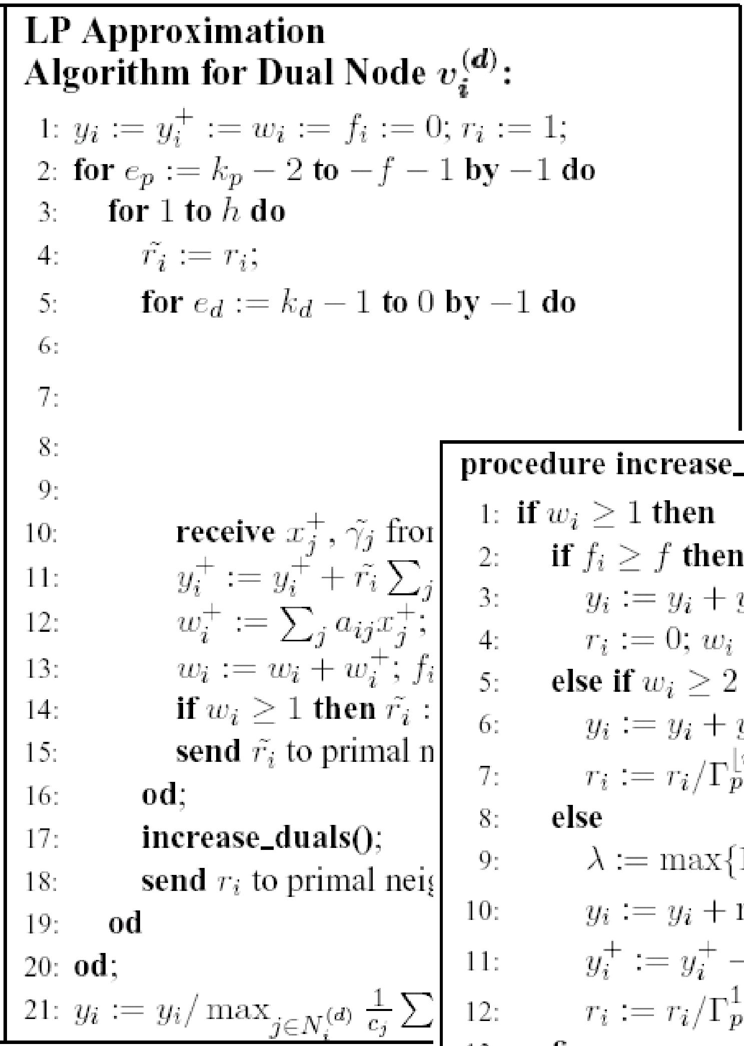


Phase A Algorithm



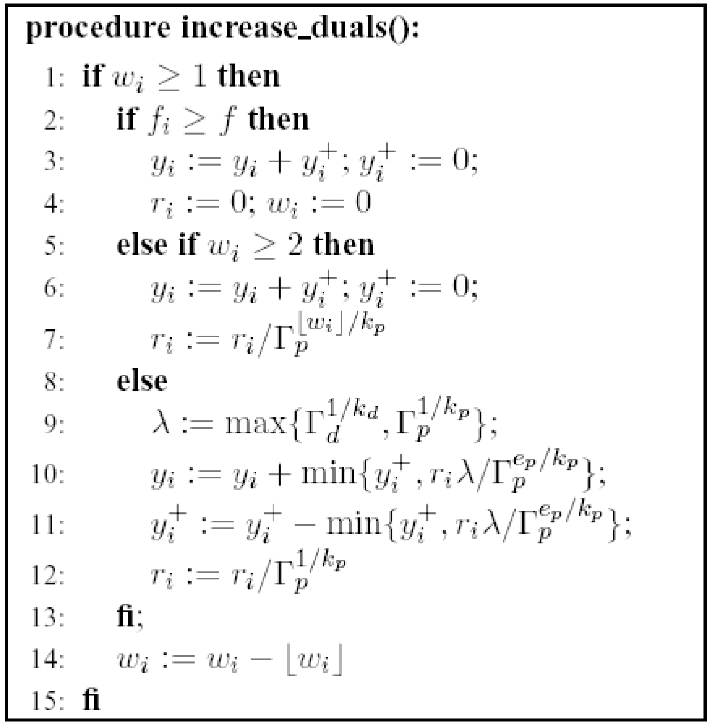
LP Approximation Algorithm for Primal Node $v_i^{(p)}$:

```
1:  $x_i := 0$;
2:  for $e_p := k_p - 2$ to $-f - 1$ by $-1$ do
3:      for $1$ to $h$ do
4:          $(* \gamma_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} r_j *)$
5:          for $e_d := k_d - 1$ to $0$ by $-1$ do
6:              $\tilde{\gamma}_i := \frac{c_{\max}}{c_i} \sum_j a_{ji} \tilde{r}_j$;
7:              if $\tilde{\gamma}_i \geq 1/\Gamma_p^{e_p/k_p}$ then
8:                  $x_i^+ := 1/\Gamma_d^{e_d/k_d}$; $x_i := x_i + x_i^+$;
9:              fi;
10:             send $x_i^+$, $\tilde{\gamma}_i$ to dual neighbors;
11:
12:
13:
14:
15:             receive $\tilde{r}_j$ from dual neighbors
16:         od;
17:
18:         receive $r_j$ from dual neighbors
19:     od
20: od;
21: $x_i := x_i / \min_{j \in N_i^{(p)}} \sum_\ell a_{j\ell} x_\ell$
```



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

```
1:  $y_i := y_i^+ := w_i := f_i := 0$; $r_i := 1$;
2:  for $e_p := k_p - 2$ to $-f - 1$ by $-1$ do
3:      for $1$ to $h$ do
4:          $\tilde{r}_i := r_i$;
5:          for $e_d := k_d - 1$ to $0$ by $-1$ do
6:
7:
8:
9:
10:             receive $x_j^+$, $\tilde{\gamma}_j$ fro
11:             $y_i^+ := y_i^+ + \tilde{r}_i \sum_j$
12:             $w_i^+ := \sum_j a_{ij} x_j^+$;
13:             $w_i := w_i + w_i^+$; $f_i$
14:             if $w_i \geq 1$ then $\tilde{r}_i$ :
15:             send $\tilde{r}_i$ to primal n
16:         od;
17:         increase_duals();
18:         send $r_i$ to primal nei
19:     od
20: od;
21: $y_i := y_i / \max_{j \in N_i^{(d)}} \frac{1}{c_j} \sum$
```



procedure increase_duals():

```
1:  if $w_i \geq 1$ then
2:      if $f_i \geq f$ then
3:          $y_i := y_i + y_i^+$; $y_i^+ := 0$;
4:          $r_i := 0$; $w_i := 0$
5:      else if $w_i \geq 2$ then
6:          $y_i := y_i + y_i^+$; $y_i^+ := 0$;
7:          $r_i := r_i / \Gamma_p^{\lfloor w_i \rfloor / k_p}$
8:      else
9:          $\lambda := \max\{\Gamma_d^{1/k_d}, \Gamma_p^{1/k_p}\}$;
10:         $y_i := y_i + \min\{y_i^+, r_i \lambda / \Gamma_p^{e_p/k_p}\}$;
11:         $y_i^+ := y_i^+ - \min\{y_i^+, r_i \lambda / \Gamma_p^{e_p/k_p}\}$;
12:         $r_i := r_i / \Gamma_p^{1/k_p}$
13:     fi;
14:     $w_i := w_i - \lfloor w_i \rfloor$
15: fi
```

Result after Phase A

- Distributed Approximation for Linear Program
- Instead of the optimal values x_i^* at nodes, nodes have $x_i^{(\alpha)}$, with

$$\sum_{i=1}^{n} x_i^{(\alpha)} \leq \alpha \cdot \sum_{i=1}^{n} x_i^*$$

- The value of α depends on the number of rounds k (the locality)

$$\alpha \leq (\Delta + 1)^{c/\sqrt{k}}$$

- The analysis is rather intricate… ☺

Phase B Algorithm

Each node applies the following algorithm:

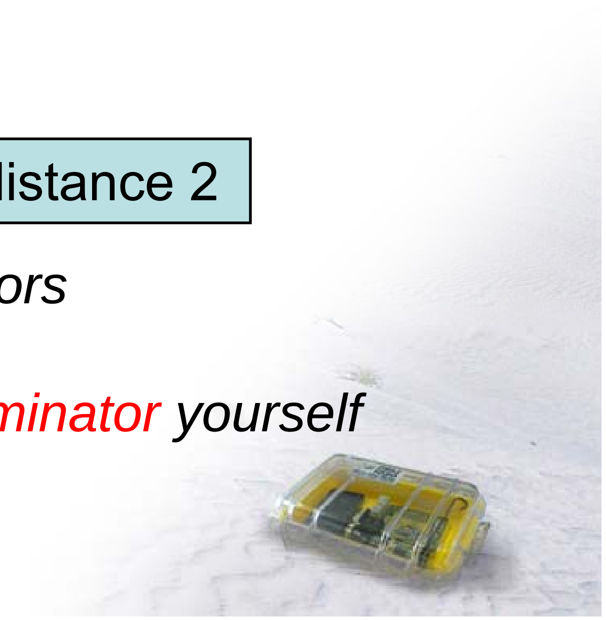
1. *Calculate* $\delta_i^{(2)}$ *(= maximum degree of neighbors in distance 2)*

2. *Become a dominator (i.e. go to the dominating set) with probability*

$$p_i := \min\{1, \underbrace{x_i^{(\alpha)}}_{\text{From phase A}} \cdot \ln(\underbrace{\delta_i^{(2)}}_{\text{Highest degree in distance 2}} + 1)\}$$

3. *Send status (dominator or not) to all neighbors*

4. *If no neighbor is a dominator, become a dominator yourself*

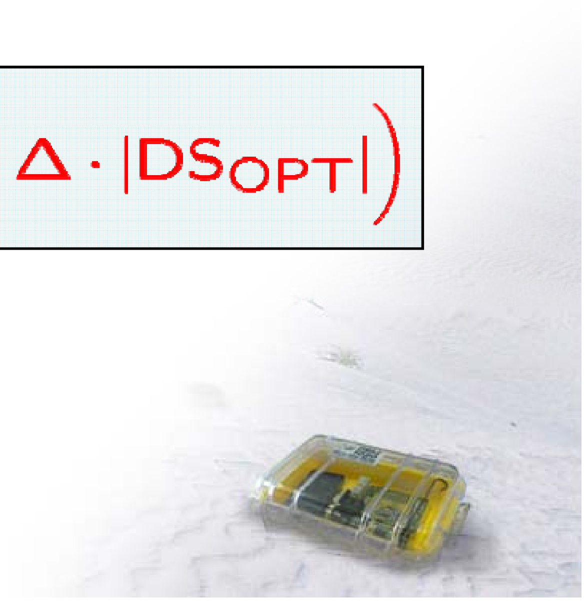


Result after Phase B

- Randomized rounding technique
- Expected number of nodes joining the dominating set in step 2 is bounded by $\alpha \log(\Delta+1) \cdot |DS_{OPT}|$.
- Expected number of nodes joining the dominating set in step 4 is bounded by $|DS_{OPT}|$.

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

- Phase C → essentially the same result for CDS

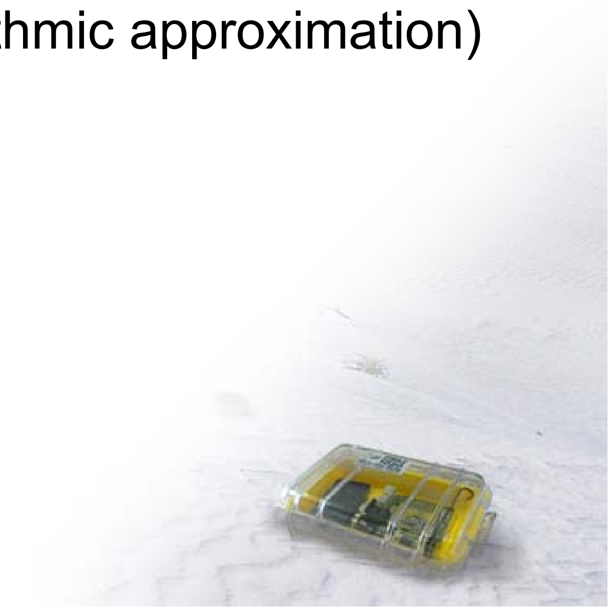


Results

- First time/approximation tradeoff. First algorithm which achieves a non-trivial approximation ratio in constant time (even for UDG!)

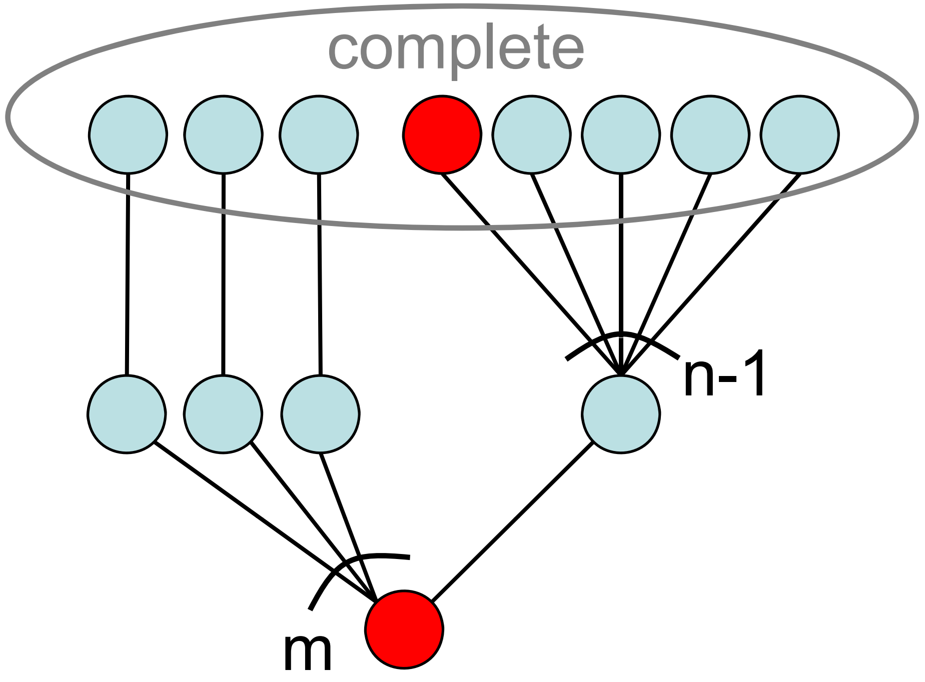
- Improved version
  - $O(\log^2\Delta / \varepsilon^4)$ time for a $(1+\varepsilon)$-approximation of phase A with logarithmic sized messages.
  - An improved and generalized distributed randomized rounding technique for phase B (constant time, logarithmic approximation)
  - Works for quite general linear programs.

- Is it any good…?

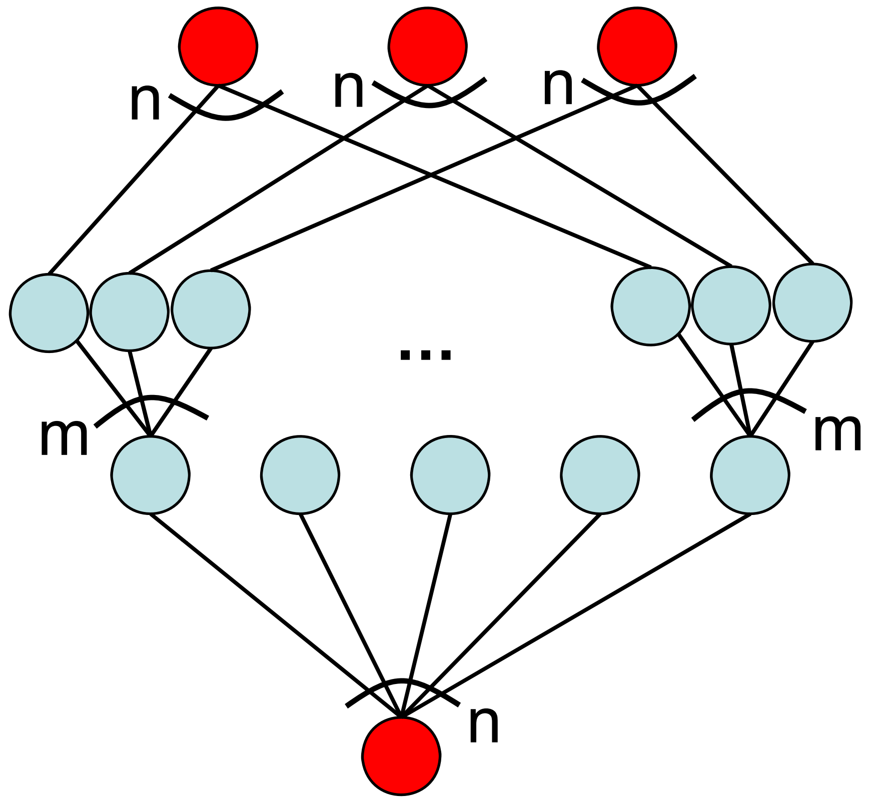


Lower Bound for Dominating Sets: Intuition…

- Two graphs ($m << n$). Optimal dominating sets are marked red.



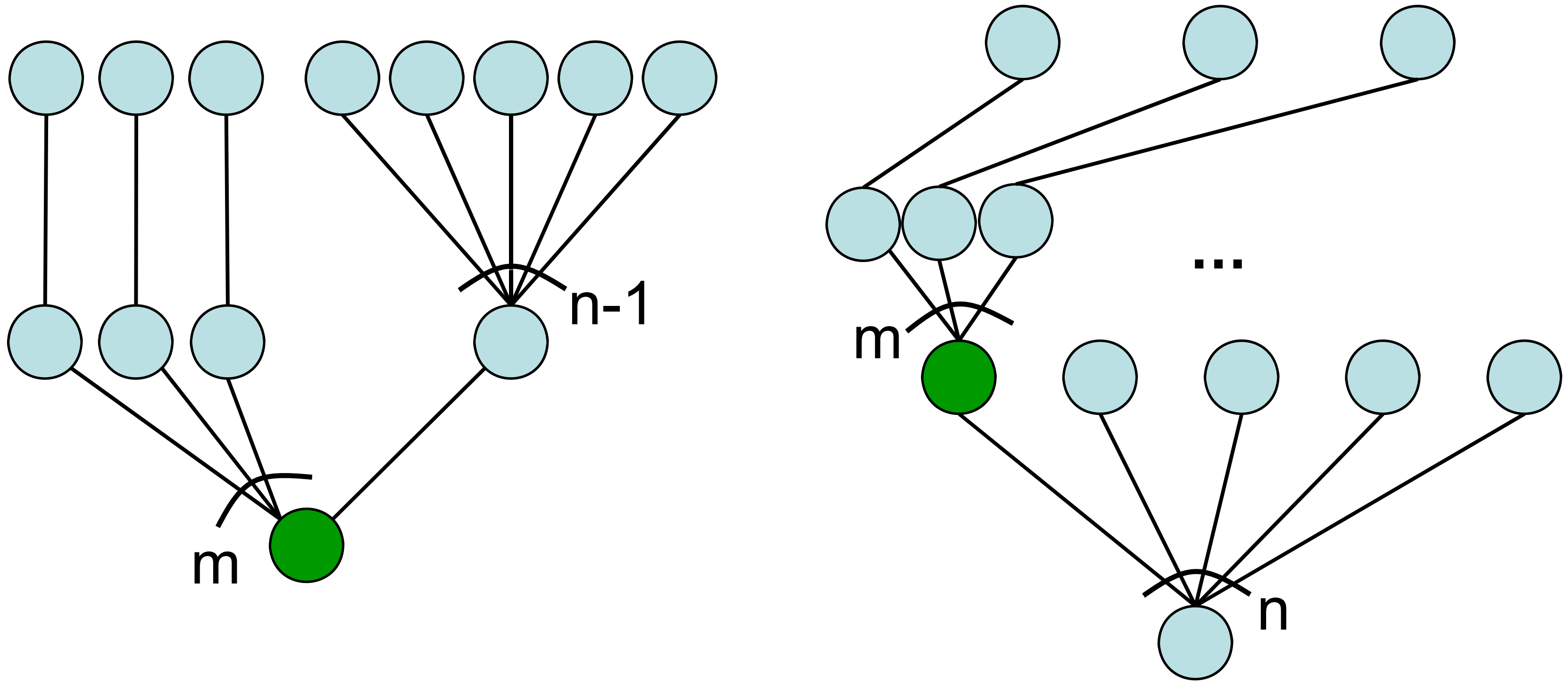
$|DS_{OPT}| = 2.$



$|DS_{OPT}| = m+1.$

Lower Bound for Dominating Sets: Intuition…

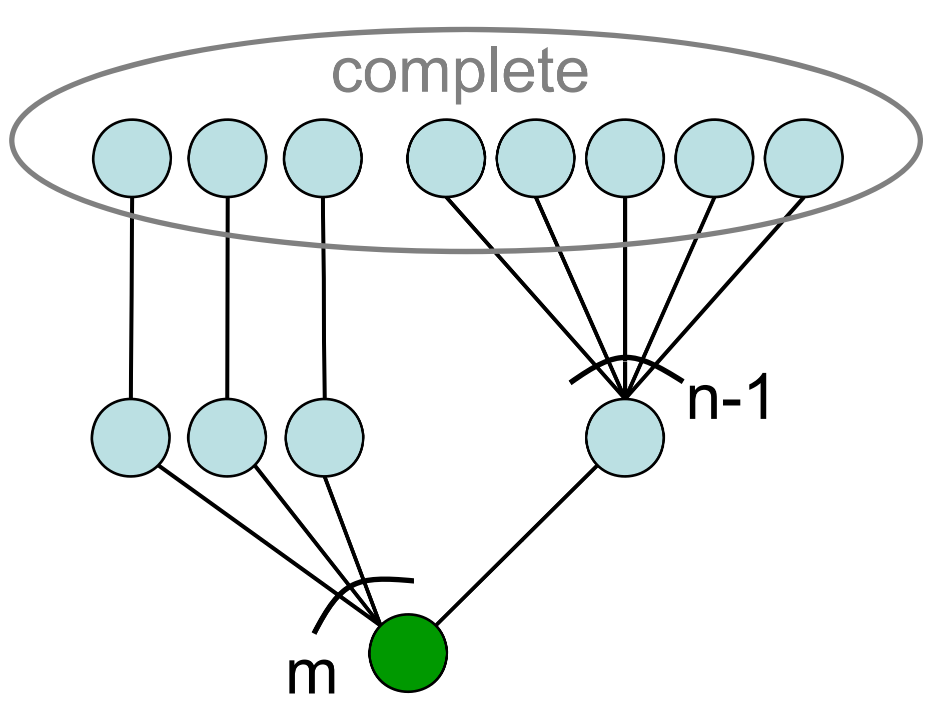
- In local algorithms, nodes must decide only using local knowledge.
- In the example green nodes see exactly the same neighborhood.

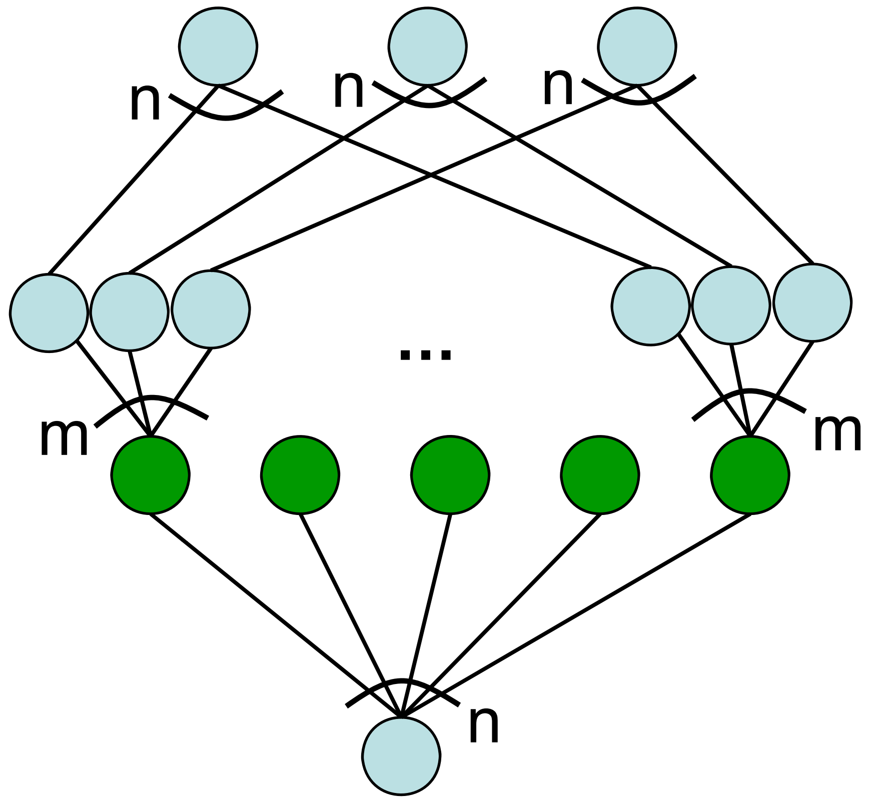


- So these green nodes must decide the same way!

Lower Bound for Dominating Sets: Intuition…

- But however they decide, one way will be devastating (with $n = m^2$)!





$|DS_{OPT}| = 2.$

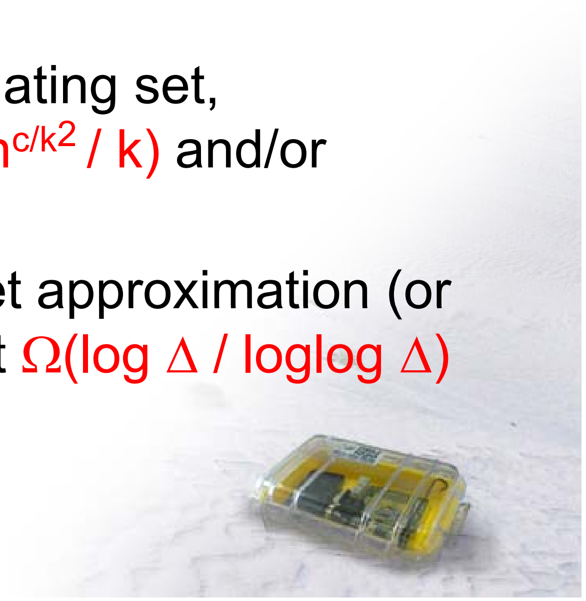
$|DS_{OPT\text{ without green}}| \geq m.$

$|DS_{OPT}| = m+1.$

$|DS_{OPT\text{ with green}}| > n$

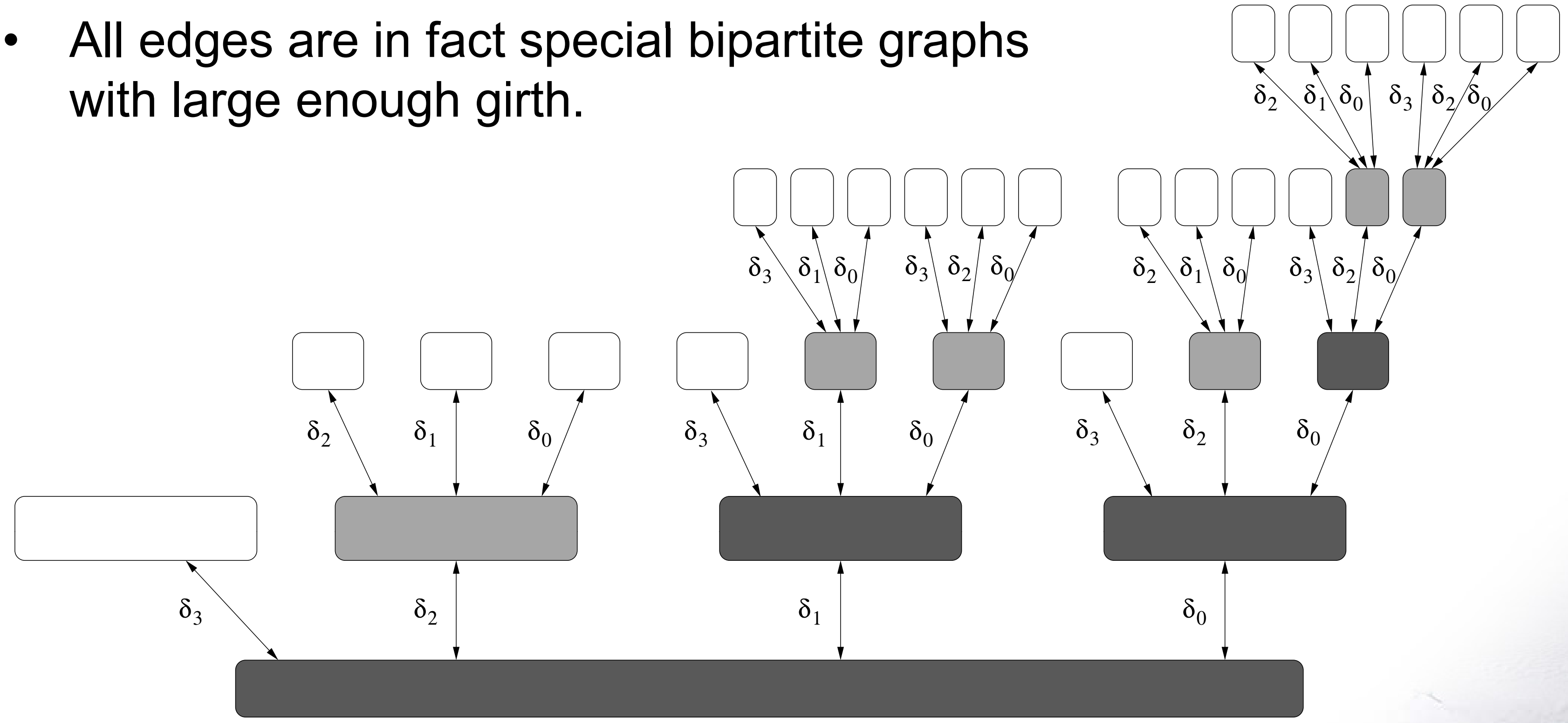
The Lower Bound

- Lower bound
  - Model: In a network/graph G (nodes = processors), each node can exchange a message with all its neighbors for k rounds. After k rounds, node needs to decide.
  - We construct the graph such that there are nodes that see the same neighborhood up to distance k. We show that node ID’s do not help, and using Yao’s principle also randomization does not.
  - Results: Many problems (vertex cover, dominating set, matching, etc.) can only be approximated $\Omega(n^{c/k^2} / k)$ and/or $\Omega(\Delta^{1/k} / k)$.
  - It follows that a polylogarithmic dominating set approximation (or maximal independent set, etc.) needs at least $\Omega(\log \Delta / \log\log \Delta)$ and/or $\Omega((\log n / \log\log n)^{1/2})$ time.

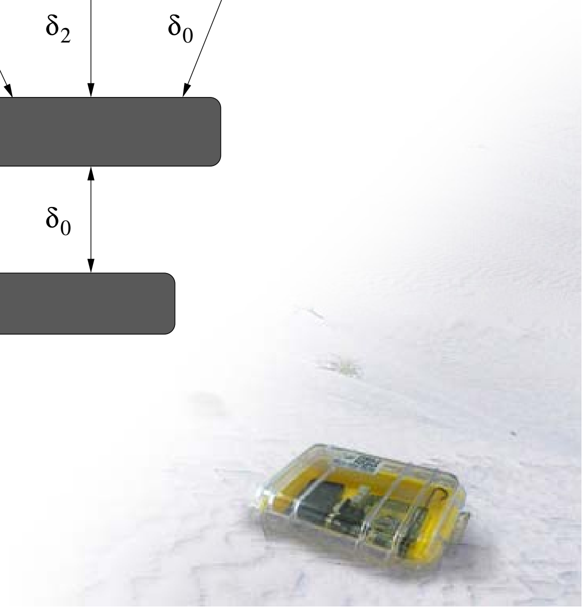


Graph Used in Dominating Set Lower Bound

- The example is for k = 3.
- All edges are in fact special bipartite graphs with large enough girth.

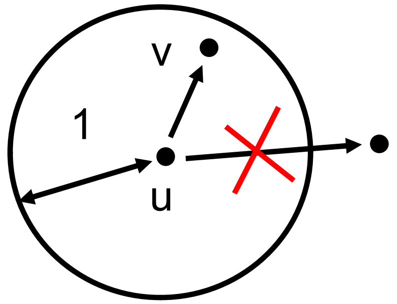


δ_2 δ_1 δ_0 δ_3 δ_2 δ_0
δ_3 δ_1 δ_0 δ_3 δ_2 δ_0
δ_2 δ_1 δ_0 δ_3 δ_2 δ_0
δ_2 δ_1 δ_0
δ_3 δ_1 δ_0
δ_3 δ_2 δ_0
δ_3 δ_2 δ_1 δ_0

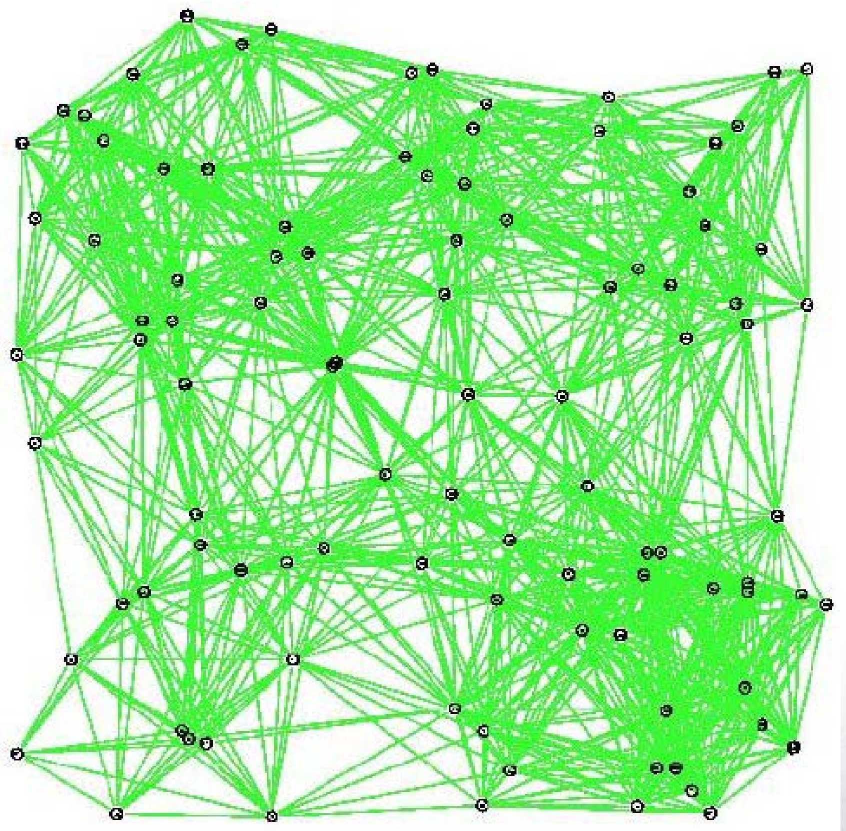


Better and faster algorithm

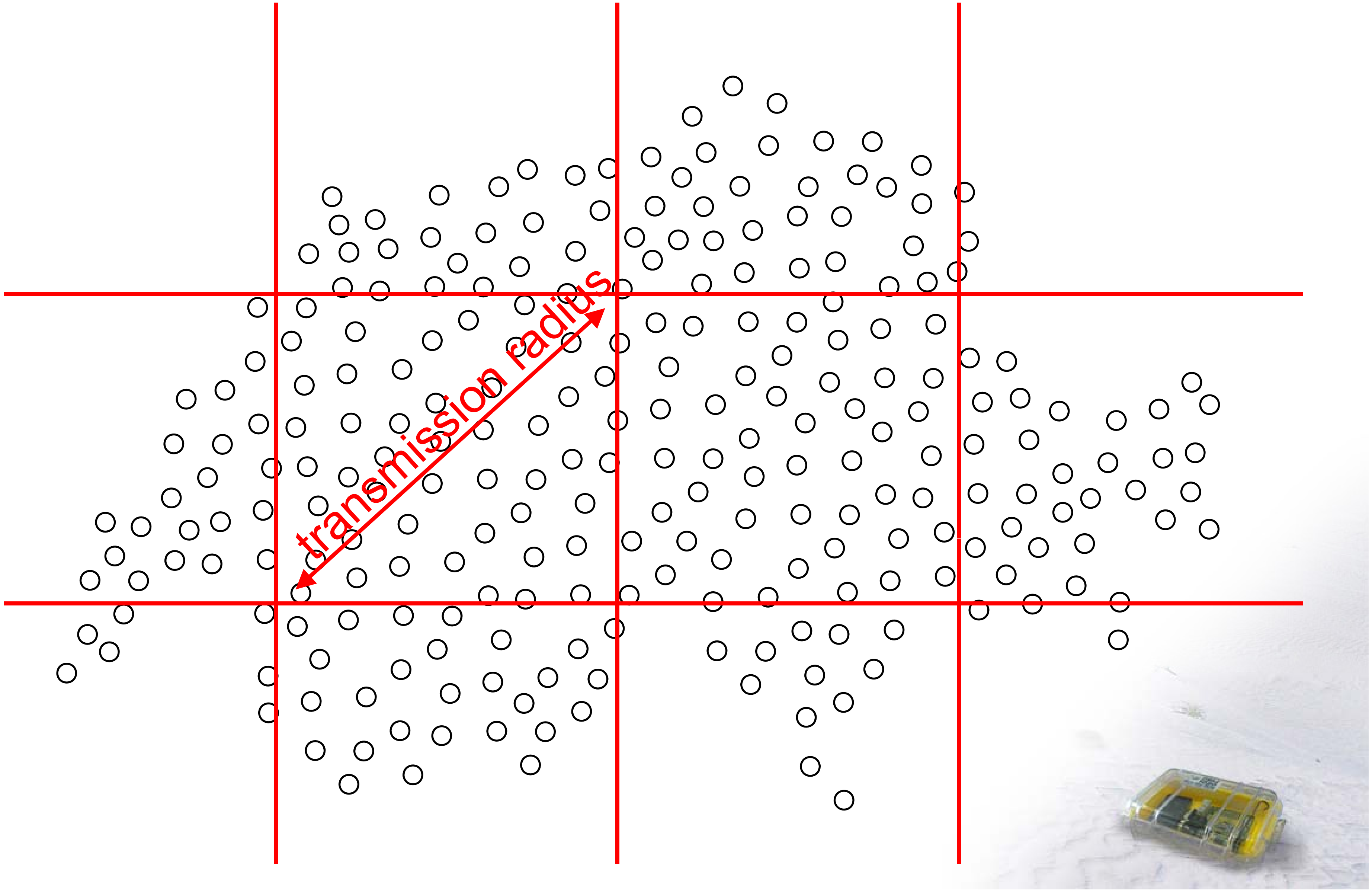
- Assume that graph is a unit disk graph (UDG)



- Assume that nodes know their positions (GPS)



Then…



Grid Algorithm

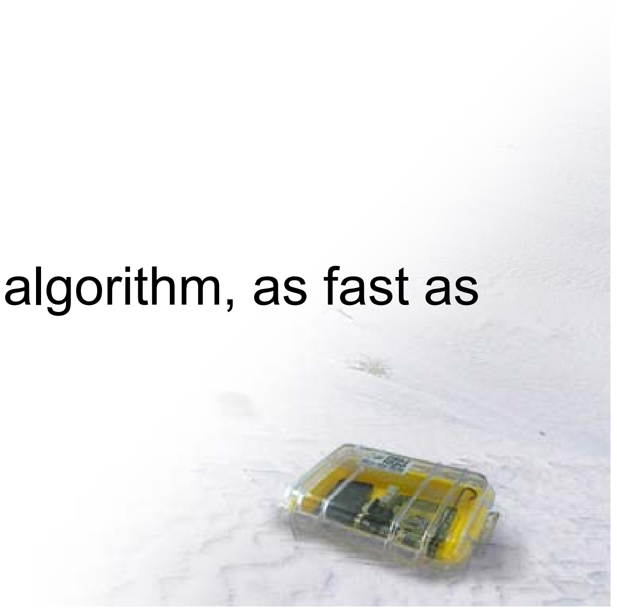
1. Beacon your position

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, then simply “loop” through algorithm, as fast as your application/mobility wants you to.



Example: Comparison of Two Algorithms for Dominating Set

Algorithm 1

- Algorithm c S
- k^2+O(1 sions/node
- O(Δ) approximation
- plex!
- ance OK

General Graph!
No Position Information!

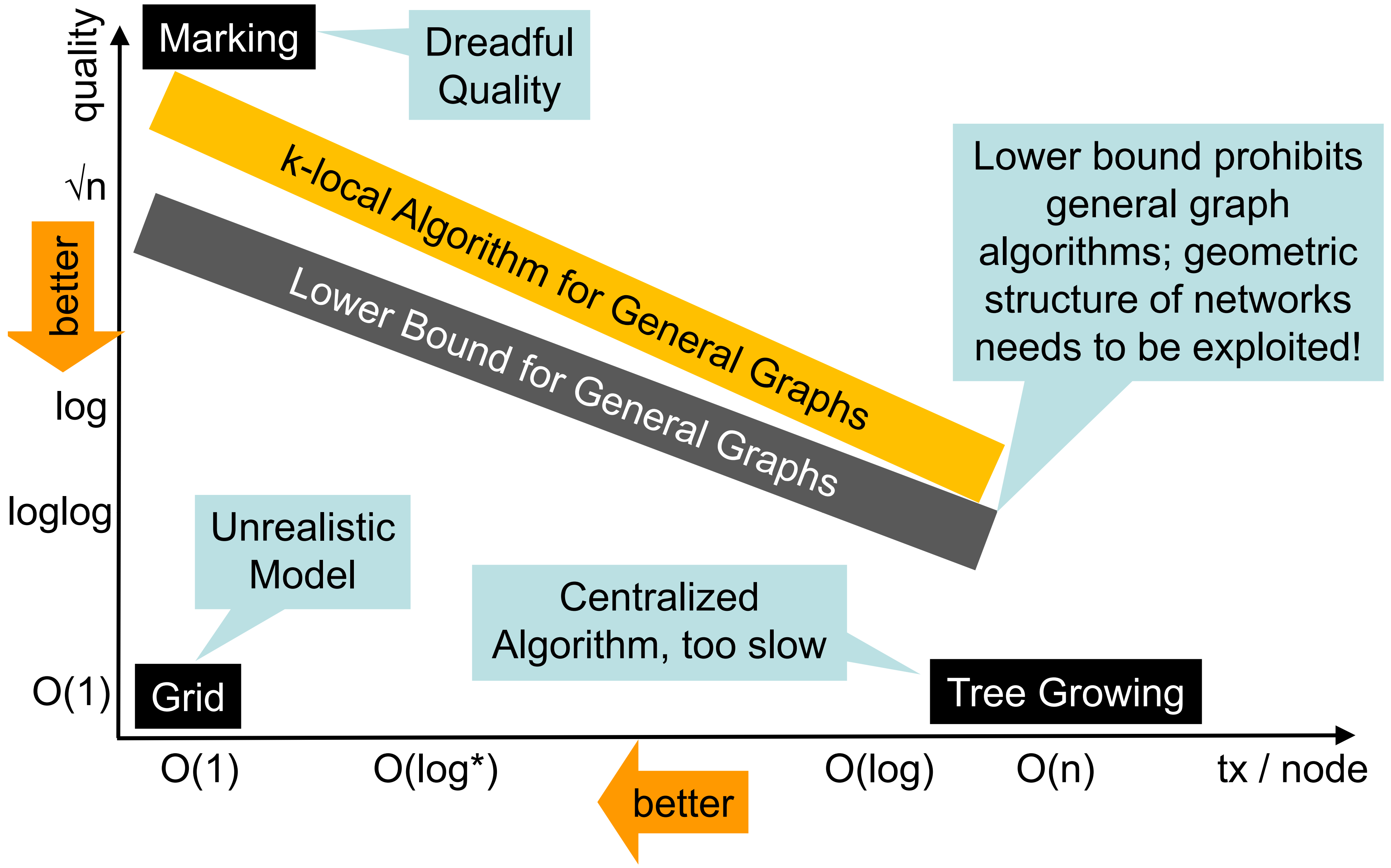
Algorithm 2

- Algorith s DS
- 1 tr n/node
- O ximation
- formance great!
- etter than lower bound!!

Unit Disk Graph Only!
Requires GPS Device!

The model determines the distributed complexity of a problem

Algorithms seen so far... all suck!!


quality
√n
log
loglog
O(1)
better
Marking
Dreadful Quality
k-local Algorithm for General Graphs
Lower Bound for General Graphs
Lower bound prohibits general graph algorithms; geometric structure of networks needs to be exploited!
Unrealistic Model
Grid
Centralized Algorithm, too slow
Tree Growing
O(1)
O(log*)
O(log)
O(n)
tx / node
better

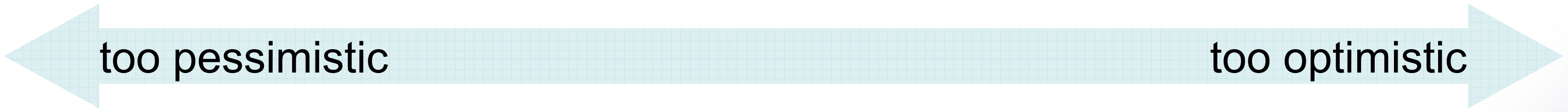
Let’s talk about models…

- 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

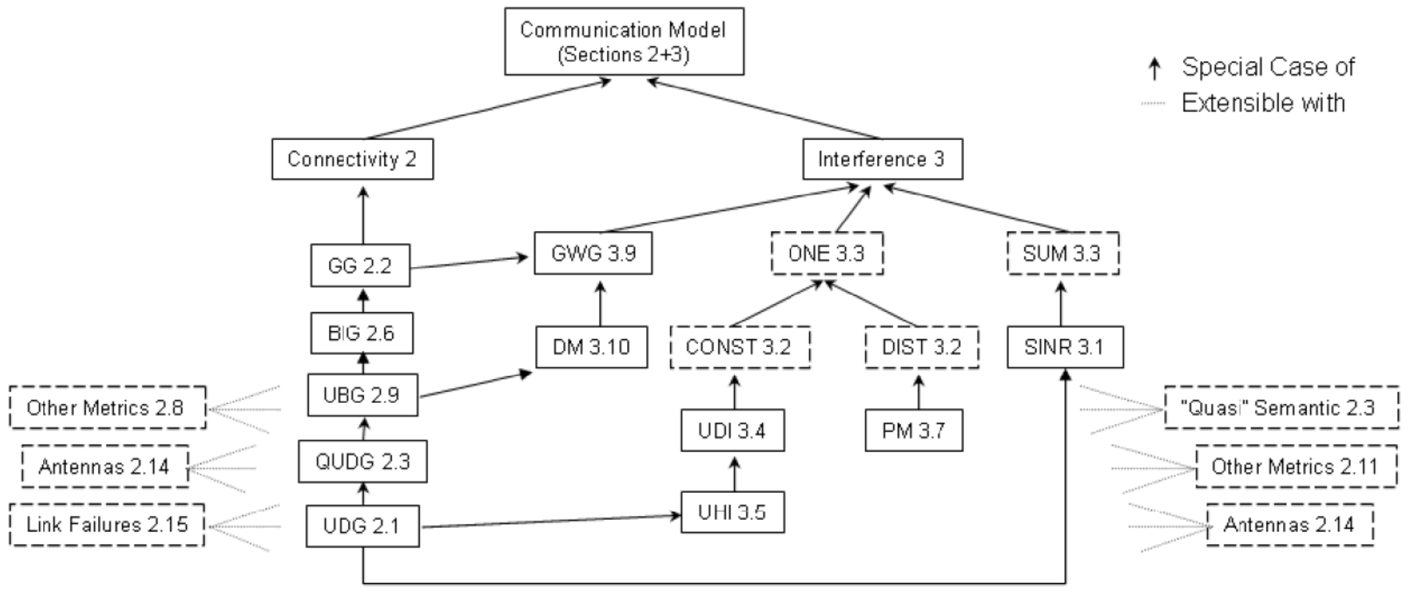


Let‘s look at models in between these extremes!

Why are models needed?

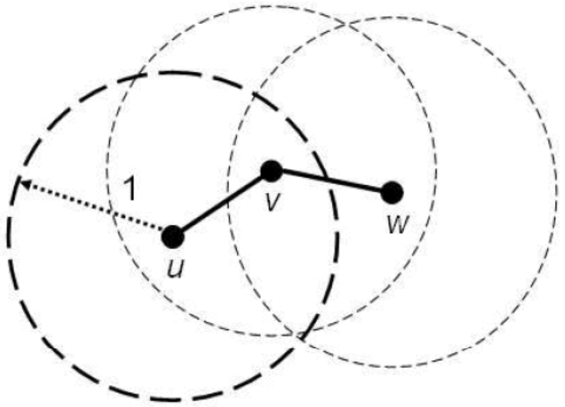
- Formal models help us understanding a problem

- 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

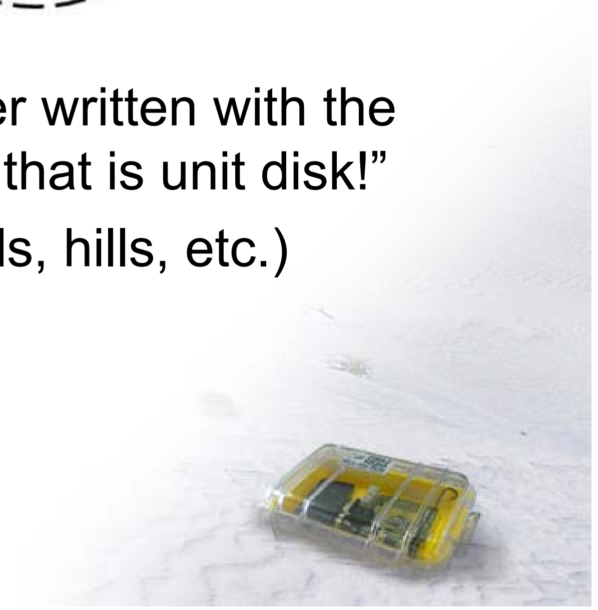


Unit Disk Graph (UDG)

- 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 $\{u,v\} \in E \Leftrightarrow |u,v| \leq 1$

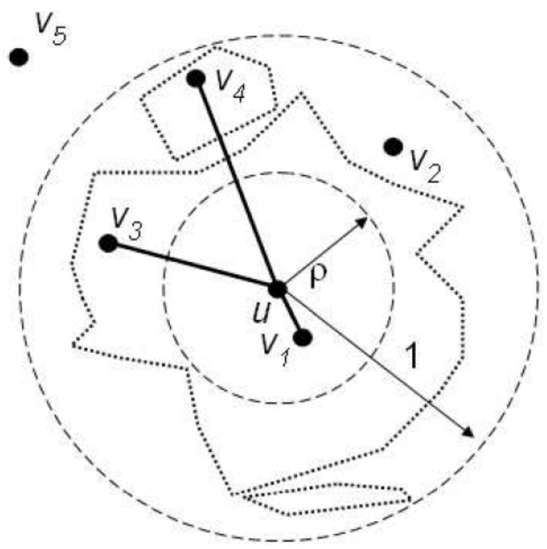


+ Very simple, allows for strong analysis
– Not realistic: "If you gave me $100 for each paper written with the unit disk assumption, I still could not buy a radio that is unit disk!"
– Particularly bad in obstructed environments (walls, hills, etc.)
- Natural extension: 3D UDG



Quasi Unit Disk Graph (UDG)

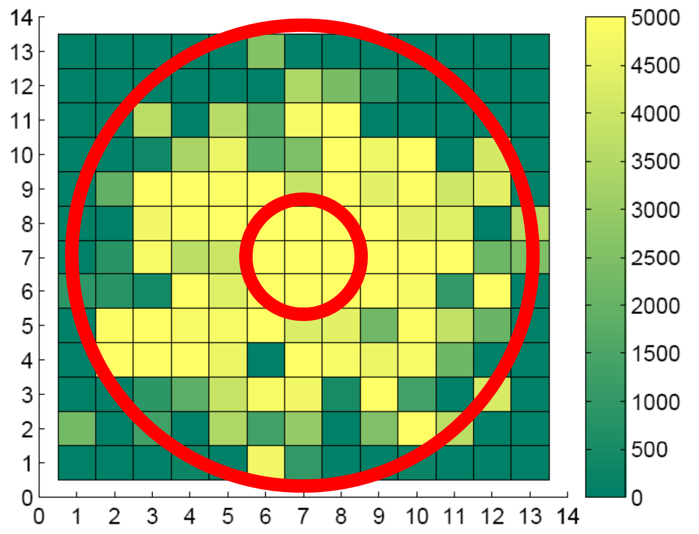
- Two radii, 1 and ρ, with $\rho \leq 1$
  - $|u,v| \leq \rho \Leftrightarrow \{u,v\} \in E$
  - $1 < |u,v| \Leftrightarrow \{u,v\} \notin E$
  - $\rho < |u,v| \leq 1 \Leftrightarrow$ it depends!
    - … on an adversary
    - … on probabilistic model
    - …



+ Simple, analyzable
+ More realistic than UDG

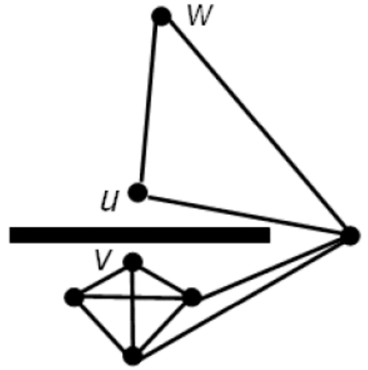
– Still bad in obstructed environments (walls, hills, etc.)

- Natural extension: 3D QUDG



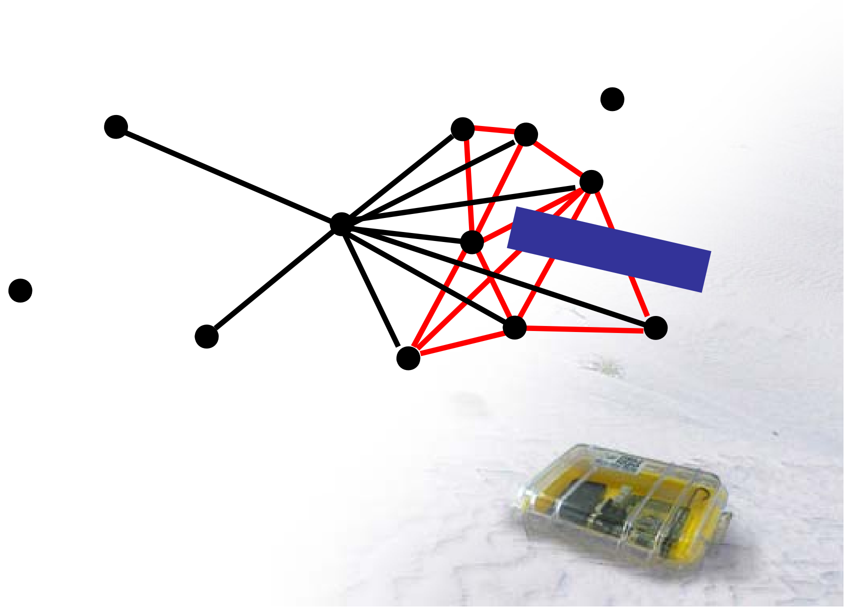
Bounded Independence Graph (BIG)

- How realistic is 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. $O(r^2)$ or $O(r^3)$



Unit Ball Graph (UBG)

- $\exists$ metric (V,d) with constant doubling dimension.

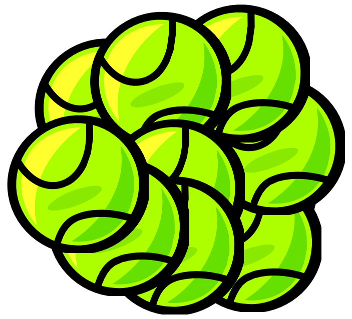
- Metric: Each edge has a distance d, with

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)

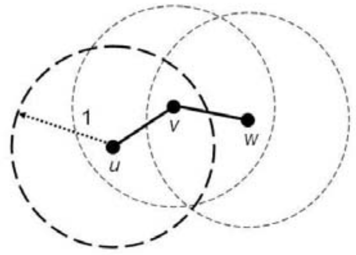
- Doubling dimension: log(#balls of radius r/2 to cover ball of radius r)
  - Constant: you only need a constant number of balls of half the radius

- 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

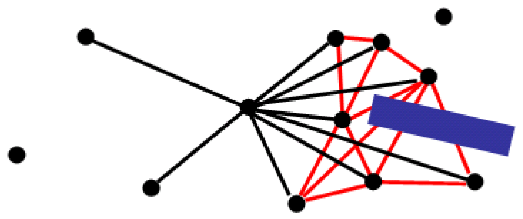
too pessimistic

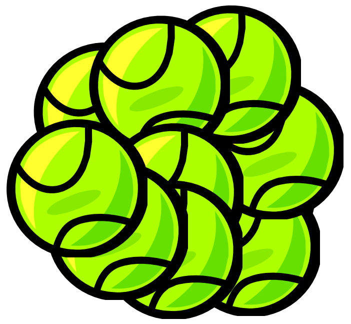
too optimistic

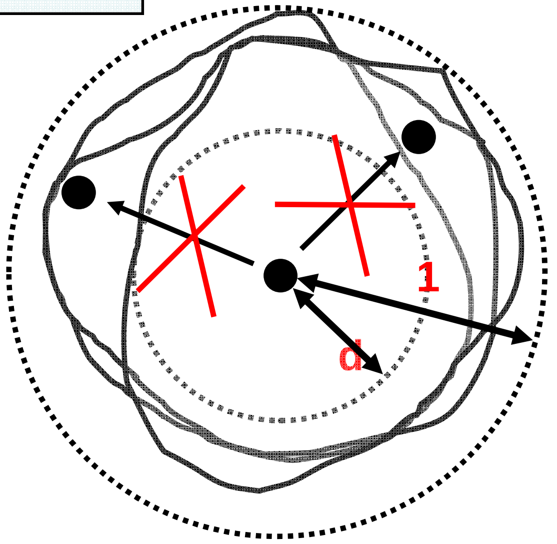
Bounded Independence

Unit Ball Graph

Quasi UDG

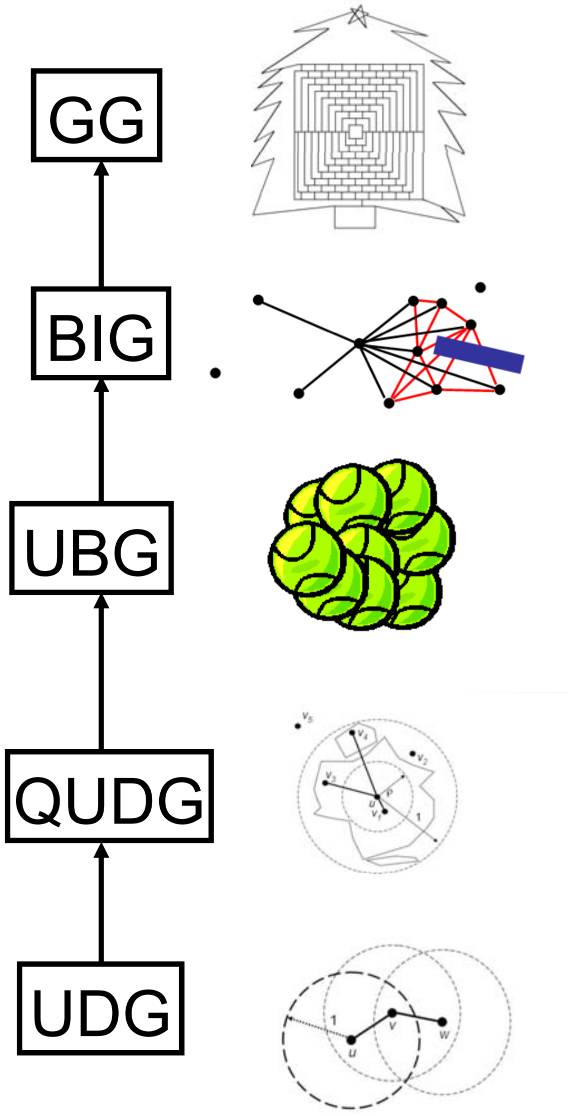






Models are related

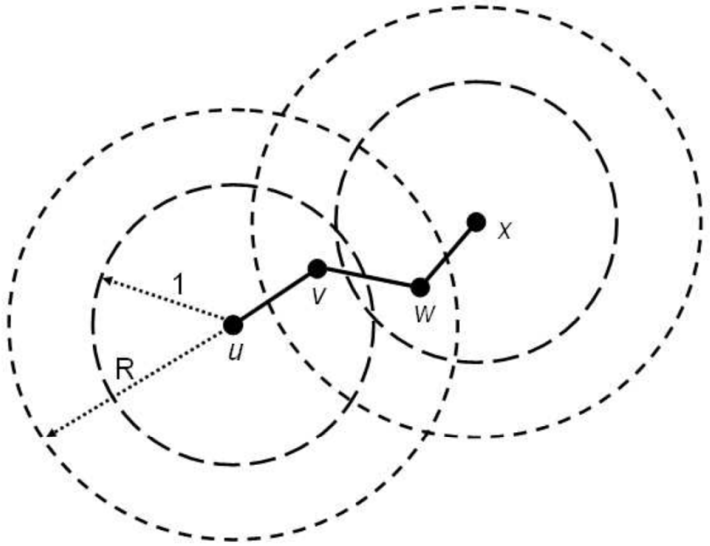
- BIG is special case of general graph, BIG $\subseteq$ GG
- UBG $\subseteq$ BIG because the size of the independent sets of any UBG is polynomially bounded
- QUDG(constant ρ) $\subseteq$ UBG
- QUDG(ρ=1) = UDG



Beyond Connectivity: Protocol Model (PM)

- For lower layer protocols, a model needs to be specific about interference. A simplest interference model is an extention of the UDG. In the protocol model, a transmission by a node in at most distance 1 is received iff there is no conflicting transmission by a node in distance at most R, with R $\geq$ 1, sometimes just R = 2.

+ Easy to explain
– Inherits all major drawbacks from the UDG model
– Does not easily allow for designing distributed algorithms
– Lots of interfering transmissions just outside the interference radius R do not sum up.
- Can be extended with the same extensions as UDG, e.g. QUDG



Hop Interference (HI)

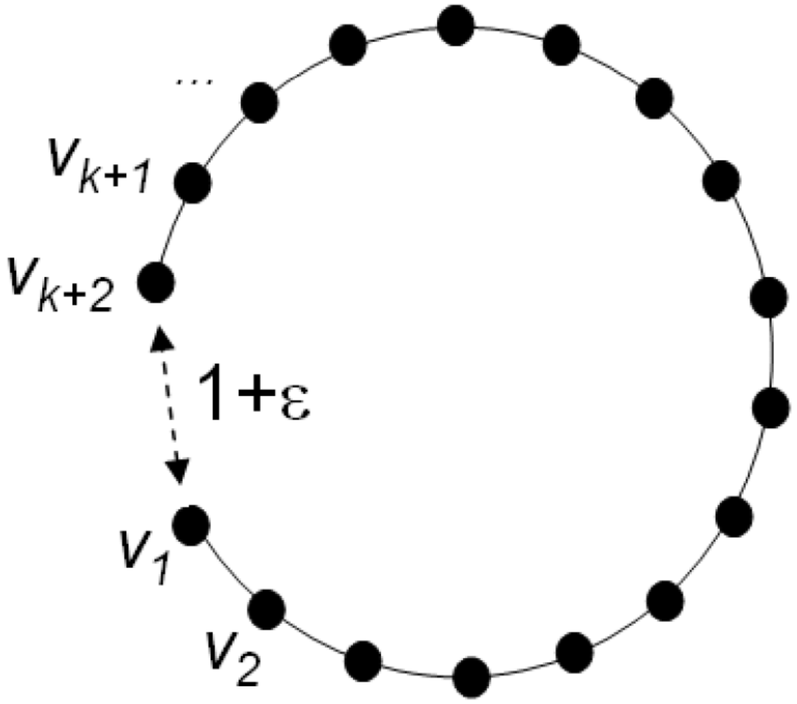
- An often-used interference model is hop-interference. Here a UDG is given. Two nodes can communicate directly iff they are adjacent, and if there is no concurrent sender in the k-hop neighborhood of the receiver (in the UDG). Sometimes k=2.

- Special case of the protocol model, inheriting all its drawbacks

+ Simple
+ Allows for distributed algorithms

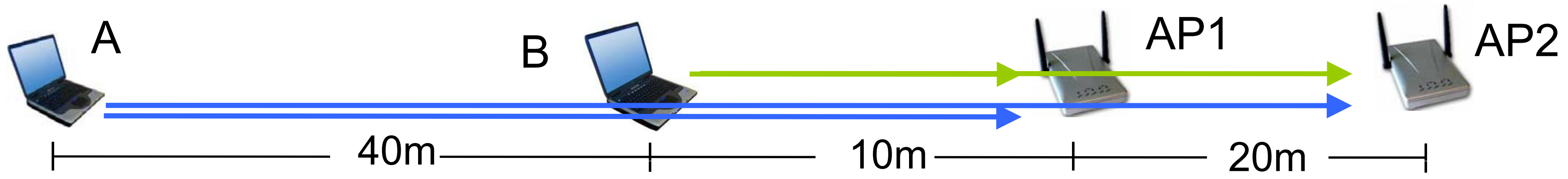
– A node can be close but not produce any interference (see pic)

- Can be extended with the same extensions as UDG, e.g. QUDG



Models Beyond Graphs

- Clients A and B want to send (max. rate x kb/s)
- Assume there is a single frequency
- What total throughput („spatial reuse“) can be achieved...?



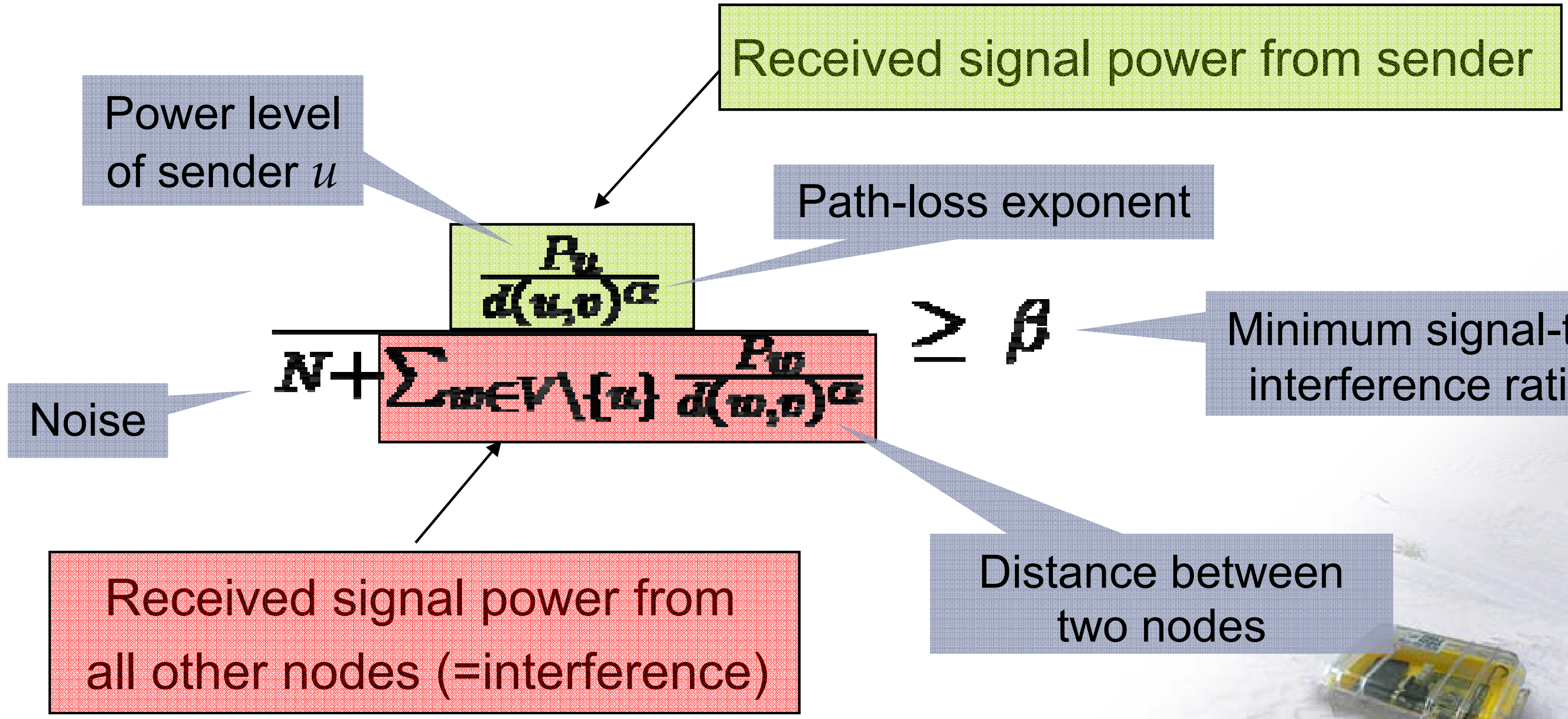
Total throughput at most: x kb/s

In graph-based models…

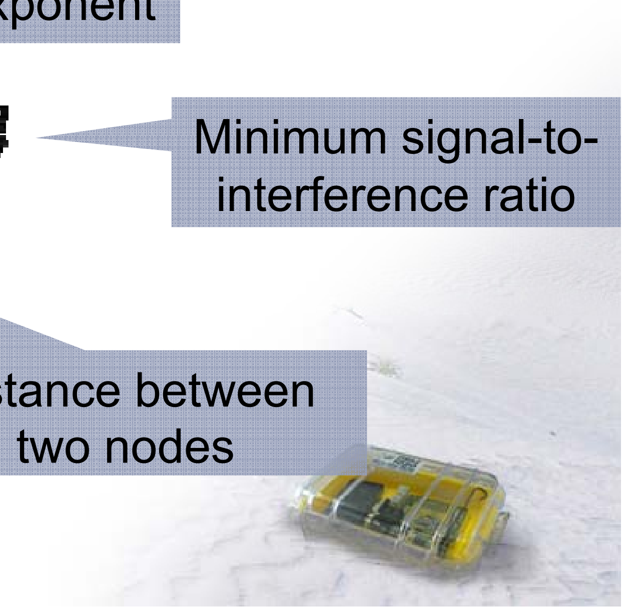
no spatial reuse seems possible…

Signal-to-Interference-Plus-Noise Ratio (SINR, Physical M.)

- Communication theorists study complex fading and signal-to-noise-plus-interference (SINR)-based models
- Simplest case:
  → packets can be decoded if SINR is larger than β at receiver

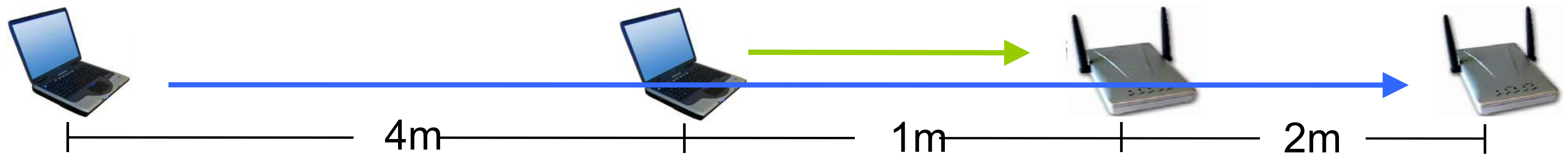

Received signal power from sender
Power level of sender u
Path-loss exponent
Minimum signal-to-interference ratio
Noise
Received signal power from all other nodes (=interference)
Distance between two nodes

$$\frac{\frac{P_u}{d(u,v)^{\alpha}}}{N+\sum_{w\in V\setminus\{u\}}\frac{P_w}{d(w,v)^{\alpha}}} \geq \beta$$



SINR Example

A sends to AP2, B sends to AP1 → (max. rate x kb/s)



- Assume a single frequency (and no fancy decoding techniques!)
- Let α=3, β=3, and N=10nW
- Set the transmission powers as follows P_B= -15 dBm and P_A= 1 dBm

SINR of A at AP2: $\frac{1.26mW/(7m)^3}{0.01\mu W + 31.6\mu W/(3m)^3} \approx 3.11 \geq \beta$

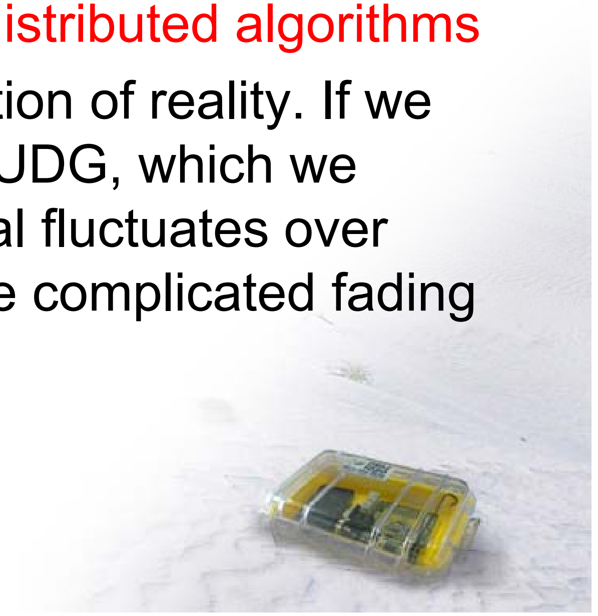
SINR of B at AP1: $\frac{31.6\mu W/(1m)^3}{0.01\mu W + 1.26mW/(5m)^3} \approx 3.13 \geq \beta$

A total throughput of 2x kb/s is possible !

SINR Discussion

+ In contrast to other low-layer models such as PM the SINR model allows for interference that does sum up. This is certainly closer to reality. However, SINR is not reality. In reality, e.g., competing transmissions may even cancel themselves, and produce less interference. In that sense the SINR model is worse than reality.

– SINR is complicated, hard to analyze
– Similarly as PM, SINR does not really allow for distributed algorithms
– Despite being complicated, it is a total simplification of reality. If we remove the “I” from the SINR model, we have a UDG, which we know is not correct. Also, in reality, e.g. the signal fluctuates over time. Some of these issues are captures by more complicated fading channel models.

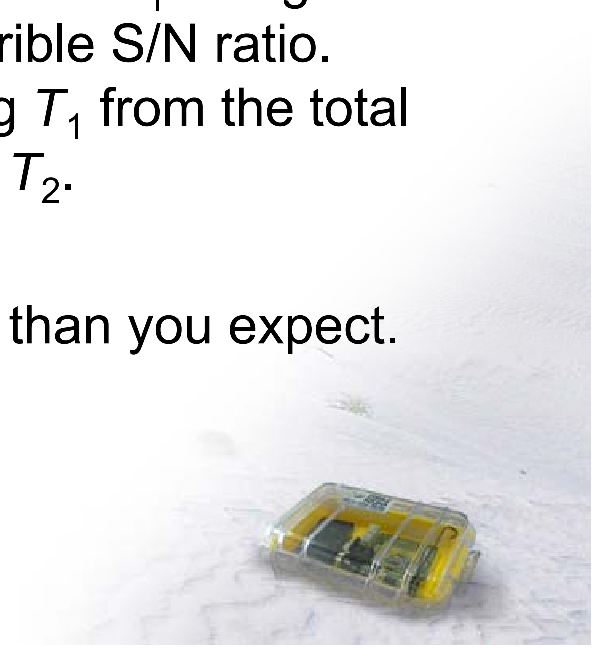


More on SINR

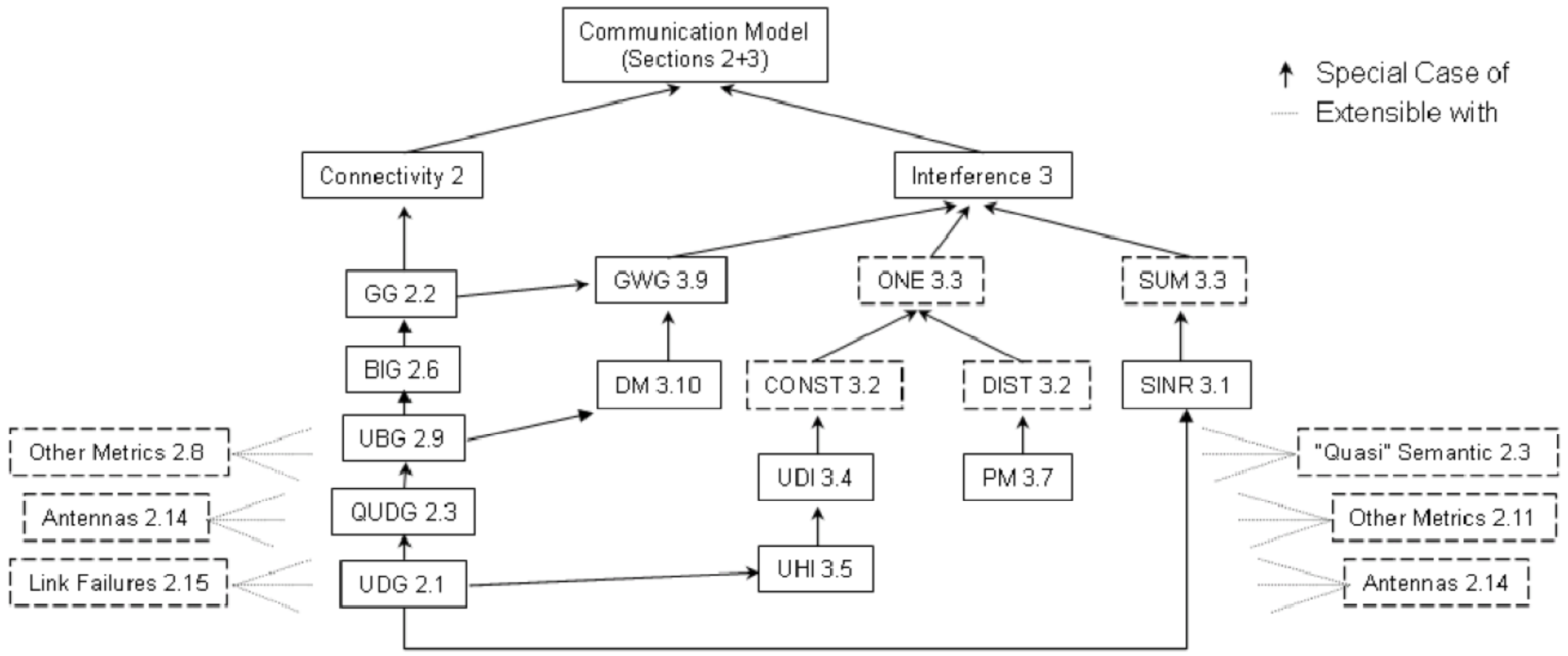
- Often there is more than a single threshold β, that decides whether reception is possible or not. In many networks, a higher S/N ratio allows for more advanced modulation and coding techniques, allowing for higher throughput (e.g. Wireless LAN)

- However, even more is possible: For example, assume that a receiver is receiving two transmissions, transmission T_1 being much stronger than transmission T_2. Then T_2 has a terrible S/N ratio. However, we might be able to subtract the strong T_1 from the total signal, and with $T - T_1 = T_2$, and hence also get T_2.

- These are just two examples of how to get more than you expect.



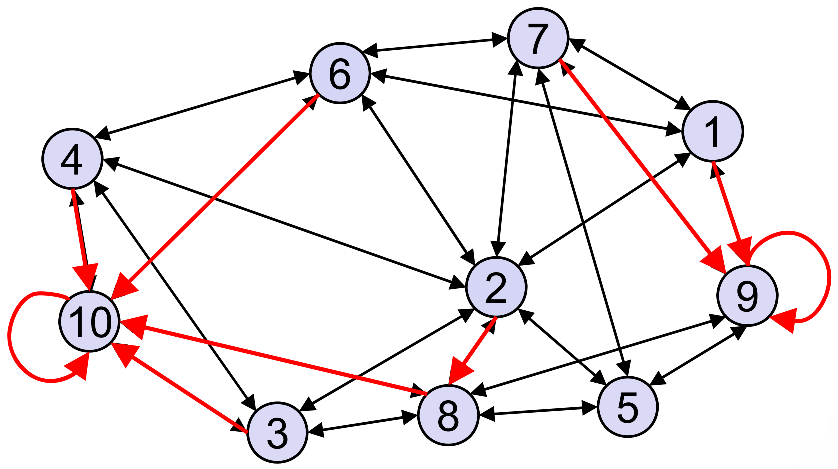
Overview of some models



- Try to proof correctness in an as “high” as possible model
- For efficiency, a more optimistic (“lower”) model might be fine
- Lower bounds should be proved in low models

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 (extremely local!)

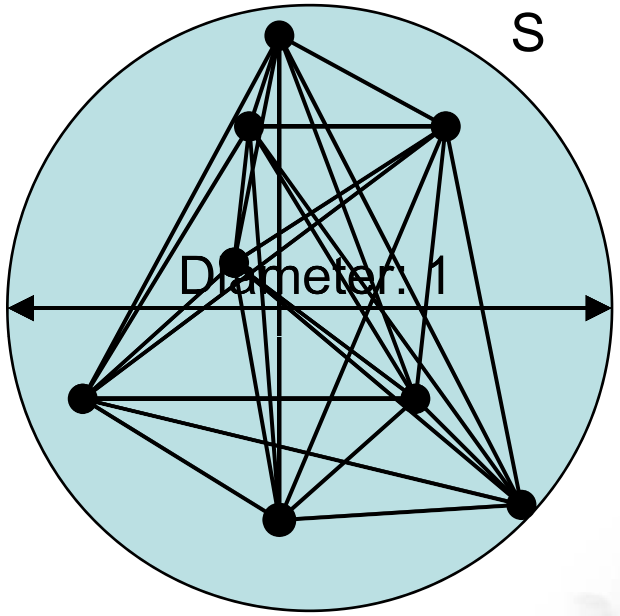


“Largest ID” Algorithm, Analysis I

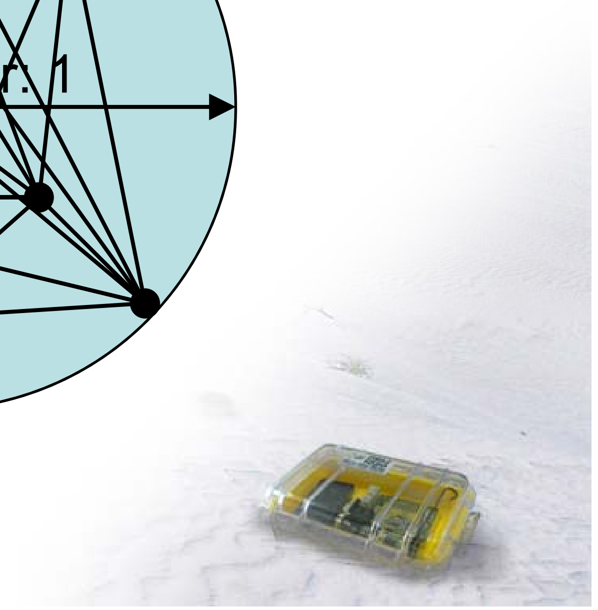
- To simplify analysis: assume graph is UDG
  (same analysis works for UBG based on doubling metric)

- We look at a disk S of diameter 1:

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

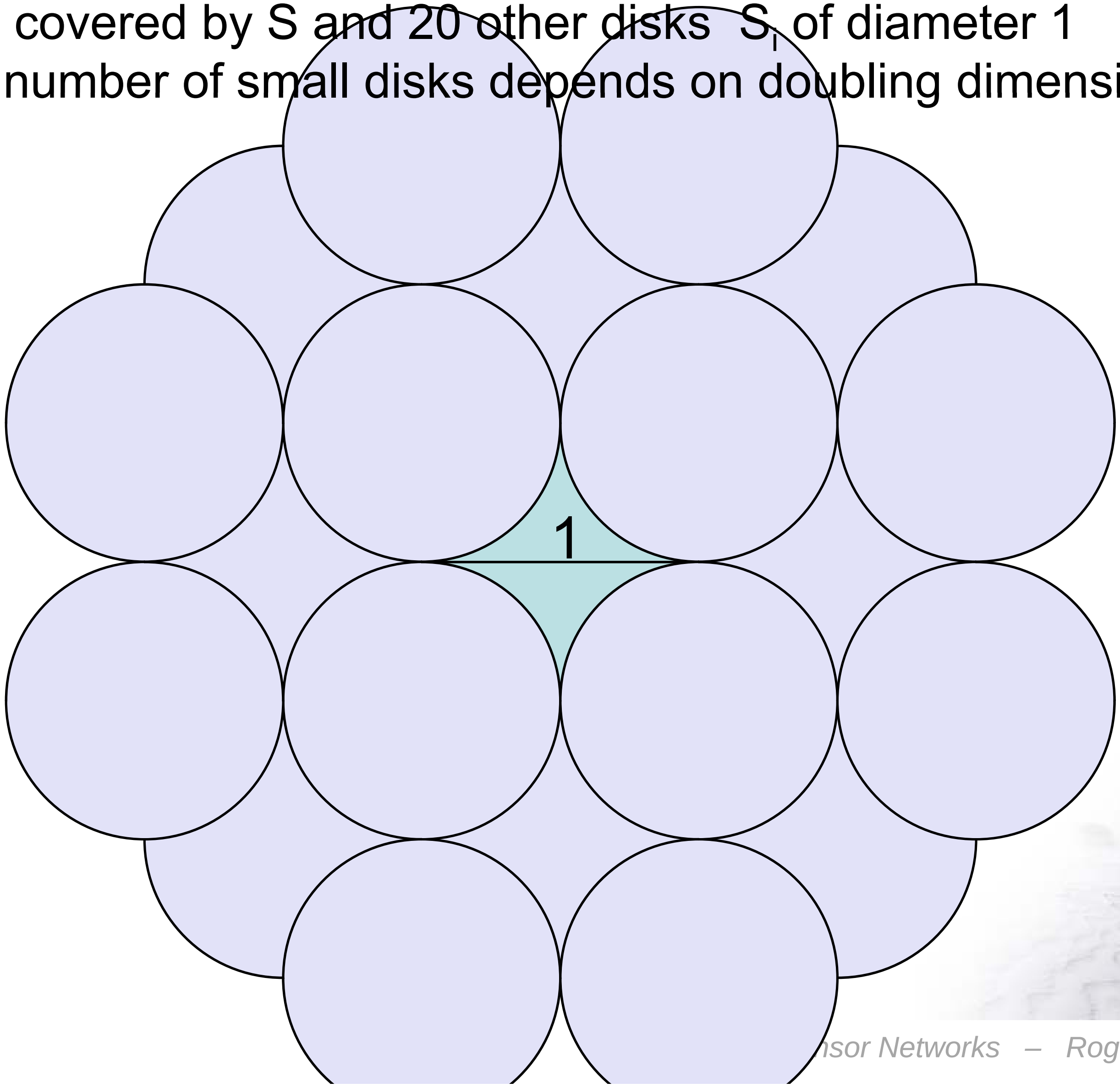


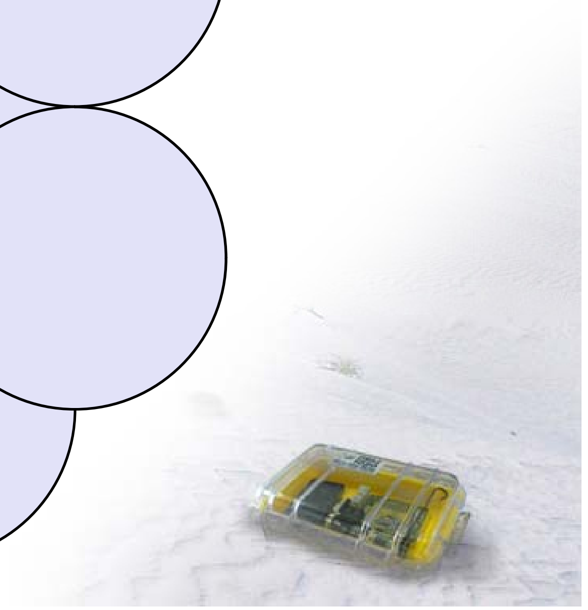
How many nodes in S are selected for the DS?



“Largest ID” Algorithm, Analysis II

- 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)



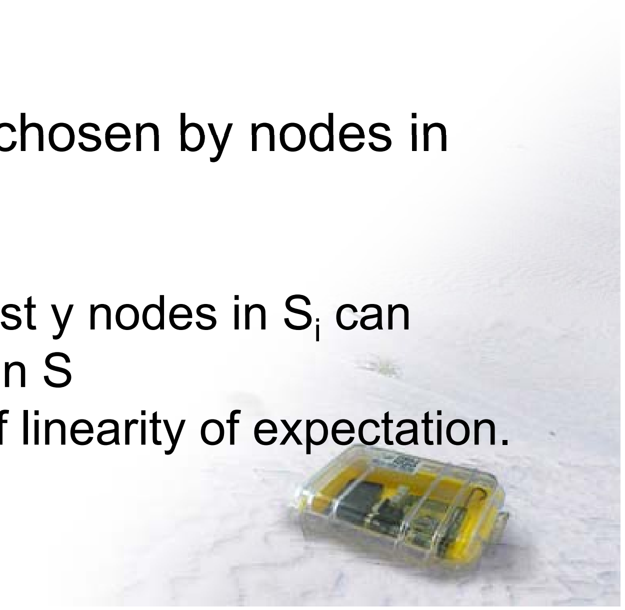


“Largest ID” Algorithm: Analysis III

- How many nodes in S are chosen by nodes in a disk S_i?
- x = # of nodes in S, y = # of nodes in 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: $\dfrac{1}{1+y}$
- Therefore, the expected number of nodes in S chosen by nodes in S_i is at most:

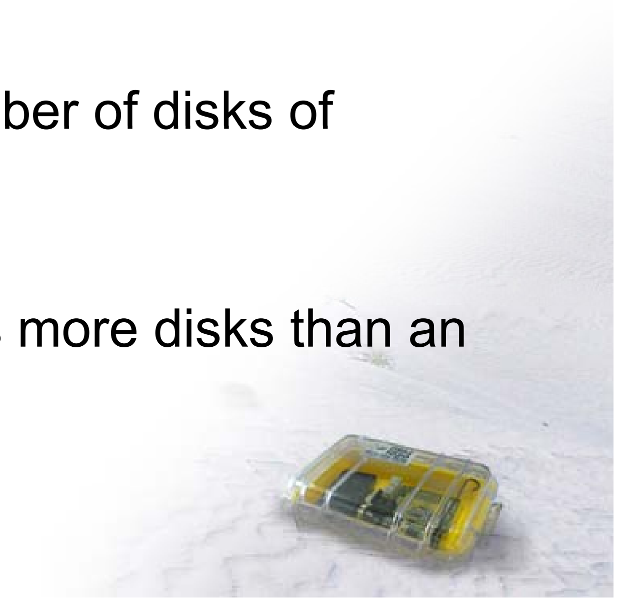
$$\min\left\{y, \frac{x}{1+y}\right\}$$

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



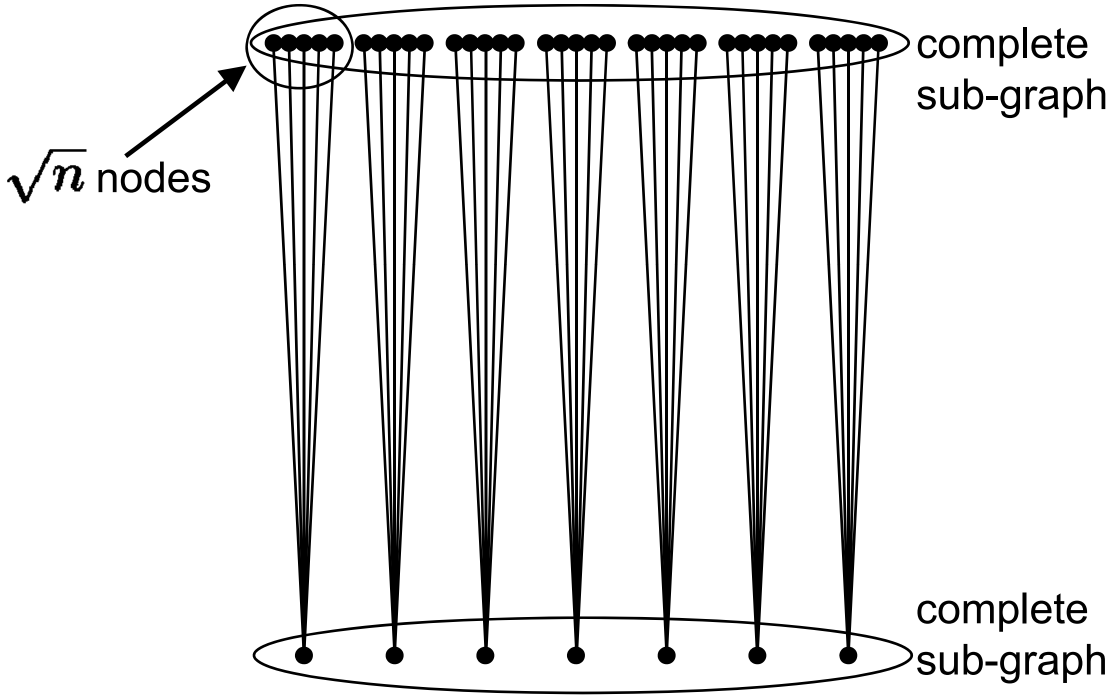
“Largest ID” Algorithm, Analysis IV

- From $x \leq n$ and $y \leq n$, it follows that: $\min\left\{y, \frac{x}{1+y}\right\} \leq \sqrt{n}$
- 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” alg:

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

Iterative "Largest ID" Algorithm

- Assume that nodes know the distances to their neighbors:

```
all nodes are active;
for i := k to 1 do
    $\forall$ act. nodes: select act. node with largest ID in dist. $\leq 1/2^i$;
    selected nodes remain active
od;
DS = set of active nodes
```

- Set of active nodes is always a DS (computing CDS also possible)
- Number of rounds: k
- Approximation ratio $n^{(1/2^k)}$

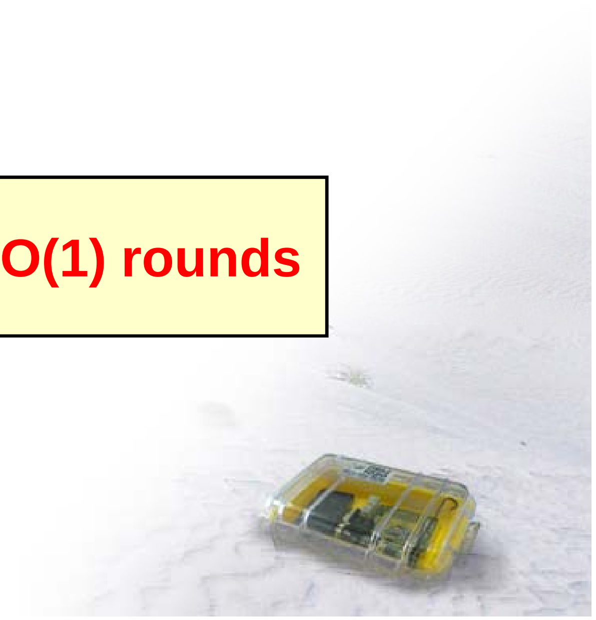
- For k=O(loglog *n*), approximation ratio = O(1)

Iterative “Largest ID” Algorithm, Remarks

- Possible to do everything in O(1) rounds
  (messages get larger, local computations more complicated)

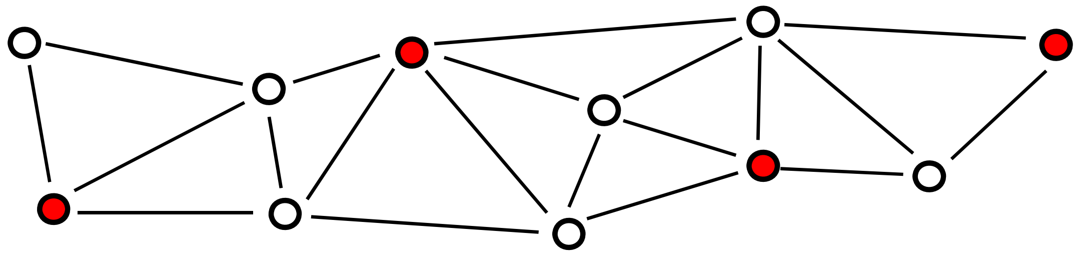
- If we slightly change the algorithm such that largest radius is 1/4:
  - Sufficient to know IDs of all neighbors, distances to neighbors, and distances between adjacent neighbors
  - Every node can then locally simulate relevant part of algorithm to find out whether or not to join DS

UBG w/ distances: O(1) approximation in O(1) rounds



Maximal Independent Set I

- Maximal Independent Set (MIS):
  (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

Maximal Independent Set II

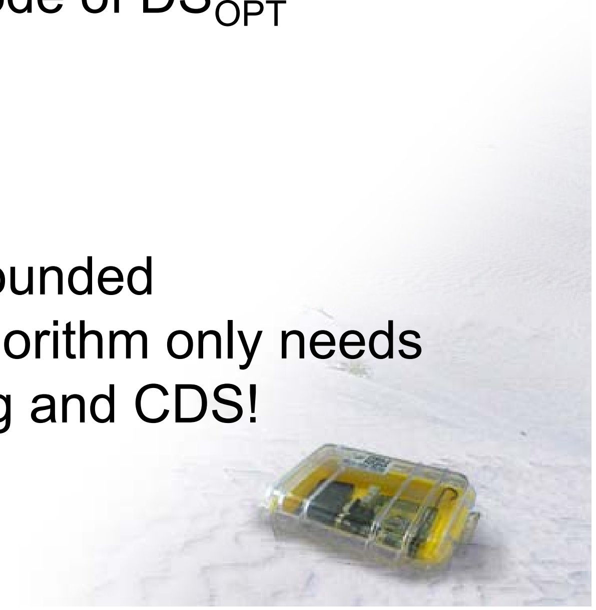
- Lemma:

  On bounded independence graphs: **$|MIS| \leq O(1) \cdot |DS_{OPT}|$**

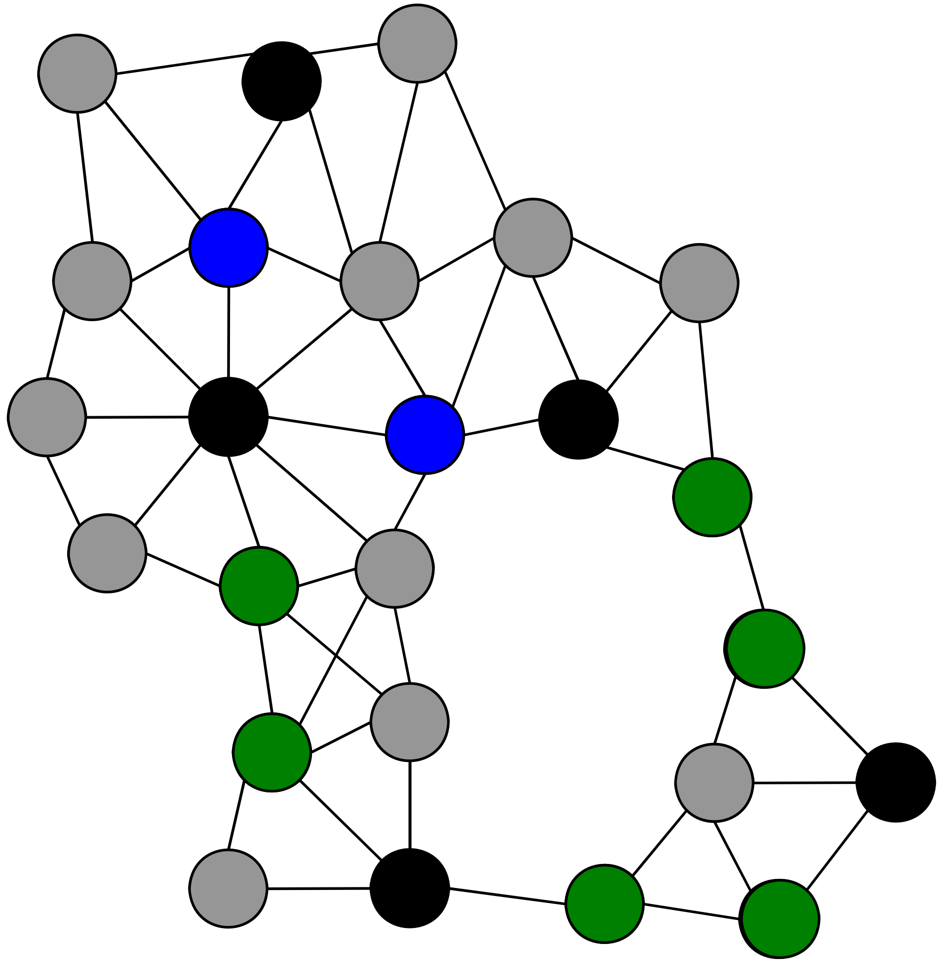
- Proof:
  1. Assign every MIS node to an adjacent node of DS_{OPT}
  2. $u \in DS_{OPT}$ has at most f(1) neighbors $v \in MIS$
  3. At most f(1) MIS nodes assigned to every node of DS_{OPT}

  $\rightarrow |MIS| \leq f(1) \cdot |DS_{OPT}|$

- Recently a lot of progress to compute MIS on bounded independece graphs. Indeed the best known algorithm only needs O(log* *n*) time! This algorithm also gives coloring and CDS!



MIS (DS) → CDS



MIS gives a dominating set.
But it is not connected.

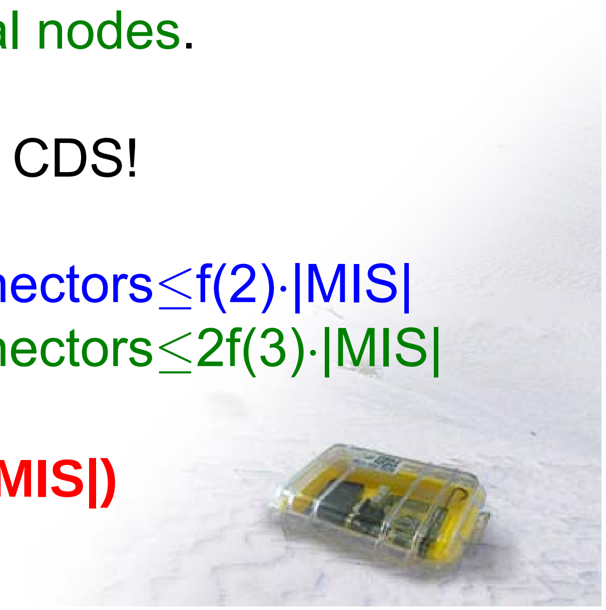
Connect any two MIS nodes
which can be connected by
one additional node.

Connect unconnected MIS nodes
which can be conn. by two
additional nodes.

This gives a CDS!

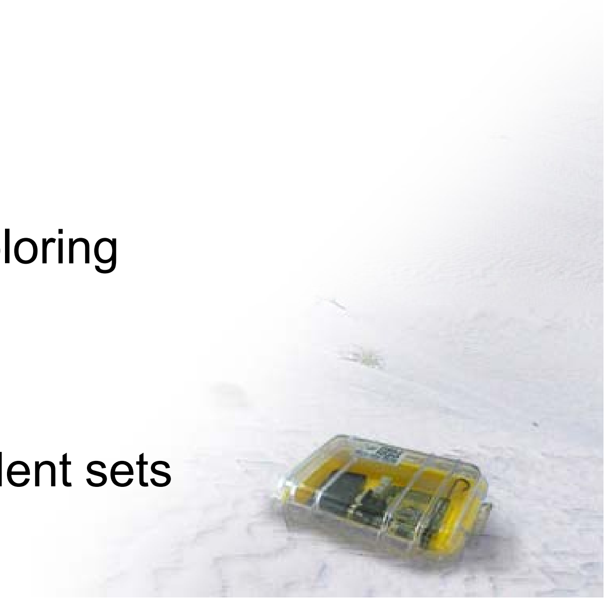
#2-hop connectors$\leq$f(2)·|MIS|
#3-hop connectors$\leq$2f(3)·|MIS|

|CDS| = O(|MIS|)

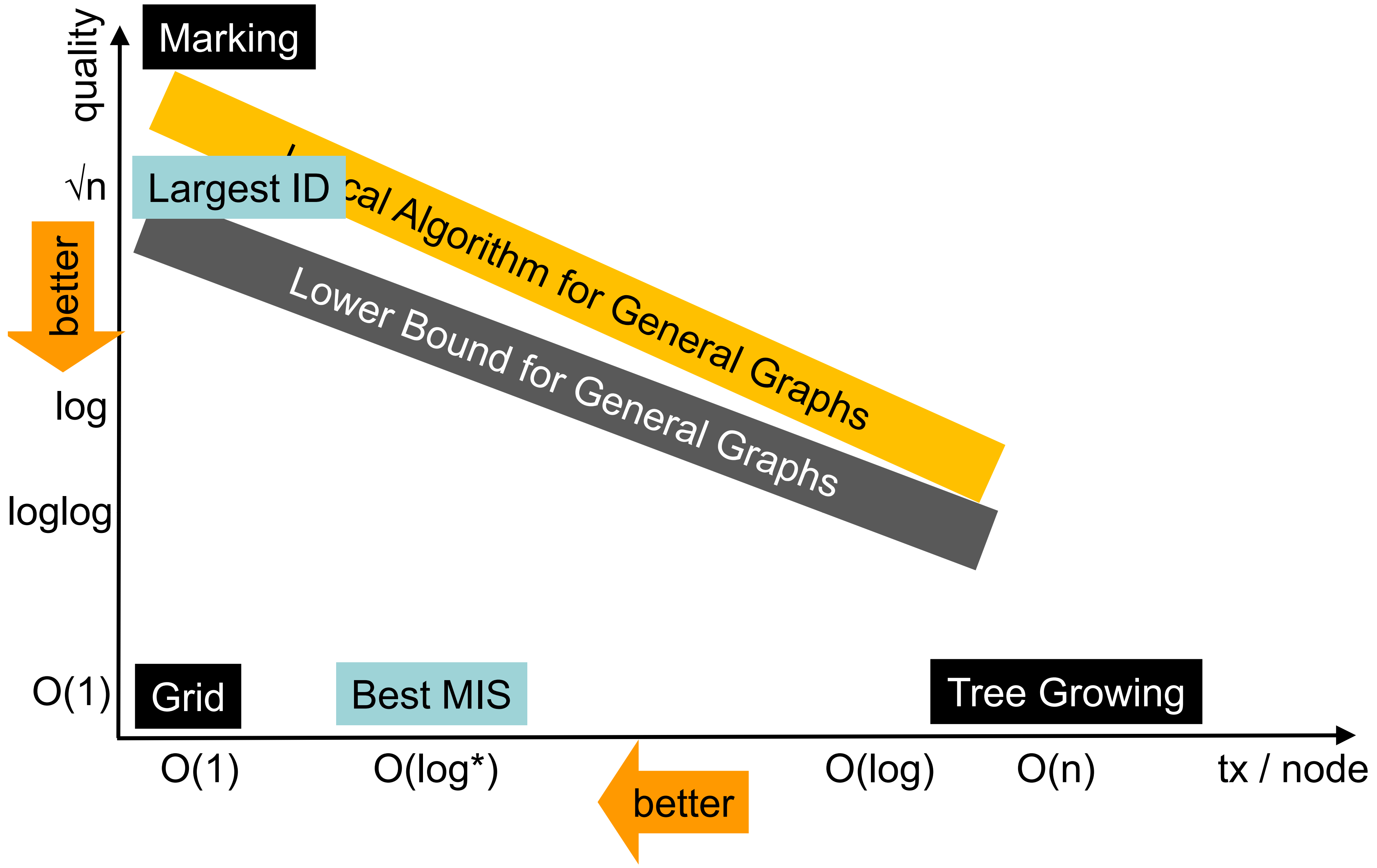


Some other interesting structures from graph theory

- We have already seen
  - (Connected) Dominating Set
  - Maximal Independent Set
- Maximum Independent Set (MaxIS)
  - An independent set with maximal cardinalty
  - Essentially „impossible“ to compute unless P ≈ NP
- $(\Delta+1)$- or $O(\Delta)$-Coloring
  - Can be used for MAC layer
  - A MIS algorithm can be misused to compute a coloring
- (Connected) Domatic Partition
  - Is to dominating sets what coloring is to independent sets

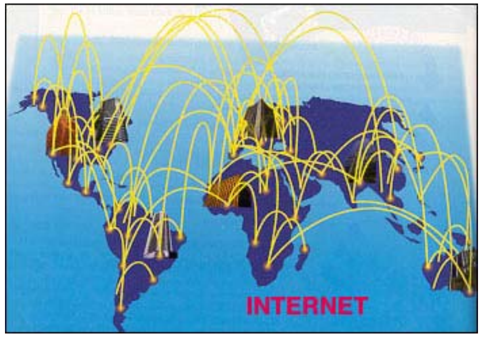


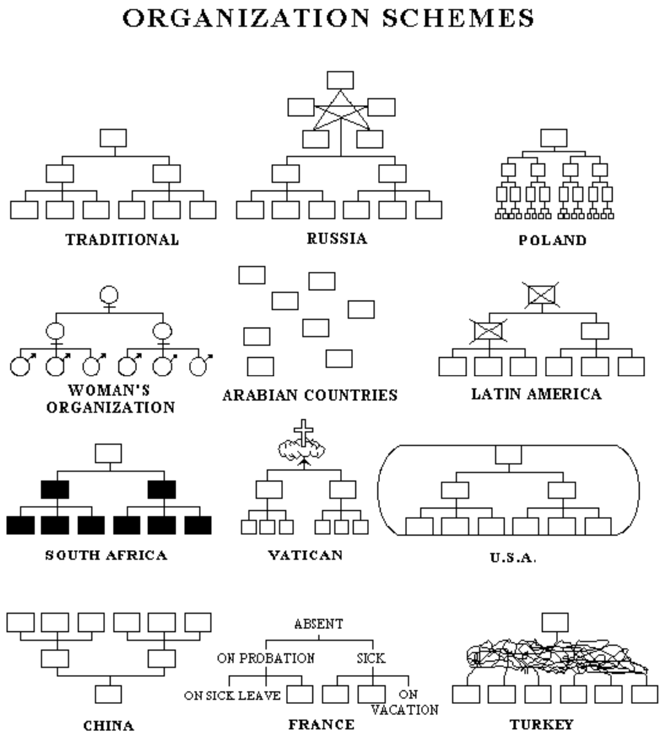
I would use one of these two algorithms...



A Theory of “Locality”?

- Ad hoc and sensor networks
- The largest network in the world?!?





- Managing organizations? Society?!?

A Simple “Localized” Algorithm

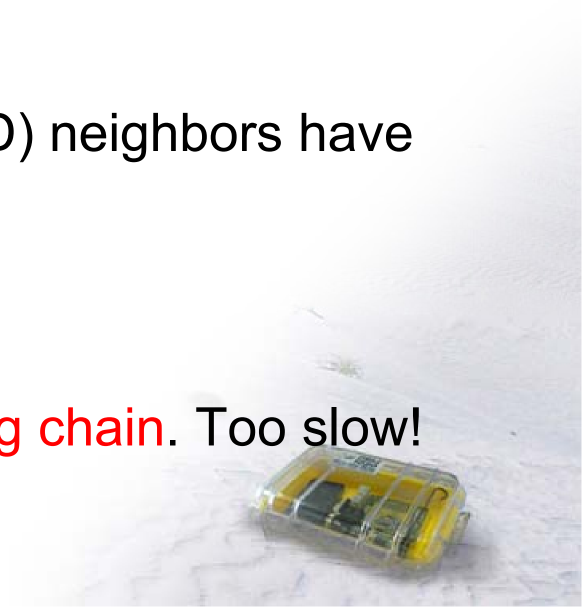
- Classic greedy algorithm:

- “Always choose node with most non-dominated neighbors.”
- The solution is a log-approximation (which is asymptotically optimal, unless P $\approx$ NP).

- Distributed version:

1. Wait until higher-degree (same degree: higher-ID) neighbors have decided not to join dominating set.
2. Join dominating set and tell neighbors.

- Problem: This algorithm can have a linear waiting chain. Too slow!

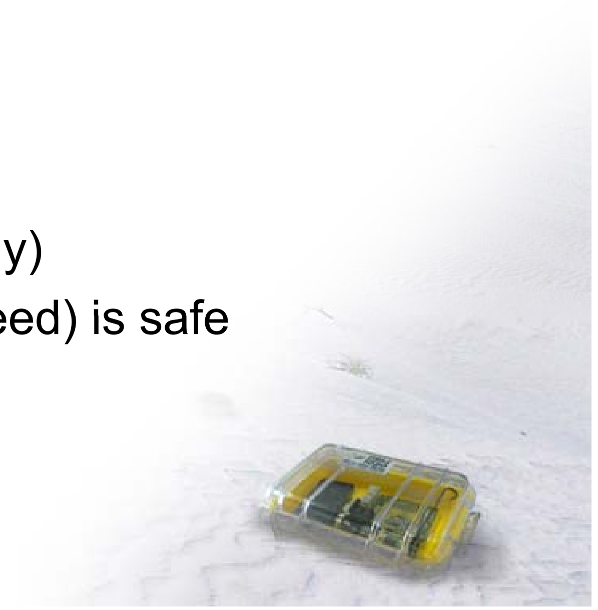


Are Localized/Local Algorithms Practical?!?

- Localized algorithm: Causality chain, butterfly effect

- Local algorithm: Synchronous communication rounds
  - Quite high demand to MAC layer
  - In reality messages get lost, due to fading, noise, and interference
  - In reality not all neighbors receive a message (hidden terminal problem)
  - In reality nodes might crash and restart (shabby power supply)

- Smells like self-stabilization
  - Messages might get lost, duplicated, or corrupted
  - Node memory/state might get corrupted (RAM only)
  - However, ROM (program, initialization, random seed) is safe



How to turn any local into a self-stabilizing algorithm

- Local algorithm:

- Initialize (local) variables
- Phase
  - Compute message from variables
  - Transmit message
  - Receive messages from neighbors
  - Recompute variables
  - Decision? If not → go to next phase

Receive	Variables	Transmit
-	Init	Out0
In1	Phase1	Out1
In2	Phase2	Out2
…	…	…

- Self-stabilizing algorithm:

- Simply keep transmitting <Out0,Out1,Out2, …> in one single message. (For many local algorithms, this message can be encoded to save space.)

- And keep checking whether your memory is still ok.

- It works! Adversarial memory corruptions are local only.

- [Awerbuch, Varghese, FOCS 91]

Algorithm Classes

$$\begin{tikzcd}
& \text{Global Algorithm} \arrow[d] & \\
& \text{Distributed Algorithm} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Local} & \text{Localized} & \text{Unstructured}
\end{tikzcd}$$

For some problems we don't even understand the non-distributed case

"Reiceive msg X → Transmit msg Y"
Every global algo can be distributed

Local

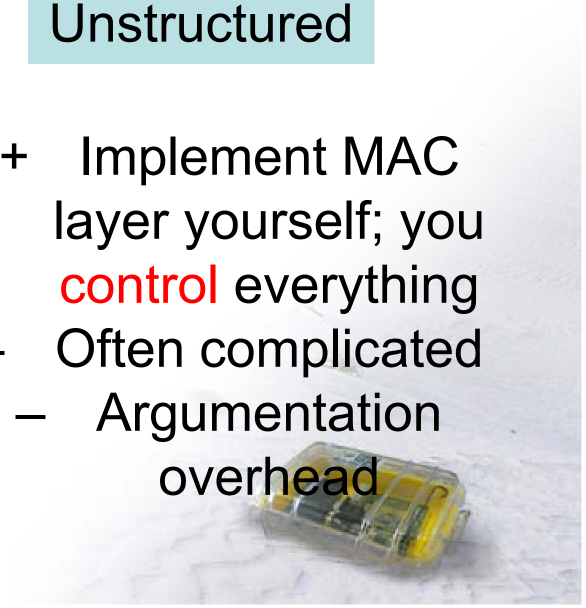
+ Node can only communicate with neighbors k times.
+ Strict time bounds
– Synchronous model

Localized

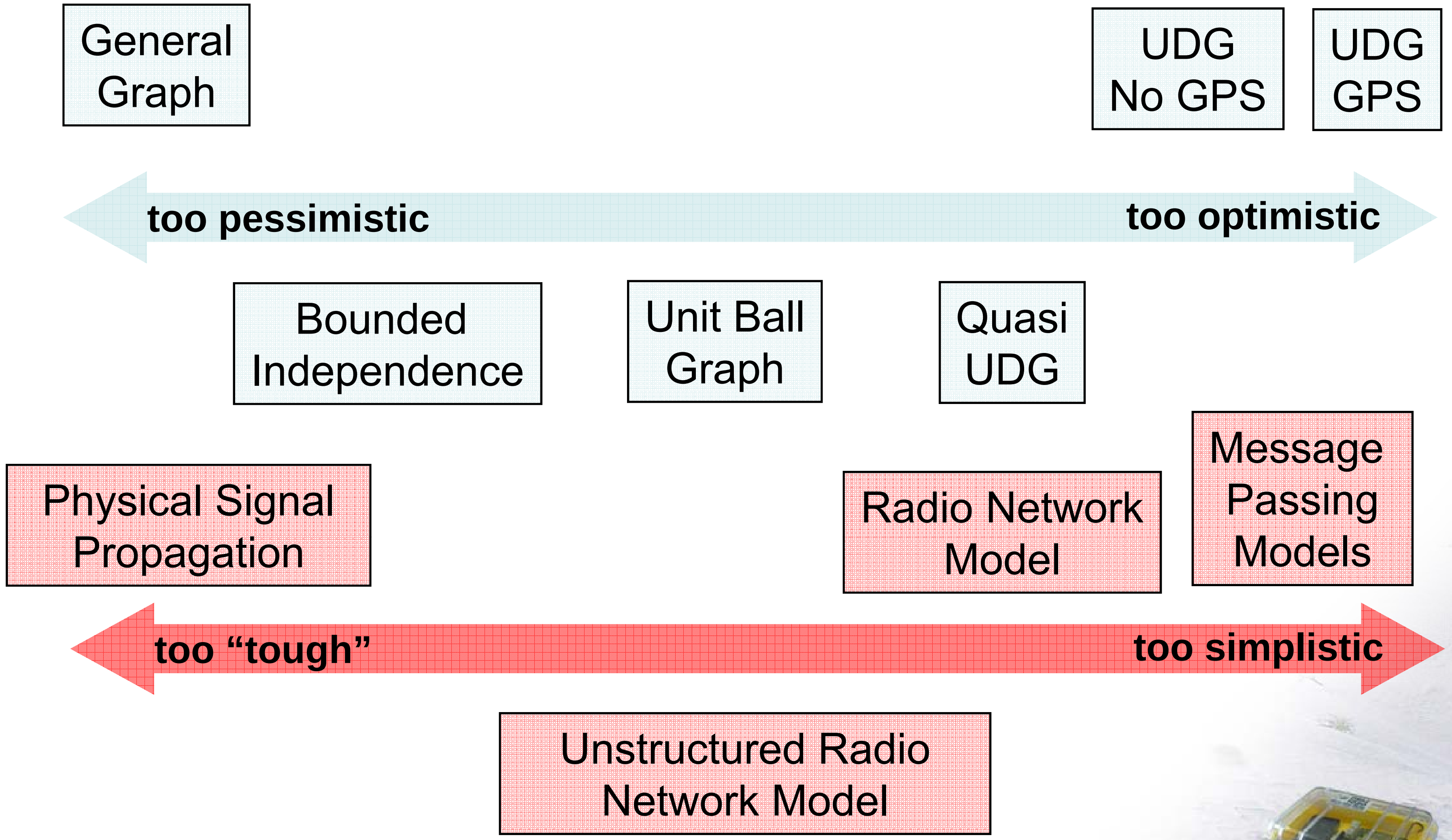
+ Often simple
– Nodes can wait for neighbor actions
– Often linear chain of causality

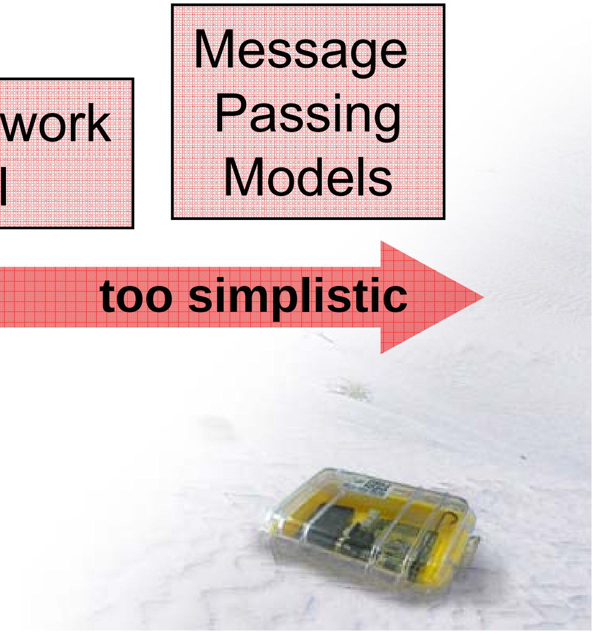
Unstructured

+ Implement MAC layer yourself; you control everything
– Often complicated
– Argumentation overhead



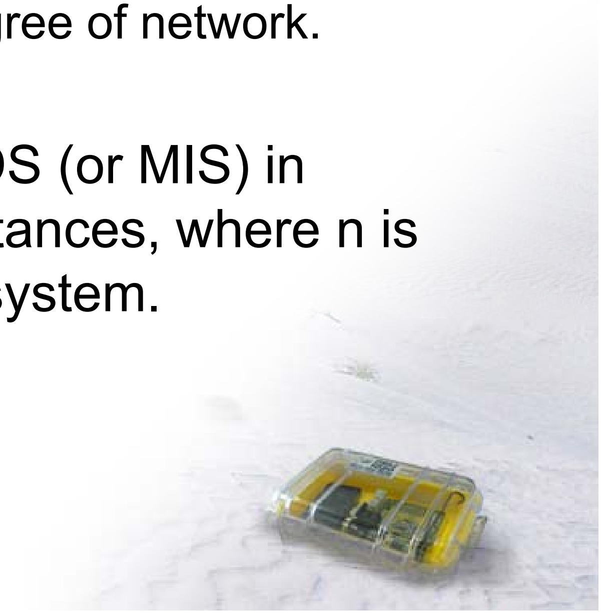
Another Model Dimension



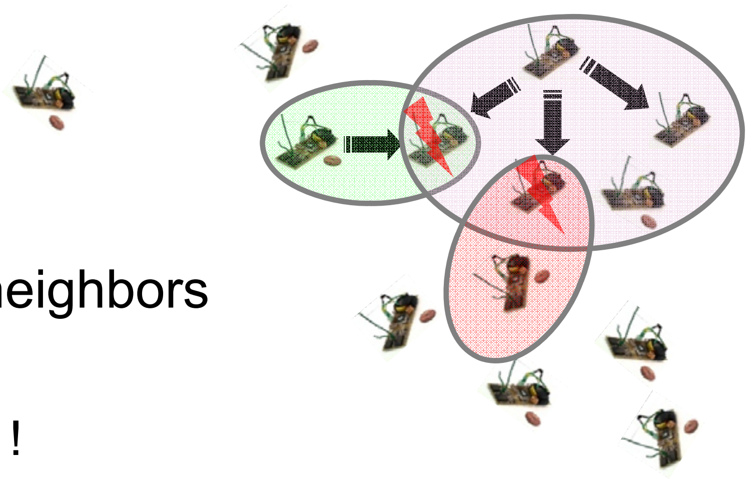


Clustering for Unstructured Radio Networks

- "Big Bang" (deployment) of a sensor and/or ad-hoc network:
  - Nodes wake up asynchronously (very late, maybe)
  - Neighbors unknown
  - Hidden terminal problem
  - No global clock
  - No established MAC protocol
  - No reliable collision detection
  - Limited knowledge of the number of nodes or degree of network.
- We have randomized algorithms that compute DS (or MIS) in polylog(n) time even under these harsh circumstances, where n is an upper bound on the number of nodes in the system.



Unstructured Radio Network Model



- Multi-Hop
- No collision detection
  - Not even at the sender!
- No knowledge about (the number of) neighbors
- Asynchronous Wake-Up
  - Nodes are not woken up by messages !
- Unit Disk Graph (UDG) to model wireless multi-hop network
  - Two nodes can communicate iff Euclidean distance is at most 1
- Upper bound n for number of nodes in network is known
  - This is necessary due to $\Omega(n / \log n)$ lower bound
    [Jurdzinski, Stachowiak, ISAAC 2002]

Unstructured Radio Network Model

- Can MDS and MIS be solved efficiently in such a harsh model?

There is a MIS algorithm with running time $O(\log^2 n)$ with high probability.

- And there is a matching lower bound.

Summary

- Sensor networks are an excellent application for distributed algorithms

- We need to study new network topologies
  - Network models between geometry and graph theory (BIG, UBG)
  - Interference models such as SINR

- We need to study new algorithmic paradigms
  - Distributed → Localized → Local → Self-Stabilizing → Unstructured

Thank You!

Questions & Comments?

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich