

# Ad hoc and Sensor Networks

## Topology control

# Goals of this chapter

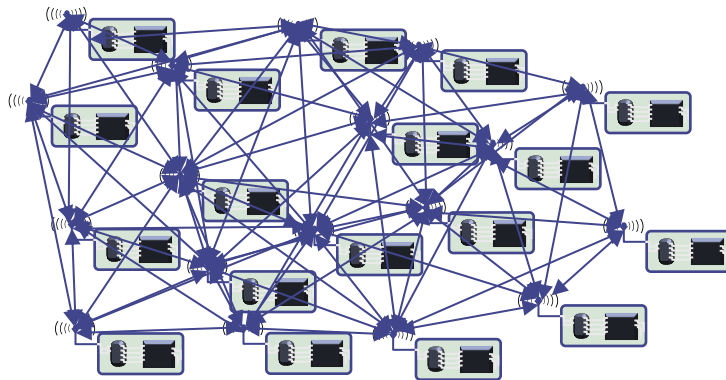
- Networks can be too dense – too many nodes in close (radio) vicinity
- This chapter looks at methods to deal with such networks by
  - Reducing/controlling transmission power
  - Deciding which links to use
  - Turning some nodes off
- Focus is on basic ideas, some algorithms
  - Complexity results are only very superficially covered

# Overview

- ***Motivation, basics***
- Power control
- Backbone construction
- Clustering
- Adaptive node activity

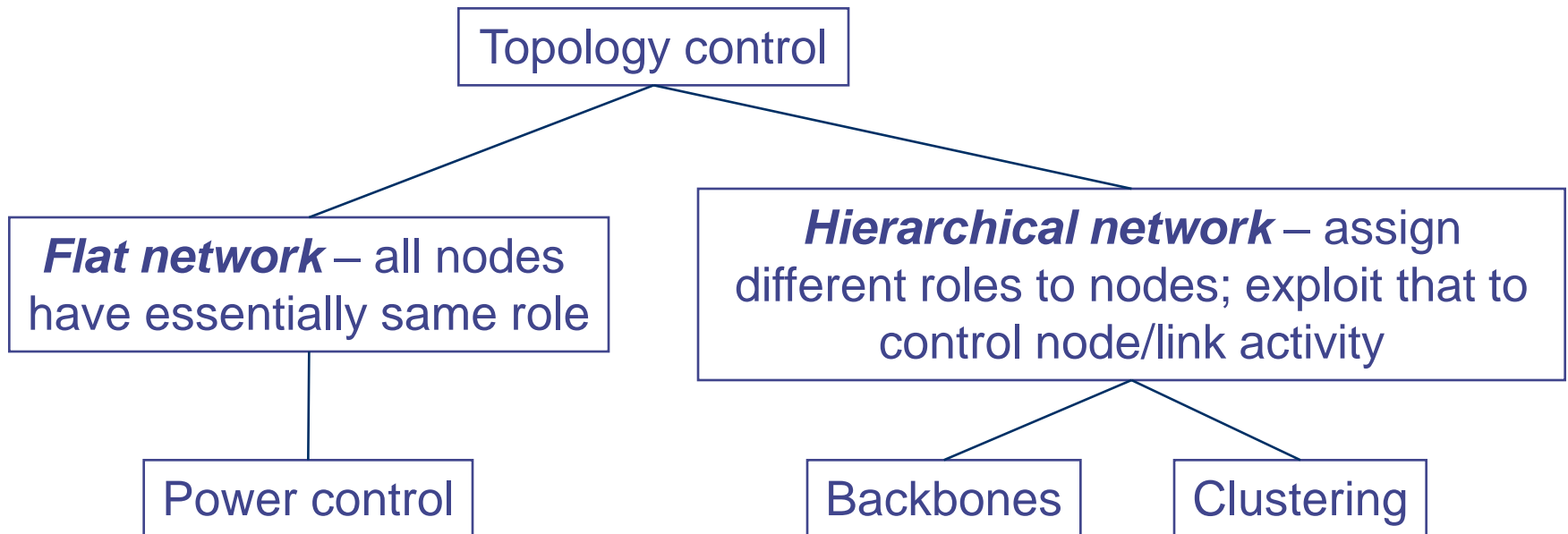
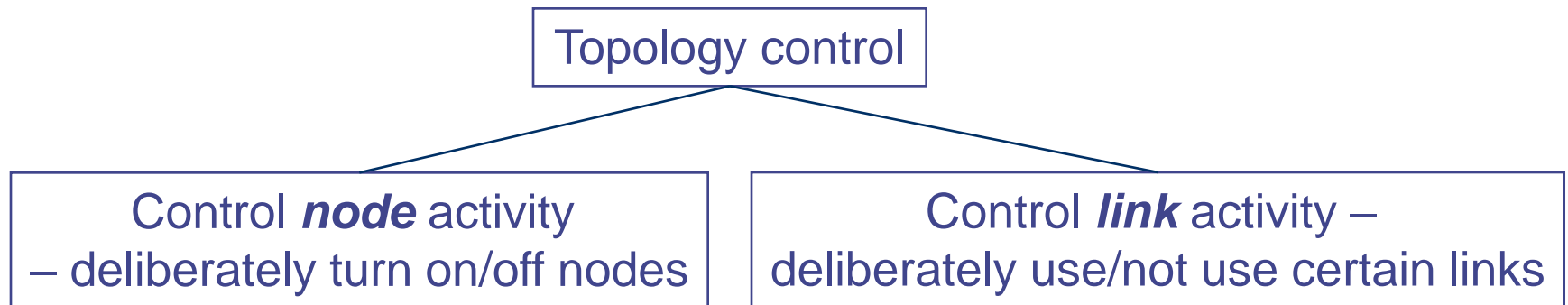
# Motivation: Dense networks

- In a very dense networks, too many nodes might be in range for an efficient operation
  - Too many collisions/too complex operation for a MAC protocol, too many paths to chose from for a routing protocol, ...



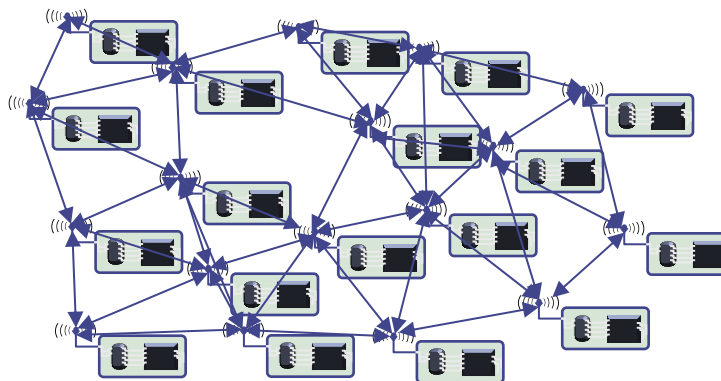
- Idea: Make *topology* less complex
  - **Topology**: Which node is able/allowed to communicate with which other nodes
  - Topology control needs to maintain invariants, e.g., connectivity

# Options for topology control



# Flat networks

- Main option: Control transmission power
  - Do not always use maximum power
  - Selectively for some links or for a node as a whole
  - Topology looks “thinner”
  - Less interference, ...

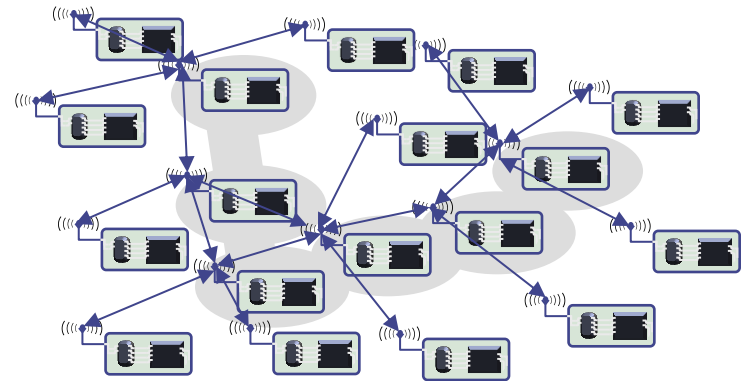


- Alternative: Selectively discard some links
  - Usually done by introducing hierarchies

# Hierarchical networks – backbone

- Construct a **backbone** network

- Some nodes “control” their neighbors – they form a (minimal) **dominating set**
- Each node should have a controlling neighbor
- Controlling nodes have to be connected (backbone)
- Only links within backbone and from backbone to controlled neighbors are used



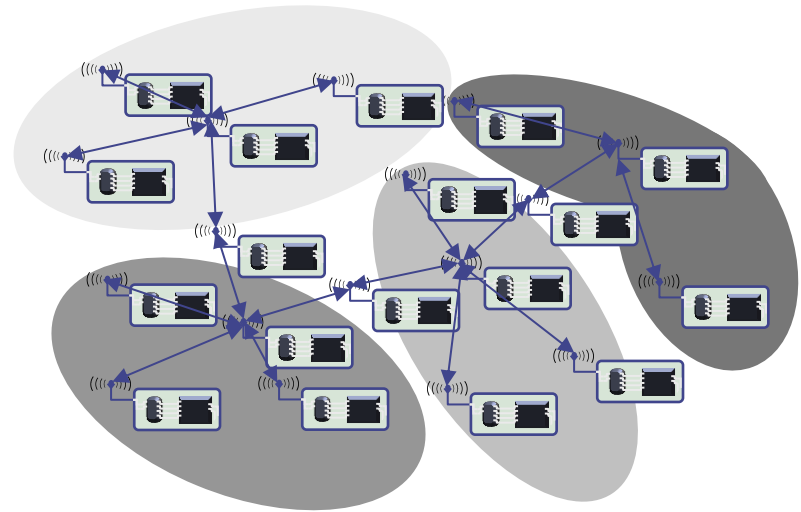
- Formally: Given graph  $G=(V,E)$ , construct  $D \subseteq V$  such that

$$\forall v \in V : v \in D \vee \exists d \in D : (v, d) \in E$$

# Hierarchical network – clustering

- Construct **clusters**

- Partition nodes into groups (“clusters”)
- Each node in exactly one group
  - Except for nodes “bridging” between two or more groups
- Groups can have **clusterheads**
- Typically: all nodes in a cluster are direct neighbors of their clusterhead
- Clusterheads are also a dominating set, but should be separated from each other – they form an **independent set**



- Formally: Given graph  $G=(V,E)$ , construct  $C \subseteq V$  such that

$$\forall v \in V - C : \exists c \in C : (v, c) \in E$$

$$\forall c_1, c_2 \in C : (c_1, c_2) \notin E$$

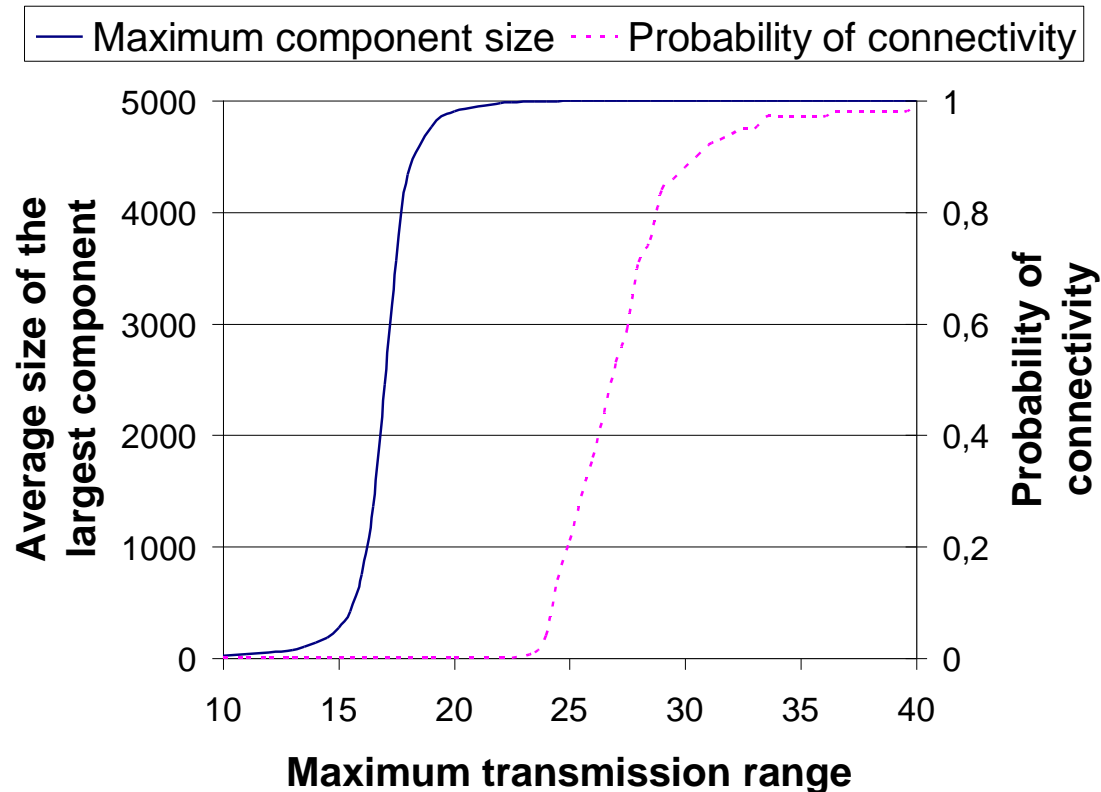


# Aspects of topology-control algorithms

- **Connectivity** – If two nodes connected in  $G$ , they have to be connected in  $G^0$  resulting from topology control
- **Stretch factor** – should be small
  - **Hop stretch factor**: how much longer are paths in  $G^0$  than in  $G$ ?
  - **Energy stretch factor**: how much more energy does the most energy-efficient path need?
- **Throughput** – removing nodes/links can reduce throughput, by how much?
- Robustness to mobility
- Algorithm overhead

# Example: Price for maintaining connectivity

- Maintaining connectivity can be very “costly” for a power control approach
- Compare power required for connectivity compared to power required to reach a very big maximum component



# Overview

- Motivation, basics
- ***Power control***
- Backbone construction
- Clustering
- Adaptive node activity

# Power control – magic numbers?

- Question: What is a good power level for a node to ensure “nice” properties of the resulting graph?
- Idea: Controlling transmission power corresponds to controlling the number of neighbors for a given node
- Is there an “optimal” number of neighbors a node should have?
  - Is there a “magic number” that is good irrespective of the actual graph/network under consideration?
- Historically,  $k=6$  or  $k=8$  had been suggested as such “magic numbers”
  - However, they optimize progress per hop – they do **not** guarantee connectivity of the graph!!
  - ! Needs deeper analysis

# Controlling transmission range

- Assume all nodes have identical transmission range  $r=r(|V|)$ , network covers area  $A$ ,  $V$  nodes, uniformly distr.
- Fact: Probability of connectivity goes to zero if:

$$r(|V|) \leq \sqrt{\frac{(1-\epsilon)A \log |V|}{\pi|V|}}, \text{ for any } \epsilon > 0$$

- Fact: Probability of connectivity goes to 1 for

$$r(|V|) \geq \sqrt{\frac{A(\log |V| + \gamma_{|V|})}{\pi|V|}}$$

if and only if  $\gamma_{|V|} \rightarrow 1$  with  $|V|$

- Fact (uniform node distribution, density  $\rho$ ):

$$P(G \text{ is } k\text{-connected}) \approx \left( 1 - \sum_{l=0}^{k-1} \frac{(\rho\pi r^2)^l}{l!} e^{-\rho\pi r^2} \right)$$

# Controlling number of neighbors

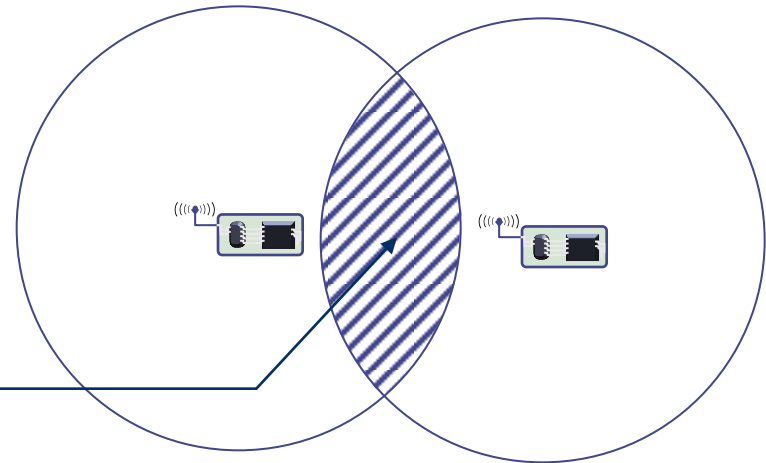
- Knowledge about range also tells about number of neighbors
  - Assuming node distribution (and density) is known, e.g., uniform
- Alternative: directly analyze number of neighbors
  - Assumption: Nodes randomly, uniformly placed, only transmission range is controlled, identical for all nodes, only symmetric links are considered
- Result: For connected network, required number of neighbors per node is  $\Theta(\log |V|)$ 
  - It is ***not a constant***, but depends on the number of nodes!
  - For a larger network, nodes need to have more neighbors & larger transmission range! – Rather inconvenient
  - Constants can be bounded

# Some example constructions for power control

- Basic idea for most of the following methods:  
Take a graph  $G=(V,E)$ , produce a graph  $G^0=(V,E^0)$  that maintains connectivity with fewer edges
  - Assume, e.g., knowledge about node positions
  - Construction should be local (for distributed implementation)

# Example 1: Relative Neighborhood Graph (RNG)

- Edge between nodes  $u$  and  $v$  if and only if there is no other node  $w$  that is closer to either  $u$  or  $v$
- Formally:  $\forall u, v \in V : (u, v) \in E'$  iff
$$\nexists w \in V : \max\{d(u, w), d(v, w)\} < d(u, v)$$
- RNG maintains connectivity of the original graph
- Easy to compute locally
- But: Worst-case spanning ratio is  $\Omega(|V|)$
- Average degree is 2.6



This region has to be empty for the two nodes to be connected

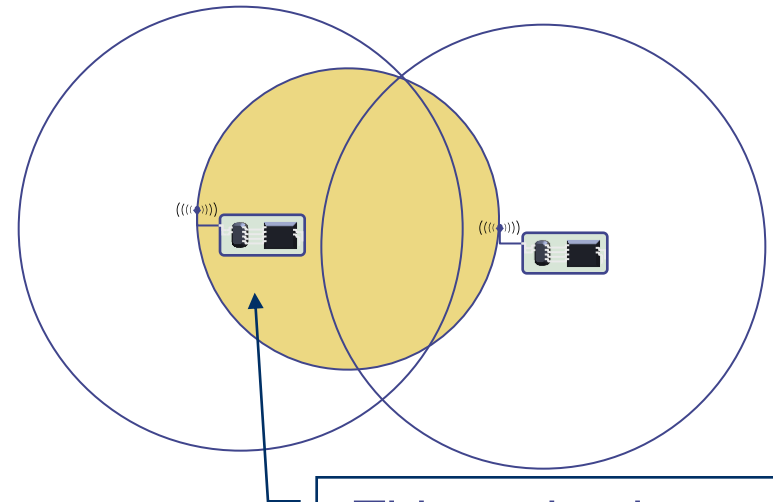


## Example 2: Gabriel graph

- Gabriel graph (GG) similar to RNG
- Difference: Smallest circle with nodes  $u$  and  $v$  on its circumference must only contain node  $u$  and  $v$  for  $u$  and  $v$  to be connected
- Formally:

$\forall u, v \in V : (u, v) \in E'$  iff

$$\nexists w \in V : d^2(u, w) + d^2(v, w) < d^2(u, v)$$



This region has to be empty for the two nodes to be connected

- Properties: Maintains connectivity, Worst-case spanning ratio  $\Omega(|V|^{1/2})$ , energy stretch  $O(1)$  (depending on consumption model!), worst-case degree  $\Omega(|V|)$

# Example 3: Delaunay triangulation

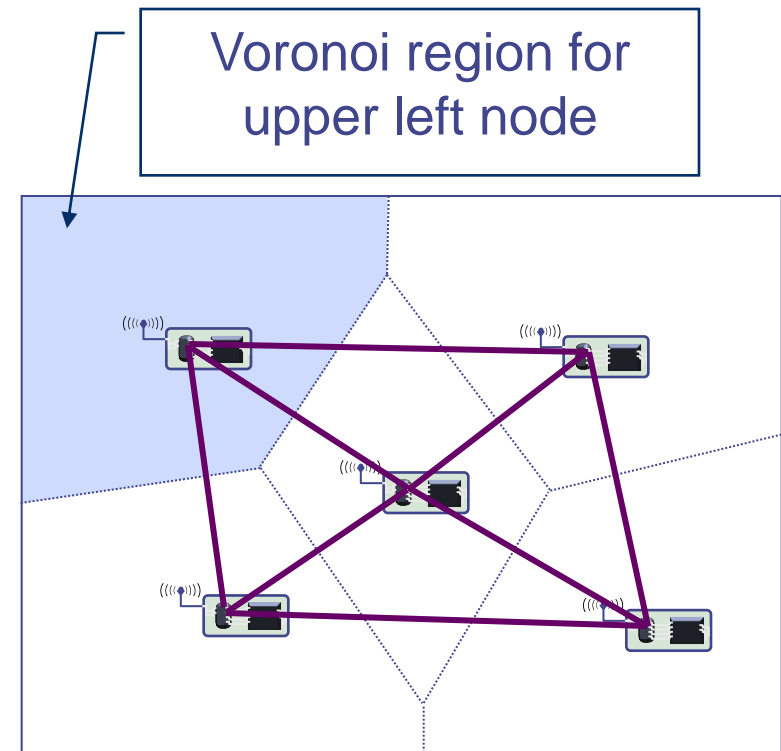
- Assign, to each node, all points in the plane for which it is the closest node

## ! *Voronoi diagram*

- Constructed in  $O(|V| \log |V|)$  time
- Connect any two nodes for which the Voronoi regions touch

## ! *Delaunay triangulation*

- Problem: Might produce very long links; not well suited for power control

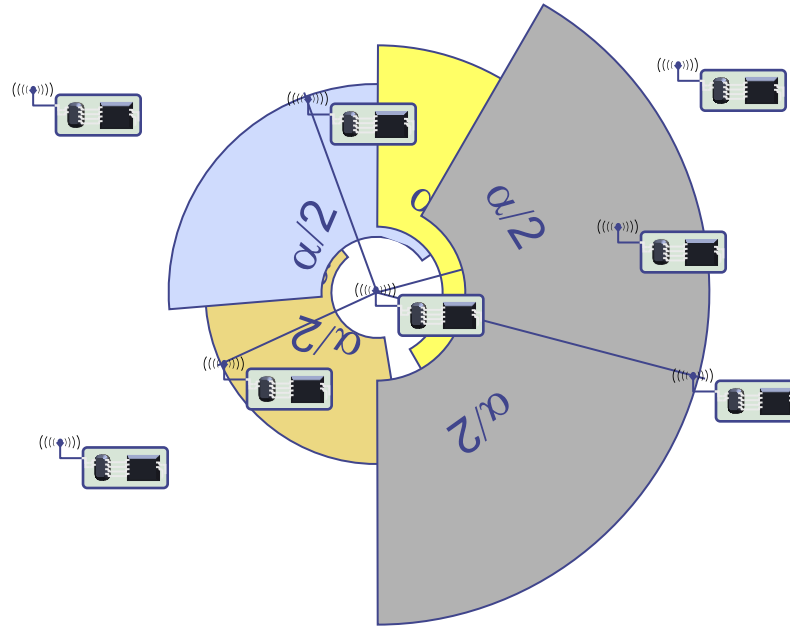


— Edges of Delaunay triangulation

# Example: Cone-based topology control

- Assumption: Distance and angle information between nodes is available
- Two-phase algorithm
- Phase 1
  - Every node starts with a small transmission power
  - Increase it until a node has sufficiently many neighbors
  - What is “sufficient”? – When there is at least one neighbor in each **cone** of angle  $\alpha$
  - $\alpha = 5/6\pi$  is necessary and sufficient condition for connectivity!
- Phase 2
  - Remove redundant edges: Drop a neighbor  $w$  of  $u$  if there is a node  $v$  of  $w$  and  $u$  such that sending from  $u$  to  $w$  directly is less efficient than sending from  $u$  via  $v$  to  $w$
  - Essentially, a local Gabriel graph construction

## Example: Cone-based topology control (2)



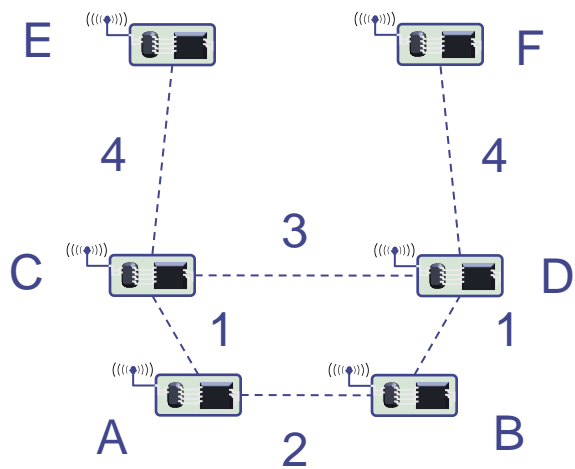
- Properties: simple, local construction
- Extensions for k-connectivity (Yao graph)
- Little exercise: What happens when  $\alpha <$  or  $>$   $5/6 \pi$ ?

# Centralized power control algorithm

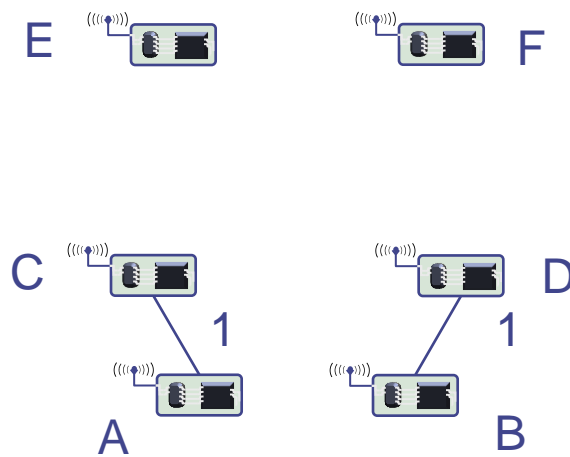
- Goal: Find topology control algorithm minimizing the *maximum* power used by any node
  - Ensuring simple or bi-connectivity
  - Assumptions: Locations of all nodes and path loss between all node pairs are known; each node uses an individually set power level to communicate with all its neighbors
- Idea: Use a centralized, greedy algorithm
  - Initially, all nodes have transmission power 0
  - Connect those two components with the shortest distance between them (raise transmission power accordingly)
- Second phase: Remove links (=reduce transmission power) not needed for connectivity
- Exercise: Relation to Kruskal's MST algorithm?

# Centralized power control algorithm

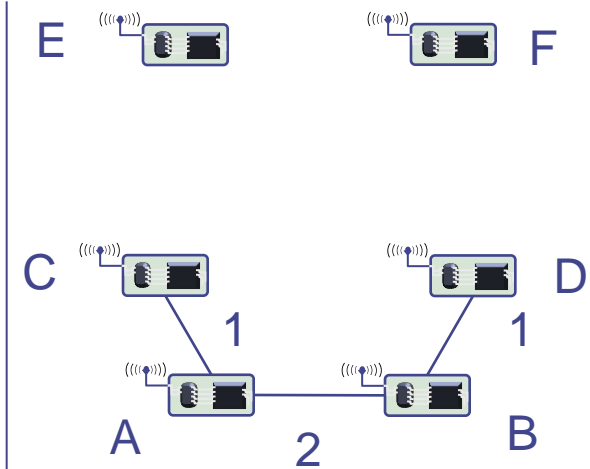
Topology



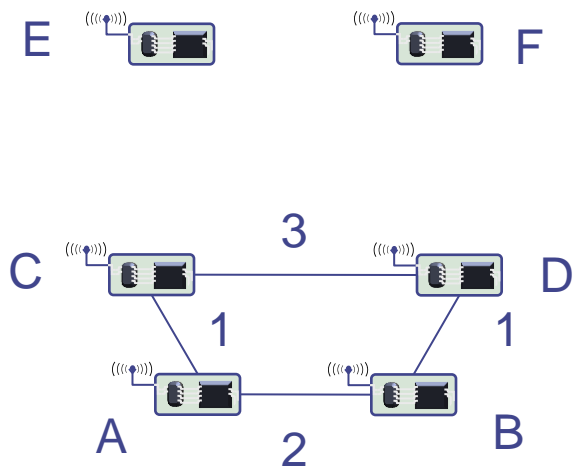
1) Connect A-C and B-D



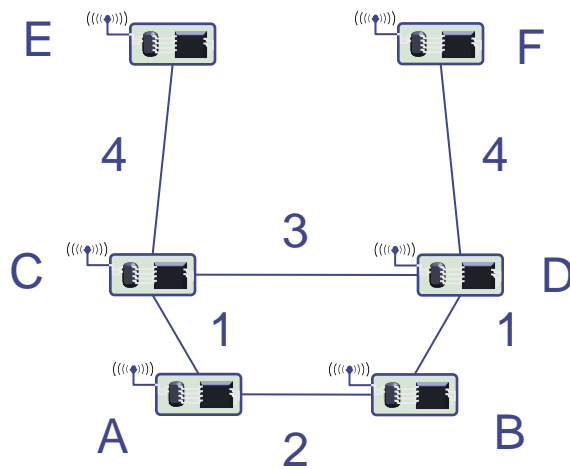
2) Connect A-B



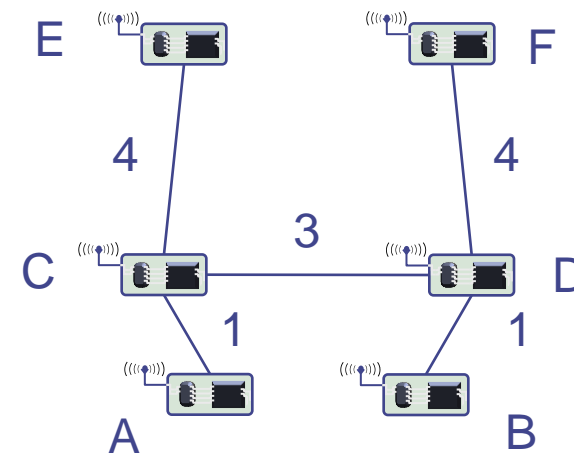
3) Connect C-D



4) Connect C-E and D-F



5) Remove edge A-B



# Overview

- Motivation, basics
- Power control
- ***Backbone construction***
- Clustering
- Adaptive node activity

# Hierarchical networks – backbones

- Idea: Select some nodes from the network/graph to form a ***backbone***
  - A connected, minimal, dominating set (MDS or MCDS)
  - Dominating nodes control their neighbors
  - Protocols like routing are confronted with a simple topology – from a simple node, route to the backbone, routing in backbone is simple (few nodes)
- Problem: MDS is an NP-hard problem
  - Hard to approximate, and even approximations need quite a few messages



# Backbone by growing a tree

- Construct the backbone as a tree, grown iteratively

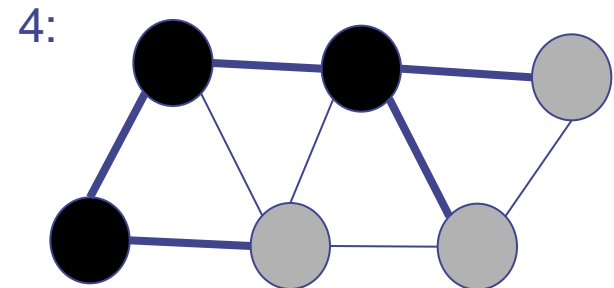
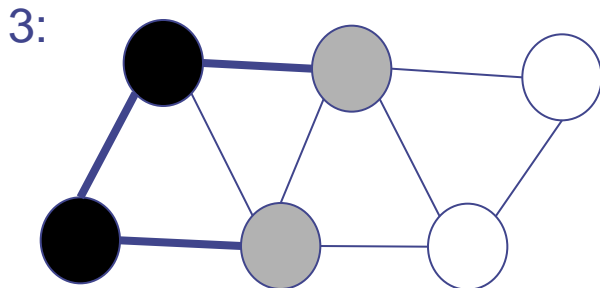
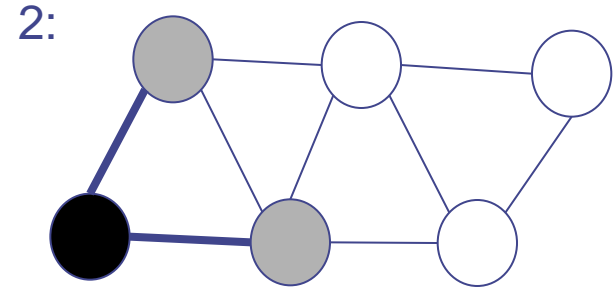
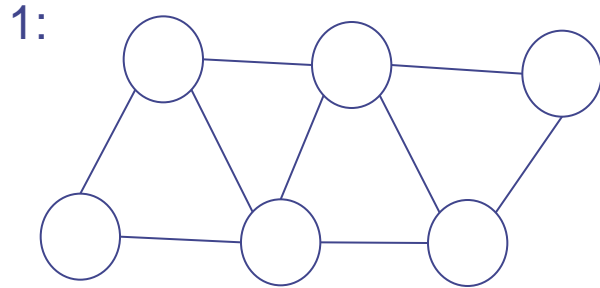
---

```
initialize all nodes' color to white
pick an arbitrary node and color it grey

while (there are white nodes) {
  pick a grey node v that has white neighbors
  color the grey node v black
  foreach white neighbor u of v {
    color u grey
    add (v,u) to tree T
  }
}
```

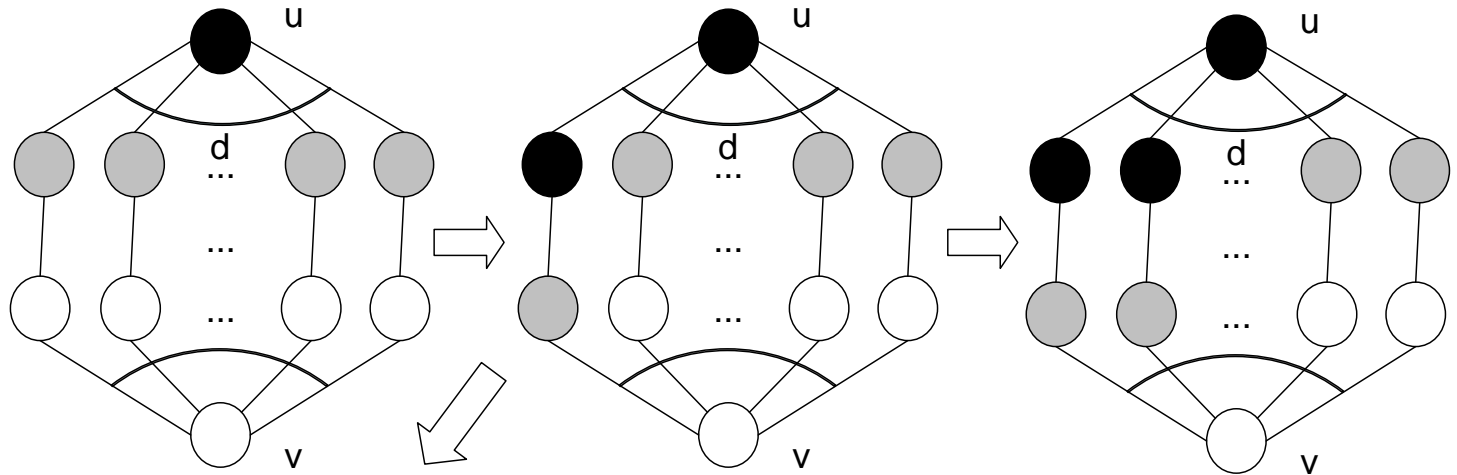
---

# Backbone by growing a tree – Example



# Problem: Which gray node to pick?

- When blindly picking any gray node to turn black, resulting tree can be very bad

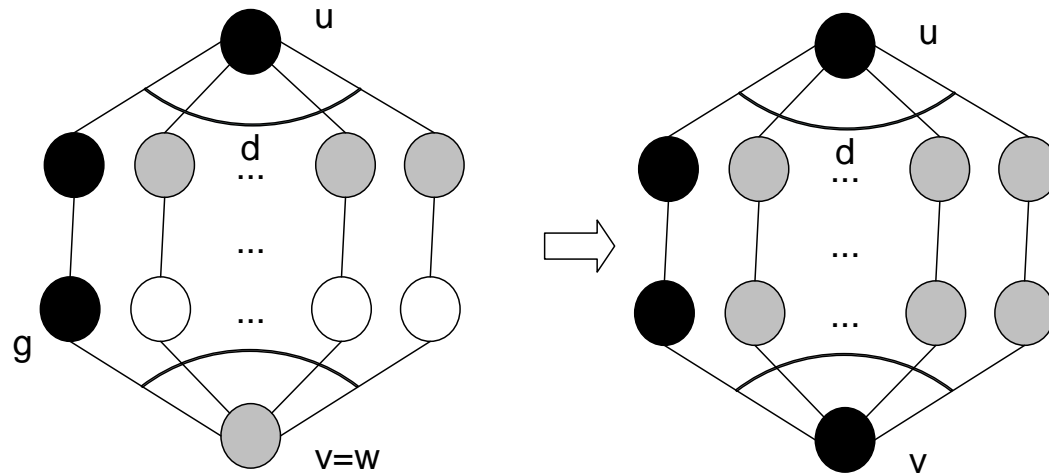


Solution:

Look ahead!

One step suffices

Look-ahead using nodes  $g$  and  $w$



# Performance of tree growing with look ahead

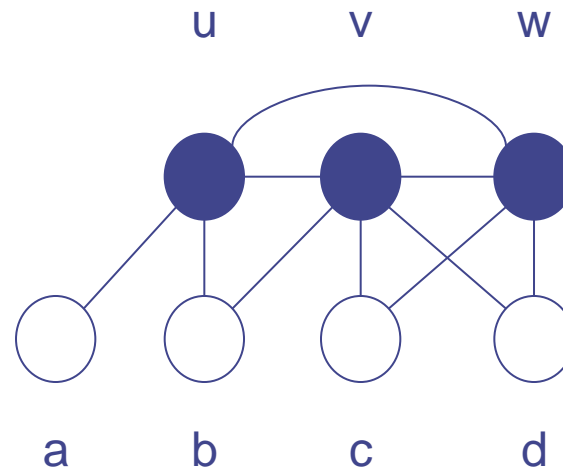
- Dominating set obtained by growing a tree with the look ahead heuristic is at most a factor  $2(1 + H(\Delta))$  larger than MDS
  - $H(k)$  harmonic function,  $H(k) = \sum_{i=1}^k 1/i \leq \ln k + 1$
  - $\Delta$  is maximum degree of the graph
- It is automatically connected
- Can be implemented in a distributed fashion as well

# Start big, make lean

- Idea: start with some, possibly large, connected dominating set, reduce it by removing unnecessary nodes
- Initial construction for dominating set
  - All nodes are initially white
  - Mark any node black that has two neighbors that are not neighbors of each other (they might need to be dominated)
  - ! Black nodes form a connected dominating set (proof by contradiction); shortest path between ANY two nodes only contains black nodes
- Needed: Pruning heuristics

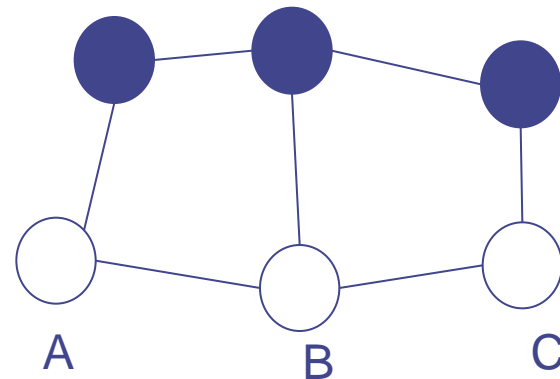
# Pruning heuristics

- Heuristic 1: Unmark node  $v$  if
  - Node  $v$  and its neighborhood are included in the neighborhood of some node marked node  $u$  (then  $u$  will do the domination for  $v$  as well)
  - Node  $v$  has a smaller unique identifier than  $u$  (to break ties)
- Heuristic 2: Unmark node  $v$  if
  - Node  $v$ 's neighborhood is included in the neighborhood of two marked neighbors  $u$  and  $w$
  - Node  $v$  has the smallest identifier of the tree nodes
- Nice and easy, but only linear approximation factor



# One more distributed backbone heuristic: Span

- Construct backbone, but take into account need to carry traffic – preserve capacity
  - Means: If two paths could operate without interference in the original graph, they should be present in the reduced graph as well
  - Idea: If the stretch factor (induced by the backbone) becomes too large, more nodes are needed in the backbone
- Rule: Each node observes traffic around itself
  - If node detects two neighbors that need three hops to communicate with each other, node joins the backbone, shortening the path
  - Contention among potential new backbone nodes handled using random backoff



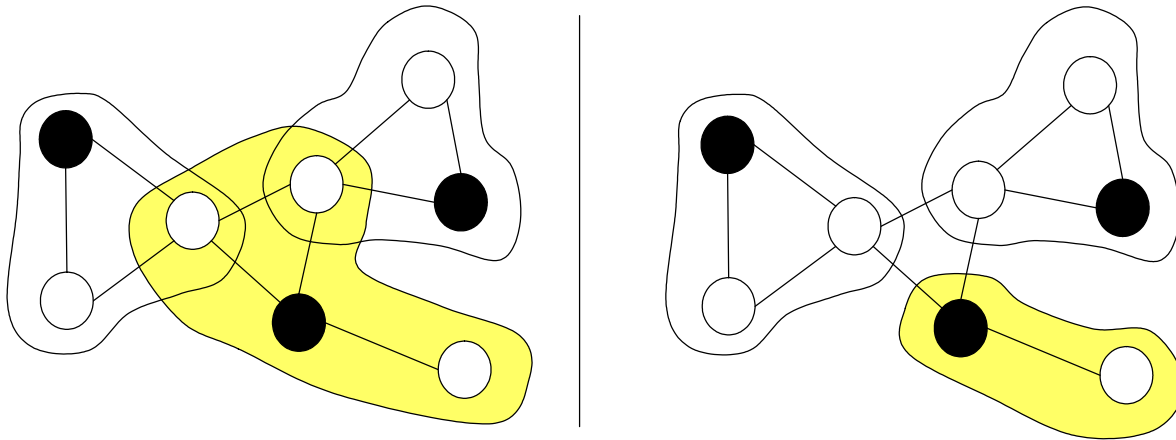
# Overview

- Motivation, basics
- Power control
- Backbone construction
- ***Clustering***
- Adaptive node activity



# Clustering

- Partition nodes into groups of nodes – **clusters**
- Many options for details
  - Are there **clusterheads**? – One controller/representative node per cluster
  - May clusterheads be neighbors? If no: clusterheads form an **independent set C**:  $\forall c_1, c_2 \in C : (c_1, c_2) \notin E$   
Typically: clusterheads form a **maximum independent set**
  - May clusters overlap? Do they have nodes in common?

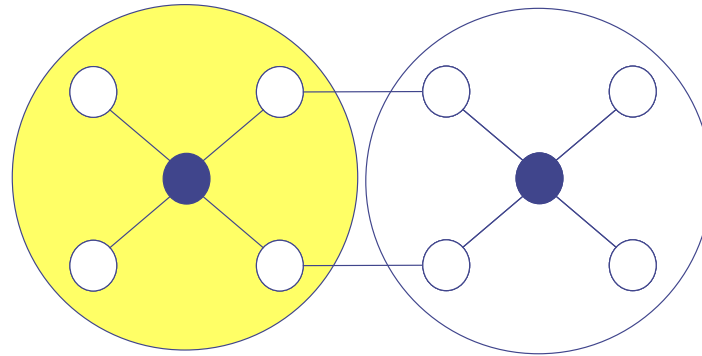


# Clustering

- Further options

- How do clusters communicate? Some nodes need to act as **gateways** between clusters

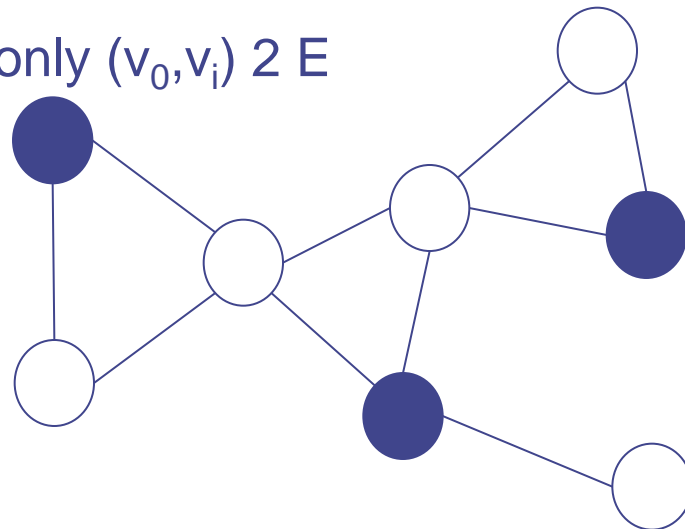
If clusters may not overlap, two nodes need to jointly act as a **distributed gateway**



- How many gateways exist between clusters? Are all active, or some standby?
- What is the maximal diameter of a cluster? If more than 2, then clusterheads are not necessarily a maximum independent set
- Is there a hierarchy of clusters?

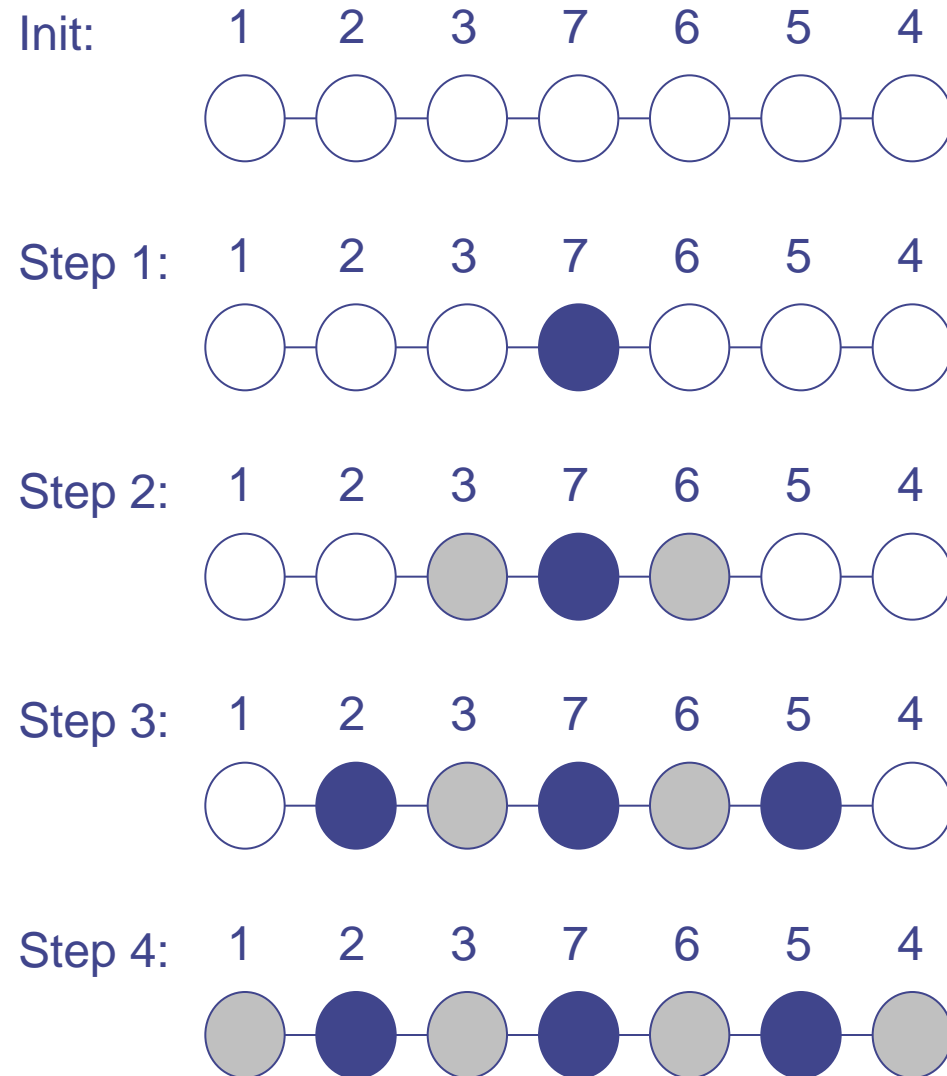
# Maximum independent set

- Computing a maximum independent set is NP-complete
- Can be approximate within  $(\Delta + 3)/5$  for small  $\Delta$ , within  $O(\Delta \log \log \Delta / \log \Delta)$  else;  $\Delta$  bounded degree
- Show: A maximum independent set is also a dominating set
- Maximum independent set not necessarily intuitively desired solution
  - Example: Radial graph, with only  $(v_0, v_i) \in E$



# A basic construction idea for independent sets

- Use some attribute of nodes to break local symmetries
  - Node identifiers, energy reserve, mobility, weighted combinations... - matters not for the idea as such (all types of variations have been looked at)
- Make each node a clusterhead that locally has the largest attribute value
- Once a node is dominated by a clusterhead, it abstains from local competition, giving other nodes a chance



# Determining gateways to connect clusters

- Suppose: Clusterheads have been found
- How to connect the clusters, how to select gateways?
  
- It suffices for each clusterhead to connect to all other clusterheads that are at most three hops
  - Resulting backbone (!) is connected
  
- Formally: Steiner tree problem
  - Given: Graph  $G=(V,E)$ , a subset  $C \subseteq V$
  - Required: Find another subset  $T \subseteq V$  such that  $S \subseteq T$  is connected and  $S \subseteq T$  is a cheapest such set
  - Cost metric: number of nodes in  $T$ , link cost
  - Here: special case since  $C$  are an independent set

# Rotating clusterheads

- Serving as a clusterhead can put additional burdens on a node
  - For MAC coordination, routing, ...
- Let this duty rotate among various members
  - Periodically reelect – useful when energy reserves are used as discriminating attribute
  - LEACH – determine an optimal percentage  $P$  of nodes to become clusterheads in a network
    - Use  $1/P$  rounds to form a period
    - In each round,  $nP$  nodes are elected as clusterheads
    - At beginning of round  $r$ , node that has not served as clusterhead in this period becomes clusterhead with probability  $P/(1-p(r \bmod 1/P))$

# Multi-hop clusters

- Clusters with diameters larger than 2 can be useful, e.g., when used for routing protocol support
- Formally: Extend “domination” definition to also dominate nodes that are at most  $d$  hops away
- Goal: Find a smallest set  $D$  of dominating nodes with this extended definition of dominance
- Only somewhat complicated heuristics exist
  
- Different tilt: Fix the **size** (not the diameter) of clusters
  - Idea: Use **growth budgets** – amount of nodes that can still be adopted into a cluster, pass this number along with broadcast adoption messages, reduce budget as new nodes are found

# Passive clustering

- Constructing a clustering structure brings overheads
  - Not clear whether they can be amortized via improved efficiency
- Question: Eat cake and have it?
  - Have a clustering structure without any overhead?
  - Maybe not the best structure, and maybe not immediately, but benefits at zero cost are no bad deal...

## ! Passive clustering

- Whenever a broadcast message travels the network, use it to construct clusters on the fly
- Node to start a broadcast: Initial node
- Nodes to forward this first packet: Clusterhead
- Nodes forwarding packets from clusterheads: ordinary/gateway nodes
- And so on... ! Clusters will emerge at low overhead

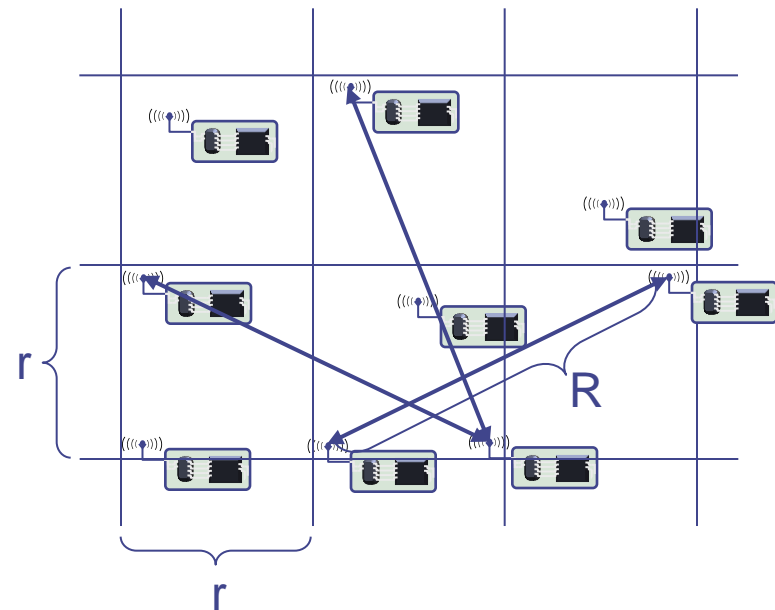


# Overview

- Motivation, basics
- Power control
- Backbone construction
- Clustering
- ***Adaptive node activity***

# Adaptive node activity

- Remaining option: Turn some nodes off deliberately
- Only possible if other nodes remain on that can take over their duties
- Example duty: Packet forwarding
  - Approach: Geographic Adaptive Fidelity (GAF)
- Observation: Any two nodes within a square of length  $r < R/5^{1/2}$  can replace each other with respect to forwarding
  - $R$  radio range
- Keep only one such node active, let the other sleep



# Conclusion

- Various approaches exist to trim the topology of a network to a desired shape
- Most of them bear some non-negligible overhead
  - At least: Some distributed coordination among neighbors, or they require additional information
  - Constructed structures can turn out to be somewhat brittle – overhead might be wasted or even counter-productive
- Benefits have to be carefully weighted against risks for the particular scenario at hand