Wireless Sensor Networks

23rd Lecture

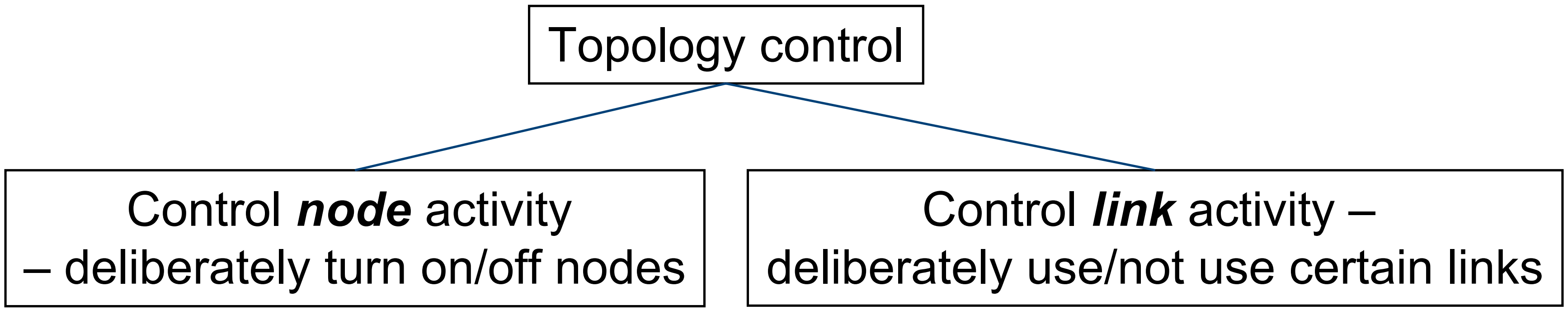
30.01.2007

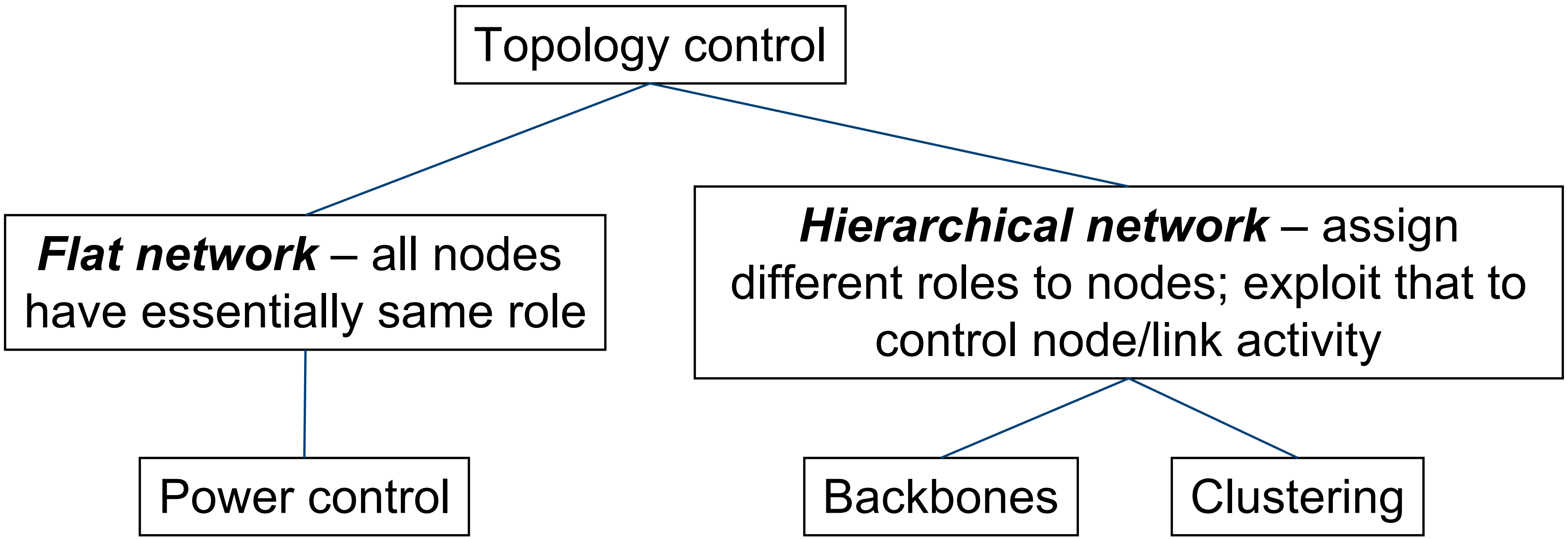
University of Freiburg
Computer Networks and Telematics
Prof. Christian Schindelhauer

Christian Schindelhauer
schindel@informatik.uni-freiburg.de

Options for topology control

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer





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)
- **Dominating Set:**
  - Given an undirected graph G=(V,E)
  - Find a minimal subset $W \subseteq V$ such that for all $u \in W$ there exists $v \in V$ with $\{u,v\} \in V$
- **Problem: MDS is an NP-hard problem**
  - Hard to approximate, and even approximations need quite a few messages
  - Polynomial approximable within c log n for some $c > 0$ only if P=NP
  - Polynomial approximable within a factor of 1 + log n.

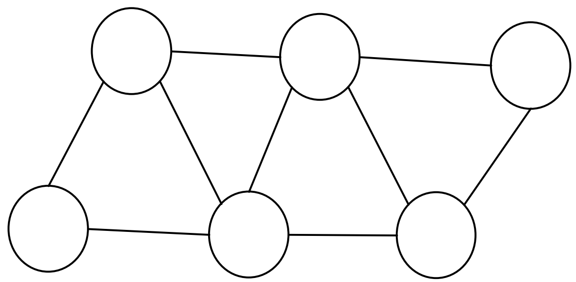
University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Backbone by growing a tree

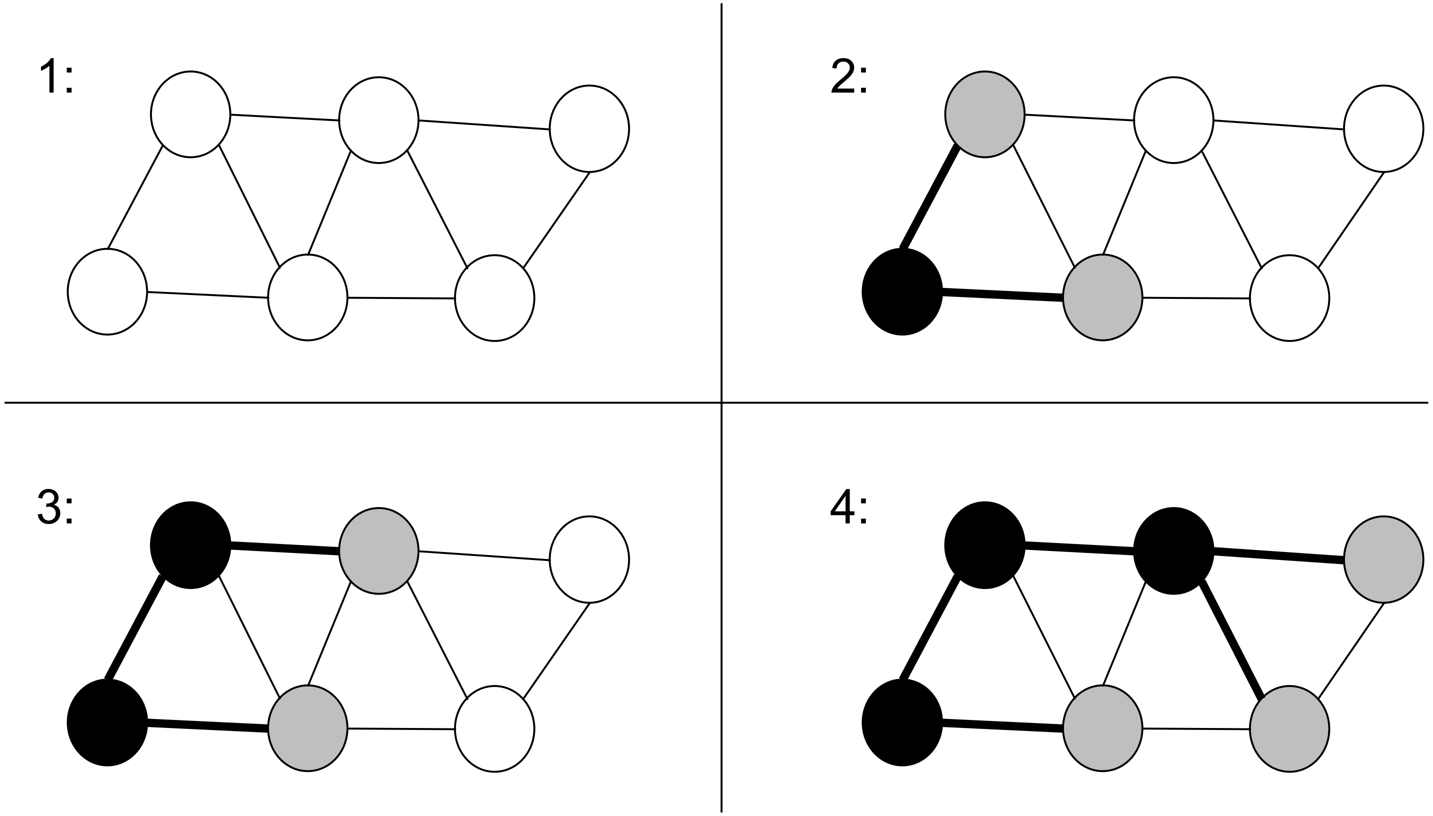
- **Construct the backbone as a tree, grown iteratively**

```
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
  }
}
```



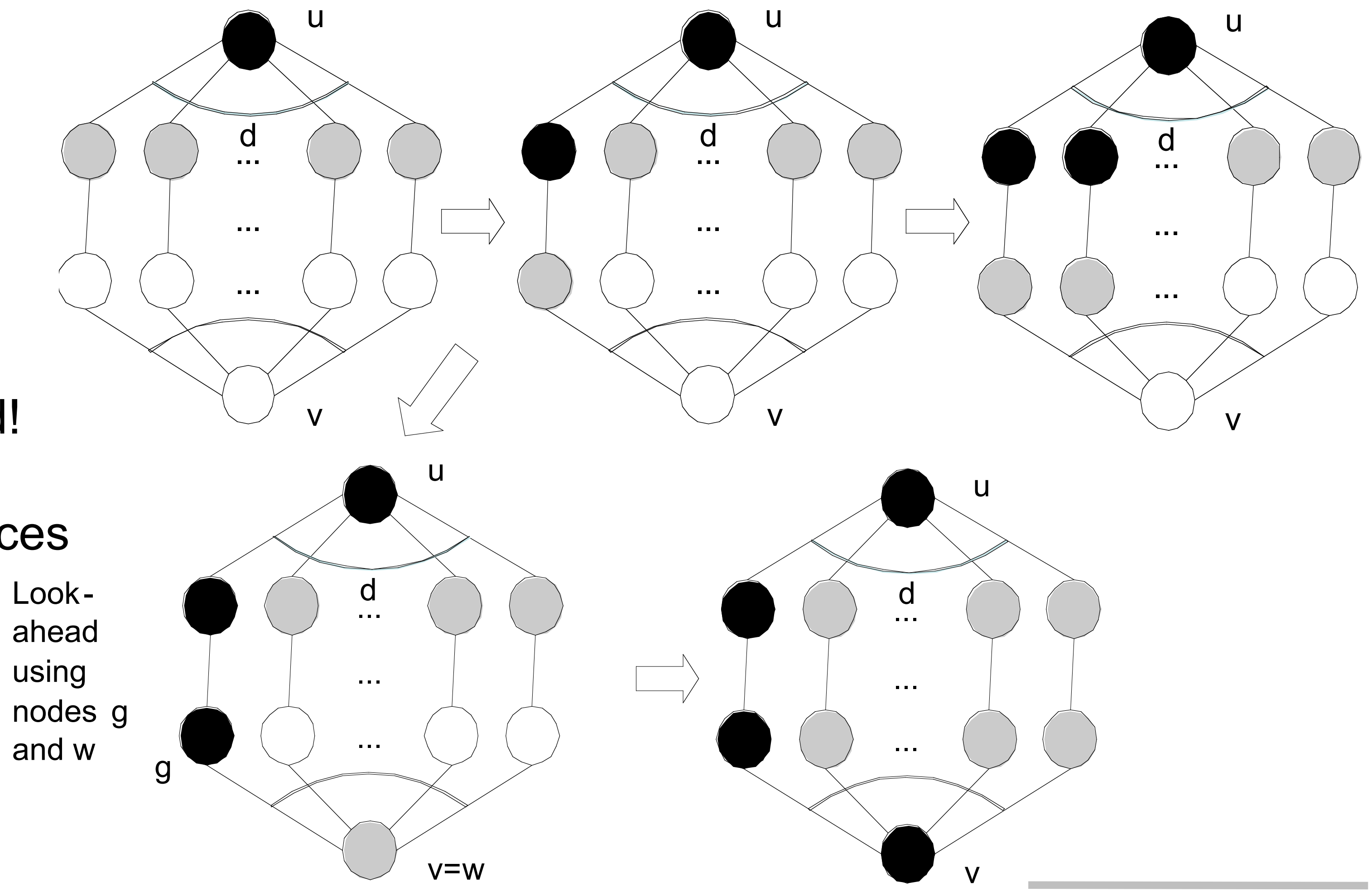
Backbone by growing a tree – Example



Problem: Which gray node to pick?

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

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



Solution:
Look ahead!
Here,
one step suffices

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(Δ)) larger than MDS**
  - H(·) harmonic function, $H(k) = \sum_{i=1}^{k} 1/i \leq \ln k + 1$
  - Δ is maximum degree of the graph
- **It is automatically connected**
- **Can be implemented in a distributed fashion as well**

Start big, make lean

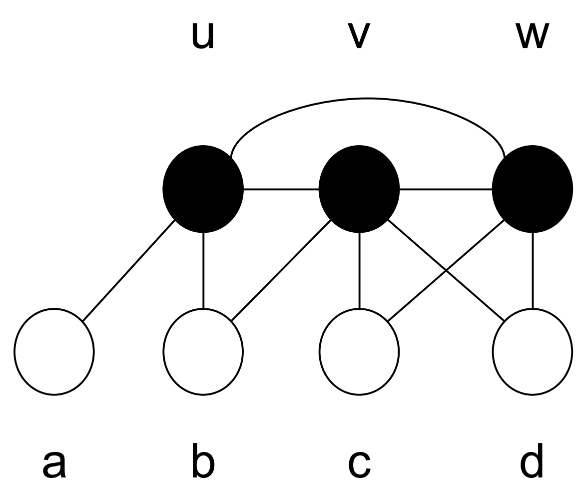
- **Idea: start with some, possibly large, connected dominating set, reduce it by removing unnecessary nodes**
- **Initial construction for dominating set**
  - All nodes are initially white
  - Mark any node black that has two neighbors that are not neighbors of each other (they might need to be dominated)

  $\rightarrow$ 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

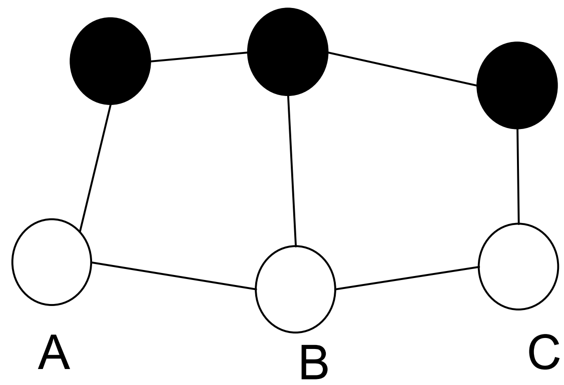
- **Heuristic 1: Unmark node v if**
  - Node v and its neighborhood are included in the neighborhood of some node marked node u (then u will do the domination for v as well)
  - Node v has a smaller unique identifier than u (to break ties)
- **Heuristic 2: Unmark node v if**
  - Node v's neighborhood 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**



University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

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



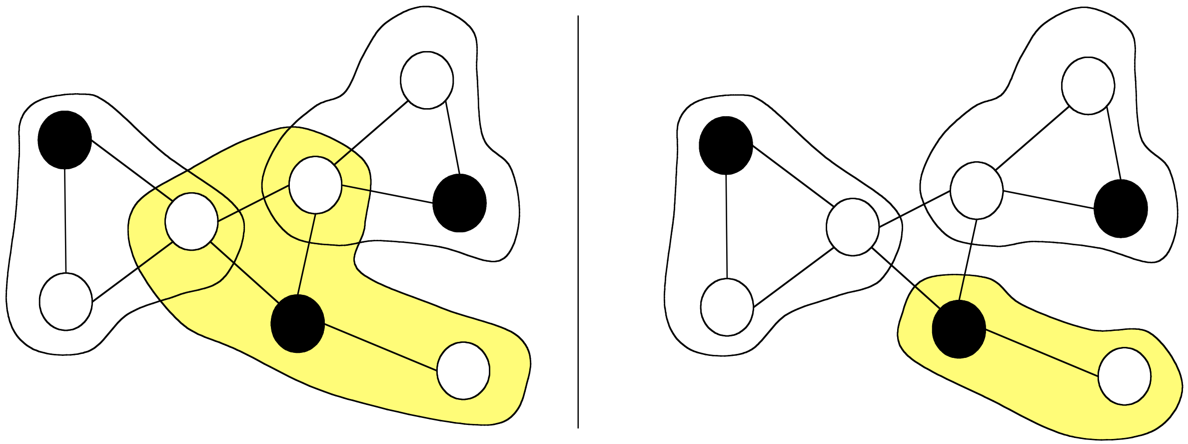
University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Overview

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

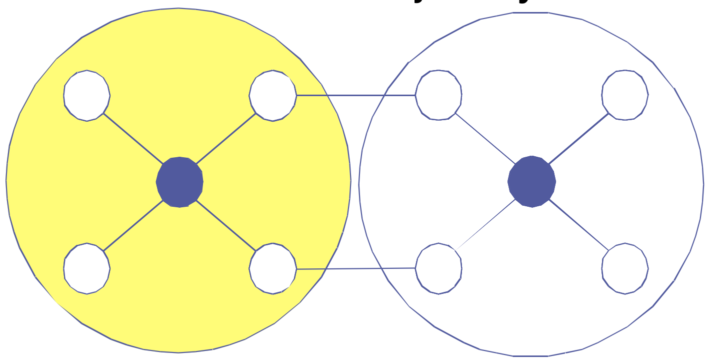
Clustering

- **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

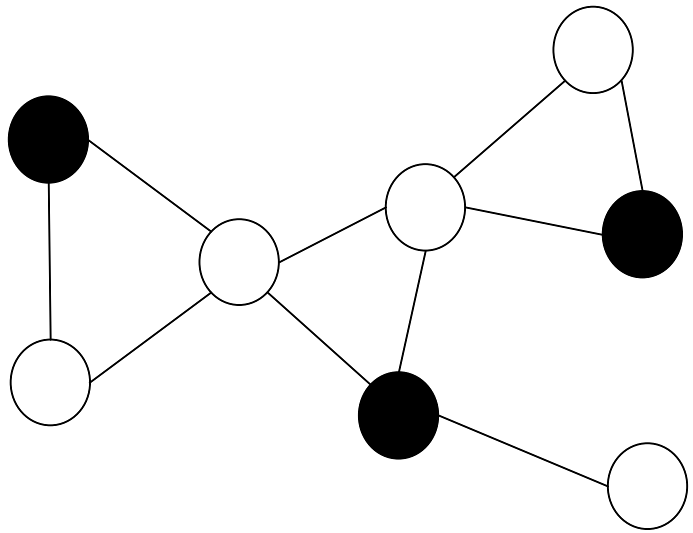
- **Further options**
  - How do clusters communicate? Some nodes need to act as ***gateways*** between clusters
    If clusters may not overlap, two nodes need to jointly act as a ***distributed gateway***



  - Many gateways may exist between clusters
    - active, 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

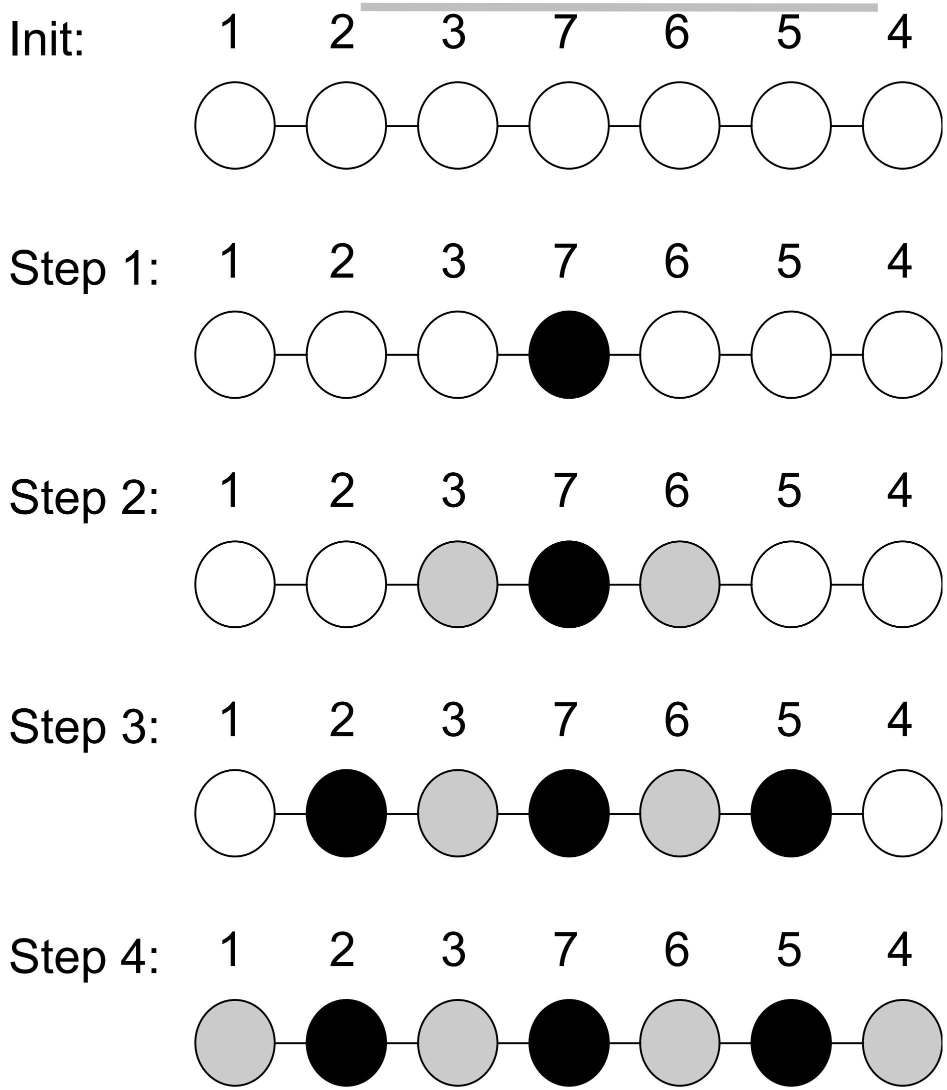
- **Computing a maximum independent set is NP-complete**
- **Can be approximate within $\Delta/6 + o(1)$ and O(Δ/ log log Δ)**
  [Halldorsson Radhakrishnan]
- **Show: A maximum independent set is also a dominating set**
- **Maximum independent set not necessarily intuitively desired solution**
  - Example: Radial graph, with only $(v_0,v_i) \in E$



A basic construction idea for independent sets

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **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

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **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 $S \cup T$ is connected and $S \cup T$ 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 mod 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

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Constructing a clustering structure brings overheads**
  - Not clear whether they can be amortized via improved efficiency
- **Question:**
  - 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

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

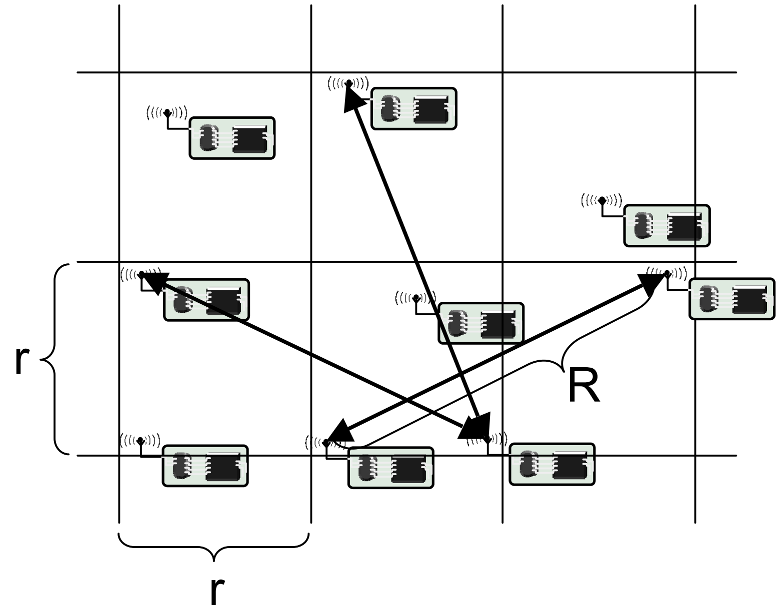
Overview

- **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

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **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**

Routing with IDs

- **In any network of diameter > 1, the routing & forwarding problem appears**
- **We will discuss mechanisms for constructing routing tables in ad hoc/sensor networks**
  - Specifically, when nodes are mobile
  - Specifically, with energy efficiency as an optimization metric
  - Specifically, when node position is available

Note: Presentation here partially follows Beraldi & Baldoni, Unicast Routing Techniques for Mobile Ad Hoc Networks, in M. Ilyas (ed.), The Handbook of Ad Hoc Wireless Networks

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Overview

- *Unicast routing in MANETs*
- **Energy efficiency & unicast routing**
- **Geographical routing**

Unicast, id-centric routing

- **Given: a network/a graph**
  - Each node has a unique identifier (ID)
- **Goal: Derive a mechanism that allows a packet sent from an arbitrary node to arrive at some arbitrary destination node**
  - The routing & forwarding problem
  - Routing: Construct data structures (e.g., tables) that contain information how a given destination can be reached
  - Forwarding: Consult these data structures to forward a given packet to its next hop
- **Challenges**
  - Nodes may move around, neighborhood relations change
  - Optimization metrics may be more complicated than “smallest hop count” – e.g., energy efficiency

Ad-hoc routing protocols

- **Because of challenges, standard routing approaches not really applicable**
  - Too big an overhead, too slow in reacting to changes
  - Examples: Dijkstra's link state algorithm; Bellman-Ford distance vector algorithm
- **Simple solution: Flooding**
  - Does not need any information (routing tables) – simple
  - Packets are usually delivered to destination
  - But: overhead is prohibitive

  → Usually not acceptable, either

→ **Need specific,** *ad hoc routing protocols*

Ad hoc routing protocols – classification

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Main question to ask:** *When* **does the routing protocol operate?**

- **Option 1: Routing protocol** *always* **tries to keep its routing data up-to-date**
  – Protocol is ***proactive*** (active before tables are actually needed) or ***table-driven***

- **Option 2: Route is only determined when actually needed**
  – Protocol operates ***on demand***

- **Option 3: Combine these behaviors**
  – ***Hybrid*** protocols

Ad hoc routing protocols – classification

- **Is the network regarded as flat or hierarchical?**
  - Compare topology control, traditional routing

- **Which data is used to identify nodes?**
  - An arbitrary identifier?
  - The ***position*** of a node?
    - Can be used to assist in ***geographic*** routing protocols because choice of next hop neighbor can be computed based on destination address
  - Identifiers that are not arbitrary, but carry some structure?
    - As in traditional routing
    - Structure akin to position, on a logical level?

Proactive protocols

- **Idea: Start from a +/- standard routing protocol, adapt it**

- **Adapted distance vector:** *Destination Sequence Distance Vector (DSDV)*
  - Based on distributed Bellman Ford procedure
  - Add ***aging*** information to route information propagated by distance vector exchanges; helps to avoid routing loops
  - Periodically send full route updates
  - On topology change, send incremental route updates
  - Unstable route updates are delayed
  - … + some smaller changes

The Shortest Path Problem

- **Given:**
  - A directed Graph G=(V,E)
  - Start node
  - and edge weights $w : E \rightarrow \mathbb{R}$
- **Define Weight of Shortest Path**
  - δ(u,v) = minimal weight w(p) of a path p from u to v
  - w(p) = sum of all edge weights w(e) of edges e of path p
- **Find:**
  - The shortest paths from s to all nodes in G
- **Solution set:**
  - is described by a tree with root s
  - Every node points towards the root s

Shortest Paths of Edsger Wybe Dijkstra

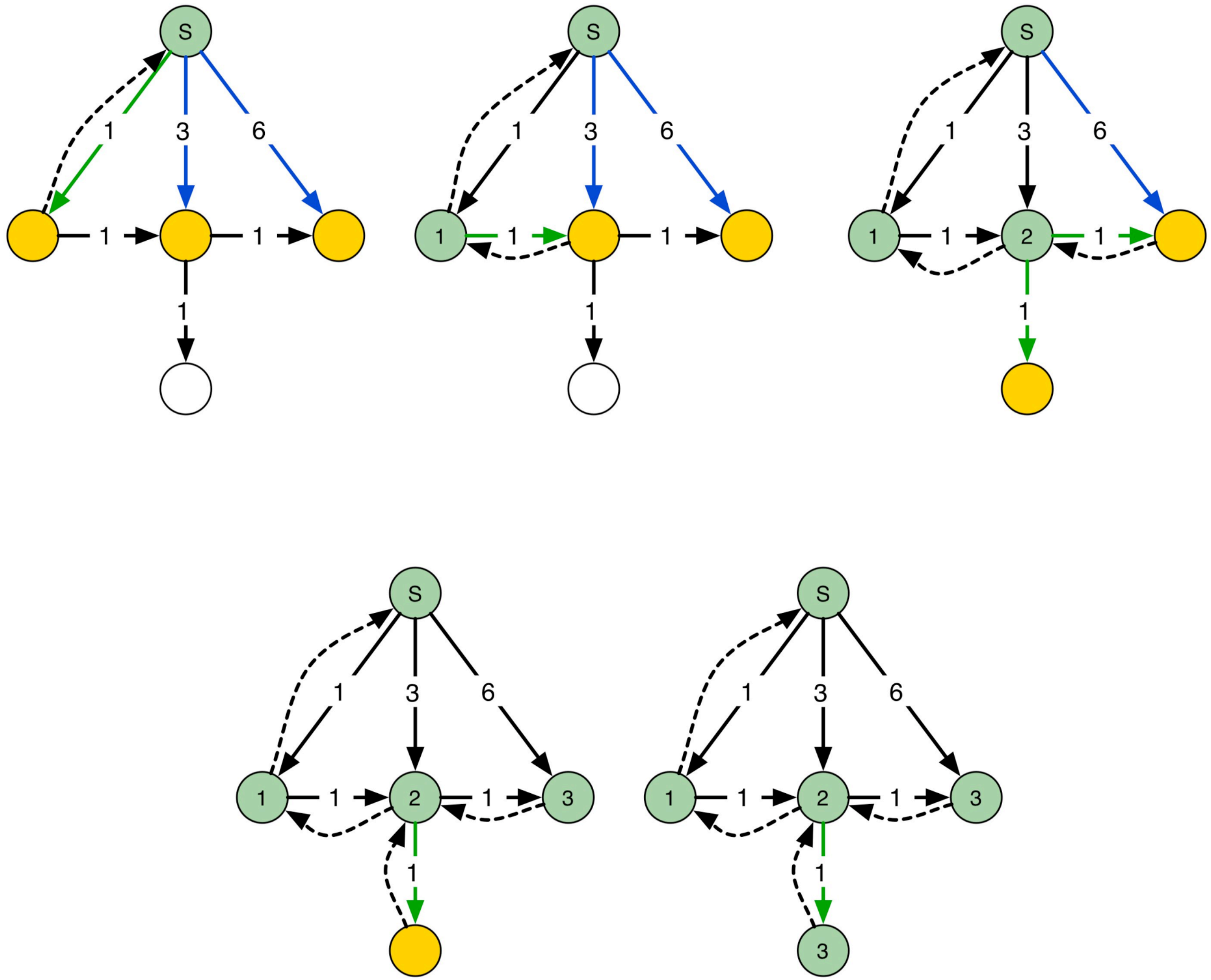
Dijkstra(G, w, s)
begin
  Init-Single-Source(G, w)
  $S \leftarrow \emptyset$
  $Q \leftarrow V$
  while $Q \neq \emptyset$ do
    $u \leftarrow$ Element aus Q mit minimalen Wert $d(u)$
    $S \leftarrow S \cup \{u\}$
    $Q \leftarrow Q \setminus \{u\}$
    for all $v \in$ Adj(u) do
      Relax(u, v)
    od
  od
end

Dijkstra's algorithm has runtime
$\Theta(|E| + |V| \log |V|)$

Init-Single-Source(G, w, s)
begin
  for all $v \in V$ do
    $d(v) \leftarrow \infty$
    $\pi(v) \leftarrow v$
  od
  $d(s) \leftarrow 0$
end

Relax(u, v)
begin
  if $d(v) > d(u) + w(u, v)$ then
    $d(v) \leftarrow d(u) + w(u, v)$
    $\pi(v) \leftarrow u$
  fi
end

Dijkstra: Example



Bellman-Ford

- Dijkstras Algorithm does not work for negative edge weights
- Bellman-Ford
  - solves shortest paths in runtime O(|V| |E|).

```
Bellman-Ford(G, w, s)
begin
    Init-Single-Source(G, w)
    loop |V| − 1 times do
        for all (u, v) ∈ E do
            Relax(u, v)
        od
    od
    for all (u, v) ∈ E do
        if d(v) > d(u) + w(u, v) then return false
    od
    return true
end
```

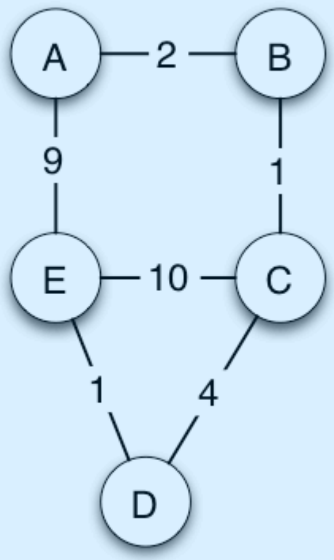
```
Init-Single-Source(G, w, s)
begin
    for all v ∈ V do
        d(v) ← ∞
        π(v) ← v
    od
    d(s) ← 0
end
```

```
Relax(u, v)
begin
    if d(v) > d(u) + w(u, v) then
        d(v) ← d(u) + w(u, v)
        π(v) ← u
    fi
end
```

Distance Vector Routing Protocol

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Distance Table Data Structure**
  - Every node has a
    - row for each target
    - column for each direct neighbor
- **Distributed Algorithm**
  - Every node communicates only with his neighbors
- **Asynchronous**
  - Nodes do not use a round model
- **Self-termination**
  - algorithm runs until no further changes occur



Distance Table für A

	von A	über B	über E	Routing Tabellen Eintrag
nach	B	2	15	B
	C	3	14	B
	D	7	10	B
	E	13	9	E

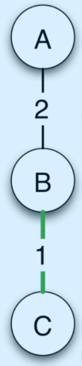
Distance Table für C

	von C	über B	über D	über E	Routing Tabellen Eintrag
nach	A	3	11	18	B
	B	1	6	16	B
	D	5	4	11	D
	E	7	5	10	D

The “Count to Infinity” - Problem

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Good news travel fast**
  - A new connection is announced quickly.
- **Bad news travel slow**
  - Connection fails
  - Neighbors increase the distance counter
  - “Count to Infinity”-Problem



Distance Table für A

		über	
	von A	B	Routing Tabellen Eintrag
nach	B	**2**	B
	C	**-**	B

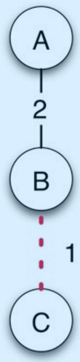
Distance Table für B

		über		
	von B	A	C	Routing Tabellen Eintrag
nach	A	2	-	A
	C	-	1	C

nach kurzer Zeit

		über	
	von A	B	Routing Tabellen Eintrag
nach	B	**2**	B
	C	**3**	B

		über		
	von B	A	C	Routing Tabellen Eintrag
nach	A	2	4	A
	C	5	1	C



		über	
	von A	B	Routing Tabellen Eintrag
nach	B	2	B
	C	3	B

		über		
	von B	A	C	Routing Tabellen Eintrag
nach	A	2	-	A
	C	5	-	A

		über	
	von A	B	Routing Tabellen Eintrag
nach	B	2	B
	C	7	B

		über		
	von B	A	C	Routing Tabellen Eintrag
nach	A	2	-	A
	C	5	-	A

		über	
	von A	B	Routing Tabellen Eintrag
nach	B	2	B
	C	7	B

		über		
	von B	A	C	Routing Tabellen Eintrag
nach	A	2	-	A
	C	9	-	A

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Link-State Protocol

- **Link State Routers**
  - exchange information using **link state packets** (LSP)
  - Every router uses a (centralized) shortest-path-algorithm
- **LSP contains**
  - ID of creator of LSP
  - Costs of all edges from the creator
  - Sequence no. (SEQNO)
  - TTL-entry (time to live)
- **Reliable Flooding**
  - The current LSP of every node are stored
  - Forwarding of LSPs to all neighbors
    - except sending nodes
  - Periodically new LSPs are generated
    - with incremented SEQNO
  - TTL is decremented after every transmission

Proactive protocols – OLSR

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Combine link-state protocol & topology control**
- *Optimized Link State Routing* **(***OLSR***)**

- **Topology control component: Each node selects a minimal dominating set for its two-hop neighborhood**
  - Called the ***multipoint relays***
  - Only these nodes are used for packet forwarding
  - Allows for efficient flooding

- **Link-state component: Essentially a standard link-state algorithms on this reduced topology**
  - Observation: Key idea is to reduce flooding overhead (here by modifying topology)

Proactive protocols – Combine LS & DS: Fish eye

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Fisheye State Routing (FSR) makes basic observation: When destination is far away, details about path are not relevant – only in vicinity are details required**
  - Look at the graph as if through a fisheye lens
  - Regions of different accuracy of routing information

- **Practically:**
  - Each node maintains topology table of network (as in LS)
  - Unlike LS: only distribute link state updates locally
  - More frequent routing updates for nodes with smaller scope

Reactive protocols – DSR

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

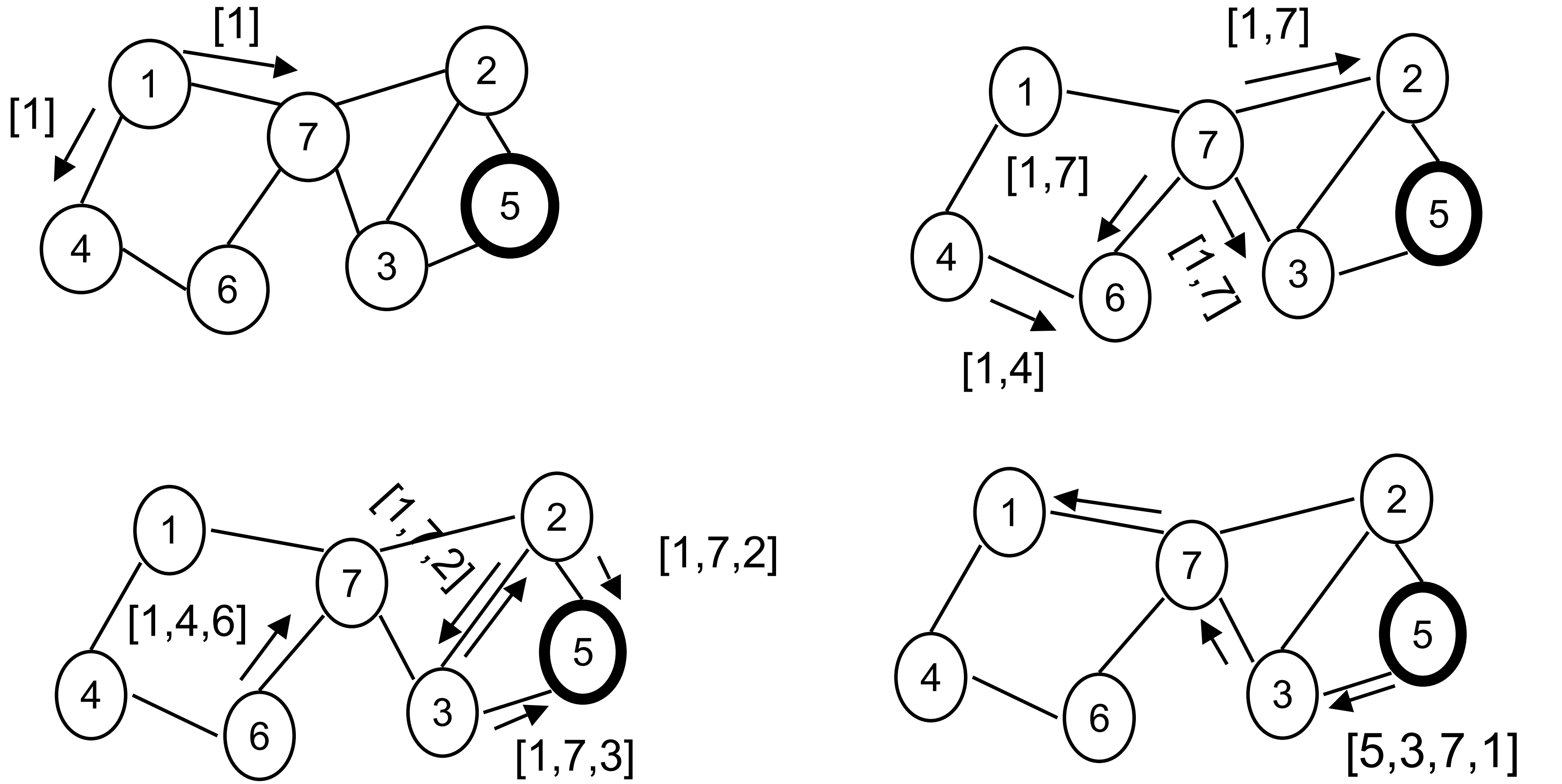
- **In a reactive protocol, how to forward a packet to destination?**
  - Initially, no information about next hop is available at all
  - One (only?) possible recourse: Send packet to ***all*** neighbors – flood the network
  - Hope: At some point, packet will reach destination and an answer is sent pack – use this answer for ***backward learning*** the route from destination to source

- **Practically:** *Dynamic Source Routing (DSR)*
  - Use separate ***route request/route reply*** packets to discover route
    - Data packets only sent once route has been established
    - Discovery packets smaller than data packets
  - Store routing information in the discovery packets

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

DSR route discovery procedure

Search for route from 1 to 5



Node 5 uses route information recorded in RREQ to send back, via ***source routing***, a route reply

DSR modifications, extensions

- **Intermediate nodes may send route replies in case they already know a route**
  - Problem: stale route caches
- **Promiscuous operation of radio devices – nodes can learn about topology by listening to control messages**
- **Random delays for generating route replies**
  - Many nodes might know an answer – reply storms
  - NOT necessary for medium access – MAC should take care of it
- **Salvaging/local repair**
  - When an error is detected, usually sender times out and constructs entire route anew
  - Instead: try to locally change the source-designated route
- **Cache management mechanisms**
  - To remove stale cache entries quickly
  - Fixed or adaptive lifetime, cache removal messages, …

Reactive protocols – AODV

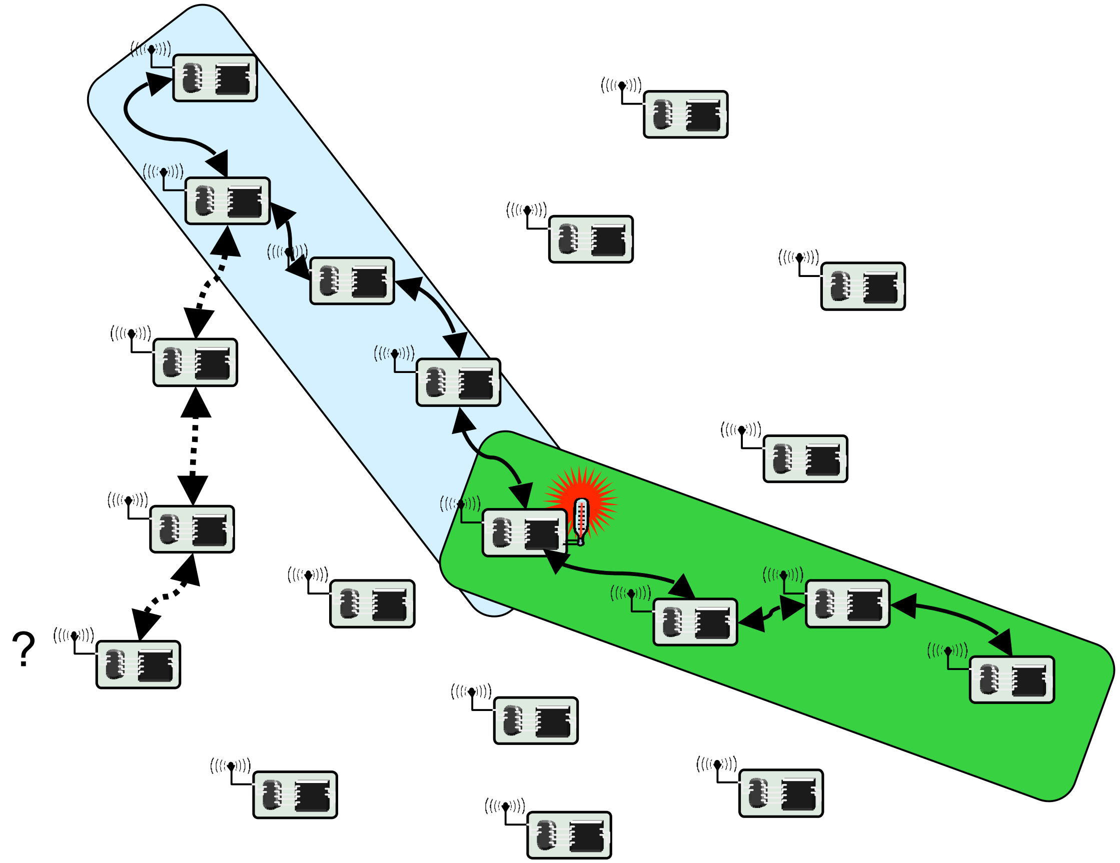
- **Ad hoc On Demand Distance Vector routing (AODV)**
  - Very popular routing protocol
  - Essentially same basic idea as DSR for discovery procedure
  - Nodes maintain routing tables instead of source routing
  - Sequence numbers added to handle stale caches
  - Nodes remember from where a packet came and populate routing tables with that information

Reactive protocols – TORA

- **Observation: In hilly terrain, routing to a river's mouth is easy – just go downhill**
- **Idea: Turn network into hilly terrain**
  - Different "landscape" for each destination
  - Assign "heights" to nodes such that when going downhill, destination is reached – in effect: orient edges between neighbors
  - Necessary: resulting directed graph has to be cycle free
- **Reaction to topology changes**
  - When link is removed that was the last "outlet" of a node, reverse direction of all its other links (increase height!)
  - Reapply continuously, until each node except destination has at least a single outlet – will succeed in a connected graph!

Alternative approach: Gossiping/rumor routing

- Turn routing problem around: Think of an "agent" wandering through the network, looking for data (events, …)
- Agent initially perform random walk
- Leave "traces" in the network
- Later agents can use these traces to find data
- Essentially: works due to high probability of line intersections



Overview

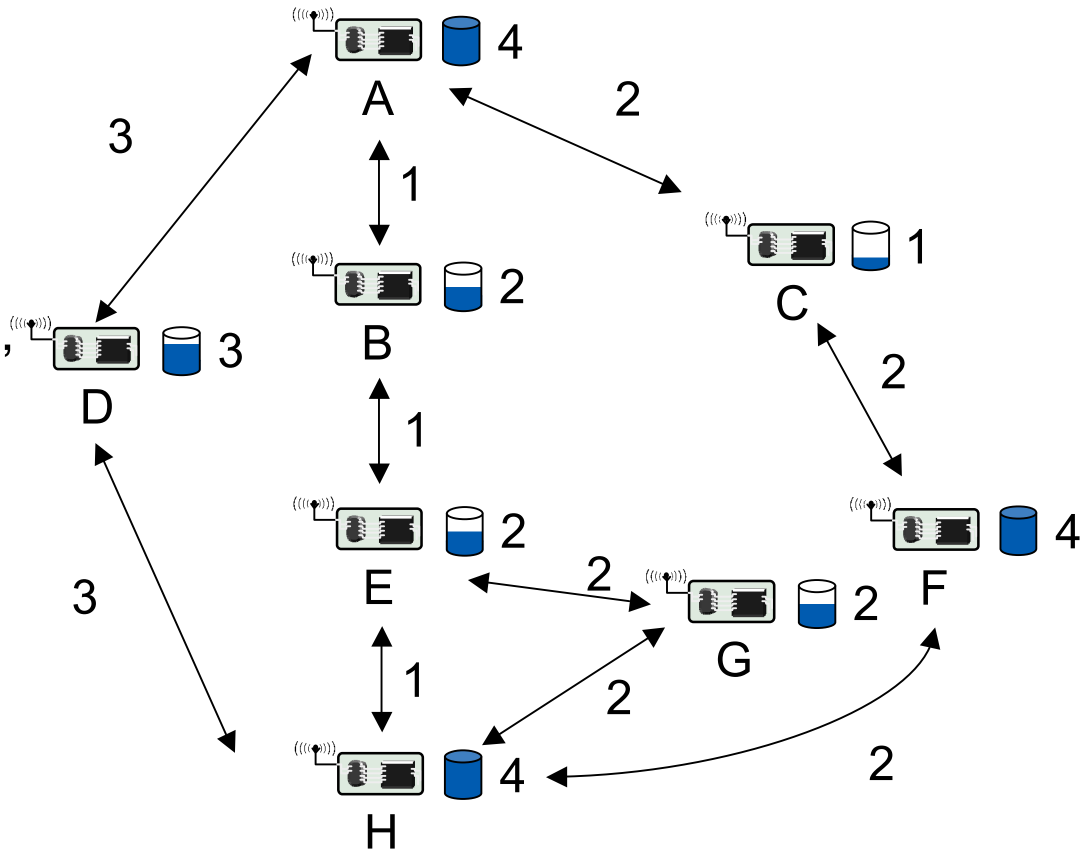
University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Unicast routing in MANETs**
- *Energy efficiency & unicast routing*
- **Geographical routing**

Energy-efficient unicast: Goals

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

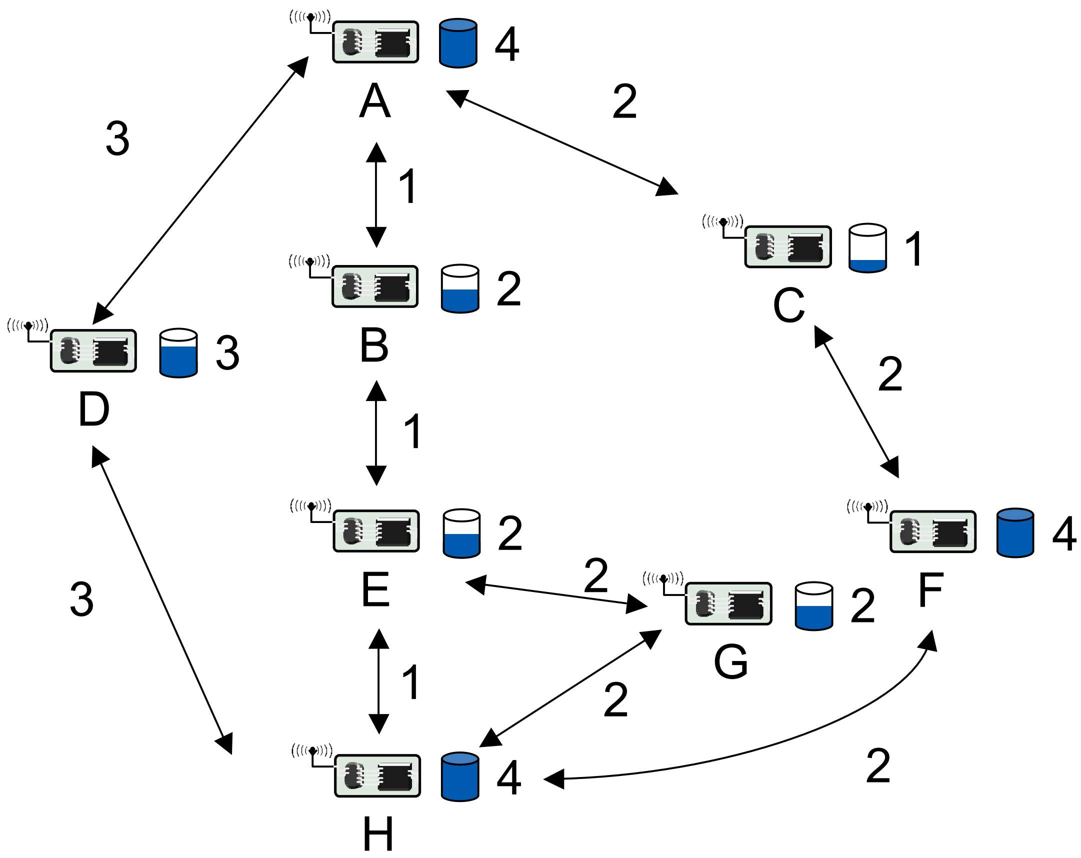
- **Particularly interesting performance metric: Energy efficiency**
- **Goals**
  - Minimize energy/bit
    - Example: A-B-E-H
  - Maximize network lifetime
    - Time until first node failure, loss of coverage, partitioning
- **Seems trivial – use proper link/path metrics (not hop count) and standard routing**



Example: Send data from node A to node H

Basic options for path metrics

- **Maximum total available battery capacity**
  - Path metric: Sum of battery levels
  - Example: A-C-F-H
- **Minimum battery cost routing**
  - Path metric: Sum of reciprocal battery levels
  - Example: A-D-H
- **Conditional max-min battery capacity routing**
  - Only take battery level into account when below a given level
- **Minimize variance in power levels**
- **Minimum total transmission power**



A non-trivial path metric

- **Previous path metrics do not perform particularly well**

- **One non-trivial link weight:**

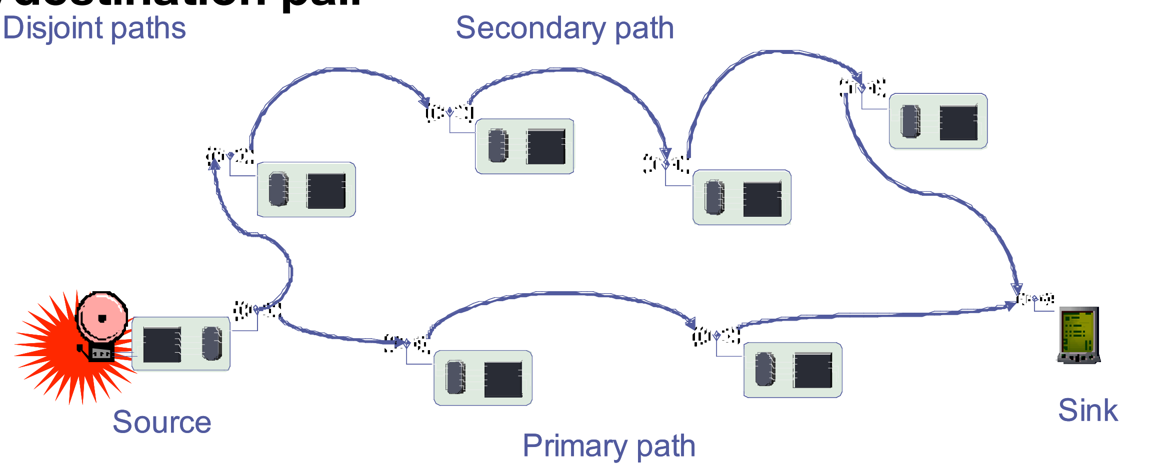
$$w_{ij} = e_{ij}(\lambda^{\alpha_i} - 1)$$

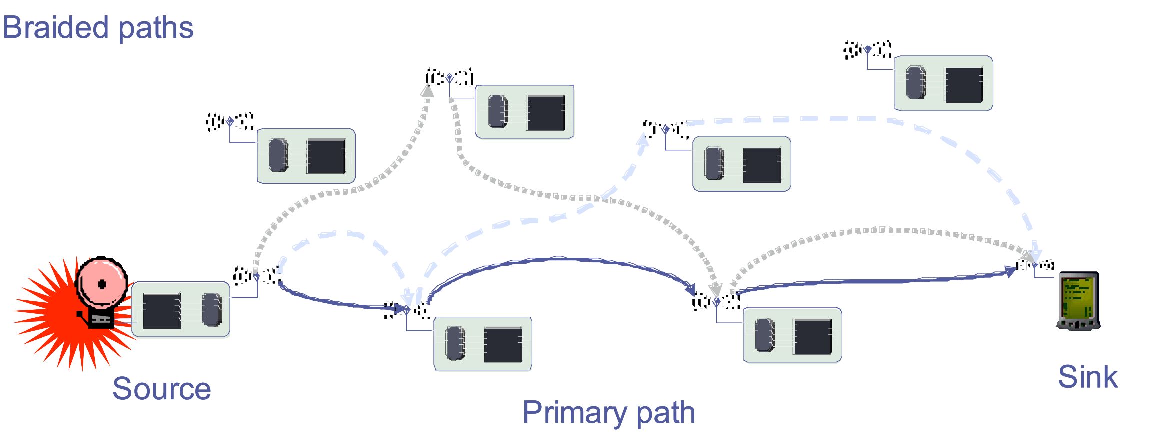
  – w_{ij} weight for link node i to node j
  – e_{ij} required energy, λ some constant, α_i fraction of battery of node i already used up
- **Path metric: Sum of link weights**
  – Use path with smallest metric

- **Properties: Many messages can be send, high network lifetime**
  – With admission control, even a competitive ratio logarithmic in network size can be shown

Multipath unicast routing

- **Instead of only a single path, it can be useful to compute multiple paths between a given source/destination pair**
  - Multiple paths can be ***disjoint*** or ***braided***
  - Used simultaneously, alternatively, randomly, …





University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

Overview

- **Unicast routing in MANETs**
- **Energy efficiency & unicast routing**
- *Geographical routing*
  - ***Position-based routing***
  - Geocasting

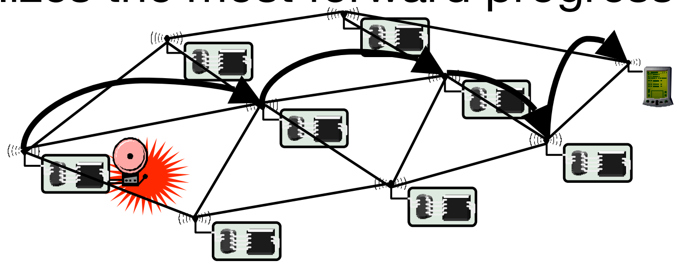
Geographic routing

- **Routing tables contain information to which next hop a packet should be forwarded**
  - Explicitly constructed
- **Alternative: Implicitly *infer* this information from physical placement of nodes**
  - Position of current node, current neighbors, destination known – send to a neighbor in the right direction as next hop
  - ***Geographic routing***
- **Options**
  - Send to any node in a given area – ***geocasting***
  - Use position information to aid in routing – ***position-based routing***
    - Might need a ***location service*** to map node ID to node position

Basics of position-based routing

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

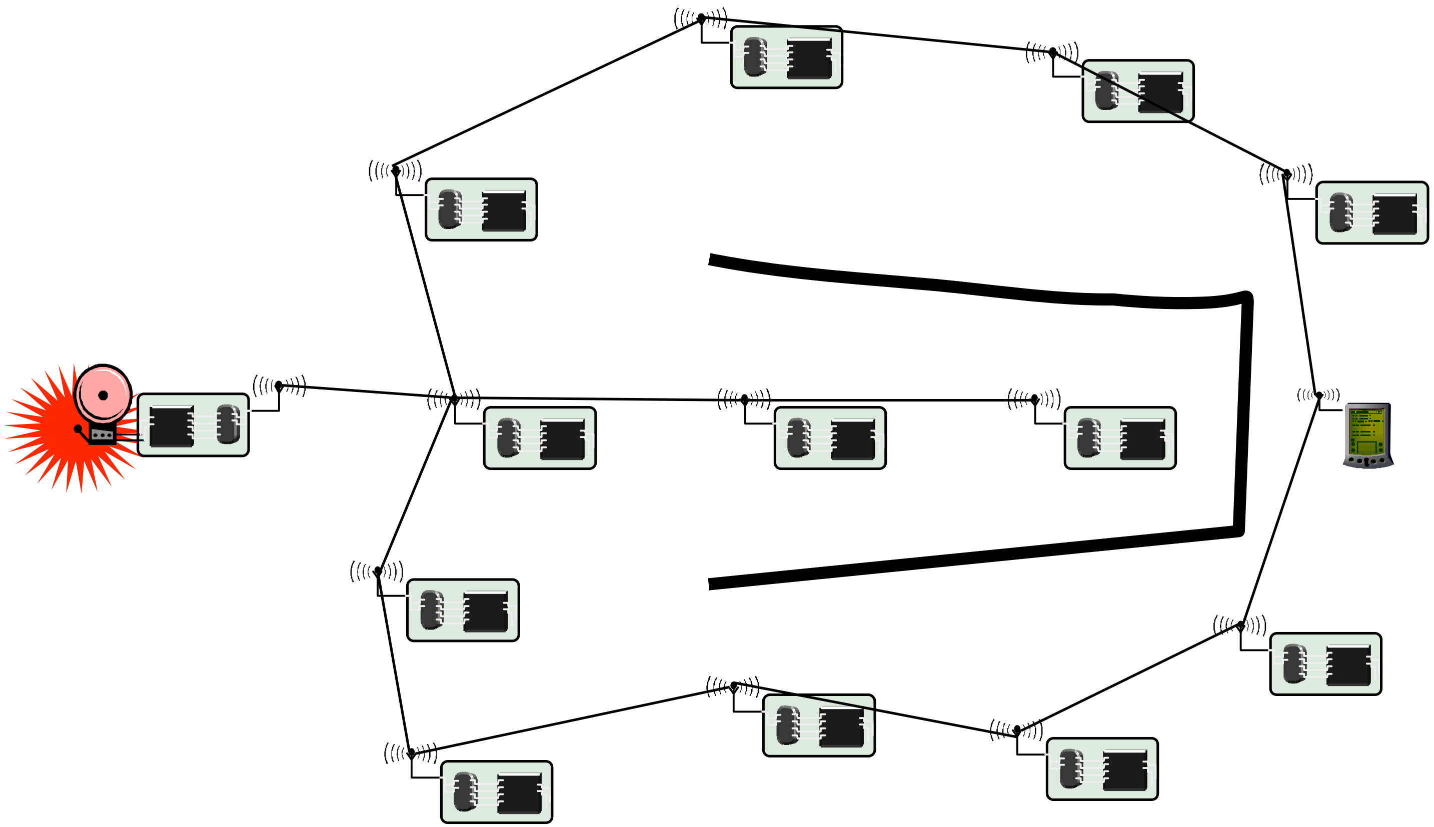
- **"Most forward within range r" strategy**
  - Send to that neighbor that realizes the most forward progress towards destination
  - NOT: farthest away from sender!



- **Nearest node with (any) forward progress**
  - Idea: Minimize transmission power
- **Directional routing**
  - Choose next hop that is angularly closest to destination
  - Choose next hop that is closest to the connecting line to destination
  - Problem: Might result in loops!

Problem: Dead ends

- Simple strategies might send a packet into a dead end

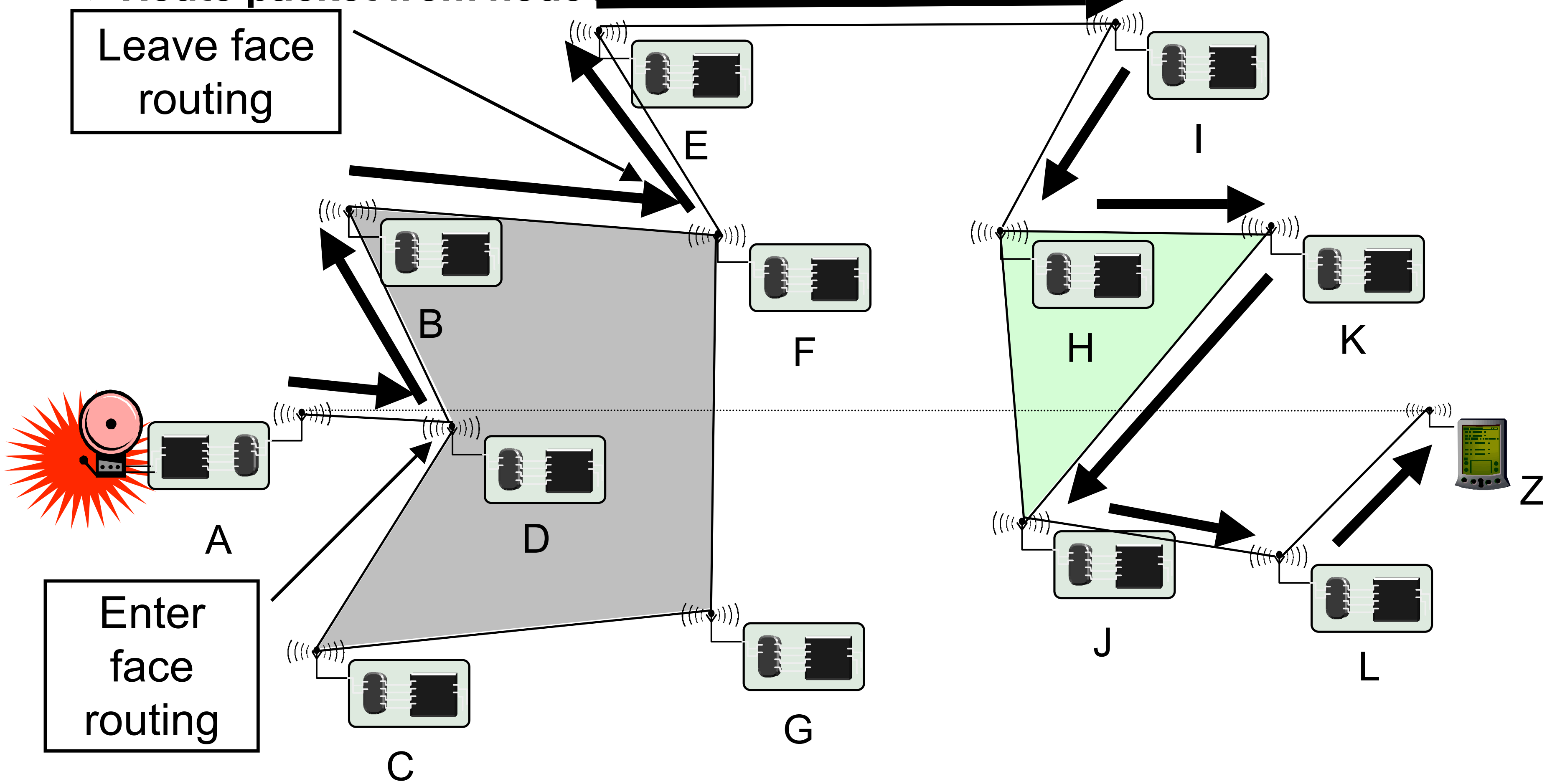


Right hand rule to leave dead ends – GPSR

- **Basic idea to get out of a dead end: Put right hand to the wall, follow the wall**
  - Does not work if on some inner wall – will walk in circles
  - Need some additional rules to detect such circles
- *Geometric Perimeter State Routing* **(***GPSR***)**
  - Earlier versions: Compass Routing II, face-2 routing
  - Use greedy, "most forward" routing as long as possible
  - If no progress possible: Switch to "face" routing
    - Face: largest possible region of the plane that is not cut by any edge of the graph; can be exterior or interior
    - Send packet around the face using right-hand rule
    - Use position where face was entered and destination position to determine when face can be left again, switch back to greedy routing
  - Requires: planar graph! (topology control can ensure that)

GPSR – Example

- Route packet from node A to node Z

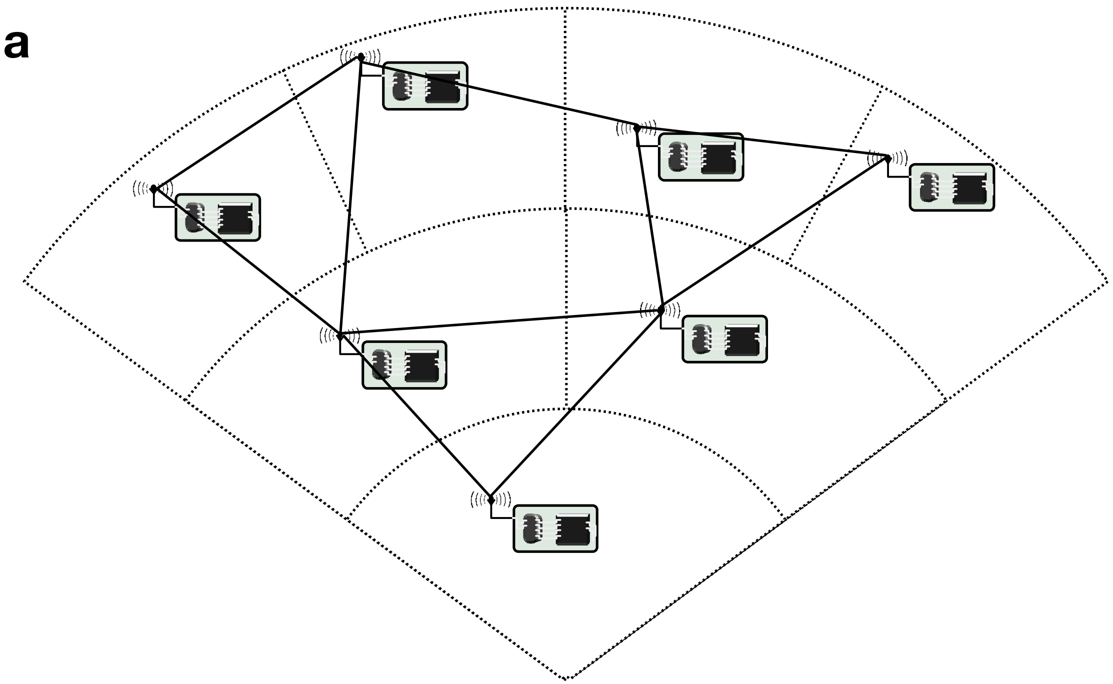


Geographic routing without positions – GEM

University of Freiburg
Institute of Computer Science
Computer Networks and Telematics
Prof. Christian Schindelhauer

- **Apparent contradiction: geographic, but no position?**
- **Construct** *virtual coordinates* **that preserve enough neighborhood information to be useful in geographic routing but do not require actual position determination**

- **Use polar coordinates from a center point**
- **Assign "virtual angle range" to neighbors of a node, bigger radius**
- **Angles are recursively redistributed to children nodes**



Thank you

and thanks to Holger Karl for the slides

University of Freiburg
Computer Networks and Telematics
Prof. Christian Schindelhauer

Wireless Sensor Networks
Christian Schindelhauer
schindel@informatik.uni-freiburg.de

23rd Lecture
31.01.2007