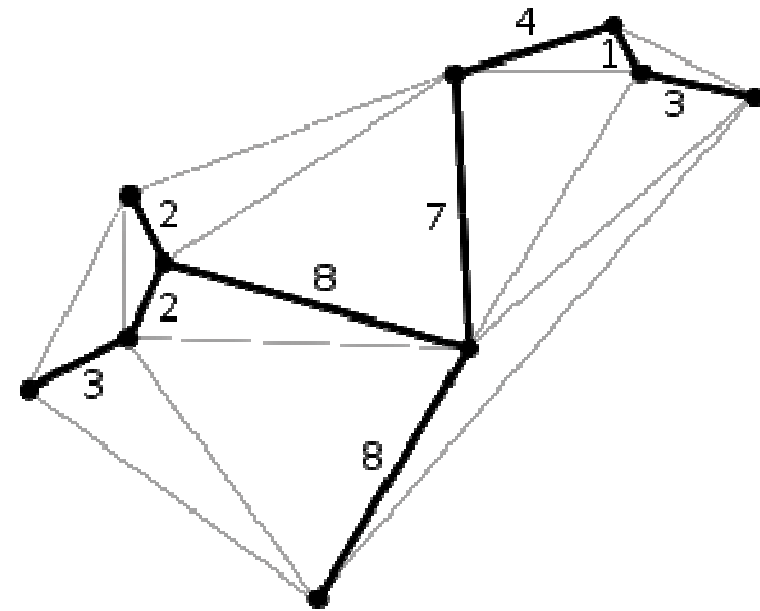


# Routing in Sensor Networks

Some algorithms

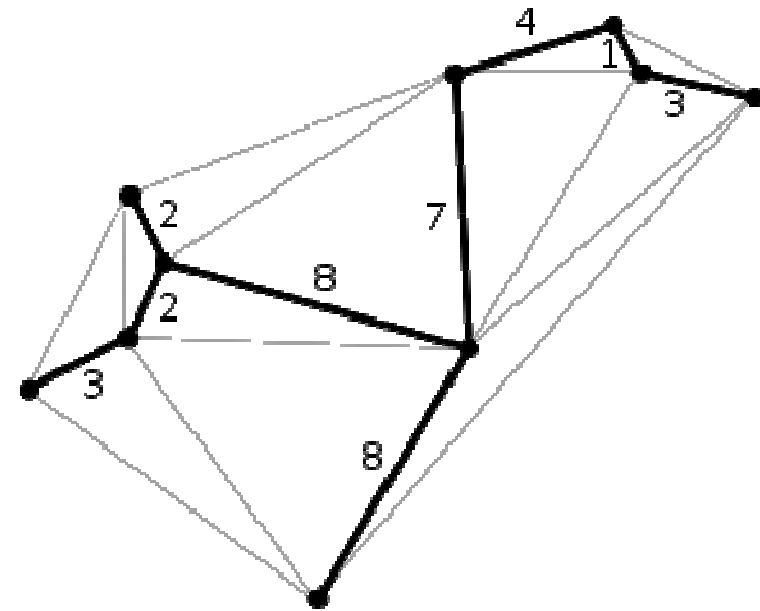
# Spanning Trees

- Spanning tree is a tree which connects all vertices
- Graphs can have many spanning trees
- Minimal spanning tree is the one with the lowest cost



# Applications

- E.g laying TV cables
- Edge costs relate to:
  - Length
  - How deep cable must be laid etc.
- Find the path with the lowest cost



# Minimal Spanning Trees

- MST is v. important in the design of data and communications networks
- Many MST algorithms exist, including ones for distributed networks, but....
  - Require large numbers of messages/time
  - Require synchronization
  - Large overheads/book keeping
- Impractical for resource constrained networks such as ad hoc WSN

# Routing in Ad-hoc WSN

- Need distributed algorithms which are energy and time efficient
- They might be sub-optimal:
  - Trade-off optimality against work done by algorithm
- Need to consider *energy complexity*:
  - *Depends on time*
  - *# messages exchanged*
  - *Radiation energy to transmit messages*

# Nearest Neighbour ST (NNT)

- Builds slightly sub-optimal tree
- Low energy complexity
- Easy to maintain dynamically
- Precludes cycles (cycle detection has high overhead)
- (closely related to nearest neighbour algorithms for TSP)

# How does it work

- Each node has a distinct id
- Each node has a unique rank
- Each node connects to the nearest node with a higher rank
- The information does not need to be updated over course of algorithm
- Algorithm is:
  - Simple
  - Local
  - Doesn't need any synchronization
  - robust

# NNT

- Every node executes following steps independently:
  - Choose a unique rank  $rank(v)$
  - Connect to nearest node  $w$  such that  $rank(w) > rank(v)$
- Random-NNT
  - $v$  chooses  $p(v)$ , a uniform random number  $[0,1]$
  - $rank(w) > rank(v)$  if:
    - $p(w) > p(v)$
    - $p(w) = p(v)$
    - and  $id(w) > id(v)$

# NNT

- Every node executes following steps independently:
  - Choose a unique rank  $rank(v)$
  - Connect to nearest node  $w$  such that  $rank(w) > rank(v)$
- Coordinate-NNT
  - $rank(v)$  = coordinate of  $v$  in a plane
  - $rank(w) > rank(v)$  if:
    - $x(w) > x(v)$
    - $x(w) = x(v)$
    - *and*  $y(w) > y(v)$

# Distributed NNT

- Assume nodes can communicate with all other nodes by increasing its transmission radius
  - Most only communicate with nearby nodes
  - Some need to communicate with distant nodes
- Algorithm exchanges 3 types of message:
  - *Request: carries rank information*
  - *Available: sent by nodes whose rank is higher than received request messages*
  - *Connect: sent by requester to nearest higher ranked node*

# Distributed NNT: Phase 1

- $i=1$
  - Repeat:
    - Set transmission radius  $r_i=2^i/\sqrt{n}$
    - If  $r_i > l$ , set  $r_i = l$
    - Broadcast  $\langle request, u, rank-info \rangle$
    - $i++$   
←
- until (receipt of available message) or ( $r_i = l$ )
- Executed by each node  $u$  independently and simultaneously
  - Recipients of broadcasts not specified
  - $l$  is max. distance between nodes

## Distributed NNT: Phase 2 and 3

- For all  $v$ , on receipt of  $\langle request, v, rank-info \rangle$  do
  - If  $rank(v) > rank(u)$ , set transmission radius to distance  $\langle u, v \rangle$
  - Send  $\langle available, u, v, \rangle$  to  $v$
- Upon receipt of available message(s):
  - Select nearest node  $v$  from senders
  - Send  $\langle connect, u, v \rangle$  to  $v$
- Phases all are all synchronized

# Ant Hoc Net

- Hybrid multi-path algorithm, consisting of reactive and proactive elements
- Doesn't maintain routes to all destinations at all times (as in wired versions)
- Sets up paths when they are needed at the start of a data session in **reactive set-up phase**:
  - **Reactive** forward ants launched by source to find a source path to the destination
  - **Backward** ants return to source to set up the paths
  - Paths set up in the form of **pheromone tables** indicating their quality

# AntHocNet

- After route set up, data-packets **routed stochastically** over different paths using the tables
- While routing is going on, **proactive forward ants** monitor, maintain and improve the paths
- The algorithm reacts to **link-failures** with either local-repair or by warning preceding nodes on the path

# Reactive Path Setup

- If a source node  $s$  wants to communicate with a destination  $d$  it broadcasts a reactive forward ant  $F$
- Each node has a pheromone table indicating the goodness of going from  $i$  to  $n$  to reach  $d$ 
  - Next hop chosen probabilistically according to pheromone
- If no pheromone available, ant is *broadcast*
  - *Proliferates over network, following different paths*

# Reactive Path Setup

- When reactive ants arrive at  $d$ , they are converted to backward ants and travel back to  $s$
- Reactive ants retrace exact path taken by forward ant
- Pheromone tables are updated according to estimated time and number of hops to source at each node

# Stochastic Data Routing

- The reactive path setup creates a number of good paths
- Data can be then forwarded according to pheromone entries
- Data is forwarded stochastically:
  - If a node has multiple next-hops, probabilistically choose according to pheromone (biased to be more greedy towards good paths)
  - Probability can be adjusted via a weighting parameter to favour load-spreading instead

# Proactive Ant Maintenance

- Pheromone **diffusion** is used to spread pheromone information across the network through periodic update messages
  - Executed by all nodes throughout their lifetime
- Proactive ant **sampling** is aimed at controlling and updating pheromone information through path sampling using proactive forward ants
  - Initiated by source node, continues while data session in process

# Proactive diffusion

- Use hello messages:
  - Identify immediate neighbours
  - Detect link-failures
- But the messages also carry pheromone information
  - Allows pheromone deposited by the reactive ants to be spread across the network
  - This is known as virtual pheromone

# Proactive Sampling

- Initiated by source nodes that send out proactive forward ants
- They follow either real or virtual pheromone:
  - Real pheromone allows the ants to update the goodness of existing routes
  - Virtual pheromone allows the ants to find new routes using ‘hints’ provided by the diffused pheromone
- Allows single route constructed by reactive ant to be extended to multiple paths

# Link Failures

- Each ant tries to maintain an updated view of its neighbours so it can detect link failures
  - Detect presence via receipt of ‘hello’ messages or other message exchange
  - If no such event in time  $t$ , or failure of a unicast-message, assume disappearance
- When this occurs:
  - Node removes neighbour from list
  - Broadcasts link-failed which contains list of destinations to which node has lost its best path and best estimated delay and # hops to destination
  - Tries to locally repair path if no other options