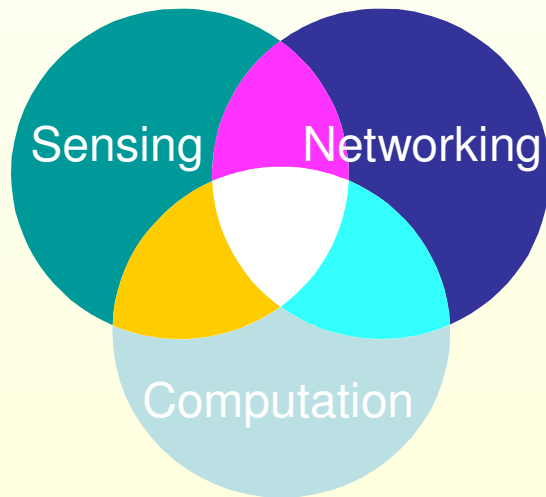
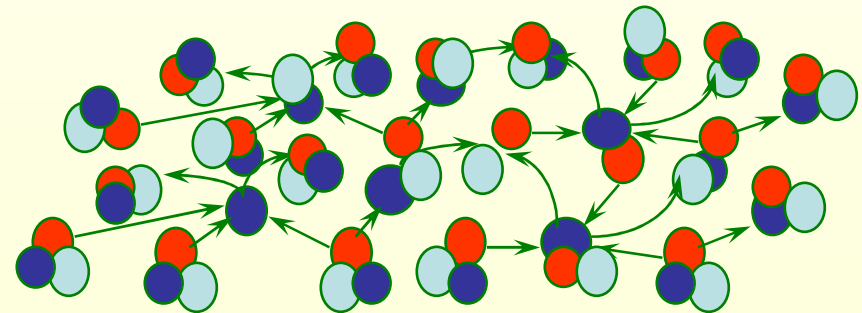


# Networking Sensors, I

---



Leonidas Guibas  
Stanford University



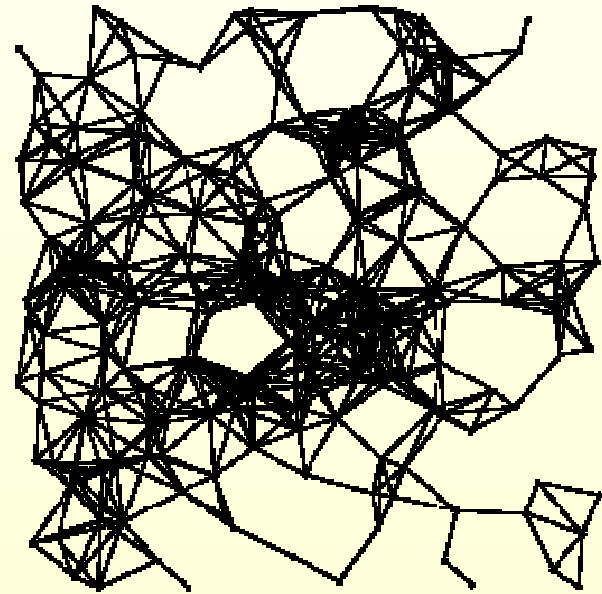
CS428

# Networking Sensors

- Networking is a crucial capability for sensor networks -- networking allows:
  - Placement of sensors close to signal sources
  - Collaborative information processing
  - Synchronization and localization
- But it is also one of the most demanding:
  - radio communication consumes the most energy
  - nodes and especially links can be unreliable and unstable; this must be mitigated by the network protocol stack

# Common Assumptions Made

- Radio range is a disk of radius  $r$  around each node. Node connectivity graph can be modeled as the *unit distance graph* (UDG) by setting  $r = 1$
- Network deployment is ad hoc, so node layout does not follow any particular topology
- Nodes operate untethered and have limited power resources. Communication must be used sparingly
- Typical sensor nodes have no mobility
- Nodes know their geographic location



# The Lower Layers of the Network Protocol Stack

# Medium Access Control

- The MAC layer of the protocol stack is concerned with the reliable transfer of information across individual physical links
- In the WSN case, the links utilize a shared medium where **contention** may occur
- WSNs have several unique characteristics that differentiate them from other networks
  - fairness issues at the node level are less important
  - events of interest typically occur in a sporadic and episodic manner; thus high variance has to be dealt with in node usage, latency times, etc.
  - The relatively fixed node topology can be exploited in synchronizing node sleep schedules, so as to reduce collision and overhearing overhead, and therefore energy use.

# SMAC [Ye, Heidemann, Estrin]

- To reduce latency and control overhead, SMAC tries to coordinate sleep schedules among neighboring nodes by periodic schedule exchanges. In general SMAC reduces
  - Collision overhead – by using RTS and CTS
  - Overhearing – by switching the radio off when the transmission is not meant for that node
  - Control overhead – by message passing
  - Idle listening – by periodic listen and sleep
- Significantly more efficient than 802.11, especially in settings with long latency and message inter-arrival times

# Periodic Listen and Sleep

- If no sensing event happens, nodes are idle for a long time, so it is not necessary to keep the nodes listening all the time
- Each node go into periodic sleep mode during which it switches the radio off and sets a timer to awaken itself later
- When the timer expires, the node wakes up and listens to see if any other node wants to talk to it

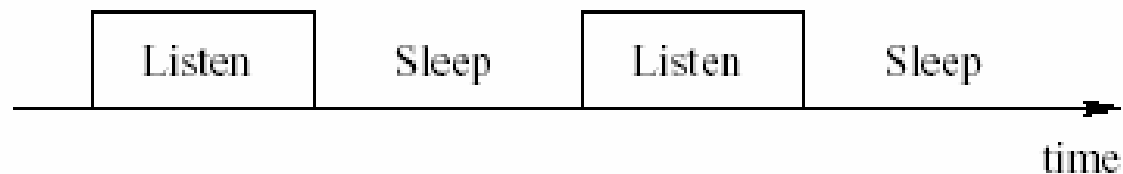


Fig. 1. Periodic listen and sleep.

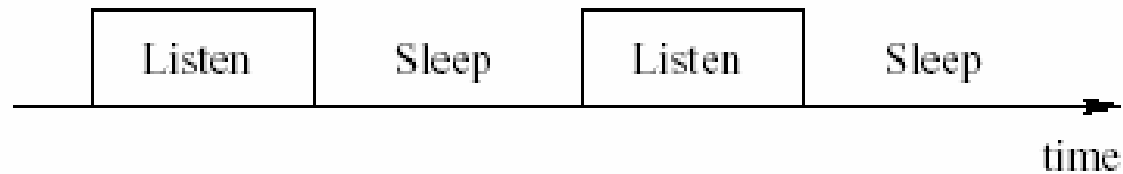


Fig. 1. Periodic listen and sleep.

- Duration of sleep and listen time can be selected based on the application scenario
- To reduce control overhead, neighboring nodes synchronize (i.e., nodes listen and sleep together) via the exchange of sleep schedules





Fig. 2. Neighboring nodes A and B have different schedules. They synchronize with nodes C and D respectively.

- Not all neighboring nodes can synchronize together
- Two neighboring nodes (A and B) can have different schedules, if they are required to synchronize with different nodes
- Note: sleep/listening times are long compared to typical clock drift between nodes – fine grain synchronization is not required (as in TDMA)

- If a node A wants to talk to node B, it just waits until B is listening
- If multiple neighbors want to talk to a node, they need to contend for the medium
- Contention mechanism is the same as that in IEEE 802.11 (using RTS and CTS)
- Nodes, after starting data transmission, do not go to periodic sleep until they finish transmission

# Choosing and Maintaining Schedules

- Each node maintains a schedule table that stores the schedules of all its known neighbors.
- To establish the initial schedule (at startup), the following steps are followed:
  - A node first listens for a certain amount of time.
  - If it does not hear a schedule from another node, it randomly chooses a schedule for itself and broadcast its schedule immediately.
  - This node is called a SYNCHRONIZER (leader).

- If a node receives a schedule from a neighbor *before* choosing its own schedule, it just follows this neighbor's schedule.
- This node is called a FOLLOWER -- it waits for a random delay (to reduce medium contention) and then broadcasts its schedule.
- If a node receives a neighbor's schedule after it selects its own schedule, it adopts both schedules (by merging their listening times).

# Rules for Adding a New Node

- Listen for a long time until an active node is discovered
- Send INTRO packet to the active node
- Active node forwards its schedule table
- Treat all the nodes on table as potential neighbors and contact them later
- If possible, follow the synchronizer's schedule, else establish a random schedule and broadcast that schedule

# Maintaining Synchronization

- Timer synchronization among neighbors is needed to prevent the clock drift.
- Done by periodic updating, using a SYNC packet (next time to sleep).
- Updating period can be quite long -- as we don't require tight synchronization.
- Synchronizer needs to periodically send SYNCs to its followers.
- If a follower has a neighbor that has a different schedule from it, it also needs to update that neighbor.

## SYNC Packet



- Time of next sleep is relative to the moment that the sender finishes transmitting the SYNC packet
- Receivers will adjust their timer counters immediately after they receive the SYNC packet
- Listen interval is divided into two parts: one for receiving SYNC and other for receiving RTS

# Timing Relationship of Possible Situations

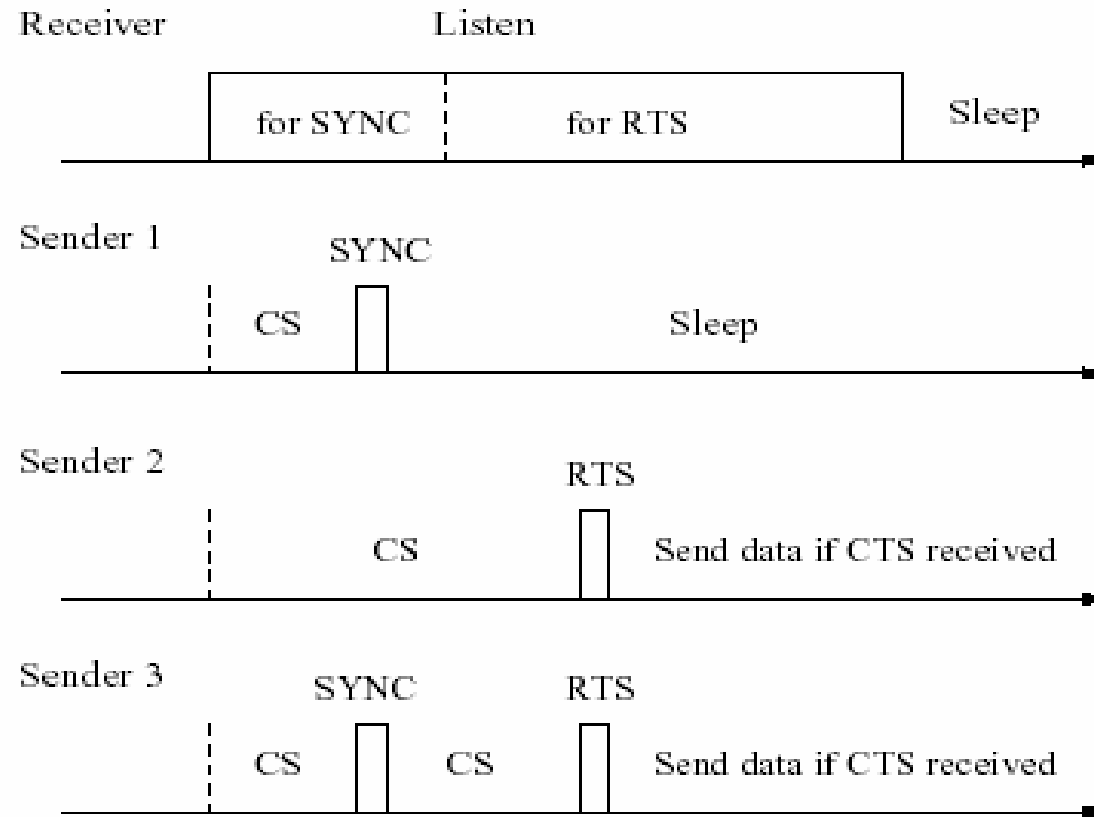


Fig. 3. Timing relationship between a receiver and different senders. CS stands for carrier sense.



# Collision Avoidance

- Similar to IEEE 802.11 using RTS/CTS mechanism
- Perform carrier sense, before initiating a transmission
- If a node fails to get the medium, it goes to sleep and wakes up when the receiver is free and listening again
- Broadcast packets are sent without RTS/CTS
- Unicast packets follow the sequence of RTS/CTS/DATA/ACK between the sender and receiver

# Overhearing Avoidance

- Duration field in each transmitted packet indicates how long the remaining transmission will be.
- So if a node hears a packet destined to another node, it knows how long it has to keep silent.
- The node records this value in network allocation vector (NAV) and sets a timer.

- When a node has data to send, it first looks at NAV.
- If NAV is not zero, then the medium is busy (virtual carrier sense).
- Medium is determined as free if both virtual and physical carrier sense indicate the medium is free.
- All immediate neighbors of both the sender and receiver should sleep after they hear RTS or CTS packet until the current transmission is over.

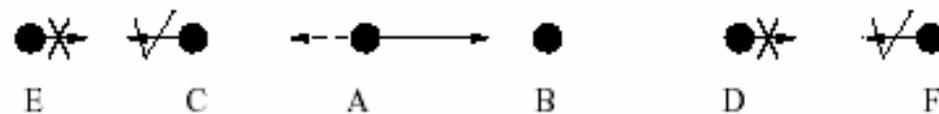


Fig. 4. Who should sleep when node A is transmitting to B?

# Message Passing

- A message is a collection of meaningful, interrelated units of data
- Transmitting a long message as a single packet is potentially expensive, as the re-transmission cost is high if a failure happens
- On the other hand, fragmentation into small packets will lead to high control overhead as each packet should contend using RTS/CTS

# SMAC Solution

- Fragment message in to small packets and transmit them as a burst (but wait for ACK after each packet)
- Advantages
  - Reduces latency of the message
  - Reduces control overhead
- Disadvantage
  - Node-to-node fairness is reduced, as nodes with small packets to send have to wait till the message burst is transmitted

# SMAC Performance

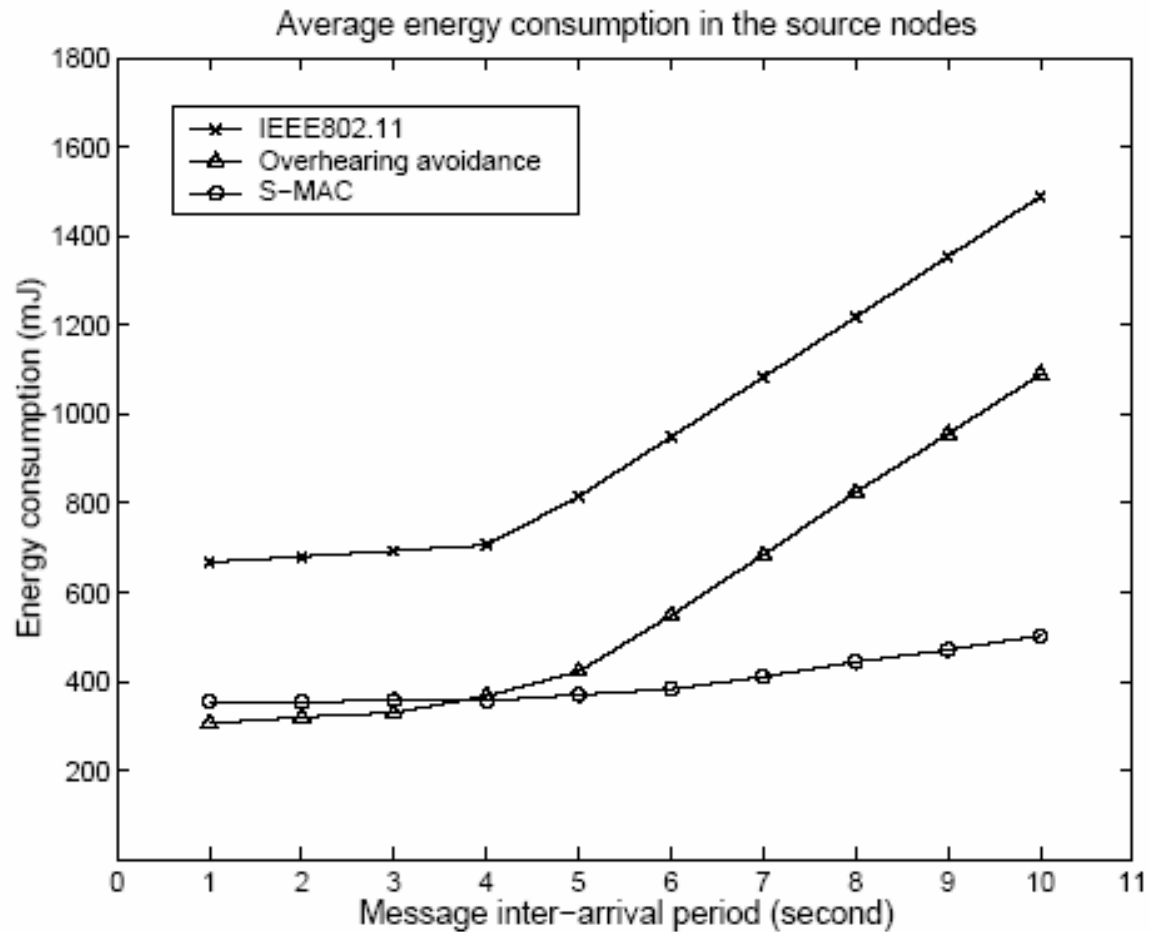
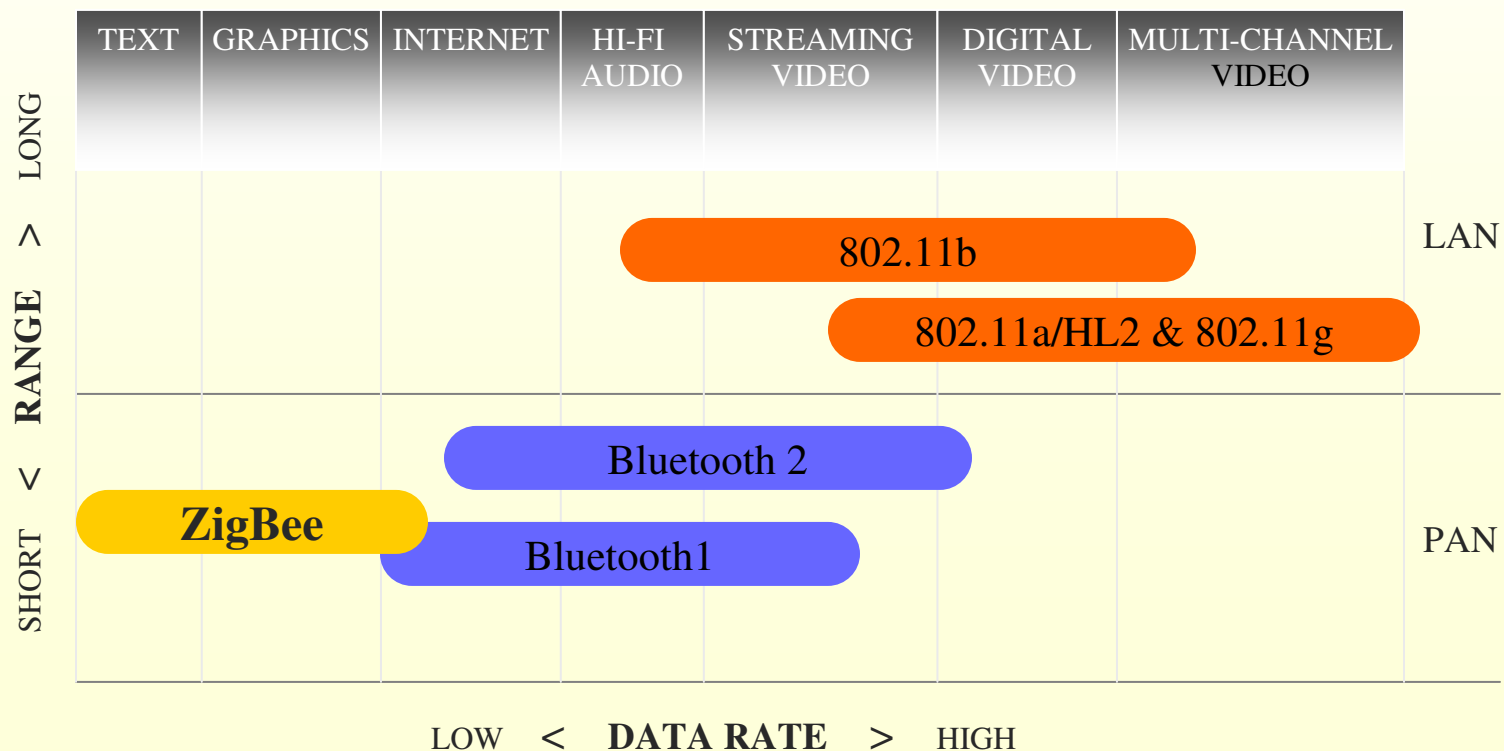


Fig. 8. Measured energy consumption in the source nodes.

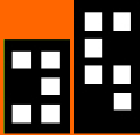
# IEEE 802.15.4 (ZigBee)

- Targeted at home and building automation and controls, consumer electronics, PC peripherals, medical monitoring, and toys




# ZigBee Application Space

security  
HVAC  
AMR  
lighting control  
access control



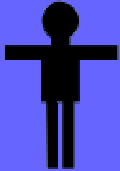
**BUILDING  
AUTOMATION**




TV  
VCR  
DVD/CD  
remote

**CONSUMER  
ELECTRONICS**

patient  
monitoring  
fitness  
monitoring




**PERSONAL  
HEALTH CARE**



mouse  
keyboard  
joystick

**PC &  
PERIPHERALS**

asset mgt  
process control  
environmental  
energy mgt



**INDUSTRIAL  
CONTROL**



**RESIDENTIAL/  
LIGHT  
COMMERCIAL  
CONTROL**

security  
HVAC  
lighting control  
access control  
lawn & garden irrigation



# IEEE 802.15.4 Basics

- 802.15.4 is a simple packet data protocol for lightweight wireless networks
  - Channel Access is via Carrier Sense Multiple Access (CSMA) with collision avoidance and optional time slotting
  - Message acknowledgement and an optional beacon structure
  - Multi-level security
  - Three bands, 27 channels specified
    - 2.4 GHz: 16 channels, 250 kbps
    - 868.3 MHz : 1 channel, 20 kbps
    - 902-928 MHz: 10 channels, 40 kbps
  - Range varies with power, up to hundreds of meters
  - Works well for
    - Long battery life, selectable latency for controllers, sensors, remote monitoring and portable electronics
  - Configured for maximum battery life, has the potential to last as long as the shelf life of most batteries

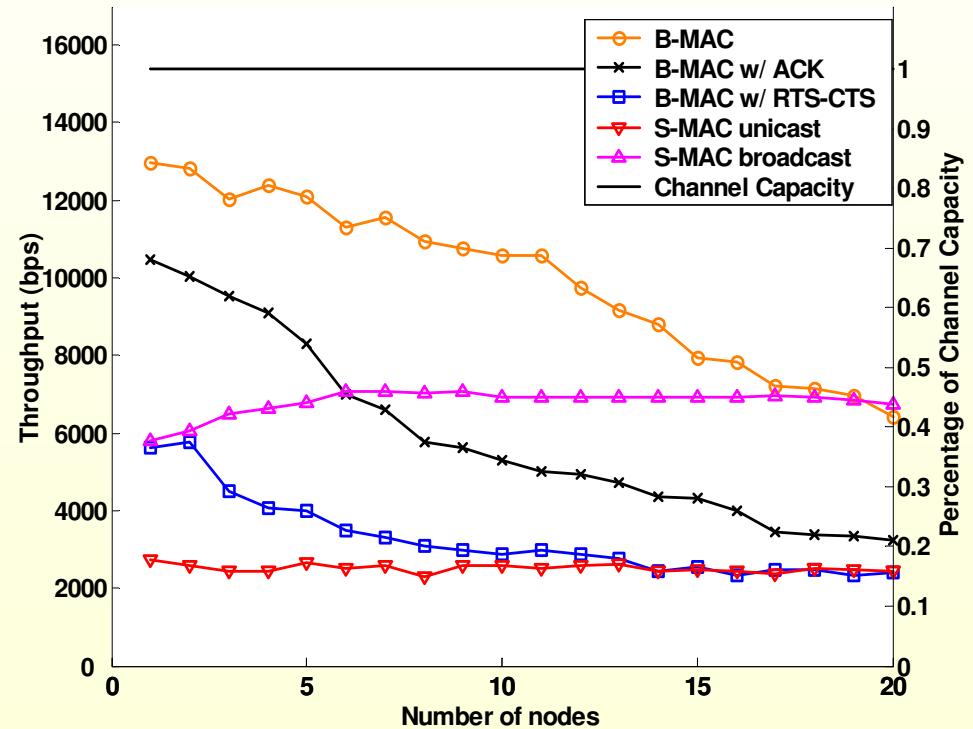
Zigbee motes operate at 250kbs, range 20-30 m indoors,  
75-100 m outdoors

# Even Newer Protocols

## ● B-MAC (Polastre, Hill, Culler – Sensys04)

● “B-MAC is about 4.5 faster than S-MAC-unicast”

- Not as fast when ACK or RTS/CTS is used
- No combined results...
- Differences less pronounced as # of nodes increases



# Routing Information in Sensor Networks

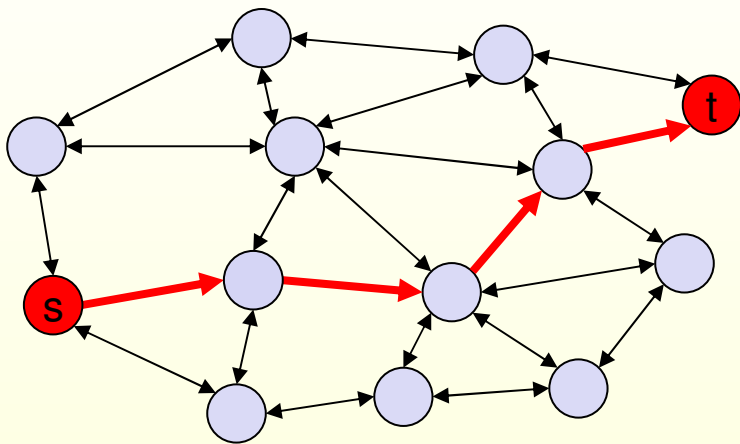
# Routing in Sensor Networks

- Routing protocols in communication networks obtain route information between pairs of nodes wishing to communicate. Such protocols can be
  - **proactive**: the protocol maintains routing tables at each node that are updated as changes in the network topology are detected
  - **reactive**: the protocol constructs paths on demand only
- Because of the high rate of topology changes, reactive protocols are much more appropriate for sensor networks
- Several such protocols have already been developed for ad hoc mobile communication networks. Examples are:
  - Ad hoc on demand distance vector routing (AODV)
  - Dynamic source routing (DSR)
    - both, however, flood the network to discover paths

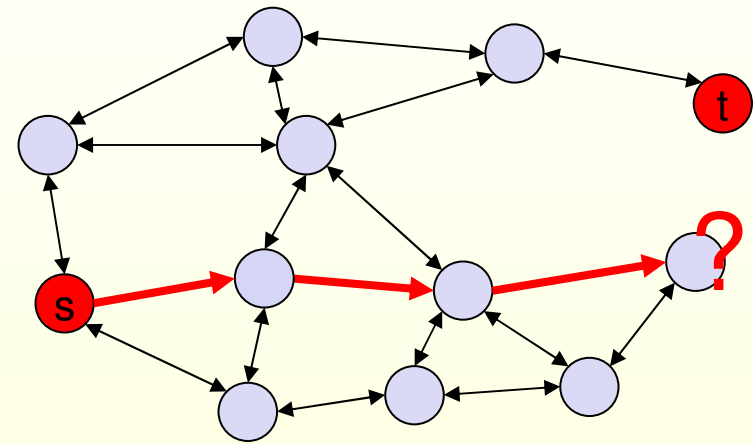
# Geographic Routing

- In sensor networks routing is frequently based on a node's attributes and sensed data, rather than on some pre-assigned network address.
- **Geographic routing** uses a node's **location** to discover paths to that node
- We assume that
  - nodes know their geographic location
  - nodes know their 1-hop neighbors
  - routing destinations are specified geographically (a point, a region)
  - each packet can hold a small amount ( $O(1)$ ) of additional routing info to record where it has been on the network
  - most of the time we will model the connectivity graph of the nodes as a unit distance graph

# Greedy Methods



In a greedy method, each node forwards a packet to its best neighbor



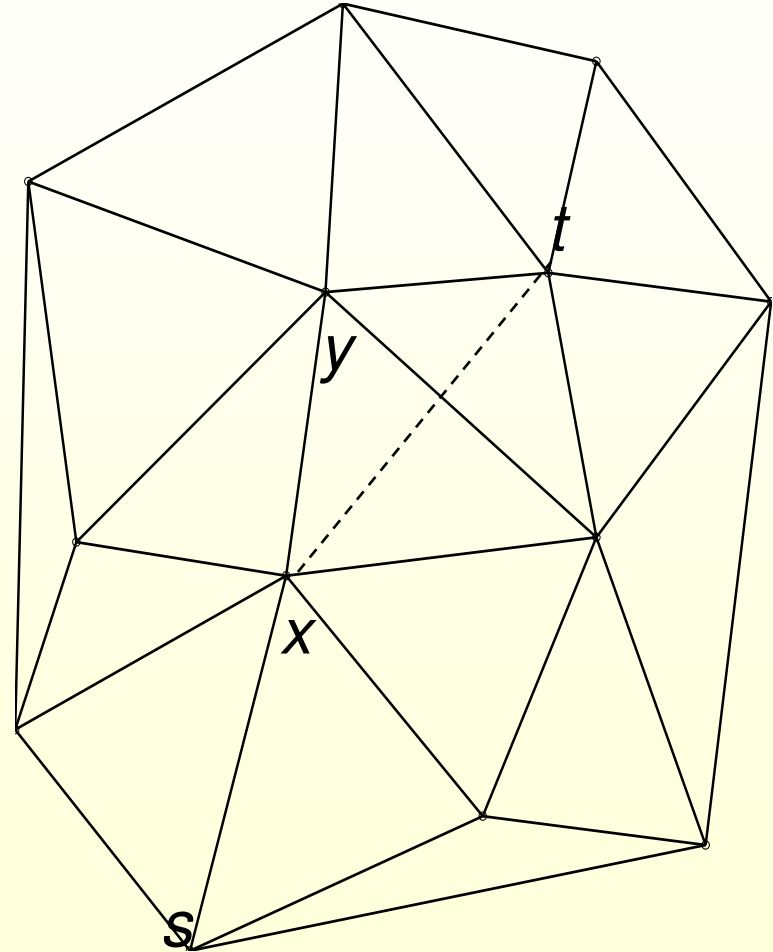
Greedy methods can get stuck at “dead-ends”

Note that no flooding is involved for route discovery

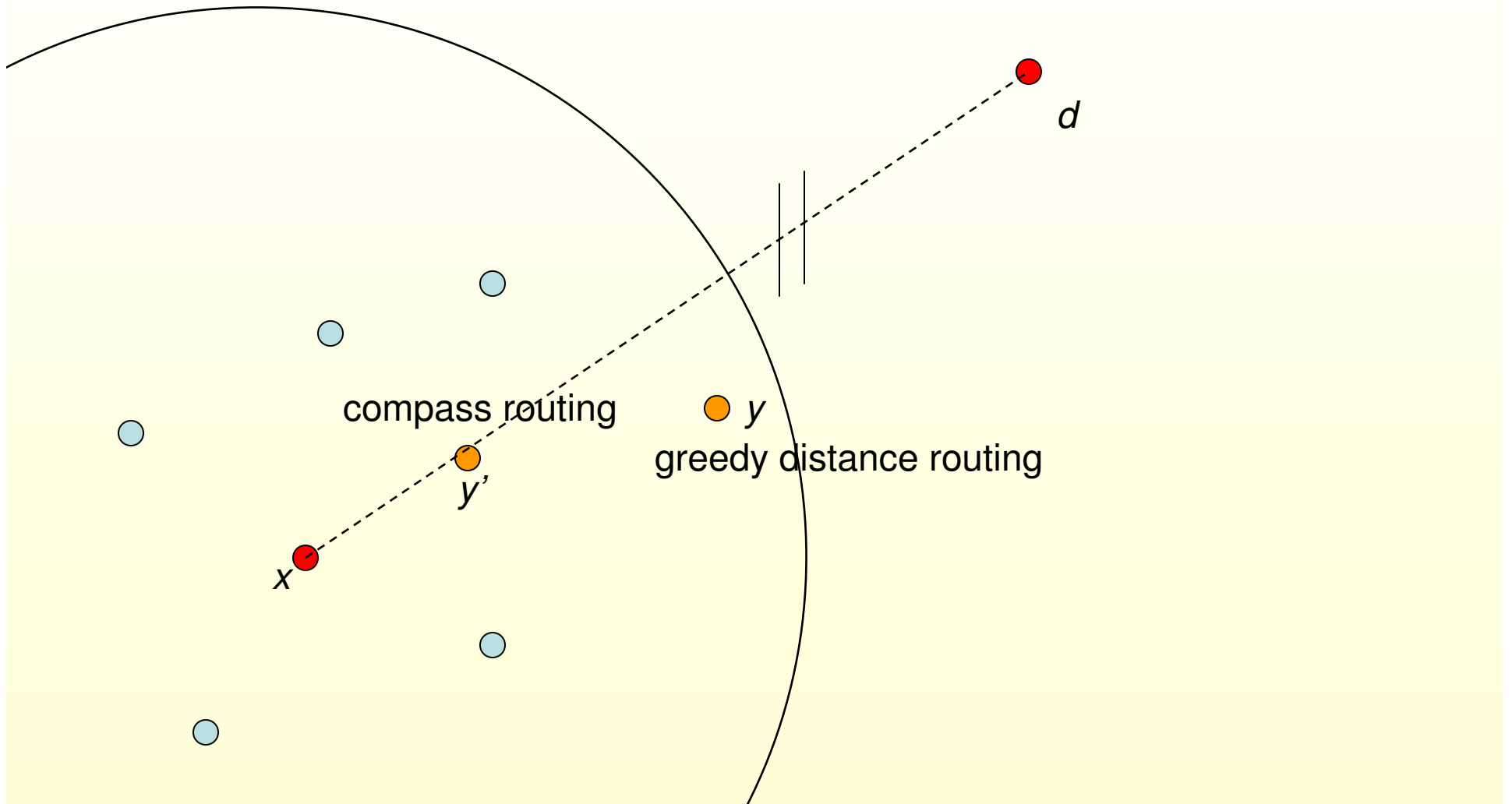
# Greedy Unicast Geographic Routing

To go from source  $s$  to destination  $t$ , at each intermediate node  $x$  advance to the neighbor  $y$  that makes most progress towards  $t$ .

- greedy distance routing (GPSR)
- compass routing



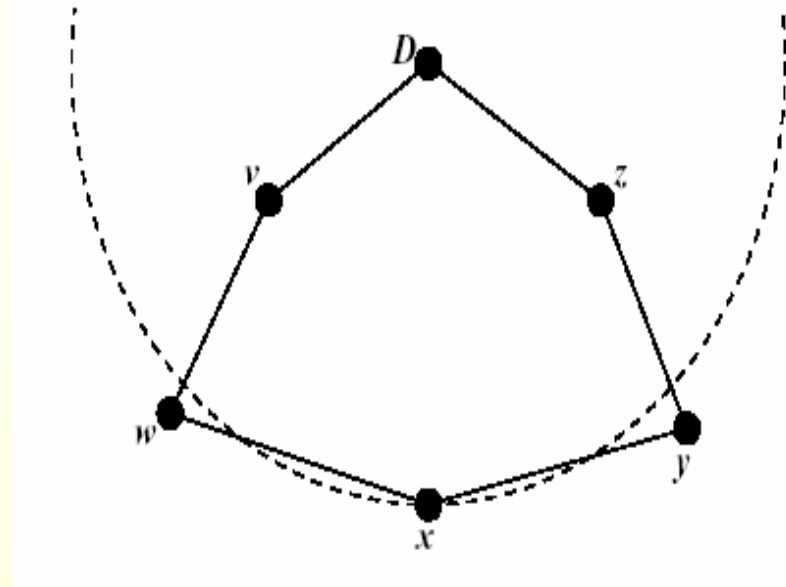
# Neighbor Choice





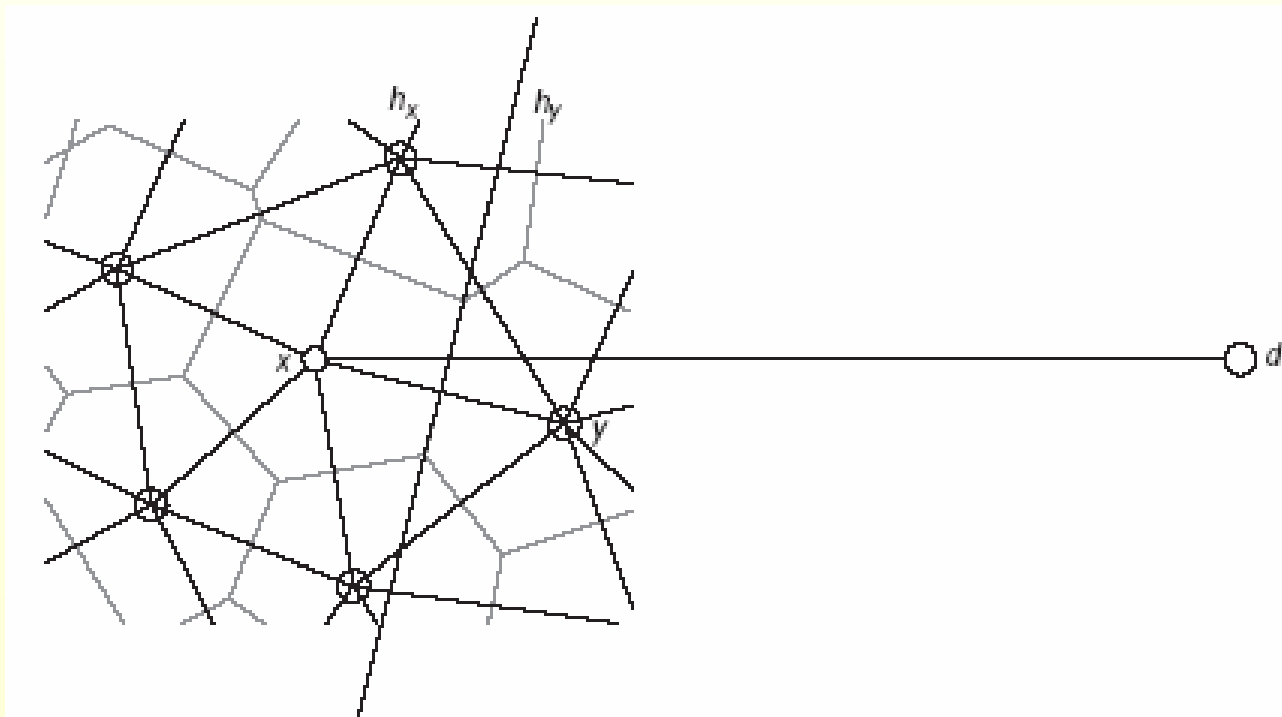
# Greedy Protocols Can Get Stuck

- The intermediate node  $x$  can be a **local optimum** towards the destination
- In general, local optima will arise if the node graph contains “holes” – areas with no sensor nodes
- To prove that such situations cannot happen we need to assume special properties about the connectivity graph  $G$



# Delaunay Triangulations (DT)

- In a Delaunay triangulation (dual to the Voronoi diagram of the nodes), packets cannot get stuck
- However, unless the nodes are spaced very closely, it is unlikely that the UDG will contain all DT edges



# Measures of Path Quality

- First and foremost, a protocol should **guarantee packed delivery**, whenever such delivery is possible
- Second, the **quality of the path** produced should be good when compared to the optimal path available. Different path costs can be used:

$$c(\pi) = \sum_{e \in \pi} l^d(e),$$

$$d=0,1,2,3,4,\dots$$

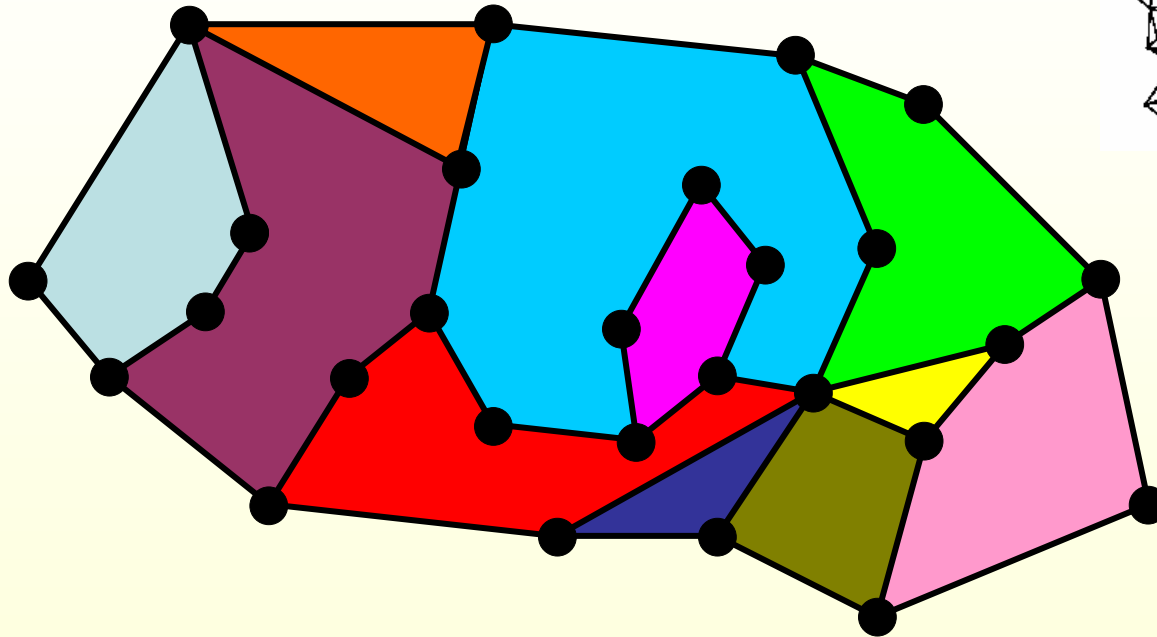
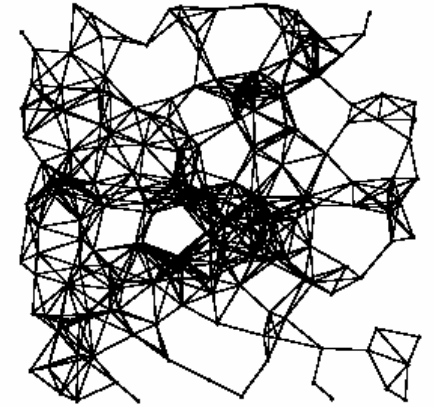
$d = 0$ , hop length

$d = 1$ , normal path length

$d = 2, 3, 4 \dots$ , energy costs

- These can be made roughly equivalent by assuming a constant node density or a minimum node spacing
  - This can be attained by a node clustering process

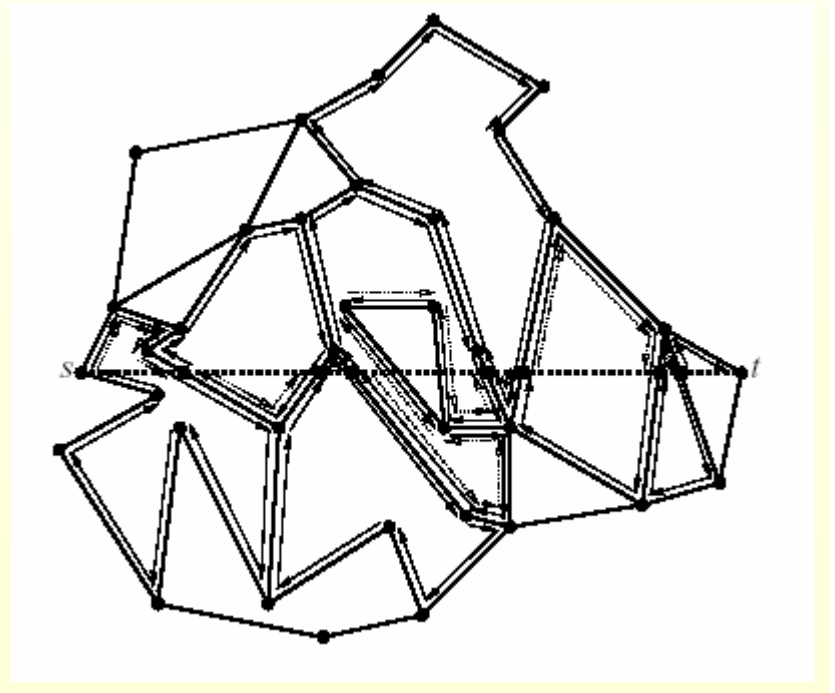
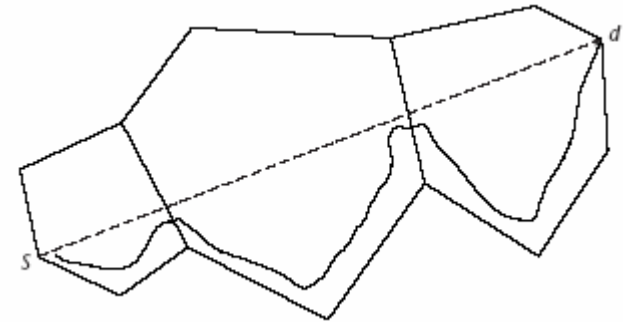
# Planarizations



A planar straight-line graph has no crossing edges. It subdivides the plane into regions called faces.

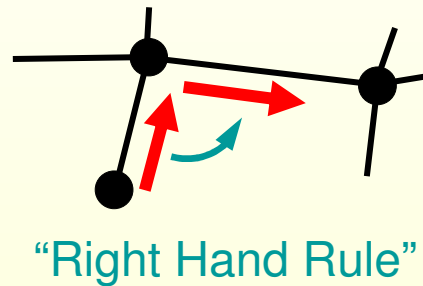
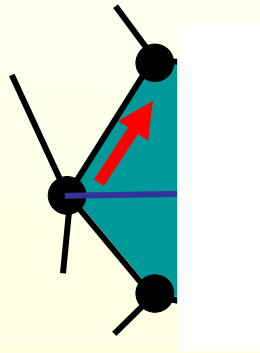
# Planarizations of the Routing Graph

- To guarantee packet delivery, *it may be advantageous to disable some connections*, so as to make the routing graph planar
- On a planar graph, **perimeter routing** guarantees delivery
- Another variant is **other face routing**
- The quality of paths can be bad, however



# Perimeter/Face Routing Properties

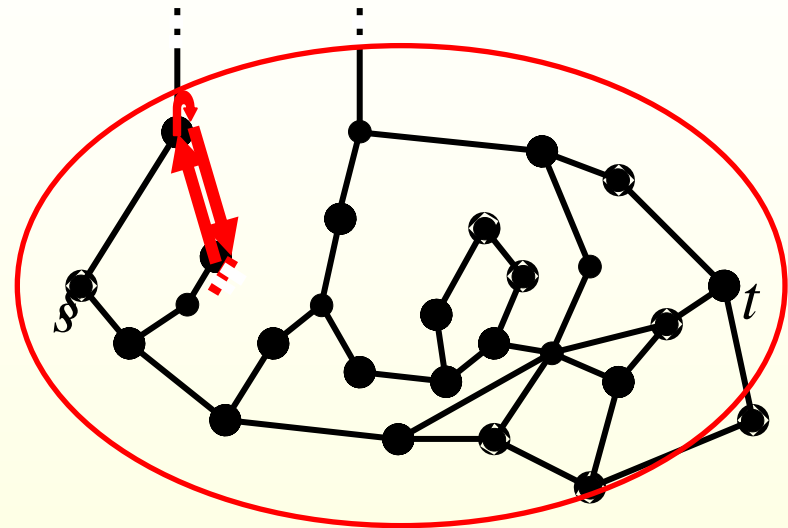
- All necessary information is stored in the message
  - Source and destination positions are given
  - Point of transition to next face needs to be chosen



- Completely local:
  - Knowledge about direct neighbors' positions sufficient
  - Faces are **implicit**, only local neighbor ordering around each node is needed

# Adaptive Algorithms

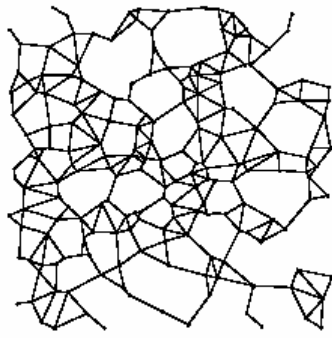
- We want the quality of paths we discover to be nearly optimal
- Alternatively, we want to discover optimal paths without searching the whole connectivity graph  $G$
- If the optimal path between  $s$  and  $t$  has length  $L$ , then every node in that path is within an ellipse with foci  $s$  and  $t$  defined by  $L$ . This ellipse limits the part of  $G$  to be searched.
- If  $L$  is not known, it can be guessed, approximately



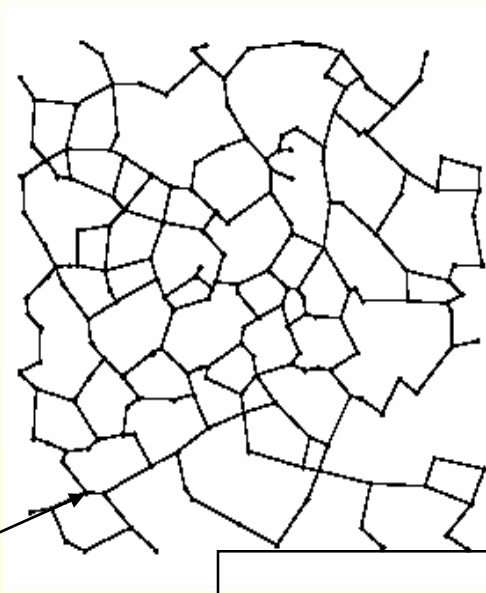
# Geometric Graphs

- defined by local rules?
- distributed construction?
- path quality (spanning property)?

$$\frac{|\pi^*|}{|st|}$$

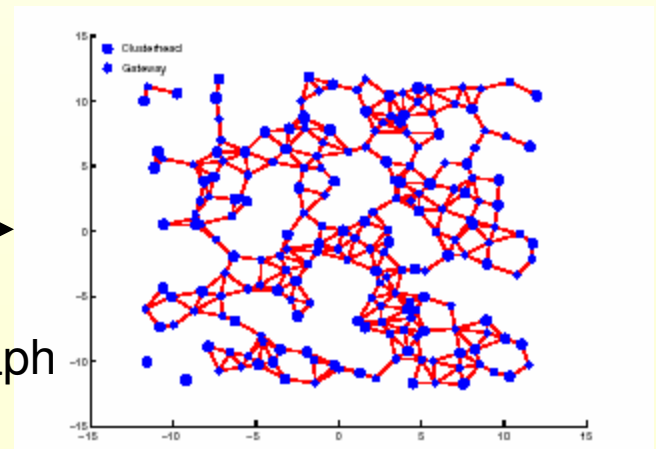
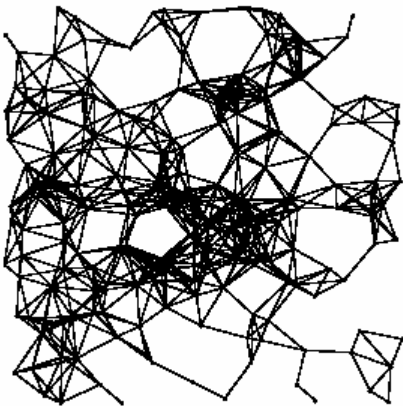


Gabriel Graph



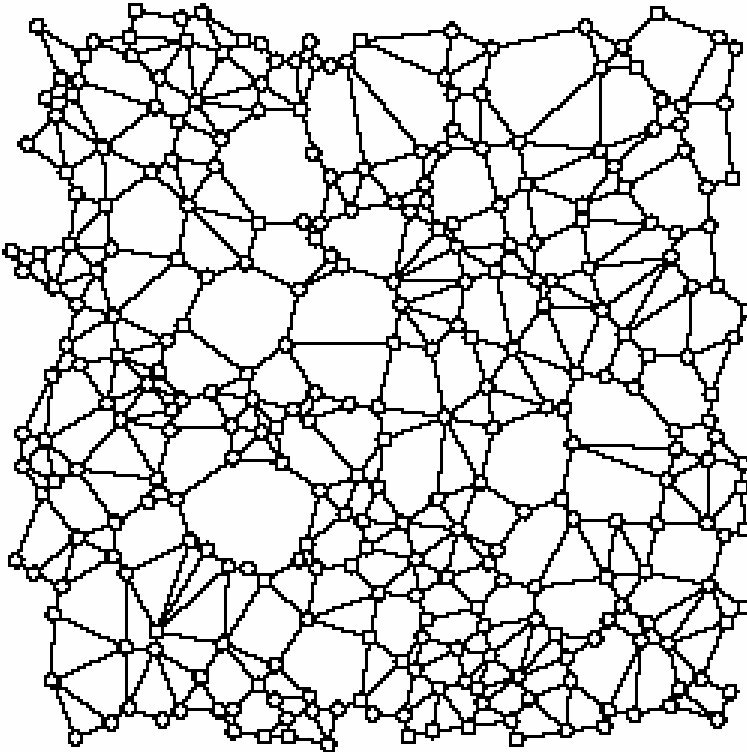
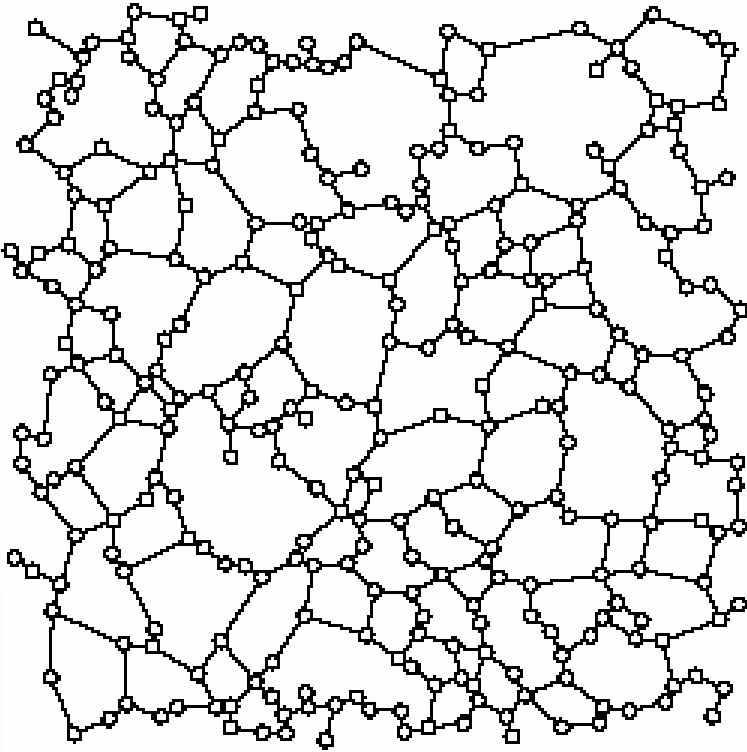
Relative Neighborhood Graph (RNG)

Restricted Delaunay Graph (RDG)

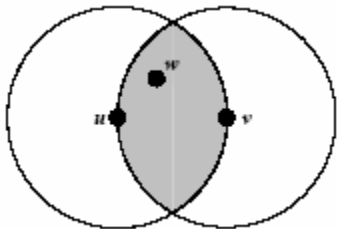




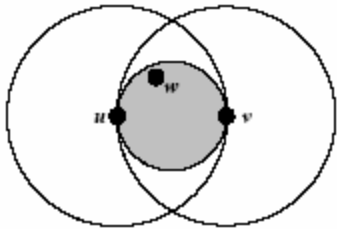
# Larger RNG and GB Examples



Relative Neighborhood Graph

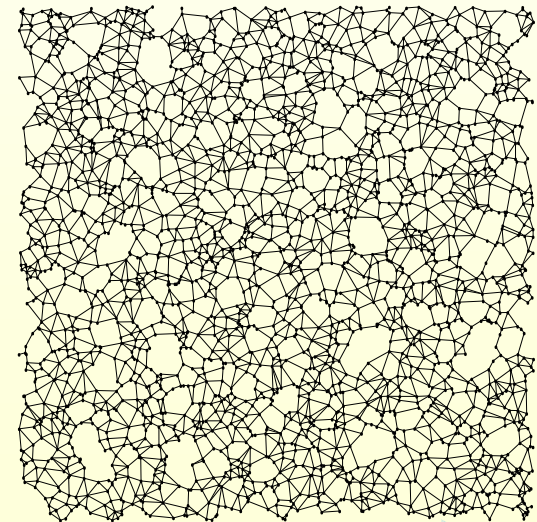
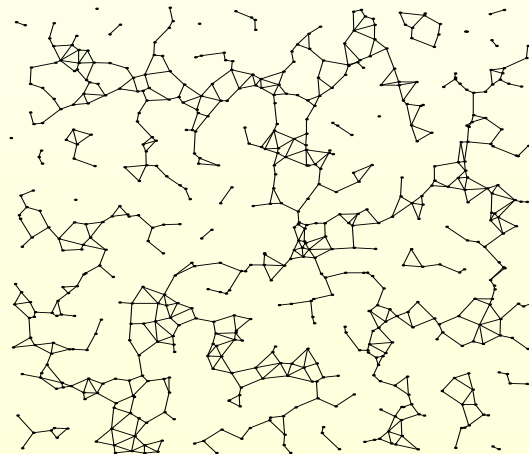
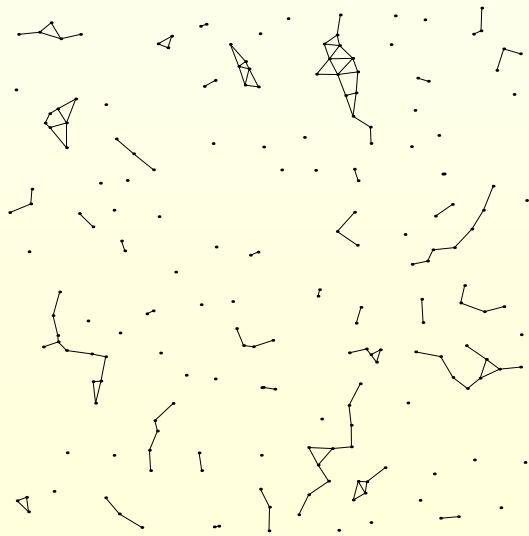


Gabriel Graph



# Average Path Quality

- Not interesting when graph not dense enough
- Not interesting when graph is too dense
- **Critical density range** (“percolation”)
  - Shortest path is significantly longer than Euclidean distance



← too sparse

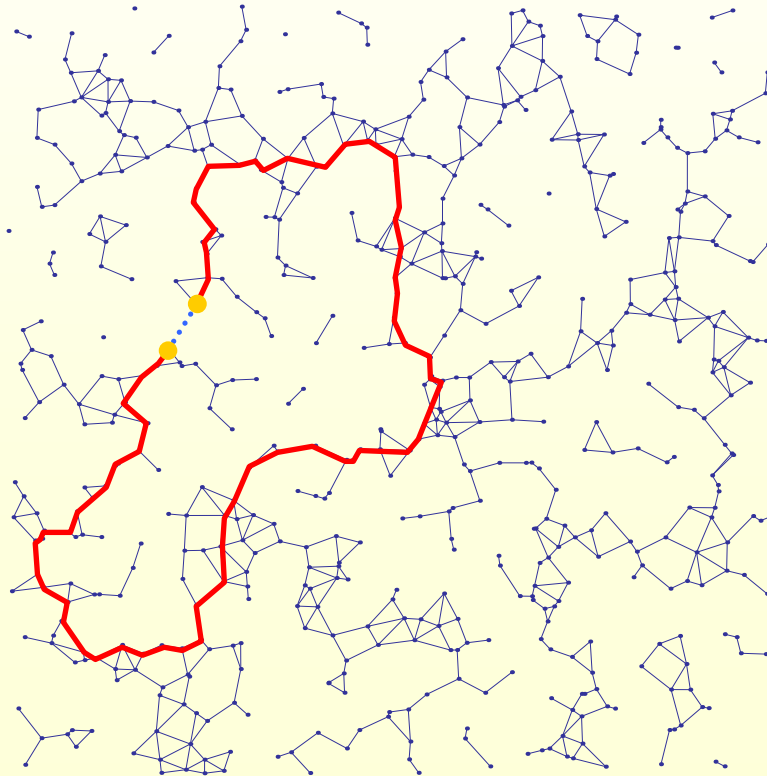
critical density

→ too dense

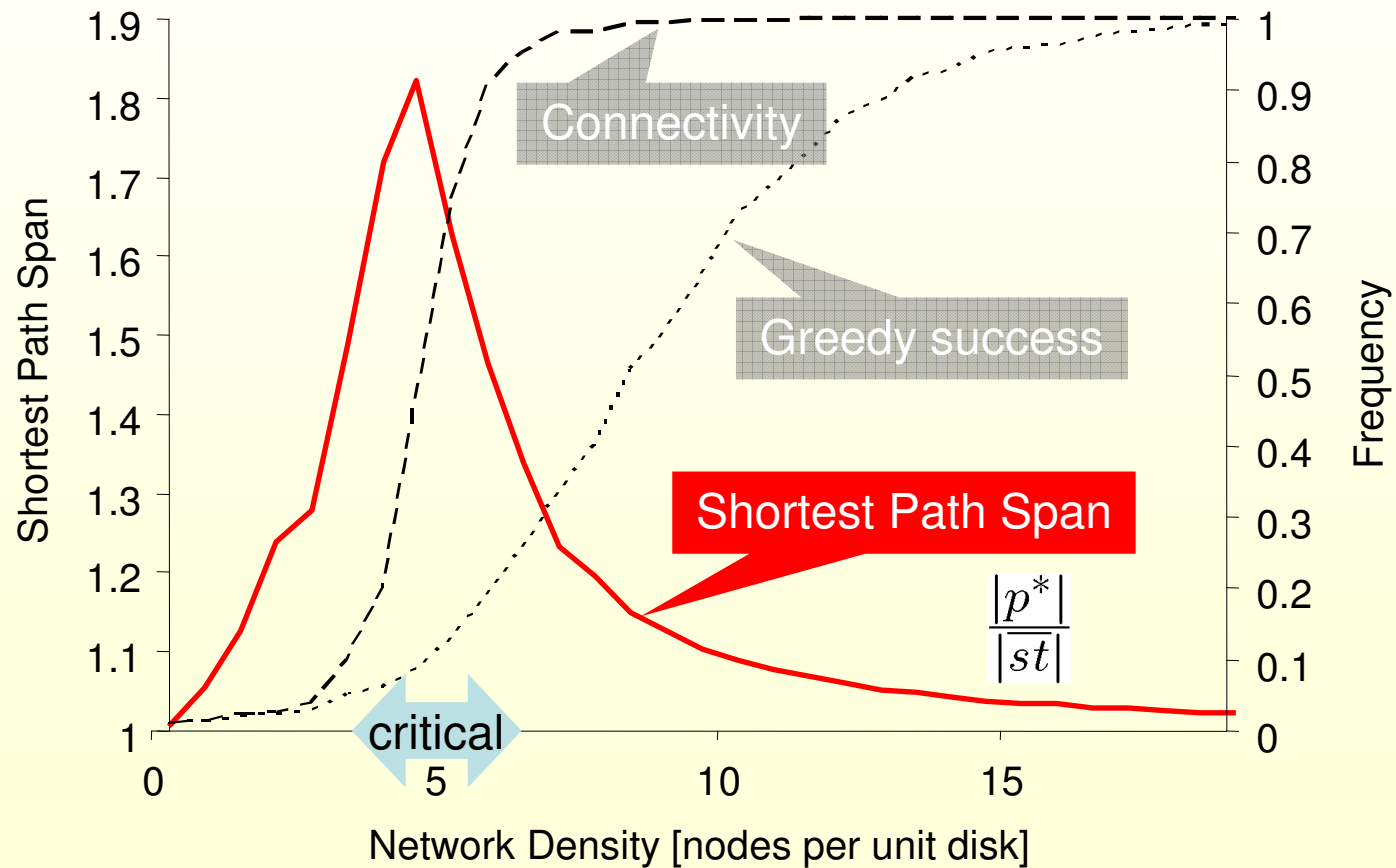
# Critical Density: Shortest Path vs. Euclidean Distance

- Shortest path is significantly longer than Euclidean distance

$$\frac{|\pi^*|}{|st|}$$



# Randomly Generated Graphs: Critical Density Range



# Combining Greedy and Planarization Strategies: Greedy Perimeter Stateless Routing (GPSR)

[Karp and Kung, 2000]: Planarize the connectivity graph  $G$

- Use greedy distance protocol on the full graph  $G$
- If stuck, switch to perimeter protocol on the planarization of  $G$ , until a node closer to the destination than the stuck node is encountered

# GPSR Performance

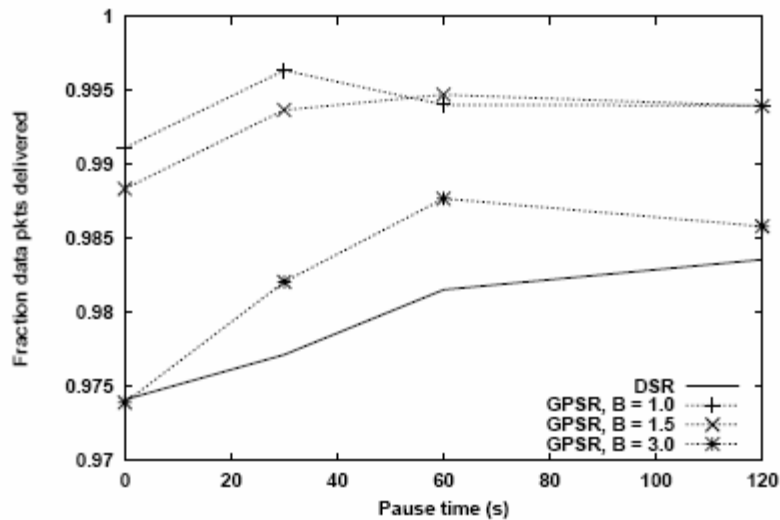


Figure 9: Packet Delivery Success Rate. GPSR with varying beacon intervals,  $B$ , compared with DSR. 50 nodes.

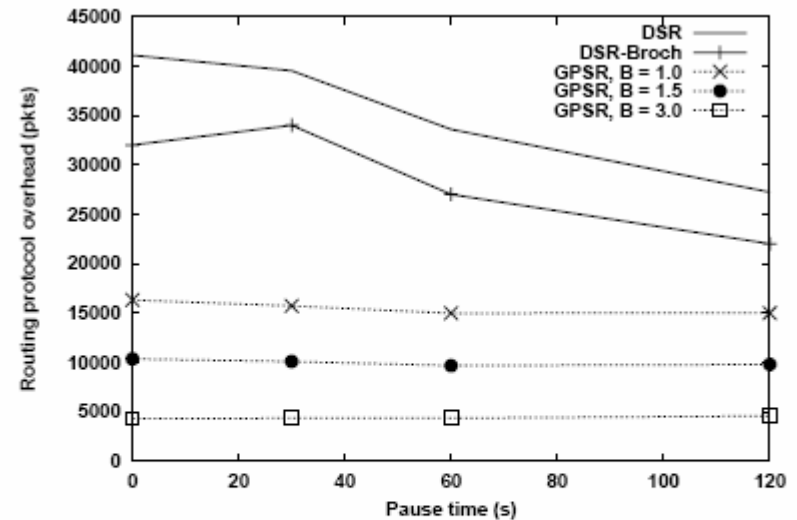


Figure 10: Routing Protocol Overhead. Total routing protocol packets sent network-wide during the simulation for GPSR with varying beacon intervals,  $B$ , compared with DSR. 50 nodes.

# Routing on a Curve

- Packets can be given a trajectory to follow in terms of a parametric curve

$$c(t) = \langle x(t), y(t) \rangle$$

- A sequence of nodes approximating  $c$  can be selected according to various criteria
- Method is robust to node failures

Figure 1: Trajectory following a sine curve

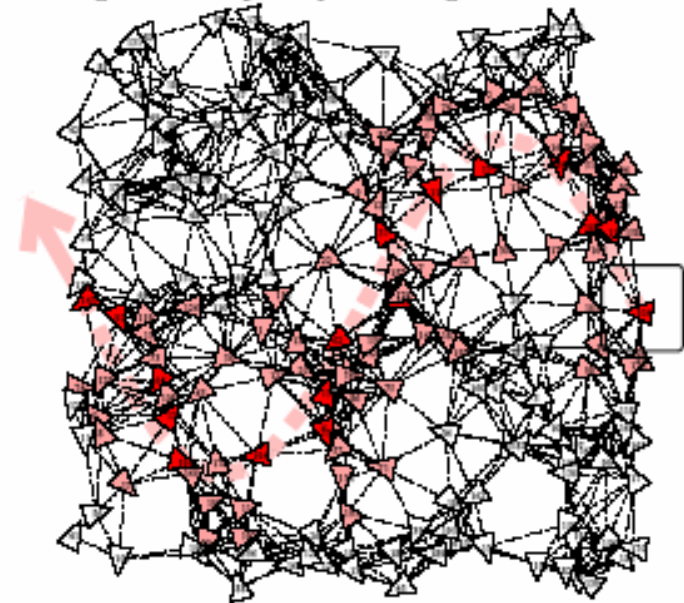
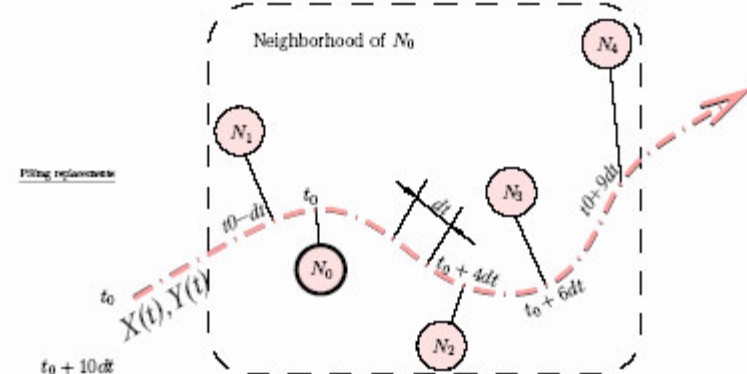


Figure 2: Forwarding on a curve



# Conclusions

- The lower levels of the protocol stack can be adapted to sensor networks so as to minimize collision and overhearing overheads and allow nodes to sleep when they are not needed.
- Knowledge of the nodes' locations enables many powerful mechanisms for message transport and route discovery that avoid expensive flooding operations yet require no routing tables or other high-maintenance data structures.



*The End*