Ad hoc Network Routing

Wireless Systems, a.a 2013/2014

Un. of Rome "La Sapienza"

Chiara Petrioli[†]

[†] *Department of Computer Science – University of Rome "Sapienza" – Italy*

Why traditional approaches have limits?

- Proactive protocols are costly in terms of overhead (the bandwidth and energy are critical resources)
- The cost of maintaining routes updated may not make sense in an environment in which
  - Medium-high mobility
  - Medium-high dynamicity (awake/asleep states)

  Motivate frequent changes in the the optimum route (requiring updates) while
  - Traffic is generally low (so the cost of maintaining always updated routes is not balanced by their use)

  If this is the scenario what can we do?

SAPIENZA
UNIVERSITÀ DI ROMA

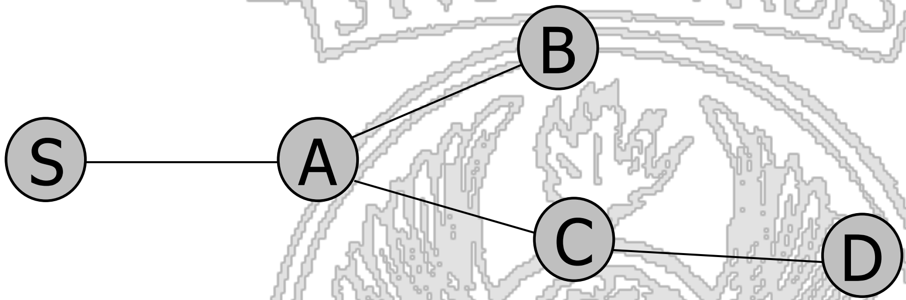
2 Primary Approaches (cont.)

- Reactive (on-demand)
  - Source build routes on-demand by “flooding”
  - Maintain only active routes
  - Route discovery cycle
  - Typically, less control overhead, better scaling properties
  - Drawback: route acquisition latency
  - Example: AODV, DSR

Ad hoc On-Demand Distance Vector Routing

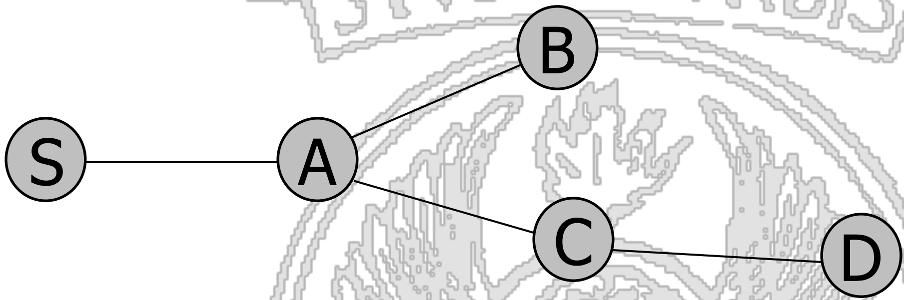
- AODV: Reactive (nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchange; a node does not have to discover/maintain a route to a destination till it is on a path to it or has to send messages to it)
- *Route discovery cycle* used for route finding
- Maintenance of *active* routes
- Sequence numbers used for loop prevention and as route freshness criteria
- Descendant of DSDV (standard distance vector approach mapped to ad hoc networks), in AODV no periodic updates but pure on-demand operation.
- Provides unicast and multicast communication

AODV: Route Discovery



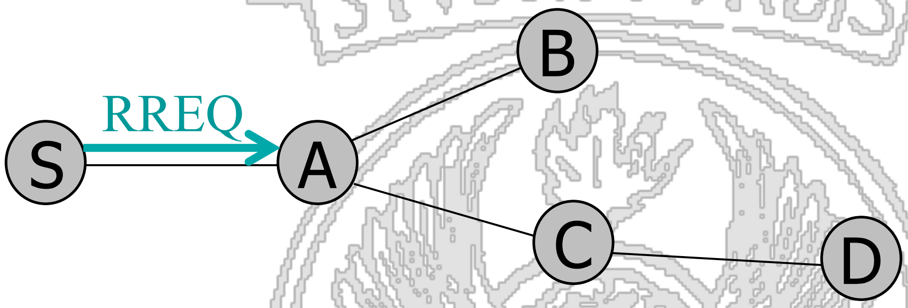
1. Node *S* needs a route to *D AND does not have routing info for it in its table*

AODV: Route Discovery



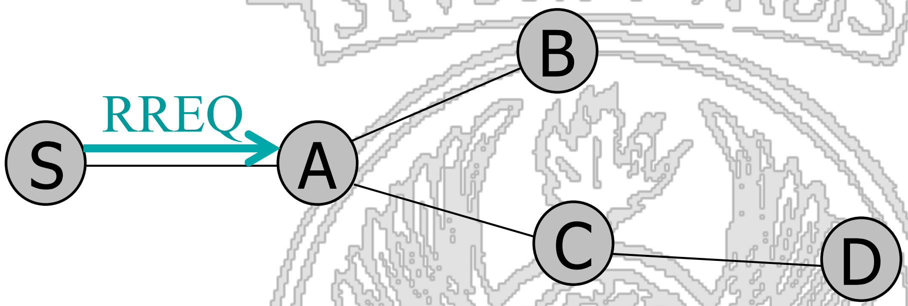
1. Node *S* needs a route to *D*
2. Creates a Route Request (RREQ)
   Enters *D*'s IP addr, seq#,
   S's IP addr, seq#
   hopcount (=0), broadcast ID

AODV: Route Discovery



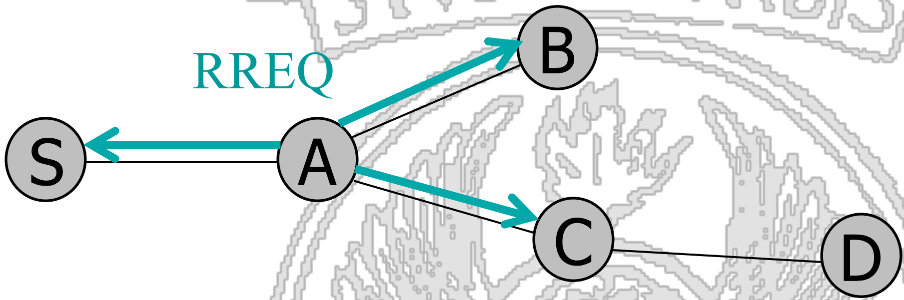
1. Node *S* needs a route to *D*
2. Creates a Route Request (RREQ)
   - Enters *D*'s IP addr, seq#,
   - *S*'s IP addr, seq#
   - hopcount (=0), broadcast ID
3. Node *S* broadcasts RREQ to neighbors

AODV: Route Discovery



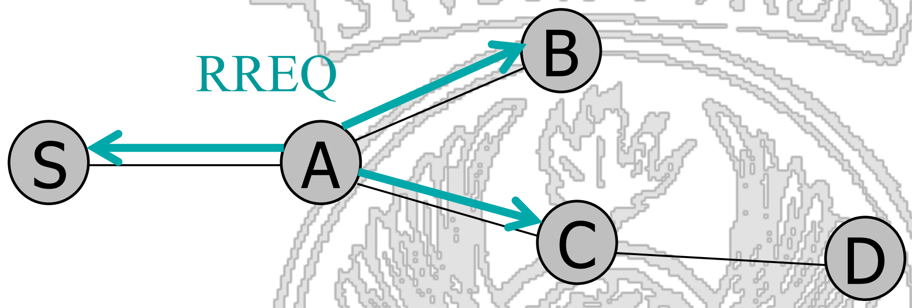
4. Node *A* receives RREQ
   - Makes reverse route entry for *S*
     dest=*S*, nexthop=*S*, hopcnt=1,expiration time for reverse path
   Source node, Source node SN,D,broadcastID also maintained
   - It has no route to *D*, so it rebroadcasts RREQ (hopcount increased)
   - If it has already received that request (same source and broadcast ID) it discards the RREQ
   - if it knows a valid path to D it will send back a reply to the source

AODV: Route Discovery



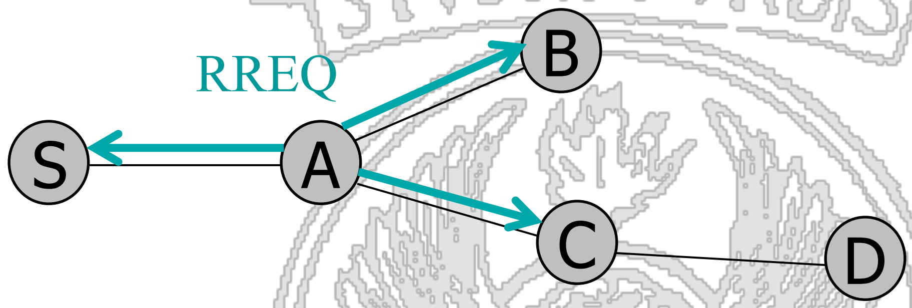
4. Node *A* receives RREQ
   - Makes reverse route entry for *S*
     dest=*S*, nexthop=*S*, hopcnt=1
   - It has no route to *D*, so it rebroadcasts RREQ

AODV: Route Discovery



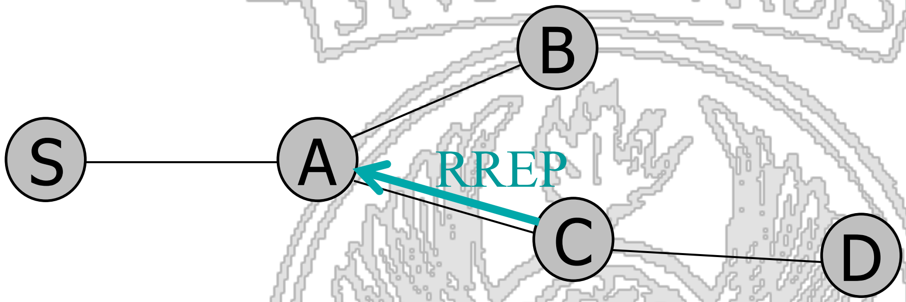
5. Node *C* receives RREQ
   - Makes reverse route entry for *S*
     dest=*S*, nexthop=*A*, hopcnt=2
   - It has a route to *D*, and the seq# for route to *D* is >= *D*'s seq# in RREQ

AODV: Route Discovery



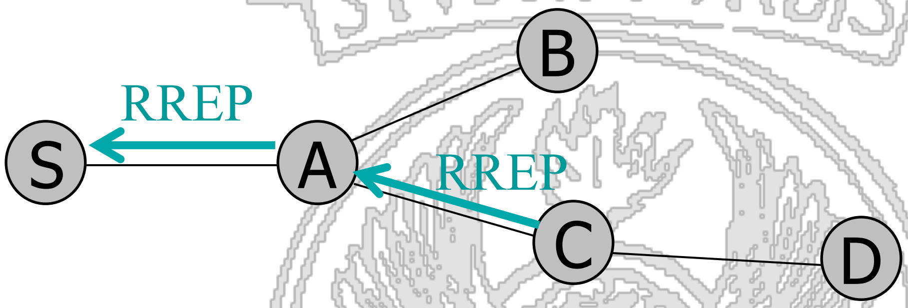
5. Node *C* receives RREQ (cont.)
   - *C* creates a Route Reply (RREP)
     Enters *D*'s IP addr, seq#
     S's IP addr, hopcount to *D* (= 1), lifetime of the forward route
   - Unicasts RREP to *A*

AODV: Route Discovery



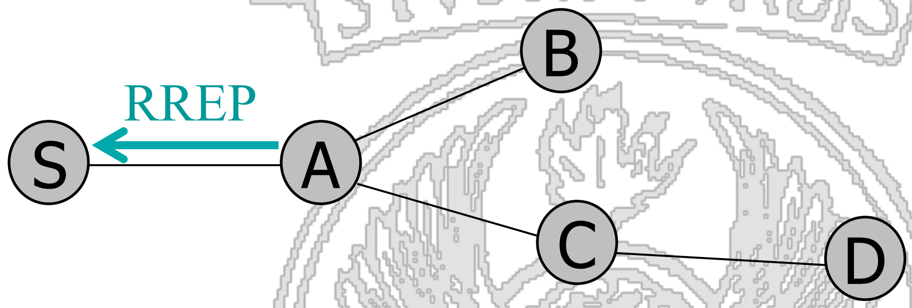
5. Node *C* receives RREQ (cont.)
   - *C* creates a Route Reply (RREP)
     Enters *D*'s IP addr, seq#
     S's IP addr, hopcount to *D* (= 1)…
   - Unicasts RREP to *A*

AODV: Route Discovery



6. Node *A* receives RREP
   - Makes forward route entry to *D*
     dest = *D*, nexthop = *C*, hopcount = 2
   - Unicasts RREP to *S*

AODV: Route Discovery

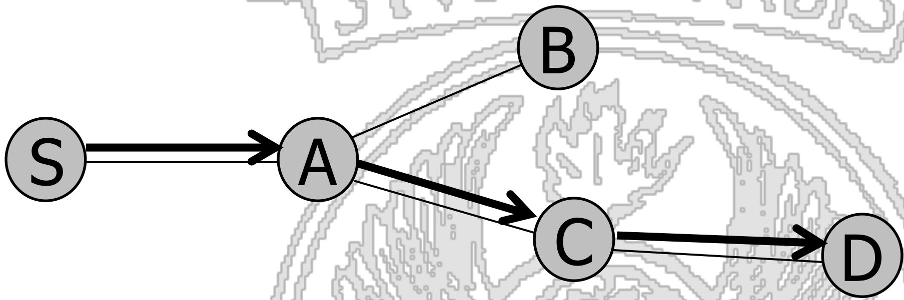


7. Node *S* receives RREP
   - Makes forward route entry to *D*
     dest = *D*, nexthop = *A*, hopcount = 3

Also the latest SN of the destination is updated when receiving the RREP

Nodes not along the path determined by the RREP will timeout after ACTIVE_ROUTE_TIMEOUT (3000ms) and will delete the reverse pointer

AODV: Route Discovery



7. Node *S* receives RREP
   - Makes forward route entry to *D*
     dest = *D*, nexthop = *A*, hopcount = 3
   - Sends data packets on route to *D*

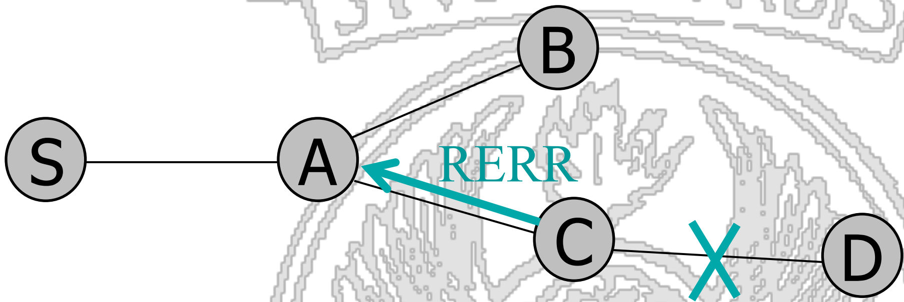
What if….

- A node receives further RREPs for the same request? (e.g. more neighbors of a node had paths to the destination in cache)?
  - upon reception of another RREP the node updates its routing information and propagates the RREP only if the RREP contains either a greater destination SN, or the same destination SN with a smaller hopcount

Other info maintained

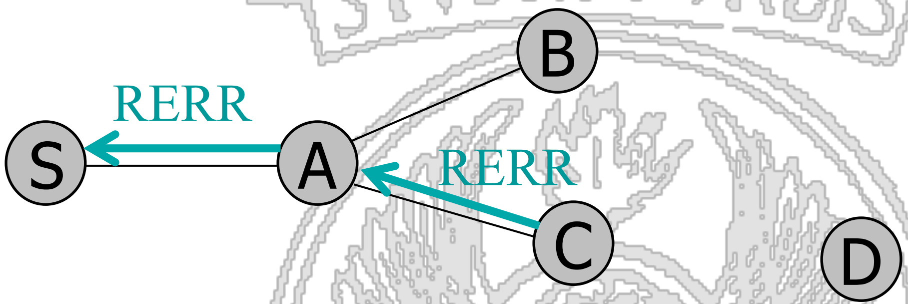
- Each node maintains the list of active neighbors, neighbors sending to a given destination through it
  - useful for route maintenance
- Routing table entries: dest,next hop, hopcount, dest SN, active neighbors for this route, expiration time for route table entry (updates each time the route is used for transmitting data → routes entries are maintained if the route is active)

AODV: Route Maintenance



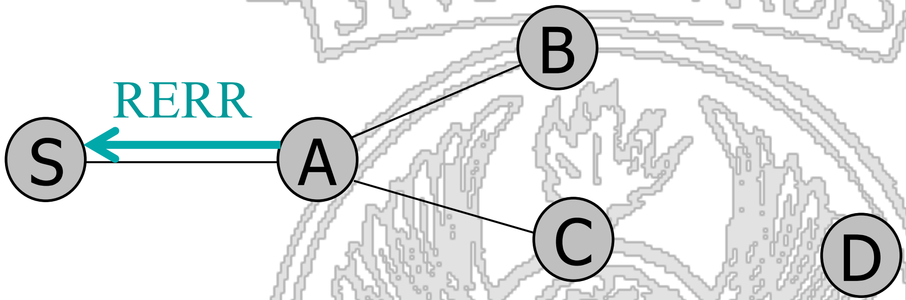
1. Link between *C* and *D* breaks
2. Node *C* invalidates route to *D* in route table
3. Node *C* creates Route Error (RERR) message
   - Lists all destinations which are now unreachable
   - Sends to upstream neighbors
   - Increases of one the SN of the destination

AODV: Route Maintenance



4. Node *A* receives RERR
   - Checks whether *C* is its next hop on route to *D*
   - Deletes route to *D*
   - Forwards RERR to *S*

AODV: Route Maintenance



5. Node *S* receives RERR
   - Checks whether *A* is its next hop on route to *D*
   - Deletes route to *D*
   - Rediscovers route if still needed (in that case it sends a RREQ with a SN which is equal to the last known destination Sequence Number +1)

AODV: Optimizations

- Expanding Ring Search
  - Prevents flooding of network during route discovery
  - Control Time To Live (TTL) of RREQ to search incrementally larger areas of network
  - Advantage: Less overhead when successful
  - Disadvantage: Longer delay if route not found immediately

AODV: Optimizations (cont.)

- Local Repair
  - Repair breaks in active routes locally instead of notifying source
  - Use small TTL because destination probably hasn’t moved far
  - If first repair attempt is unsuccessful, send RERR to source
  - Advantage: repair links with less overhead, delay and packet loss
  - Disadvantage: longer delay and greater packet loss when unsuccessful

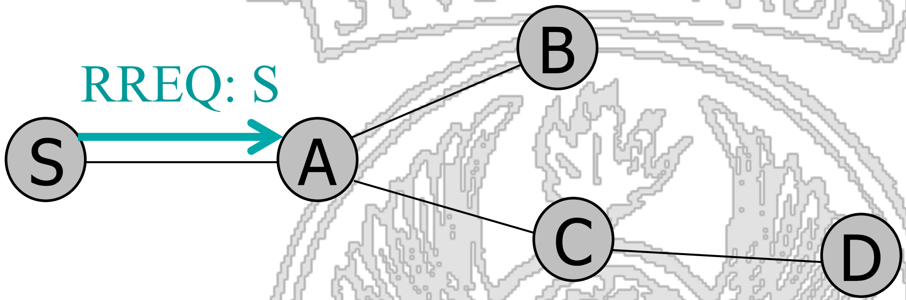
AODV: Summary

- Reactive/on-demand
- Sequence numbers used for route freshness and loop prevention
- Route discovery cycle
- Maintain only active routes
- Optimizations can be used to reduce overhead and increase scalability

Dynamic Source Routing (DSR)

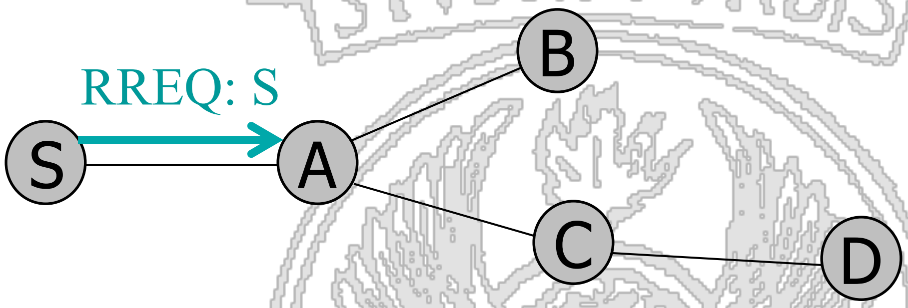
- Reactive
- *Route discovery cycle* used for route finding
- Maintenance of *active* routes
- Utilizes *source routing*

DSR: Route Discovery



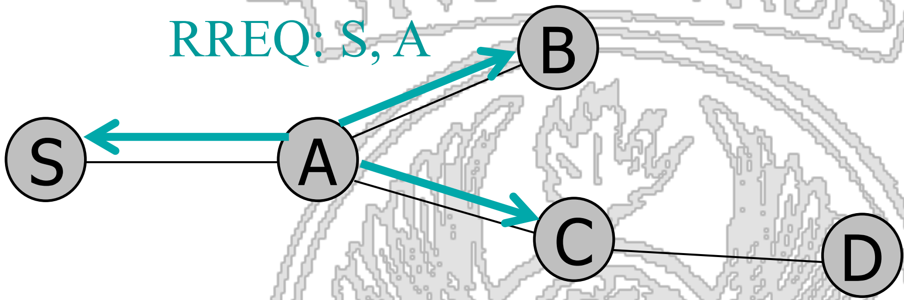
1. Node *S* needs a route to *D*
2. Broadcasts RREQ packet
    1. RREQ identifies the target of the route discovery, contains a route record in which the traversed route is accumulated, contains a pair <initiator, request id> uniquely identifying the request

DSR: Route Discovery



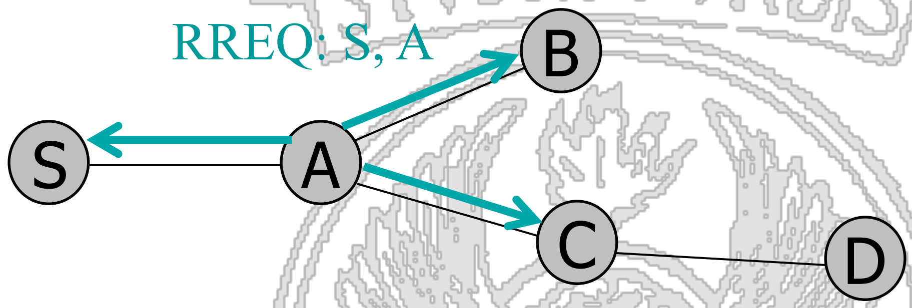
1. Node *S* needs a route to *D*
2. Broadcasts RREQ packet
3. Node *A* receives packet, has no route to *D AND is NOT D*
   - Rebroadcasts packet after adding its address to source route

DSR: Route Discovery



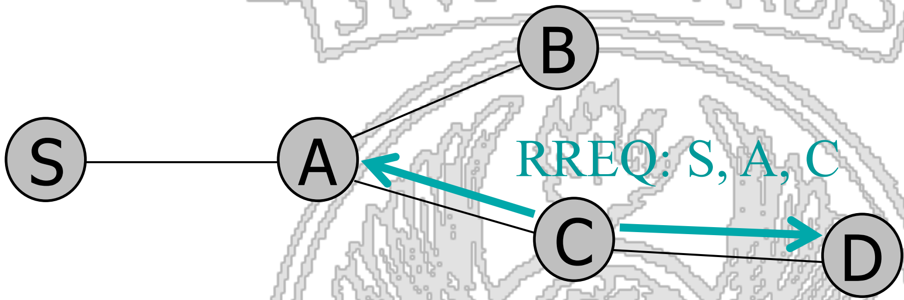
1. Node *S* needs a route to *D*
2. Broadcasts RREQ packet
3. Node *A* receives packet, has no route to *D*
   - Rebroadcasts packet after adding its address to source route

DSR: Route Discovery



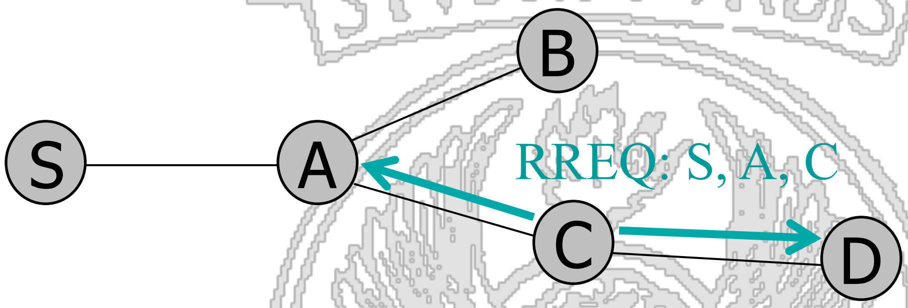
4. Node *C* receives RREQ, has no route to *D*
   – Rebroadcasts packet after adding its address to source route

DSR: Route Discovery



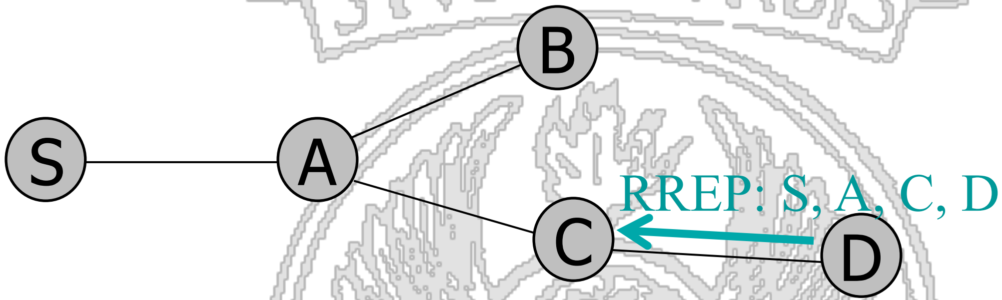
4. Node *C* receives RREQ, has no route to *D*
   – Rebroadcasts packet after adding its address to source route

DSR: Route Discovery



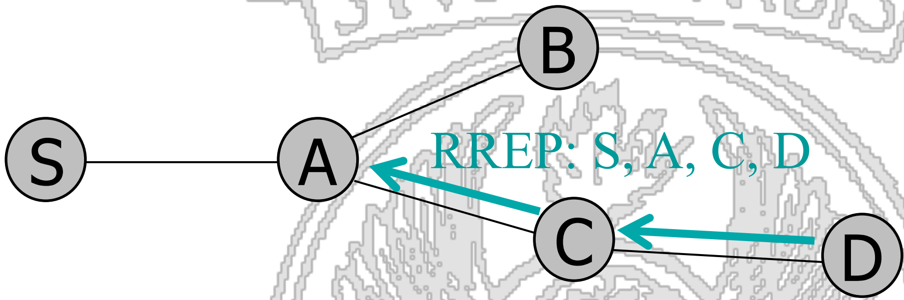
4. Node *C* receives RREQ, has no route to *D*
   – Rebroadcasts packet after adding its address to source route
5. Node *D* receives RREQ, unicasts RREP to *C*
   – Puts source route accumulated in RREQ into RREP

DSR: Route Discovery



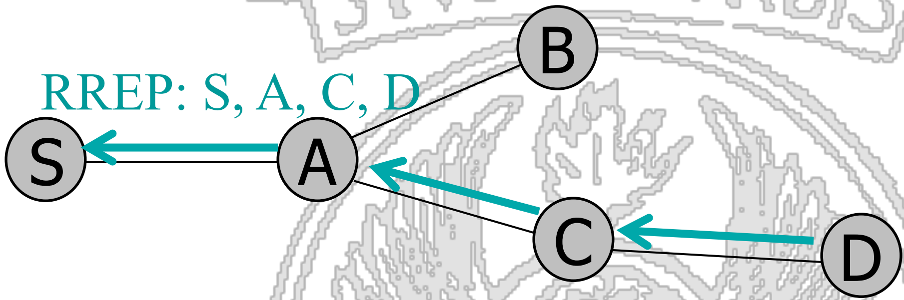
4. Node *C* receives RREQ, has no route to *D*
   – Rebroadcasts packet after adding its address to source route
5. Node *D* receives RREQ, unicasts RREP to *C*
   – Puts source route accumulated in RREQ into RREP

DSR: Route Discovery



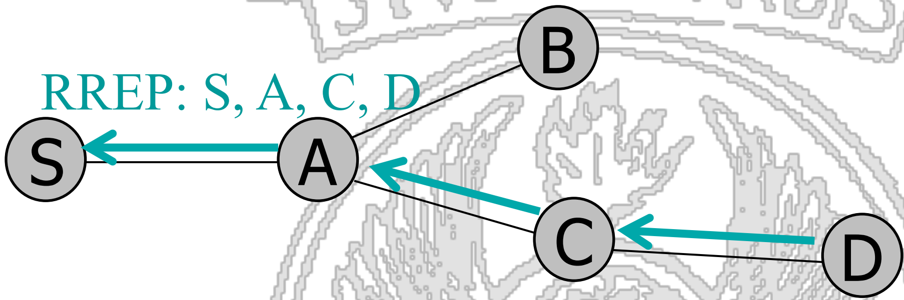
6. Node *C* receives RREP
   – Unicasts to *A*

DSR: Route Discovery



6. Node *A* receives RREP
   - Unicasts to *S*

DSR: Route Discovery



8. Node *S* receives RREP
   – Uses route for data packet transmissions

General node operation upon receiving RREQ

- If the pair <initiator address, request ID> has recently been seen, DISCARD
- If the node ID is already listed in the source route DISCARD→ avoids loops
- If I'm the destination, send a RREP
- Otrherwise, append my ID in the source route and rebroadcast (orange cases already seen in the previous slides)

Route maintenance

- The two endpoints of a failed link are transmitted to the source in a route error packet
- Upon a receiving a RERR packet a node invalidates all the routes going through that link
- If the route is invalidated and it is needed, a new route must be discovered

Optimizations (1)

- Extensive use of caching (caching source routes means that I already know all the route to intermediate destinations; discovery of a better route to an intermediate destination also brings me to improve the route to the final destination). Transmitting packets or sending back replies make me learn routes.

- A node that knows a route to a given destination (has a source route in cache) can immediately answer a RREQ
  - Broadcast storm? Each nodes waits for a time which is C*(h-1+r), r random in (0,1), h length of the route I'm advertising. Only if I haven't received other routes –listen to other routes tx in the meanwhile-I transmit mine.

Optimization (2)

- Operation in promiscuous mode (I keep discovering new routes by transmission of routes by my neighbours)

- RREQ overhead minimization: first set a TTL=1, if I do not get answer I set it to infinity

- Path shortening: if Y receives a packet by node X but in the source route we have X, B,…,C,Y, Y signals the path can be shortened (unsolicited RREP)

- What if the network is disconnected? Exponential back-off to decrease the quantity of RREQ sent

AODV and DSR Differences

- DSR uses source routing; AODV uses next hop entry
- DSR uses route cache; AODV uses route table
- DSR route cache entries do not have lifetimes;
  AODV route table entries do have lifetimes

Proactive Solutions: Drawbacks

- Updates overhead, especially in presence of high mobility
- Overhead for enforcing loop freedom
- Large routing tables
- Low *scalability*

➢ Is it really necessary to maintain a consistent view of the network topology?

Reactive Protocols: Drawbacks

- The discovery phase introduces long delays
- Route discovery and maintenance is very sensitive to node mobility
- Route caching is memory greedy
- The size of the header of a data packet can become cumbersome in approaches such as DSR (no scalability)
- Operating in promiscuous mode is energy-consuming.
- Relying on flooding based route discovery is resource consuming.

➢ Is the dependency on the network topology avoidable?

Geographically-Enabled Routing

- Outline
  - Problems with proactive approaches
  - Problems with reactive approaches
  - A new way of naming\locating the destination node: geographic routing
  - Two seminal protocols
    - ✓ DREAM & LAR
  - Geo-enable routing costs: I need to know where I am, where the destination is.

Location-Enabled Ad Hoc Routing

- Nodes are equipped with positioning system devices (e.g., Global Positioning System receivers) that make them aware of their position
- This enables "directional" routing
- Possible solutions differ on how the location information of the destination nodes is achieved

Strengths

- No need to update big routing tables, no need to piggyback routes to the packet
  - Destination position must be known at the source.
- No need to know the nodes on the way to the destination: they can be moving while the packet travels

Drawbacks

- Needs extra hardware
- Depends on the extra hardware limitation (and resource requirements)
- Scalability is an issue (indeed the problem here translates to how to maintain correct estimates of the nodes positions)

Location-Aided Routing (LAR)

- Exploits location information to limit scope of RREQ flood
- *Expected Zone*: region that is expected to hold the current location of the destination
  - Expected region determined based on potentially old location information, and knowledge of the destination's speed
- RREQs limited to a *Request Zone* that contains the Expected Zone and location of the sender node

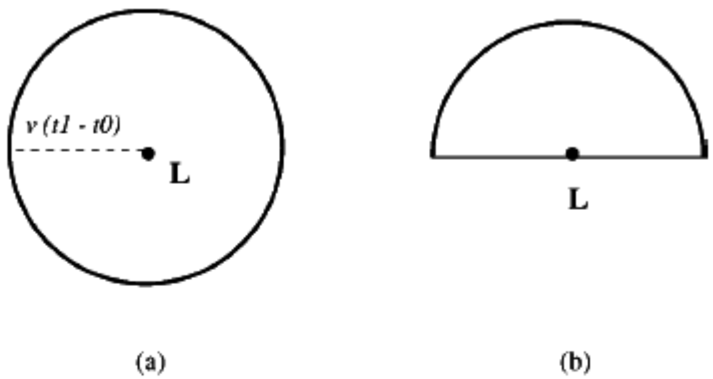


Figure 2. Examples of expected zone.

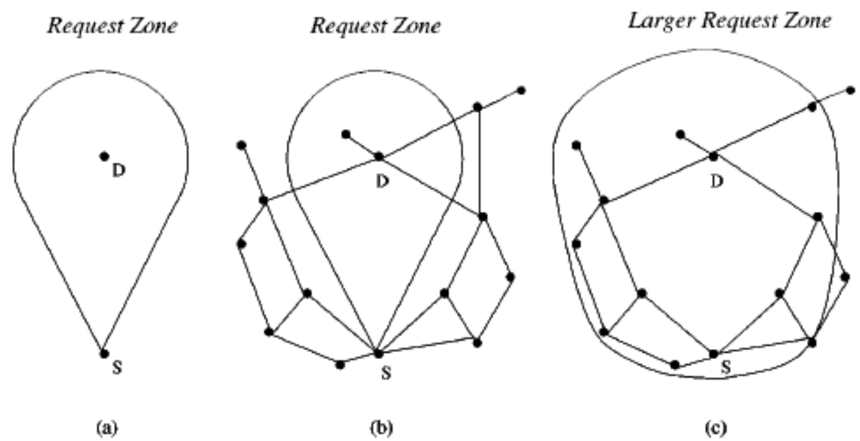


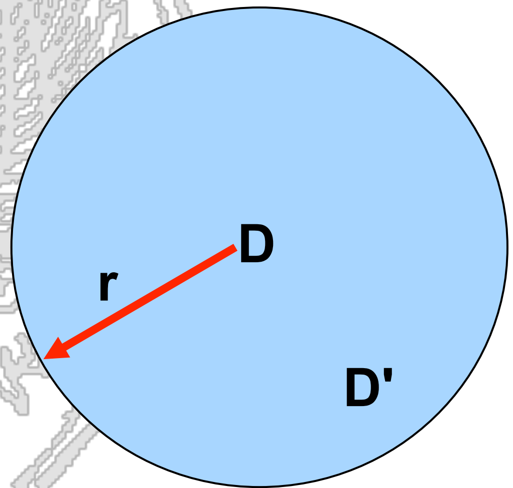
Figure 3. Request zone. An edge between two nodes means that they are neighbors.

LAR: Expected Zone

D = last known location of node D, at time t_0

D' = location of node D at current time t_1, unknown to node S

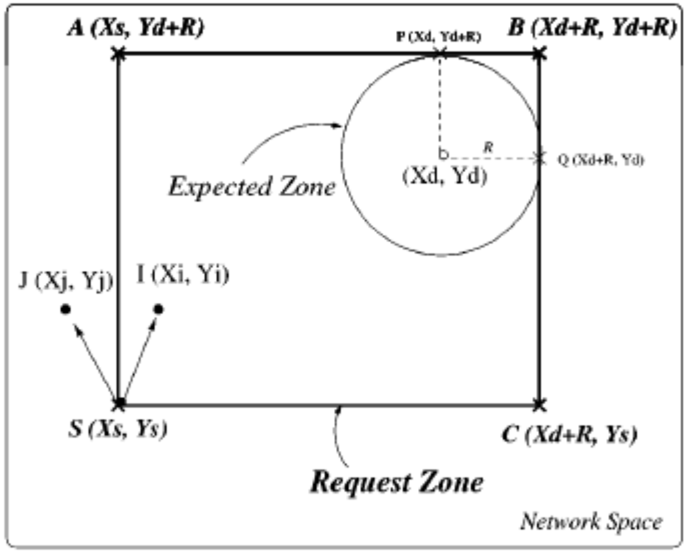
r = $(t_1 – t_0)$ * estimate of D's speed



Expected Zone

LAR-1

- The request zone is the smallest rectangle that includes the current location of the source and the expected zone
- Only nodes within the request zone forward route requests
  - Node A does not forward RREQ, but node B does
- Request zone explicitly specified in the RREQ
- Each node must know its physical location to determine whether it is within the request zone



LAR, Possible Failures

- If route discovery using the smaller request zone fails to find a route, the sender initiates another route discovery (after a timeout) using a larger request zone
  - The larger request zone may be the entire network
- Rest of route discovery protocol similar to DSR
  - The directional flooding approach proposed in the paper is used to reach the destination with a limited overhead wrt traditional flooding
  - Destination then answers with a route reply packets
  - This allows the source to compute based on tradictional reactive protocols approaches the route towards the destination

LAR, the Routing

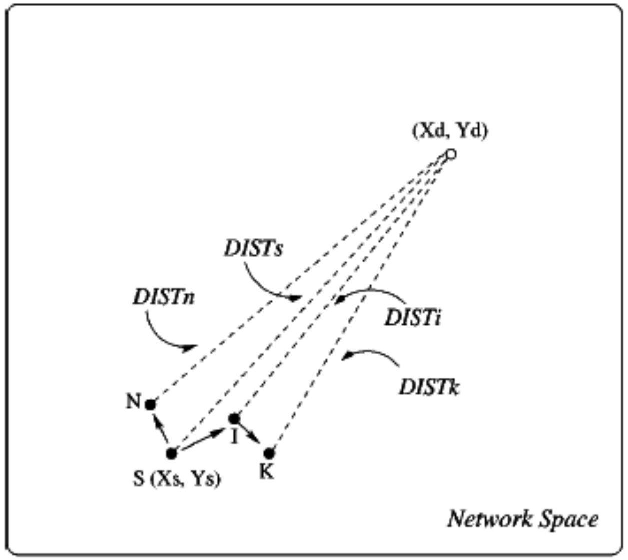
- The basic proposal assumes that, *initially*, location information for node X becomes known to Y only during a route discovery
- This location information is used for a future route discovery
  - according to the paradigm explained before
  - Updates on the node position are piggybacked in the route reply message

  → This allows to reduce overhead associated to route discovery process provided that the time between two route discovery is not too much
- Destination may also proactively distribute its location information
  - But in this case the control traffic for geographic information updates could be high
    - ✓ How to solve this issue? Later (DREAM)

LAR - 2

- Each protocol relies RREQ if it is closer to the destination than the source (greedy forwarding)

- Assume that node S knows the location (X_d, Y_d) of node D at some time t_0 – the time at which route discovery is initiated by node S is t_1, where $t_1 \geqslant t_0$. Node S calculates its distance from location (X_d, Y_d), denoted as $DIST_s$, and includes this distance with the route request message.
- The coordinates (X_d, Y_d) are also included with the route request.
- For some parameters α and β, if $\alpha(DIST_s)+\beta \geqslant DIST_i$, then node I forwards the request to its neighbors. When node I forwards the route request, it now includes $DIST_i$ and (X_d, Y_d) in the route request (i.e., it replaces the $DIST_s$ value received in the route request by $DIST_i$, before forwarding the route request).
- Else $\alpha(DIST_s)+\beta < DIST_i$. In this case, node I discards the route request.



DREAM

- Distance routing effect algorithm for mobility [Basagni+, 1998]

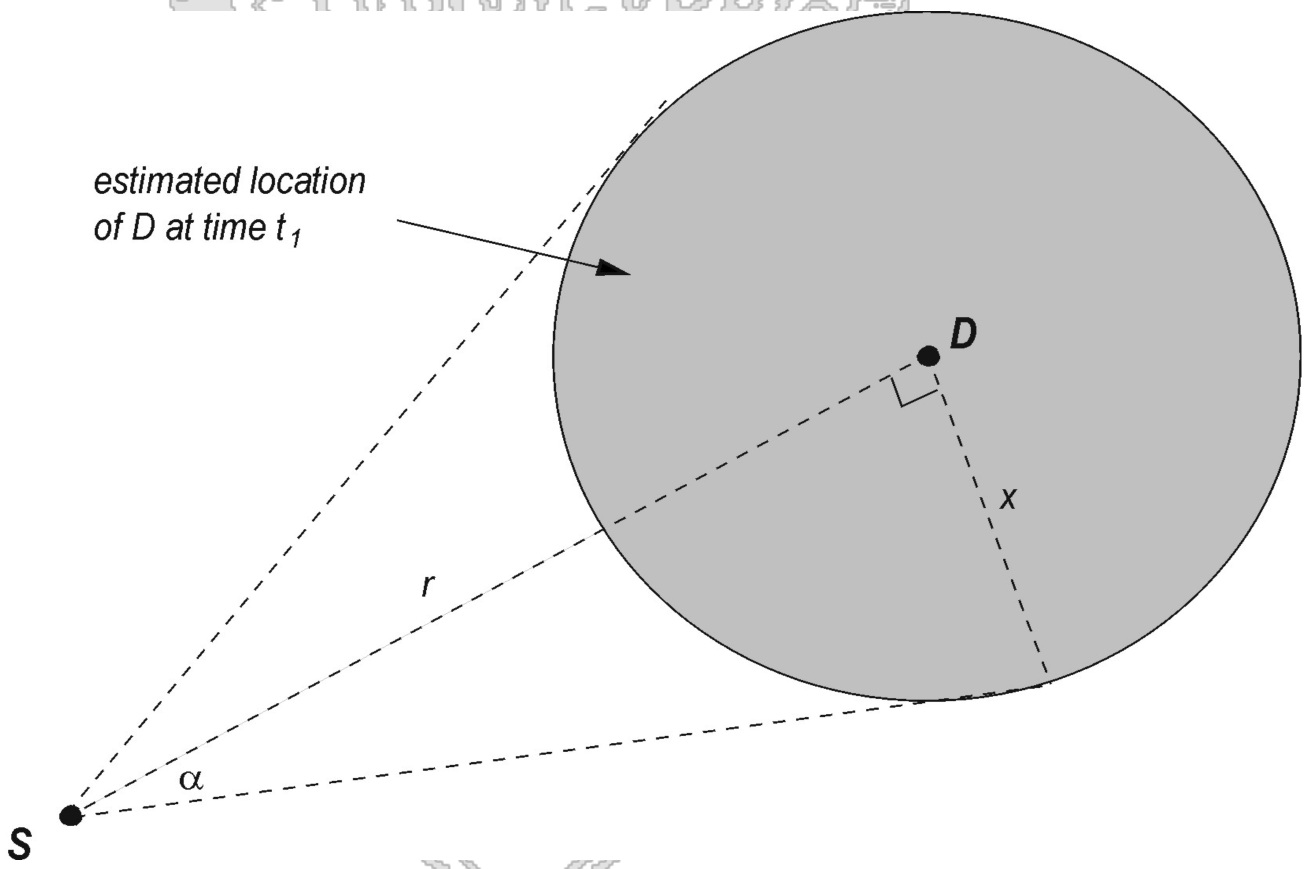
- A proactive, effective way to spread location information

- Directional routing

DREAM: Directional Routing

- Source S determines the location of destination D at time t_0 based on its location table
- Based on the current time t_1 and t_0 S determines the area in which D can be found (hence, D's direction)
- S transmits the data packet to all its neighbors in D's direction
- Each neighbor does the same till D is reached

DREAM: Routing a Data Packet



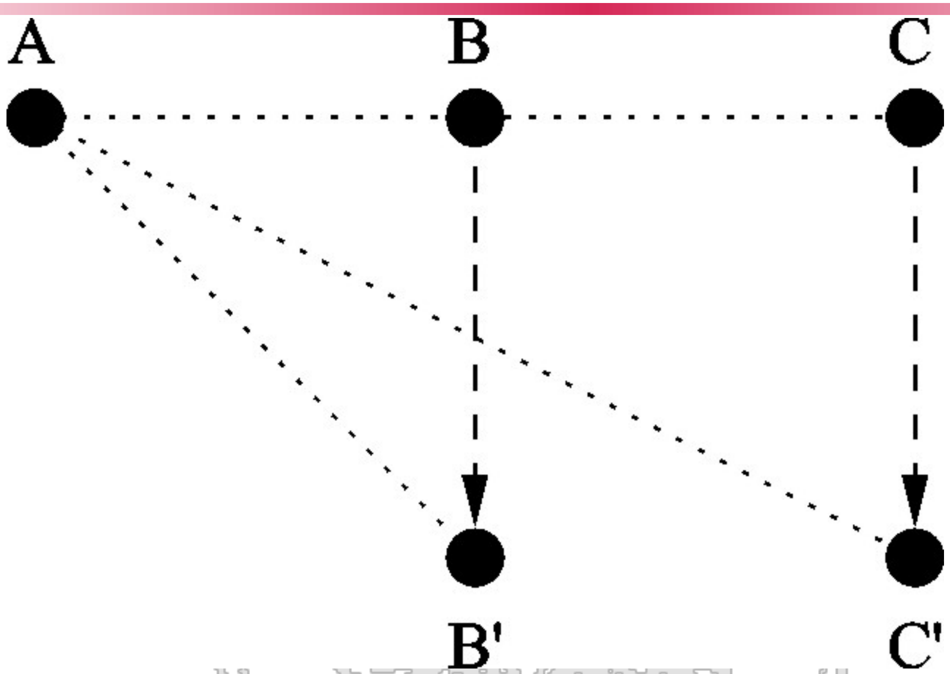
Disseminating Location Information: Problems

- Need to periodically update the location of a moving node.
  - Efficient broadcast of location information
  - Determining how far each location packet should travel
  - Determining how often a location packet should be sent

Disseminating Location Information: Solutions

- Distance effect
- Rate of updates is bound to the mobility of the node

The Distance Effect



- "Closer nodes look like they are moving faster"
- Need to receive more location updates from closer node
- Each location update packet is associated with an age that determines how far that packet must travel

DREAM: Rate of updates

- Triggered by the mobility of the nodes
- The faster the node the more updates it sends
- A plus: slow moving nodes impose little overhead

DREAM, Strengths

- First of its kind: after that, the deluge!
- Robustness: multiple routes to the destination
  - directional flooding

DREAM, Weaknesses

- It is flooding, although only directional
- It is not that scalable, geographic info updates have to be periodically transmitted (even if mechanisms to limit such overhead are enforced)

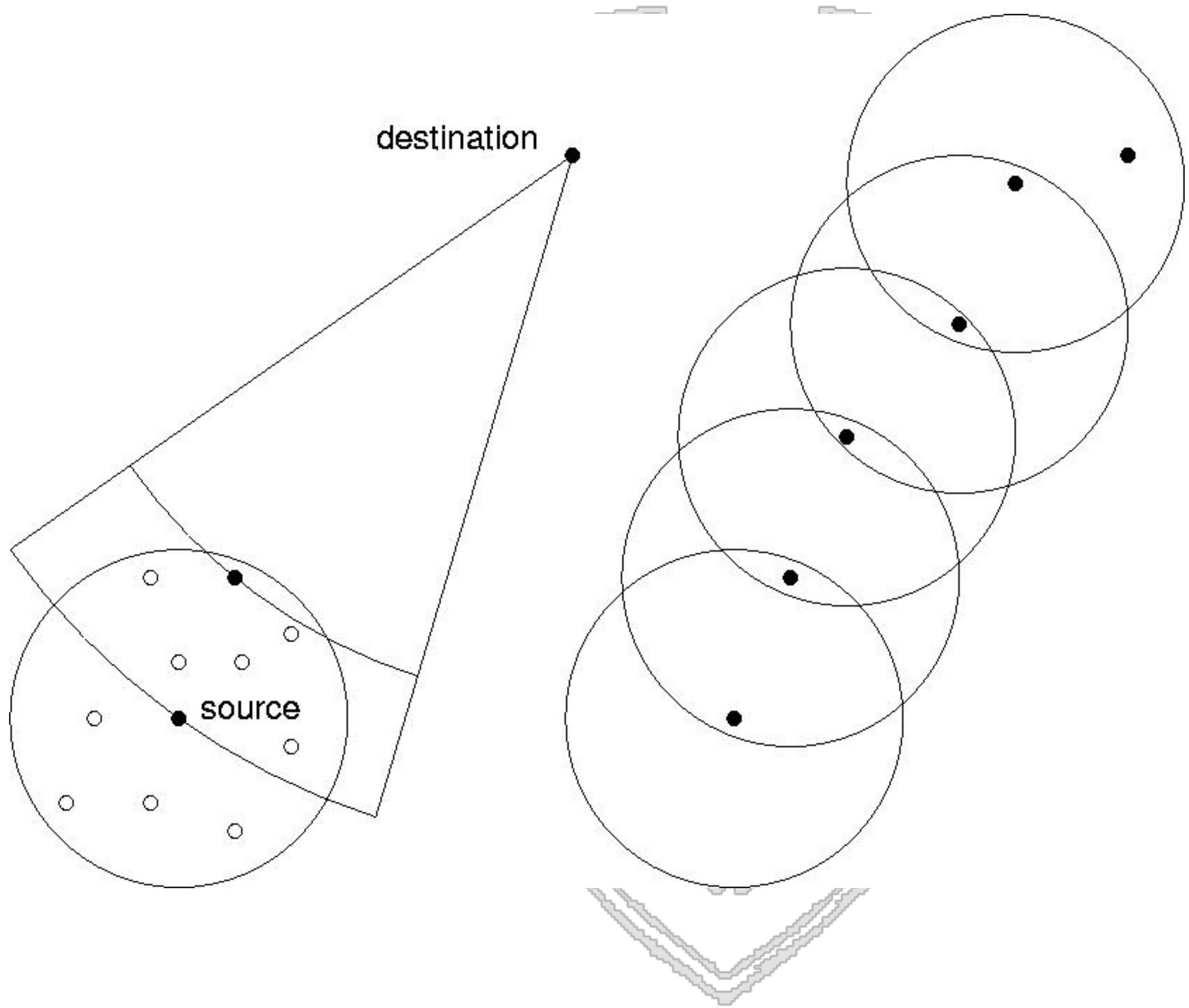
In a sensor network we seek..

- For solutions which scale
- Which are energy saving
  - Which are well integrated with awake/asleep schedules
- Which do not require to maintain routing tables
- Which are simple
- Solutions such as AODV and DSR have been proven to work well iff they exploit intensively caching and promiscuous mode operation (energy inefficient← work by L. Feeney et al, 2001) and have been shown not to scale to the volumes of nodes expected in sensor networks (work by E. Belding Royer and S.J. Lee)
- What can we use?
  - communication sensors – sink
  - Info such as localization and some level of synchronization often needed by the application (if I sense an event I have to say WHERE and WHEN it occurred, otherwise the information is not very interesting)

An example: GeRaF

- Integrates
  - geographic routing
  - awake/asleep schedule
  - MAC
- How do nodes alternate between awake and asleep states? According to a duty cycle (time ON/time ON+ OFF)

GeRaF basic idea



Geographic routing: each node needs to know its location, the destination (sink) location, and the location of whom is transmitting (communicated in the packet)
Greedy approach: tries to select relays so to advance as much as possible toward the destination at each hop

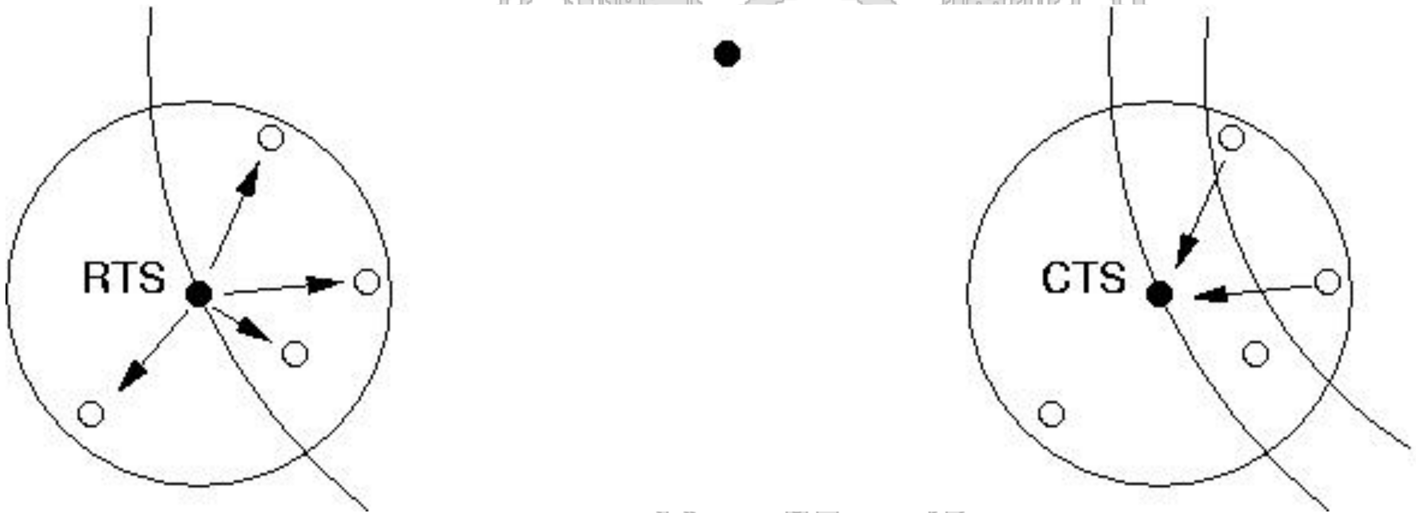
GeRaF: operations

- Main problem to be solved: how to make a contention-based scheme/routing work in the presence of sleep modes
  - Flat solution
  - Integrated MAC/routing/awake-asleep but awake-asleep schedule and routing decoupled → each node does not know its neighbor and their schedules → low overhead

- Tightly integrated with the routing layer (no clear separation really)
  - Without requiring routing tables/routing table updates
  - Based on the knowledge of the nodes location and on the knowledge of the sink location

Example of GeRaF operation

- RTS invites all awake neighbors to become a relay
- Nodes in best position should win
  - Nodes within tx range are divided in areas depending on how close they are to the final destination (the closest the better as relay)



•Need of location awareness

GeraF operation

- Node i sends RTS with the identity of the area it is polling now (starting from the closer to the sink, among the slices in which its tx range has been divided)
- Each node, upon receiving the RTS, decides whether it belongs to the polled area or not (based on location info)
- Only nodes in the polled area answer with a CTS
  - No node answers → node i polls next area (no node available for forwarding in the area-there are no nodes or they are sleeping)
  - One answer, CTS correctly received, send DATA
  - Multiple answer COLLISION, sender sends a collision packet, MAC needed to solve collision (next slide)

GeraF operation (how to handle collision)

- 1)A node receiving a collision packet tosses a coin and with probability p transmits a CTS iff it was participating to the previous phase (it had previously sent a CTS resulting in collision)
  - if only one node answers node i sends data
  - If no node answers node i asks these nodes to toss a coin again..
  - if more nodes answer COLLISION. Collision packet is sent. GO TO 1) (only the nodes which have lead to collision survive to the next phase)

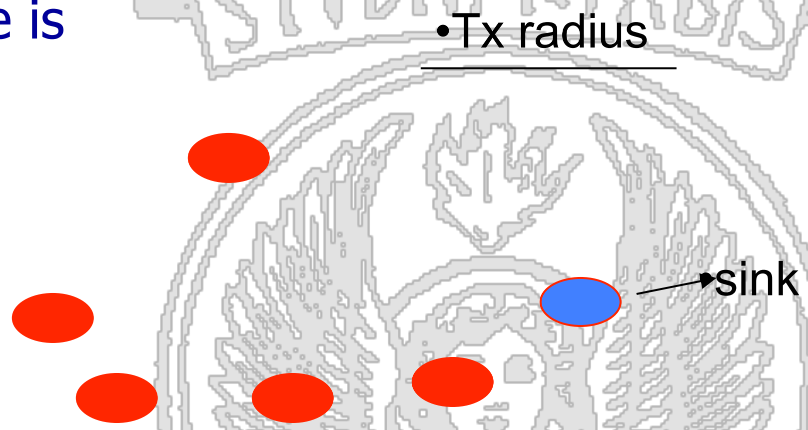
What If

- All areas are polled unsuccessfully?
  - Try again after some time (exponential backoff)
- Can I always reach the destination in this way?

What if (answer)

- No. Here is

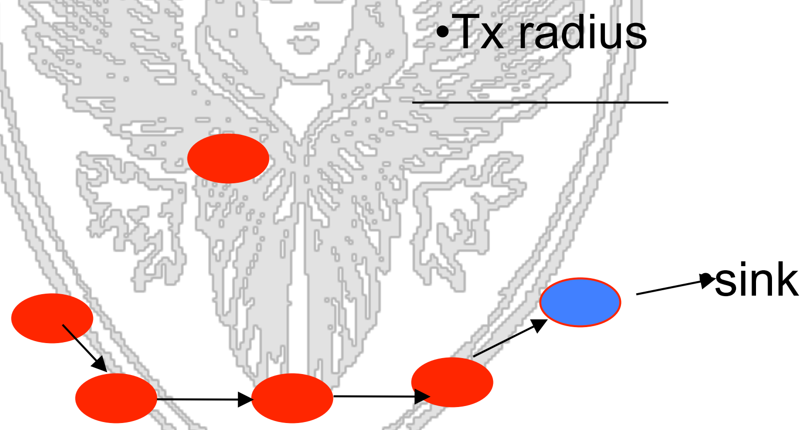
An example



- Solutions? (mechanisms have to be added to recognize the problem, do backtracking and try another route)

How to solve dead-ends

- A problem only at low density
- We can set a maximum number of attempts to find a relay. When a node fails to find a relay it starts decreasing its duty cycle/or the probability to propose itself as relay …over time nodes along paths to dead-ends are less and less selected as next hop relays and other paths able to bring to the destination are instead found
- Still..we may have problems…



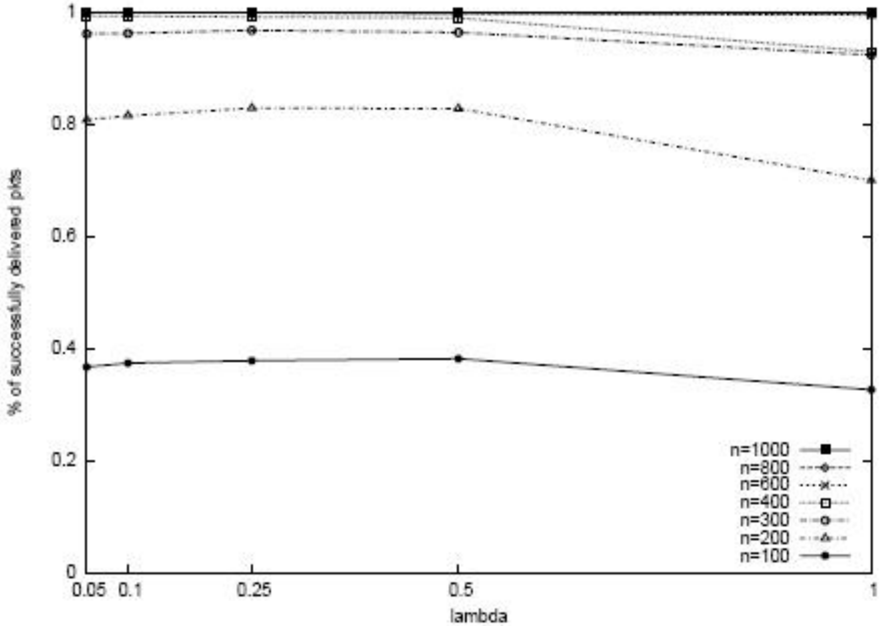
GeRaf performance

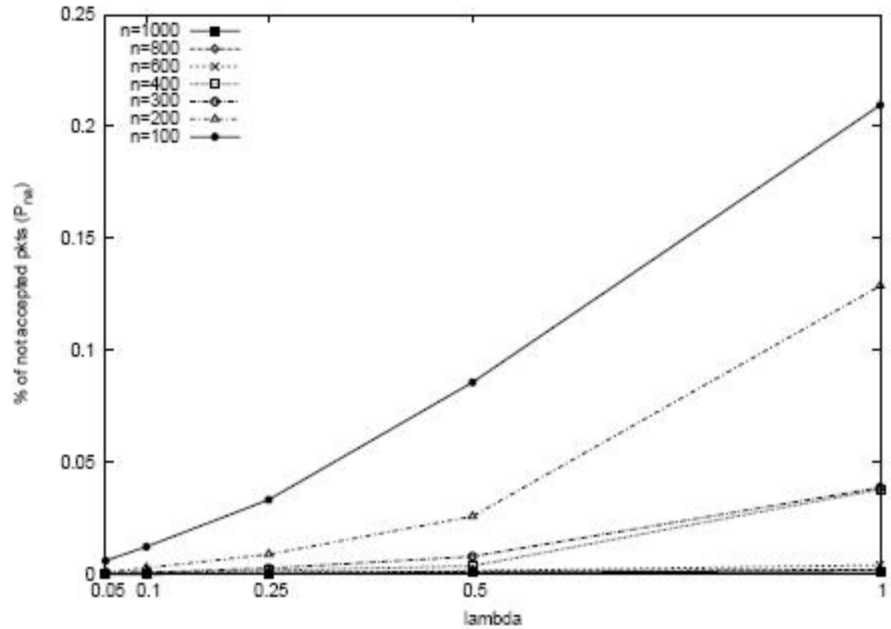
Casari, Marcucci,Nati, Petrioli, Zorzi IEEE MILCOM 2005

- square area 320m x 320m
- Transmission range=40m
- 100-1000 randomly deployed nodes (avg degree 5-50)
- Duty cycle =0.01,0.1,0.5
- Comparable costs for tx/rx/idle
- Poisson packet arrival
- Channel data rate 38Kbps

GeRaf performance, d=0.1

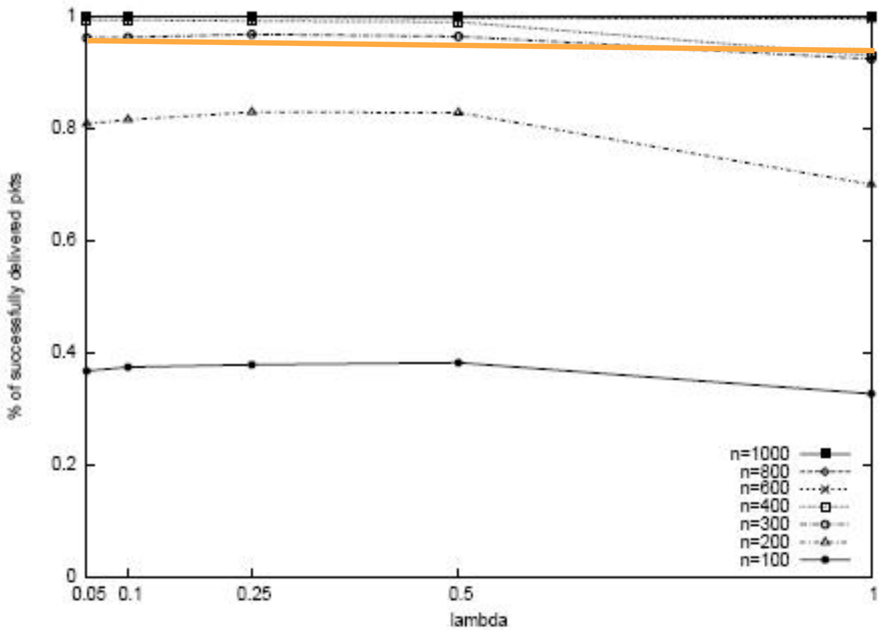
Casari, Marcucci,Nati, Petrioli, Zorzi IEEE MILCOM 2005

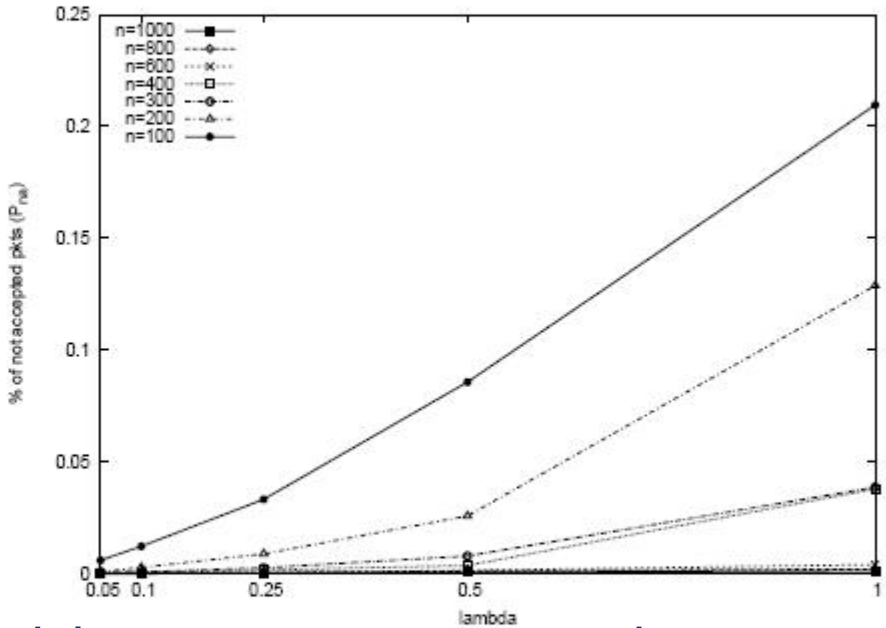




GeRaf performance, d=0.1

Casari, Marcucci,Nati, Petrioli, Zorzi IEEE MILCOM 2005





Backtracking in case of dead ends packet delivery ratio increases. When N=200 93-97% of packets are correctly delivered. Problem is instead not solved when N=100. Solution to the problem provided by ALBA-R (or better by the 'R' RAINBOW scheme of ALBA).

Localization in sensor networks

Thanks to Prof. Mani Srivastava
These slides have been derived
From his tutorial on sensor networks
Given at Rome Un. On July 2003

Localization

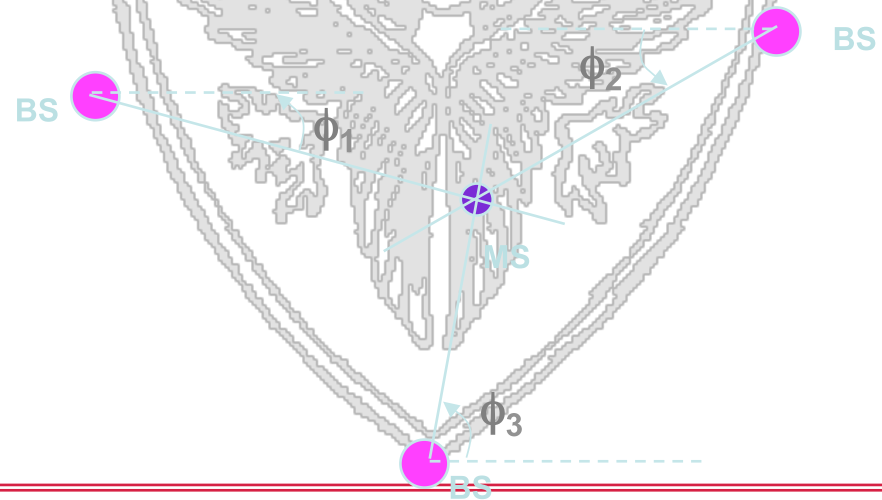
- Useful info
  - Helps with some protocols (e.g. GeraF)
  - Needed for being able to identify where events occur
- Why not just GPS (Global Positioning System) at every node?
  - High power consumption
  - Works only when LOS to satellites (not in indoor, heavy foliage…)
  - Over kill – often only relative position is needed (e.g. enough to know that relative to a coordinate system centered in the sink the event occurred in a position (x,y). Starting from relative info if some nodes have global coordinates global coordinates of events can be inferred.

Localization

- Basic step is to evaluate distance between two nodes (ranging). Different techniques depending on the available HW:
  - AoA (e.g. directional antennas)
  - RSS
  - ToA
- Range free approaches (number of hops between nodes used to estimate the distance between them without using any extra HW)

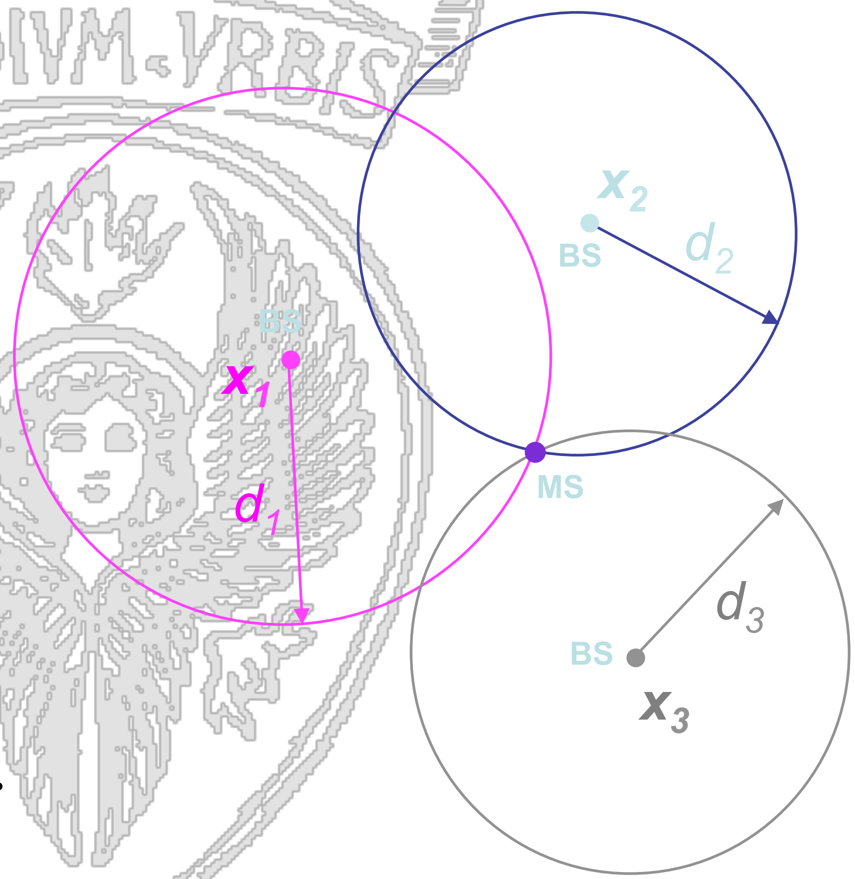
Techniques for Location Sensing (AoA)

- Measure direction of landmarks
  - Simple geometric relationships can be used to determine the location by finding the intersections of the lines-of-position
  - e.g. Radiolocation based on angle of arrival (AoA) measurements of beacon nodes (e.g. base stations)
    - ✓ can be done using directive antennas + a compass
    - ✓ **need at least two measurements**



Techniques for Location Sensing (RSS or ToA)

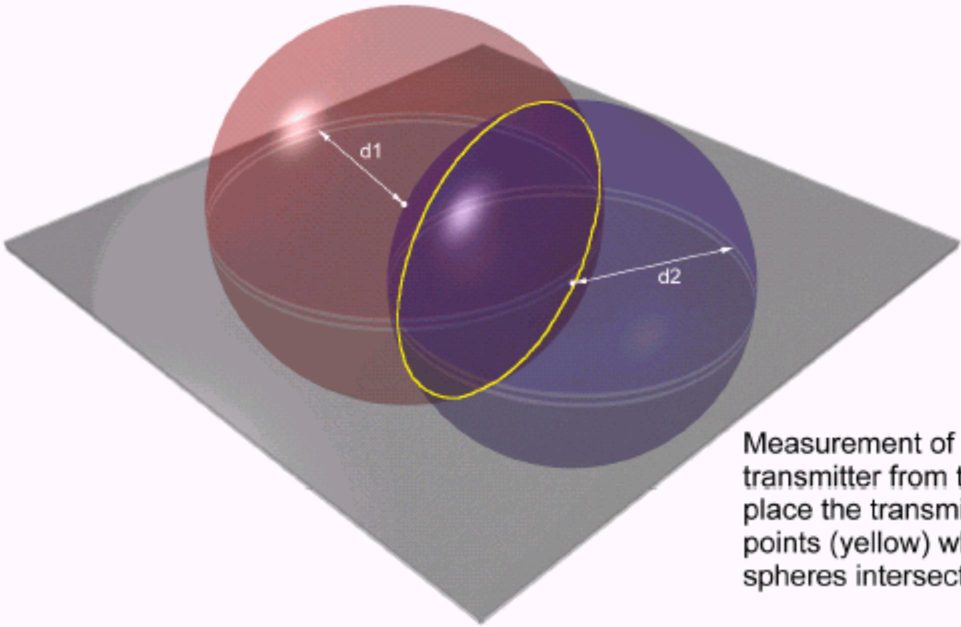
- Measure distance to landmarks, or Ranging
  - e.g. Radiolocation using signal-strength or time-of-flight
    - ✓ also done with optical and acoustic signals
  - Distance via received signal strength
    - ✓ use a mathematical model that describes the path loss attenuation with distance
      - each measurement gives a circle on which the MS must lie
    - ✓ use pre-measured signal strength contours around fixed basestation (beacon) nodes
      - can combat shadowing
      - location obtained by overlaying contours for each BS
  - Distance via Time-of-arrival (ToA)
    - ✓ distance measured by the propagation time
      - distance = time * c
    - ✓ each measurement gives a circle on which the MS must lie
    - ✓ active vs. passive
      - active: receiver sends a signal that is bounced back so that the receiver knows the round-trip time
      - passive: receiver and transmitter are separate
        - » time of signal transmission needs to be known
  - N+1 BSs give N+1 distance measurements to locate in N dimensions



- But what if the circles do not intersect due to measurement errors (e.g. due to fading etc.)?

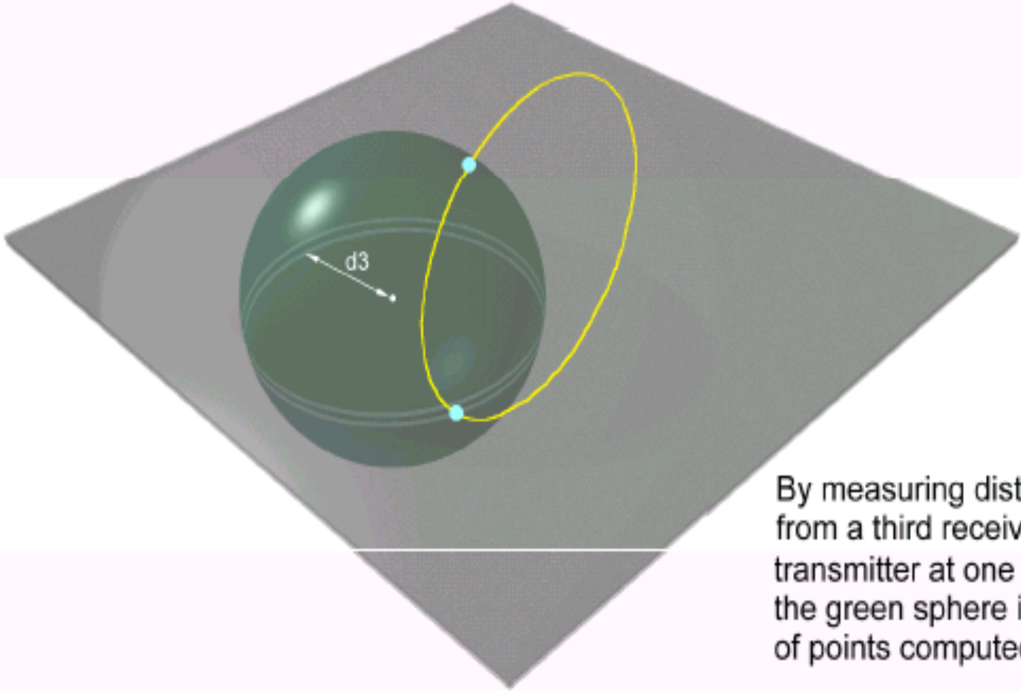
→will have to identify the best 'guess' given errors

Location in 3D



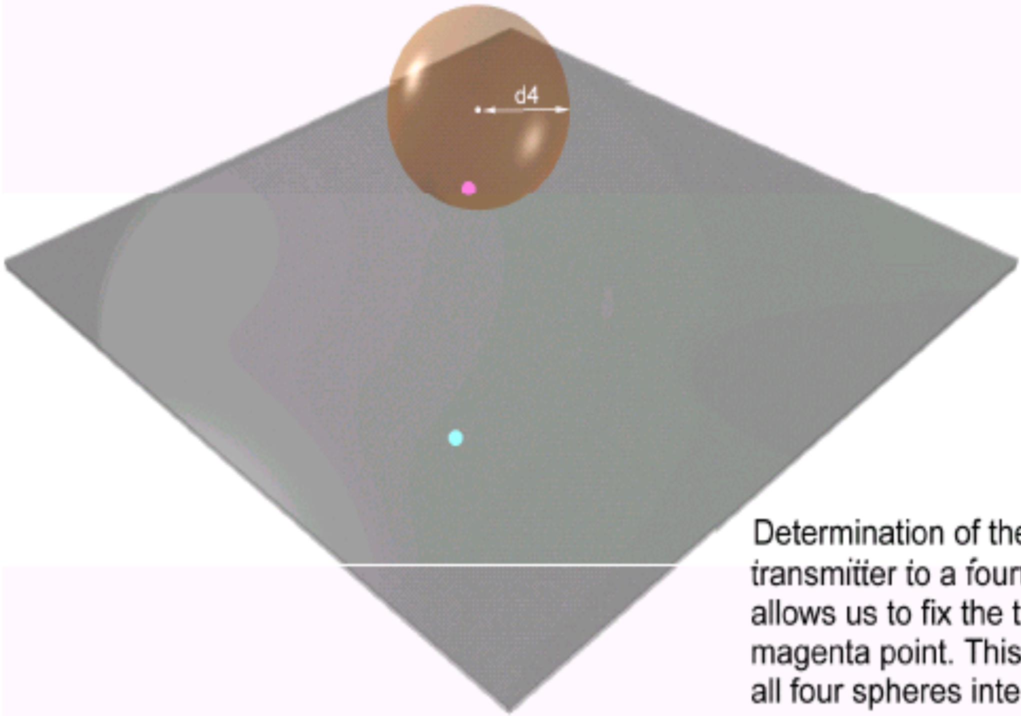
Measurement of distances d1 & d2 to the transmitter from two receivers allows us to place the transmitter on a circular locus of points (yellow) where the red and blue spheres intersect.

Location in 3D



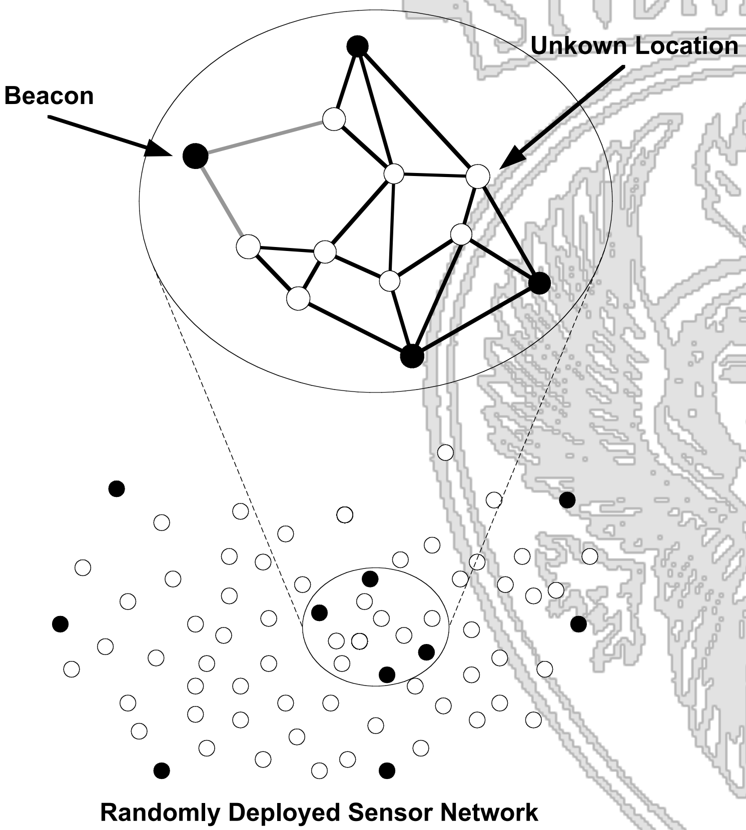
By measuring distance d3 to the transmitter from a third receiver, we can place the transmitter at one of two points (cyan) where the green sphere intersects the circular locus of points computed previously.

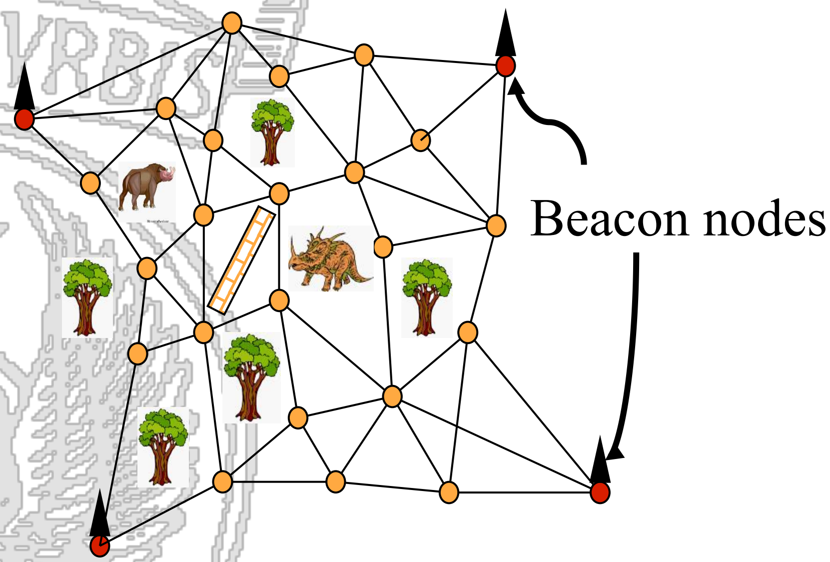
Location in 3D



Determination of the distance d4 from the transmitter to a fourth, non-copianar receiver allows us to fix the transmitter's location at the magenta point. This is the single point at which all four spheres intersect.

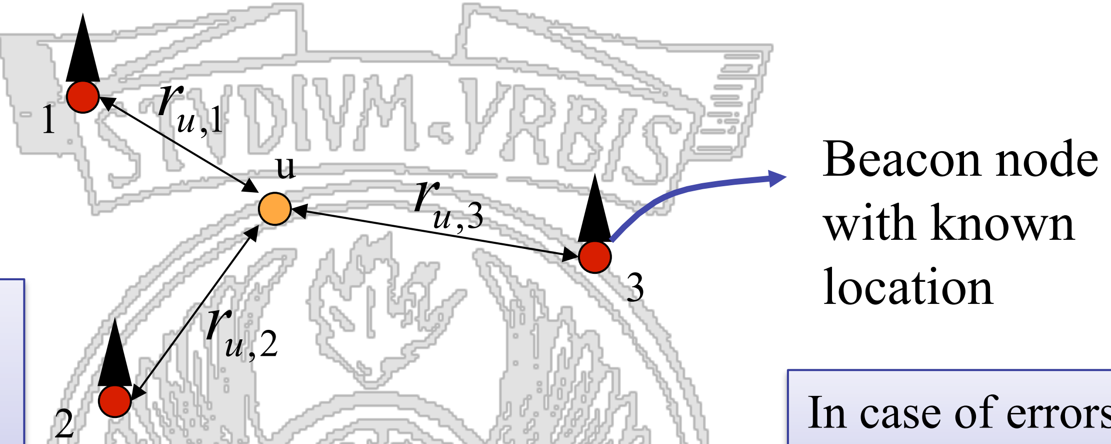
A possible solution: Absolute Localization





- A small fraction of the nodes is aware of their locations
- Rest need to collaborate to estimate their locations

Atomic Multilateration



Nodes with at least three Neighbors aware of their Location can estimate their position (triangolarization)

In case of errors metric is root mean square error

$$f_{u,i} = r_{u,i} - \sqrt{(x_i - \hat{x}_u)^2 + (y_i - \hat{y}_u)^2}$$

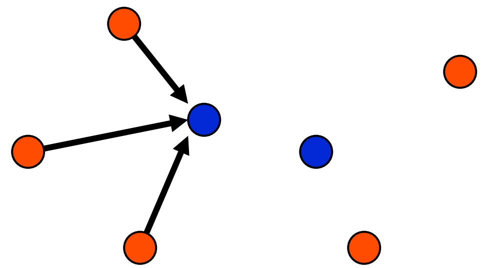
$(\hat{x}_u, \hat{y}_u)$ - initial position estimate for node u

Our objective function is:

$$F(x_u, y_u) = \min \sum f_{u,i}^2$$

Iterative Multilateration

- Nodes that estimate their locations can become beacons and help other nodes discover their locations.



- Some observations:
  - Can work for small networks, if ranging is accurate
  - Energy efficient
  - Still requires quite a lot of initial beacons
  - Suffers from error accumulation
  - **Bad geometry yields bad results => unpredictable performance**
  - Still a useful primitive for Distributed Collaborative Multilateration **First simple approach, many solutions in the literature**

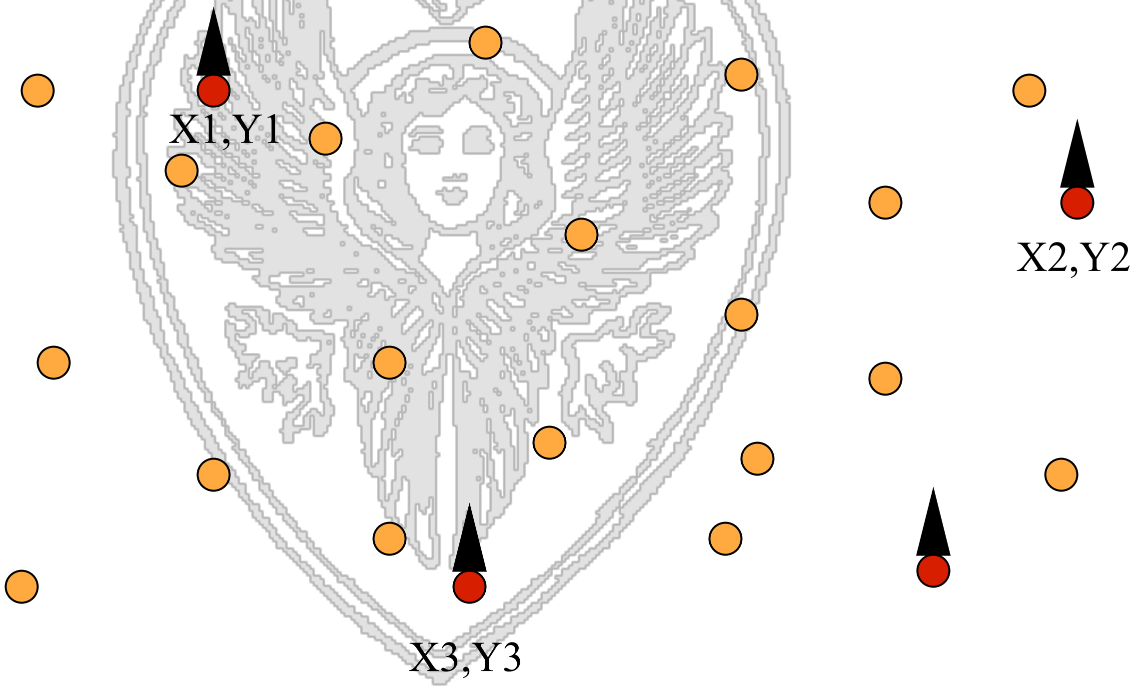
Ref: based on slides by Andreas Savvides

Range free localization

- Does not use ranging
- But only information available (e.g., discovered during routing protocols operation)

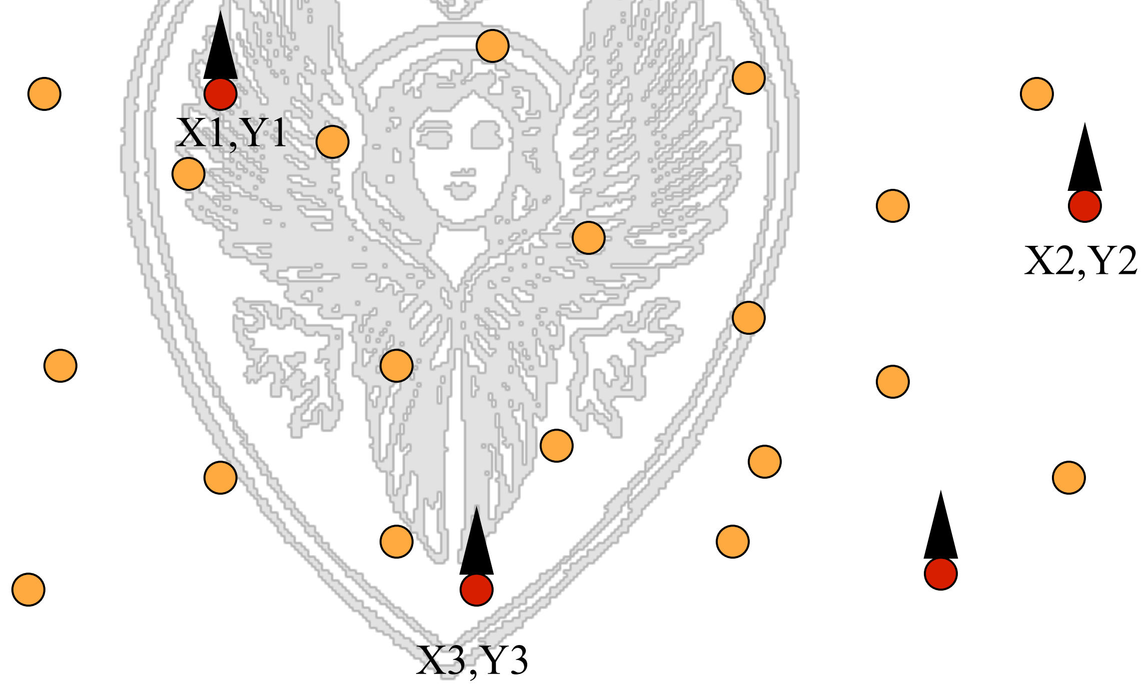
How does it work?...

- Anchors know their position according to a common coordinate system



How it works…

- All nodes compute shortest path between them and anchors
- Also anchors compute their relative distance in hops



How it works…

- Anchor A: I know my position, the position of the other anchors, our distance in hops
  - → I can therefore estimate the average length of a one hop.
- This information is used to estimate distances D(x,y), where x is a node and y is one of the anchors.
- Based on such estimated distances and on the position of the anchor nodes each node computes its own position by exploiting triangolarization.

- Pro: No extra HW
- Cons: Not very accurate