SAPIENZA
UNIVERSITÀ DI ROMA

Ad hoc Network Routing

Wireless Systems, a.a 2020/2021

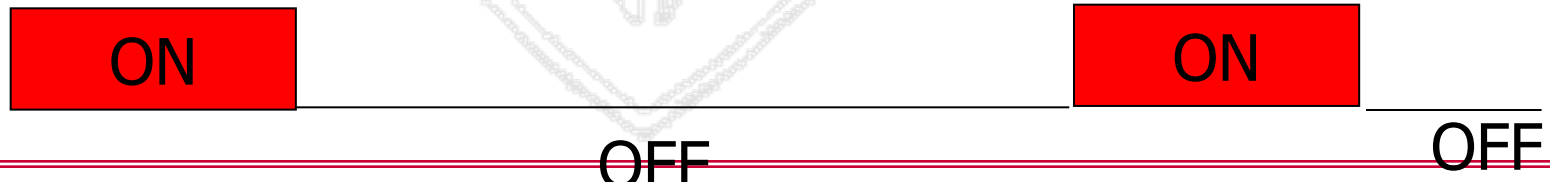
Un. of Rome "La Sapienza"

Chiara Petrioli[†]

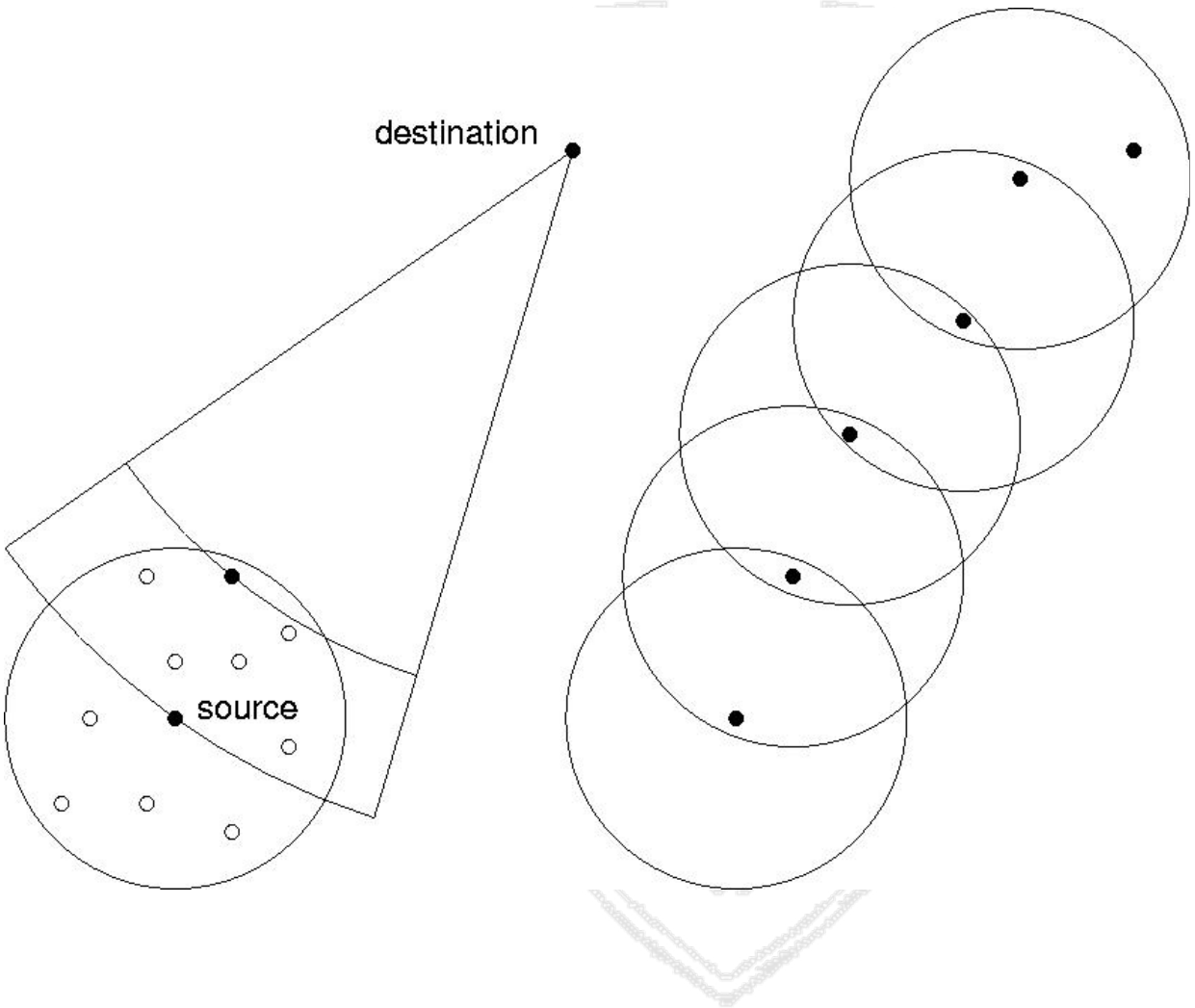
[†] *Department of Computer Science – University of Rome "Sapienza" – Italy*

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

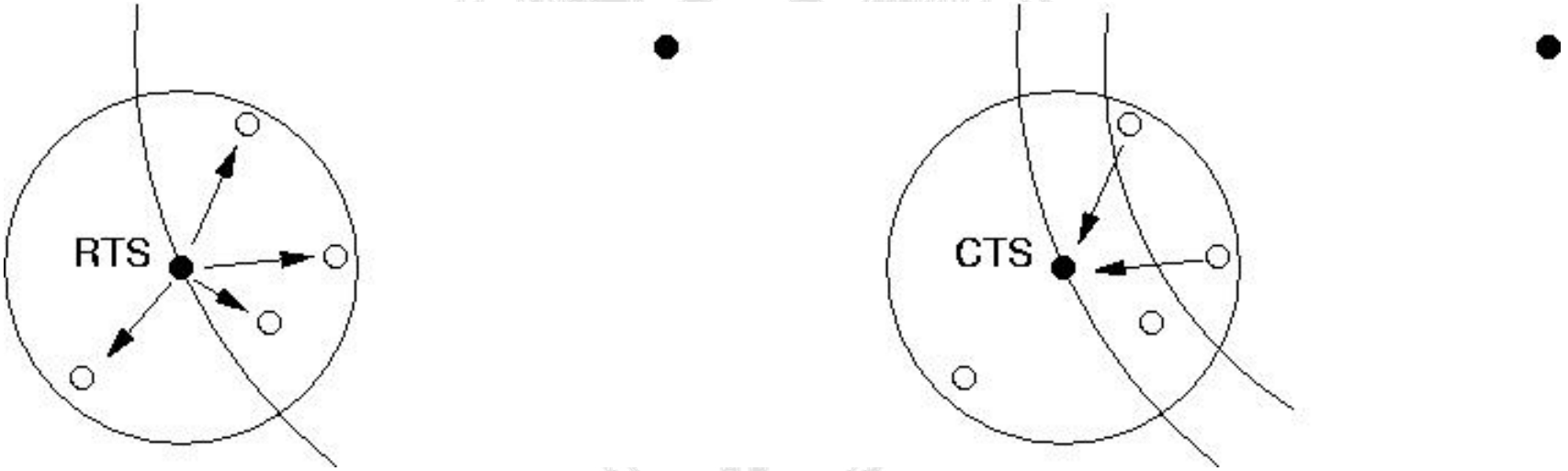
SAPIENZA
UNIVERSITÀ DI ROMA

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)

On GeRaF and cross-layering protocols

- GeRaF is also an example of a cross-layering protocol (MAC and forwarding jointly performed ← RTS/CTS packets that are needed to manage medium access control also used by potential relays to bid, implementing a forwarding policy)
- Cross-layering extensively used by solutions for IoT
- In practice in IoT systems it is enough to desynchronize transmission to avoid collisions →practical GeRaF implementations introduce a jitter in answering the RTS based on 'how good the potentialy relay' is, best relays answer first.

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)

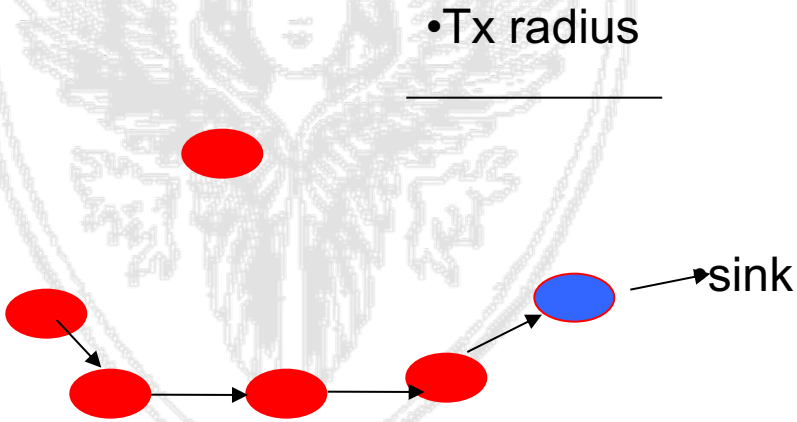
- No. Here is

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…



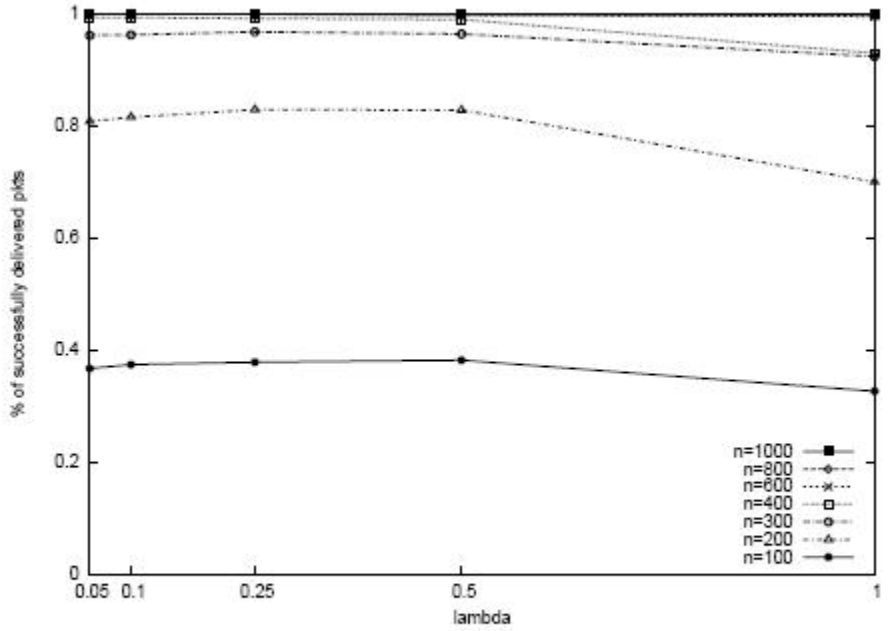
GeRaf performance

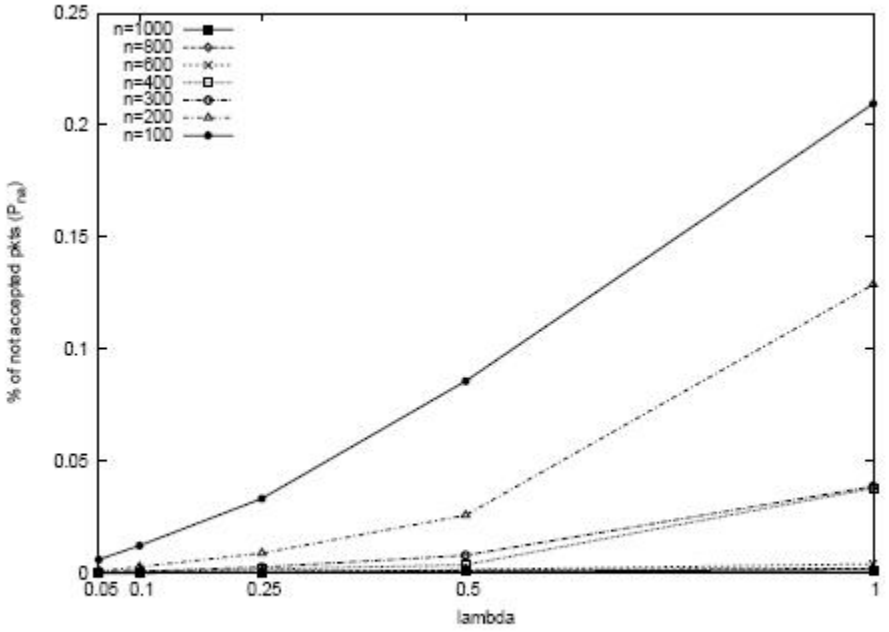
Casari, Marcucci,Nati, Petrioli, Zorzi IEEE MILCOM 2005

- 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

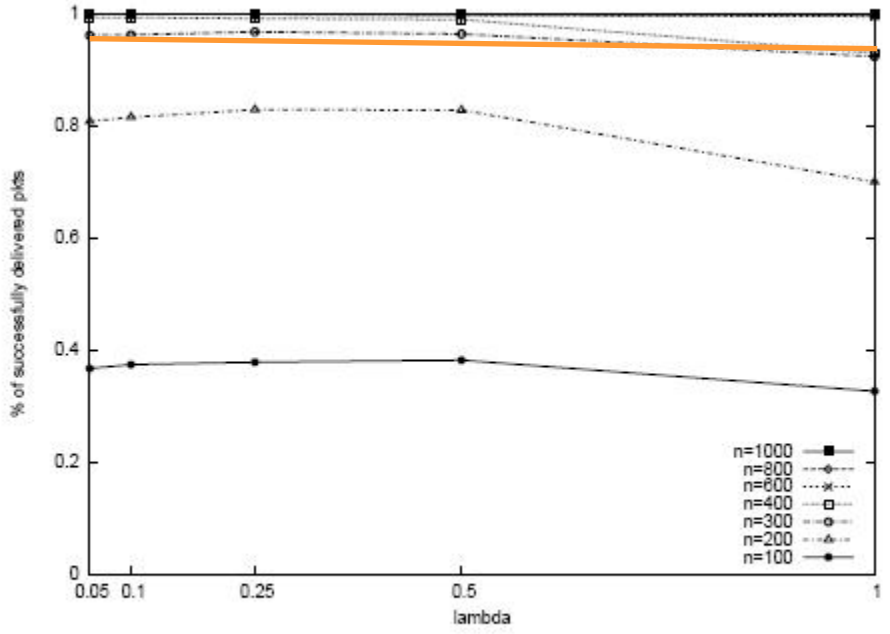
Casari. Marcucci.Nati. Petrioli. Zorzi IEEE MILCOM 2005

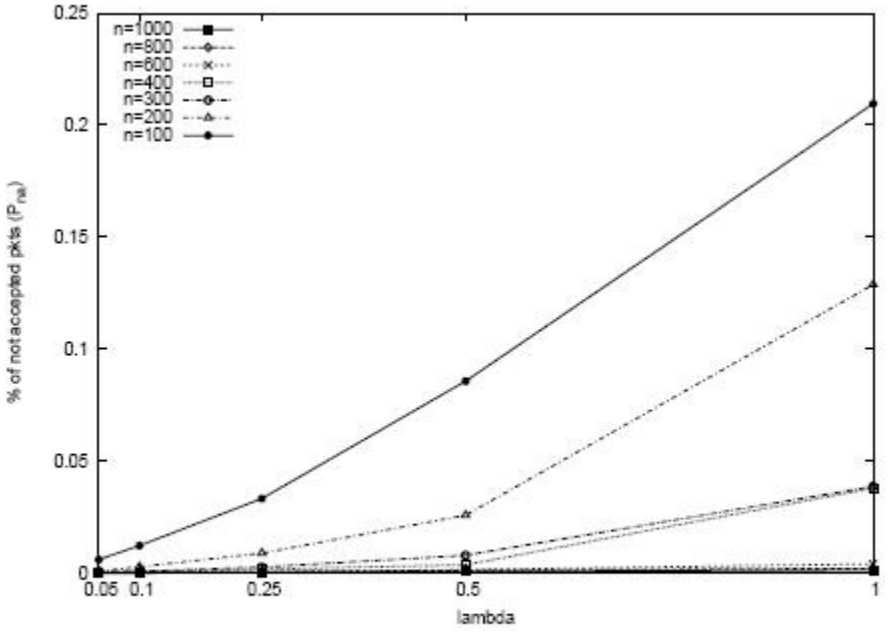




GeRaf performance, d=0.1

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





Backtracking in case of dead ends packet delivery ratio increases. When N=300 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), Petrioli et al IEEE TPDS 2014.

ALBA: a cross-layer integrated protocol stack for medium-large scale wireless sensor networks

Michele Nati, Chiara Petrioli[†]

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

Authors: P. Casari, M. Nati, C. Petrioli, M. Zorzi

This work was partially supported by the European 6th Framework Programme through the Integrated Project (IP) e-SENSE, contract number 027227.

SAPIENZA
UNIVERSITÀ DI ROMA

Outline

- Geographic routing concepts
- Handling dead ends: Related work
- Adaptive Load-Balancing Algorithm (ALBA)
- Rainbow
  - A node-coloring algorithm to route around dead ends
- Simulations settings
- Results for high and low nodal densities
- Impact of localization errors
- Conclusions and discussion

The geographic routing paradigm

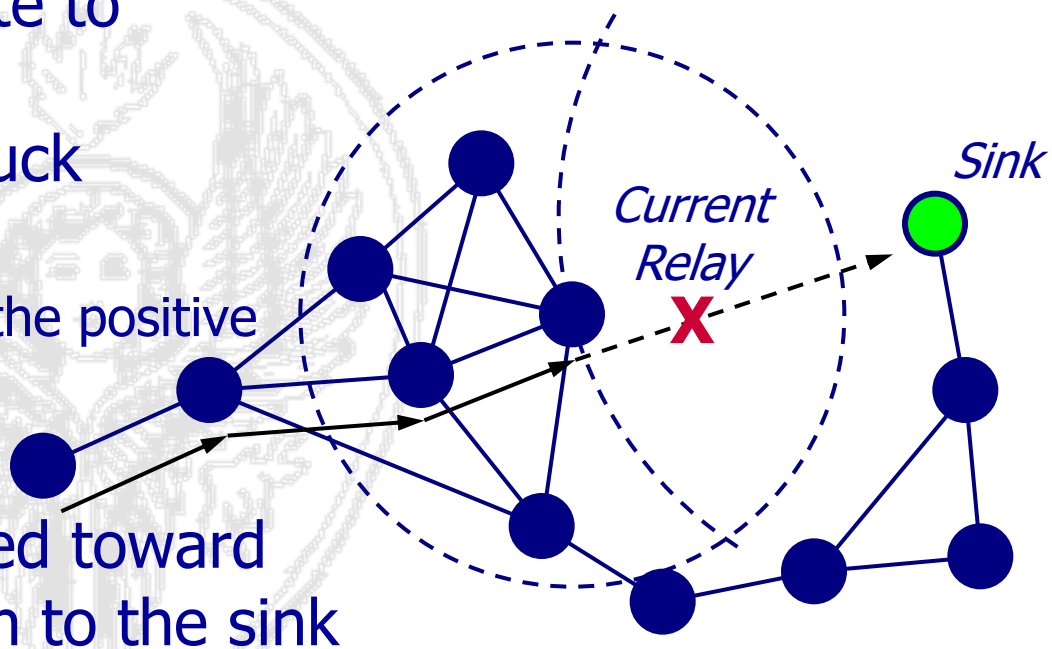
Geographic routing

"Forward the packet to a node that offers geographic advancement toward the destination"

- Pros
  - Virtually *stateless* (needs only knowledge of the source's and the destination's locations)
- Cons
  - Requires positioning estimation (BUT is it really critical?)
  - Requires mechanisms to route packets out of *dead ends*
    - ✓ The present relay is the closest to, yet not a neighbor of, the destination

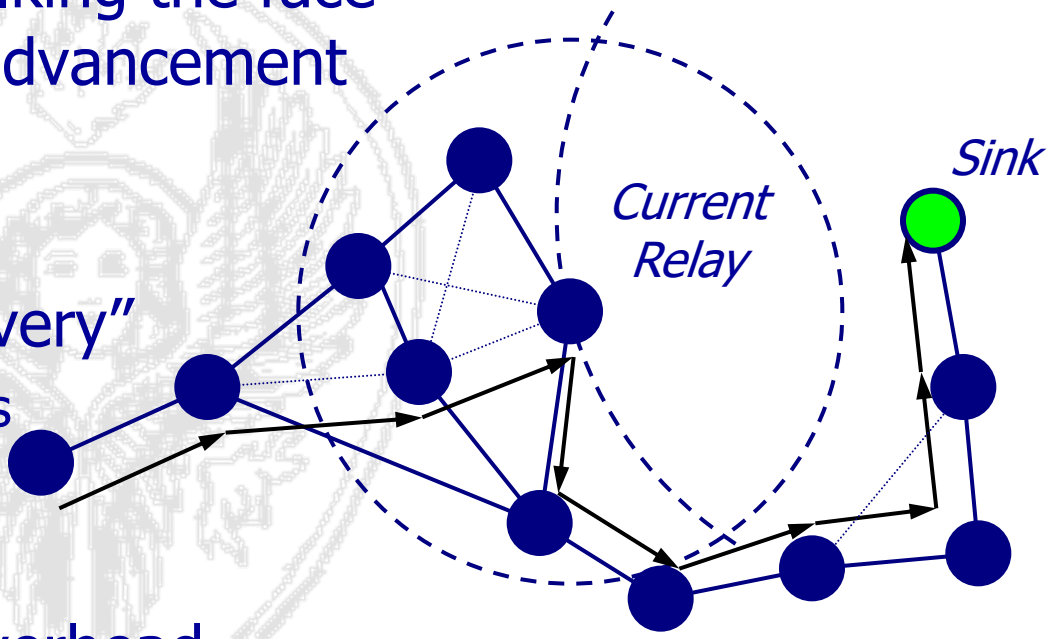
The dead end problem

- If the routing algorithm is tuned to achieve a positive advancement at each step, dead ends may occur
- In this example, a route to the sink is available but the packets get stuck at the current relay
  - There are no nodes in the positive advancement area
- Packet losses occur if data are not re-routed toward nodes that have a path to the sink



The dead end problem, 2

- Current approaches to dead end resolution include **planarizing** the network graph (the resulting graph has no cross links) and walking the face perimeters when the advancement area is empty

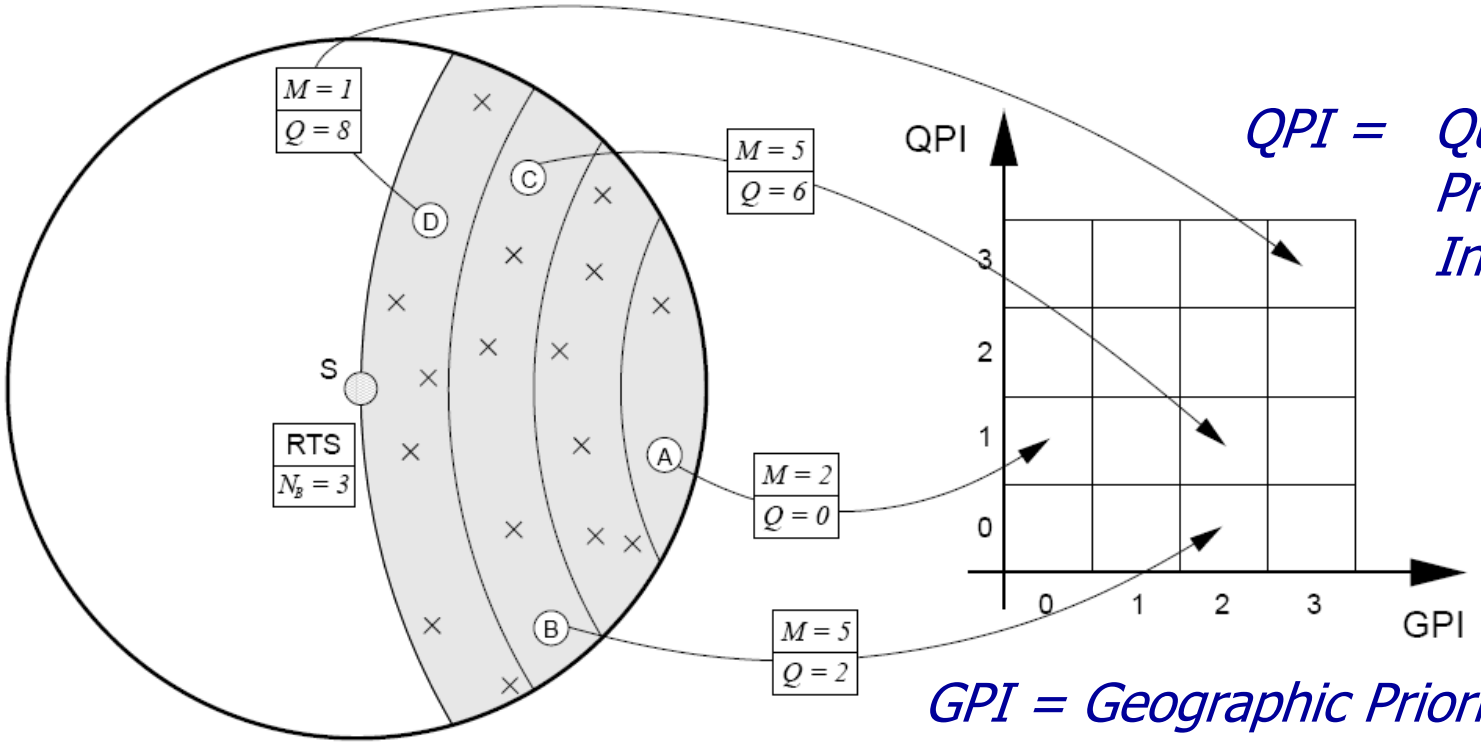


- **Pros**: "Guarantee delivery"
  - Planarization algorithms can be distributed
- **Cons**: planarization overhead, prone to location and channel errors

Our Approach: Basics

- ALBA → Adaptive Load-Balancing Algorithm
  - Integrates interest dissemination and converge-casting
  - Cross layer optimized converge-casting
    - ✓ MAC
    - ✓ Geographic Routing
    - ✓ Mechanisms to load balance traffic among nodes (to decrease the data funneling effect)
    - ✓ Schemes to distributely and efficiently deal with dead ends
- Operations:
  - Nodes forward packets in bursts (up to M_B packets sent back-to-back)
    - ✓ The length of the burst is adapted
  - Forwarders are elected based on
    - ✓ The ability to receive and correctly forward packets
      - The used metric involves the queue level, the past transmission history of the relay, and the number of packets the sender needs to transmit
    - ✓ The geographic proximity to the destination

Our Approach: Basics of the ALBA Protocol



QPI = Queue Priority Index

GPI = Geographic Priority Index

Queue level

$$QPI = \lceil (Q + N_B)/M \rceil - 1$$

Requested length of the burst

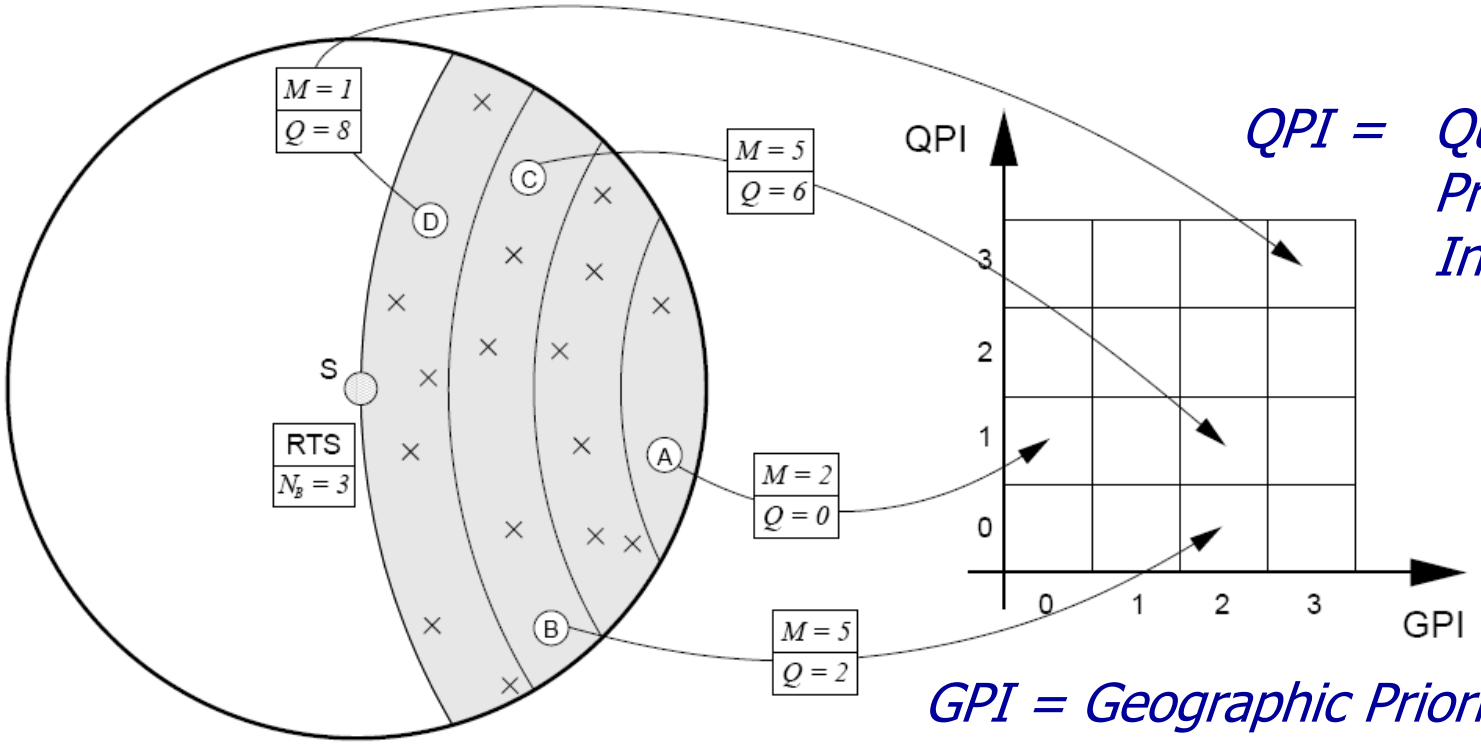
Average length of a burst the relay expects to transmit correctly

SAPIENZA
UNIVERSITÀ DI ROMA

ALBA Features

- The metric used for the choice of the relay ensures load balancing as it preferably chooses relays with
  - Low queue, especially if N_B is high
  - Good forwarding history (through M)
- Nodes employ duty-cycling to enforce energy saving
- The relay selection works in phases
  - Phase 1: Selection of the best QPI
    - ✓ Attempt 1 search for QPI=0, Attempt 2 for QPI=0,1, and so on
    - ✓ **Awaking nodes can participate in this selection phase**
  - Phase 2: Selection of the best GPI
    - ✓ Performed if more than one node with the same QPI was found
    - ✓ Awaking nodes cannot participate here (to speed up completion)
- **Still prone to dead ends**

ALBA: An example

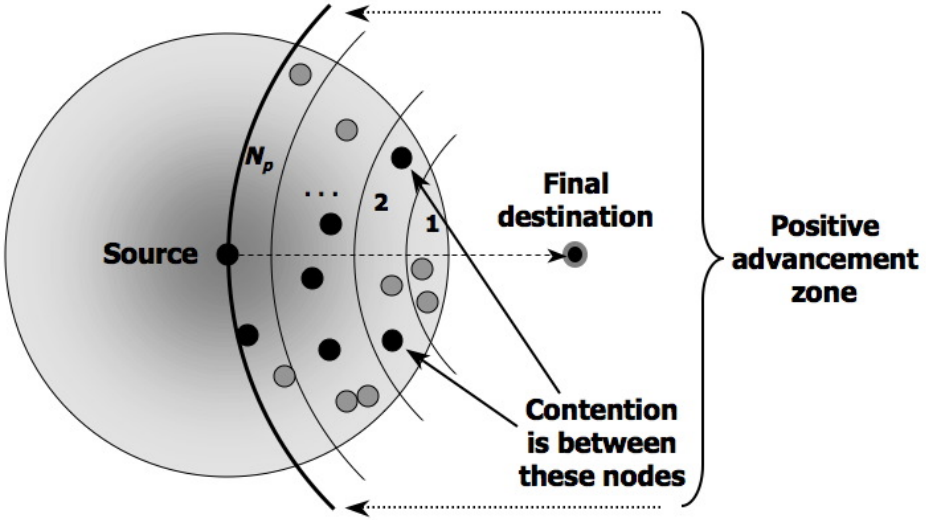


QPI = Queue Priority Index

GPI = Geographic Priority Index

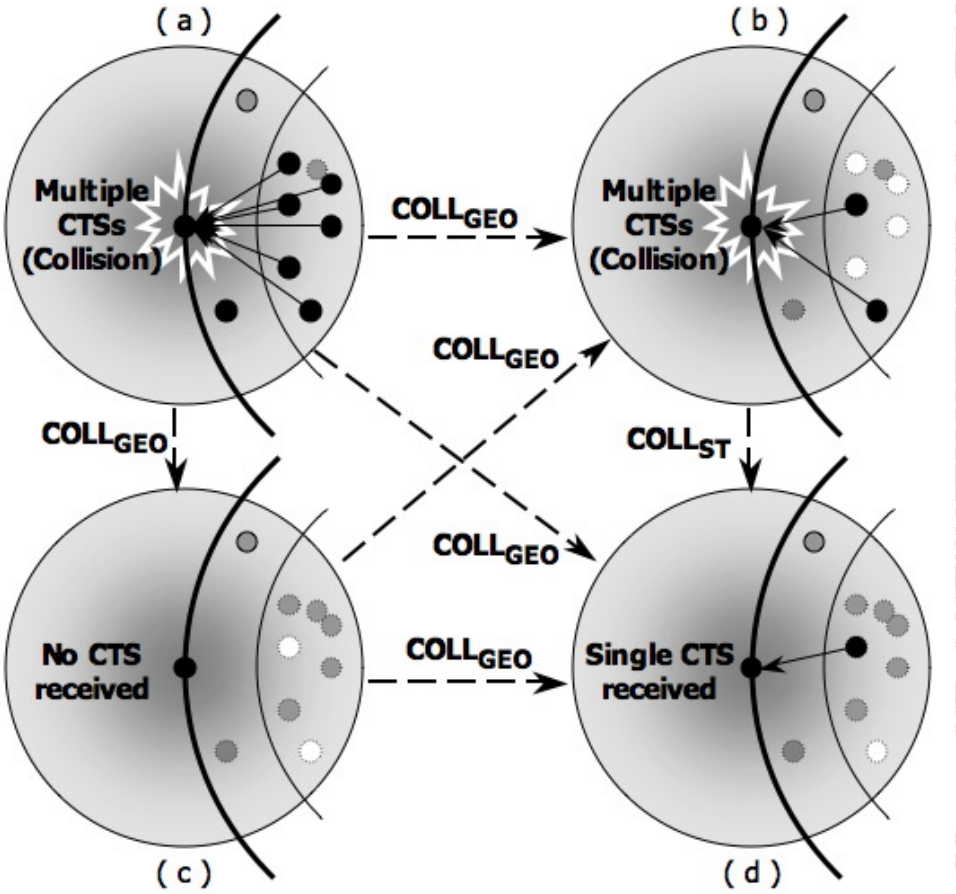
1. Node A is nearer to the sink (GPI =1) but has a low QPI (M=2); node B, is farther but has greater reliability (M) and comparable queue occupancy (Q); B has a greater QPI than A
2. In case of node B is sleeping at transmission time, node A is selected for its better GPI

Contention Mechanism



- Source nodes send a RTS msg to query relays. Relays respond with CTSs
- No response: a CONTINUE msg pings the following region
- Collision: a COLLISION msg starts the collision resolution algorithm
- CTS received: Burst of DATA transmission starts

Collision Resolution Algorithm



- Upon receiving a $COLL_{GEO}$ msg, nodes reply based on their GPI
- Upon receiving a $COLL_{ST}$ msg, nodes persist in sending CTSs with probability 0.5
- Should they all decide to stay silent, the following $COLL_{ST}$ msg enables a further decision
- Eventually, the process ends with a single valid relay being selected

SAPIENZA
UNIVERSITÀ DI ROMA

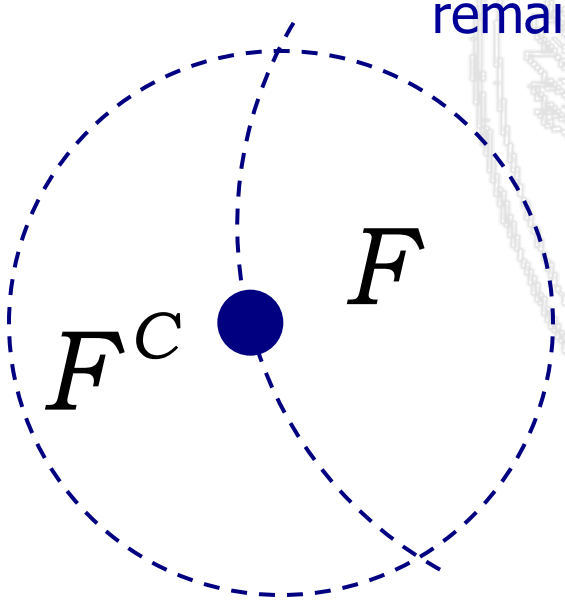
Rainbow

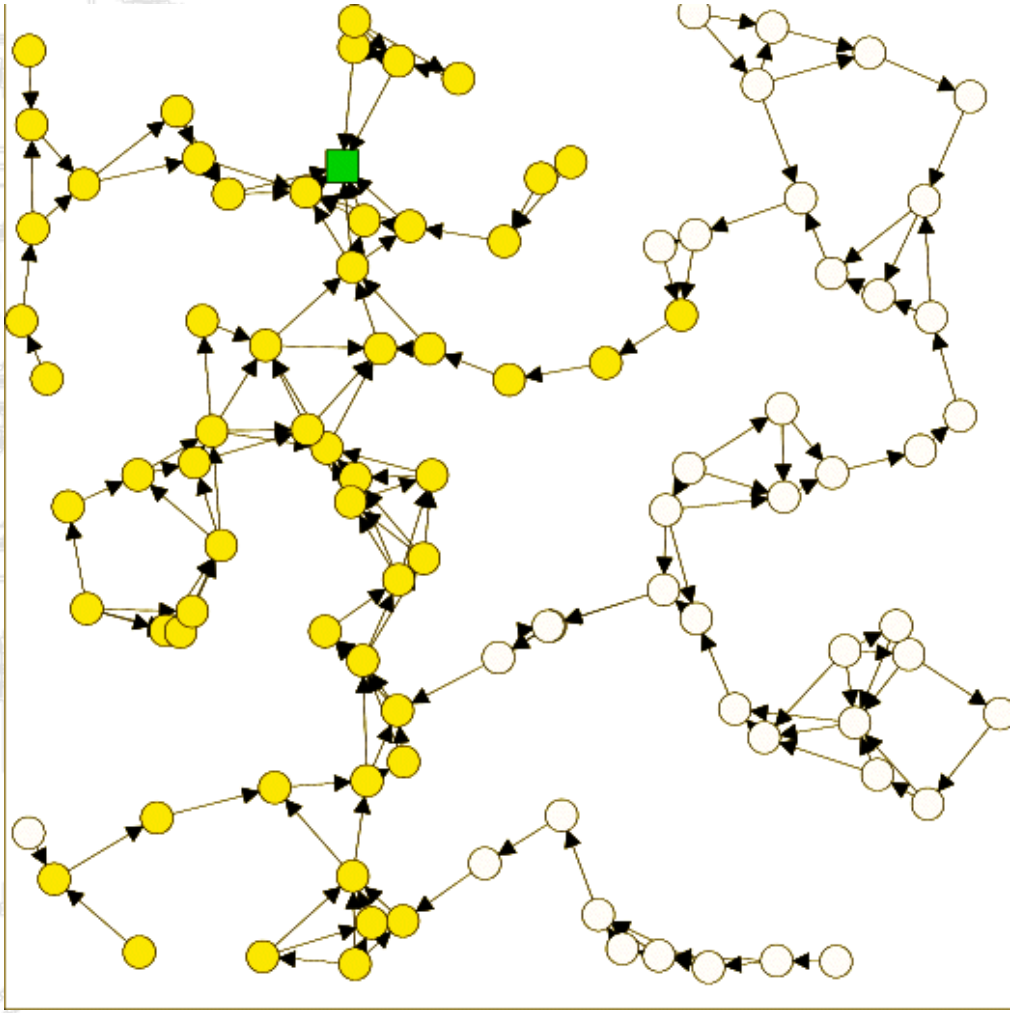
A node coloring algorithm for routing out of dead ends and around connectivity holes

- Concepts
  - In low density topologies, a method for routing around dead ends is needed
  - Nodes that recognize themselves as dead ends progressively stop volunteering as relays
  - To route traffic out of the dead end, they begin to transmit packets backward, in the negative advancement zone
  - Hopefully, a relay that has a greedy forwarding path to the sink can eventually be found
  - A recursive coloring procedure is used

Rainbow node coloring scheme – Yellow nodes

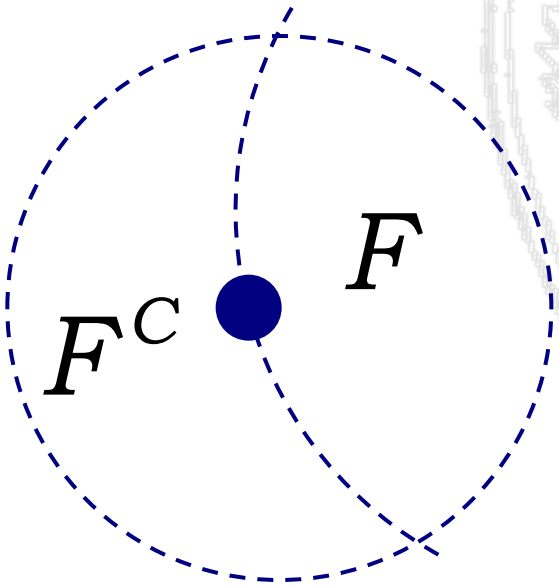
- F and F^C are the positive and negative advancement areas, respectively
- Initially, all nodes are "yellow"
- All nodes that exhibit a greedy path to the sink remain "yellow"



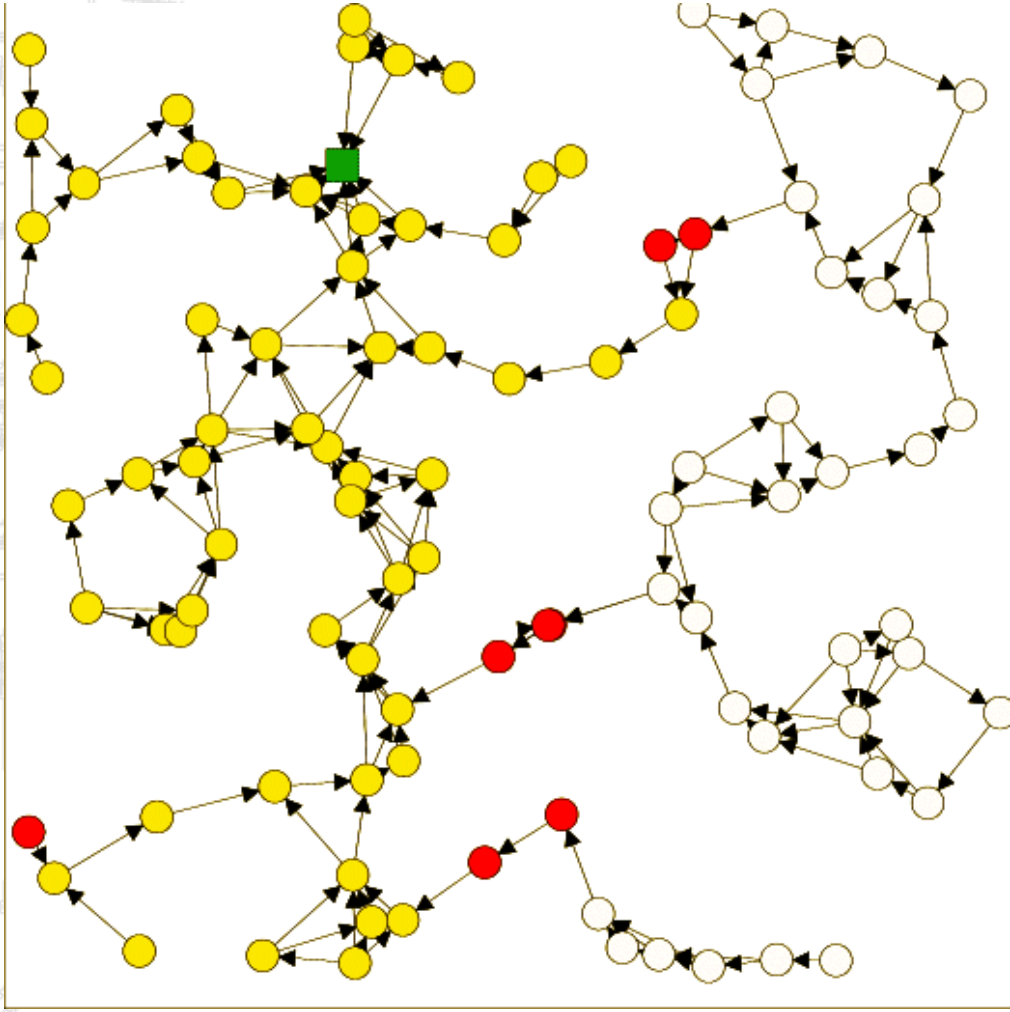


Rainbow Node Coloring Scheme: Red nodes

- If a yellow node cannot forward packets further, it switches to "red"
- From now, it looks for either "red" or "yellow" relays in F^C

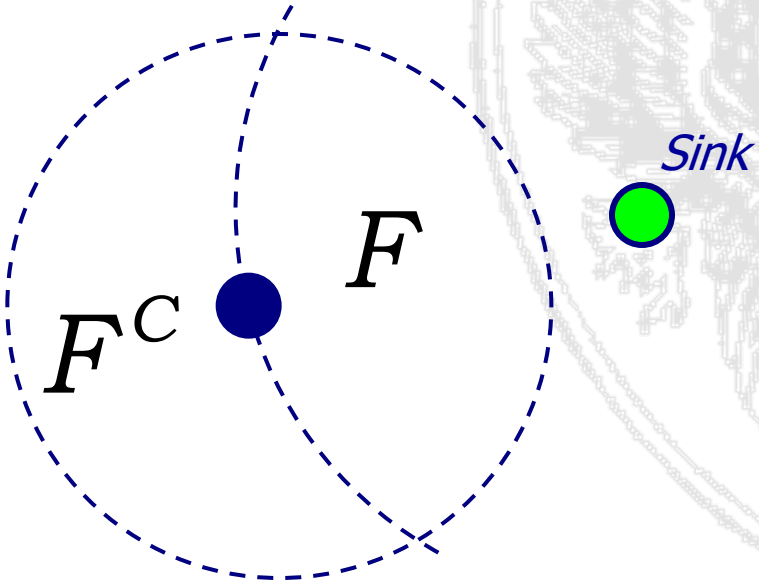


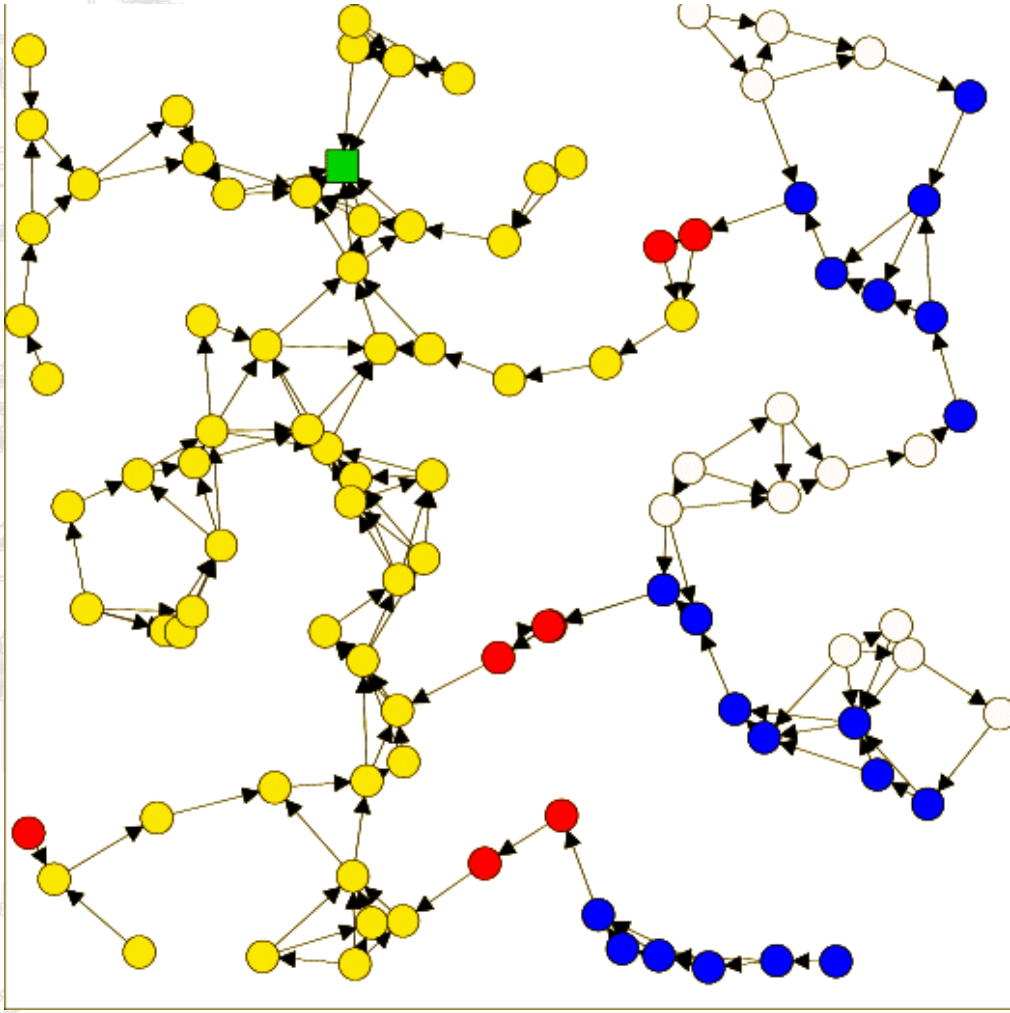
Sink



Rainbow Node Coloring Scheme: Blue Nodes

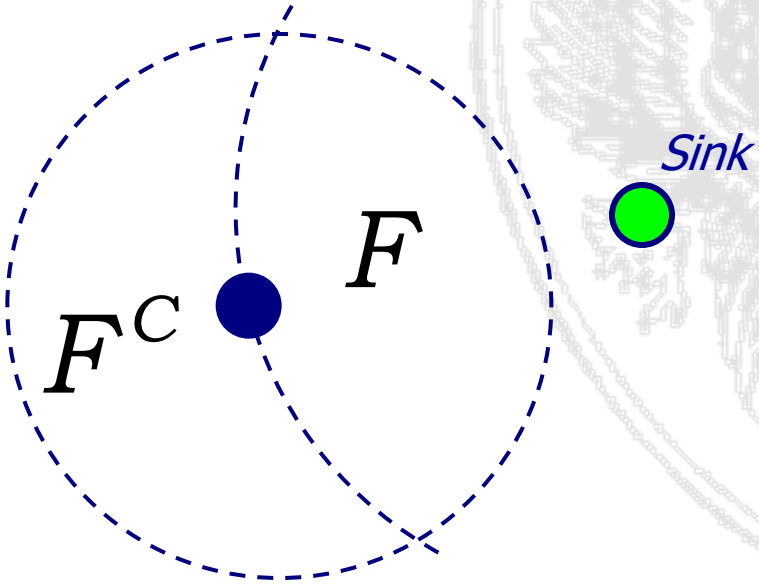
- If red nodes cannot advance packets, they turn to "blue"
- Again, they switch to look for relays in *F*
- They only look for "red" or "blue relays

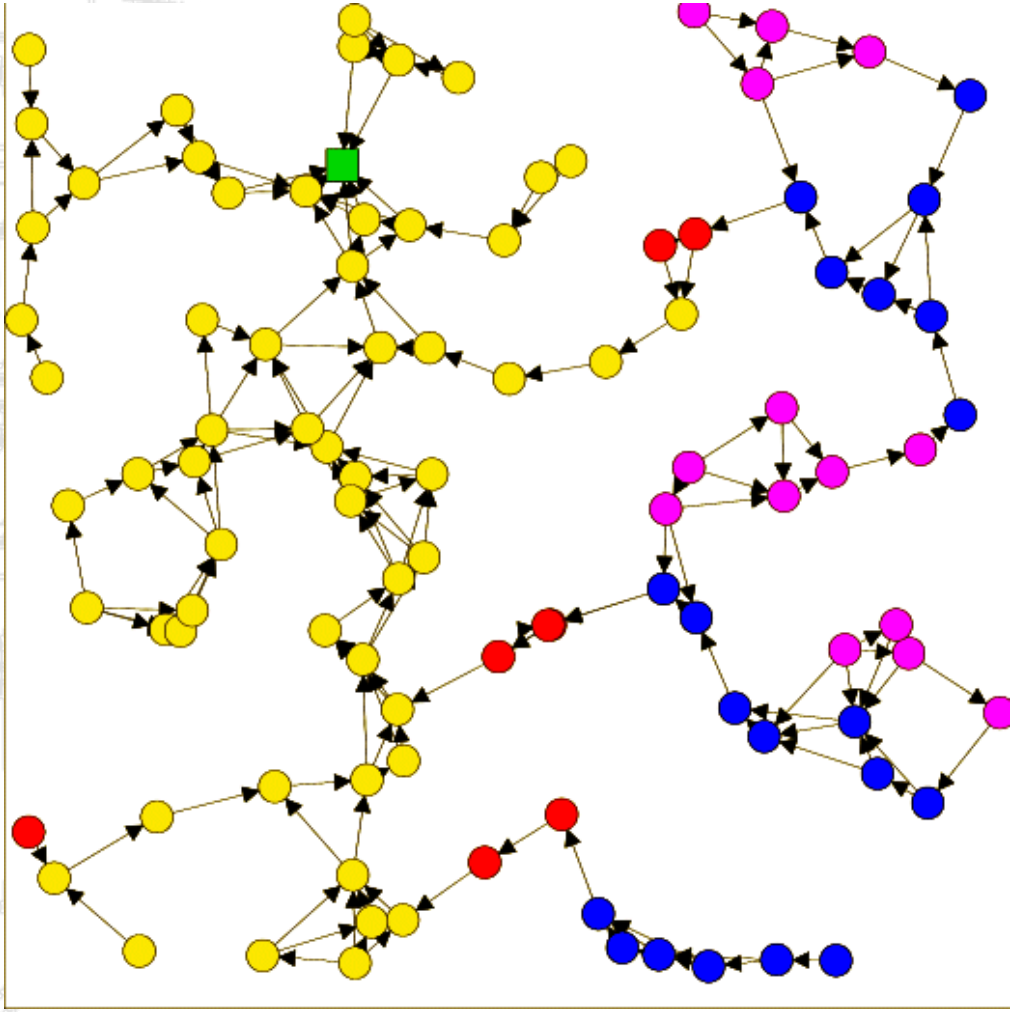




Rainbow Node Coloring Scheme: Violet nodes

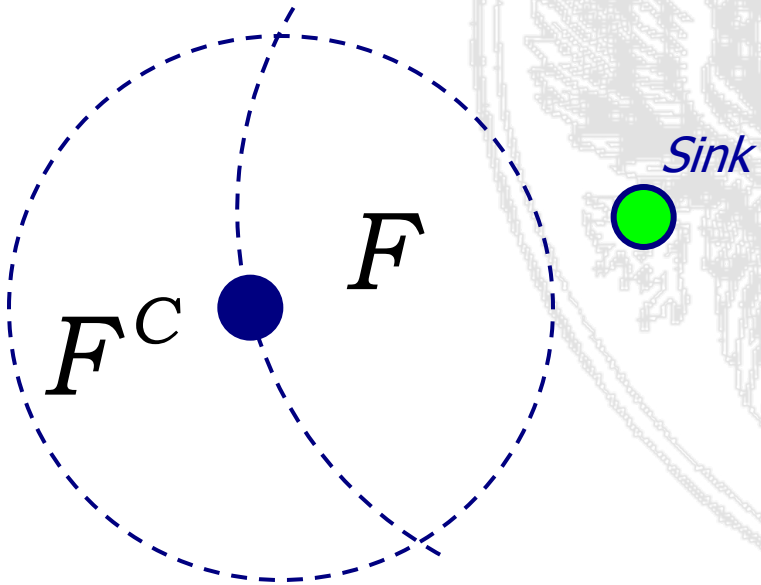
- If blue nodes still have problems finding relays they switch color again, to "violet"
- Like red nodes, they look for relays in F^C…
- …but only "blue" or "violet"





Rainbow Node Coloring Scheme: In general

- The number h of needed colors is fully general
  - The greater the number of colors, the more nodes can be connected to the converge-casting tree
- In general, given *h* labels $C_1, C_2, \ldots, C_h$ …
- The nodes switch from a label to the following one every time they perceive to be a dead end with their present label



- Nodes labeled C_1 are the only one with a greedy path to the sink
- Nodes with odd labels ($C_1, C_3, \ldots$) always look for relays in *F*
- Nodes with even labels ($C_2, C_4, \ldots$) always look for relays in F^C
- A node with label C_k always looks for C_k- or C_{k-1}-nodes, except C_1-nodes that always look for other C_1-nodes

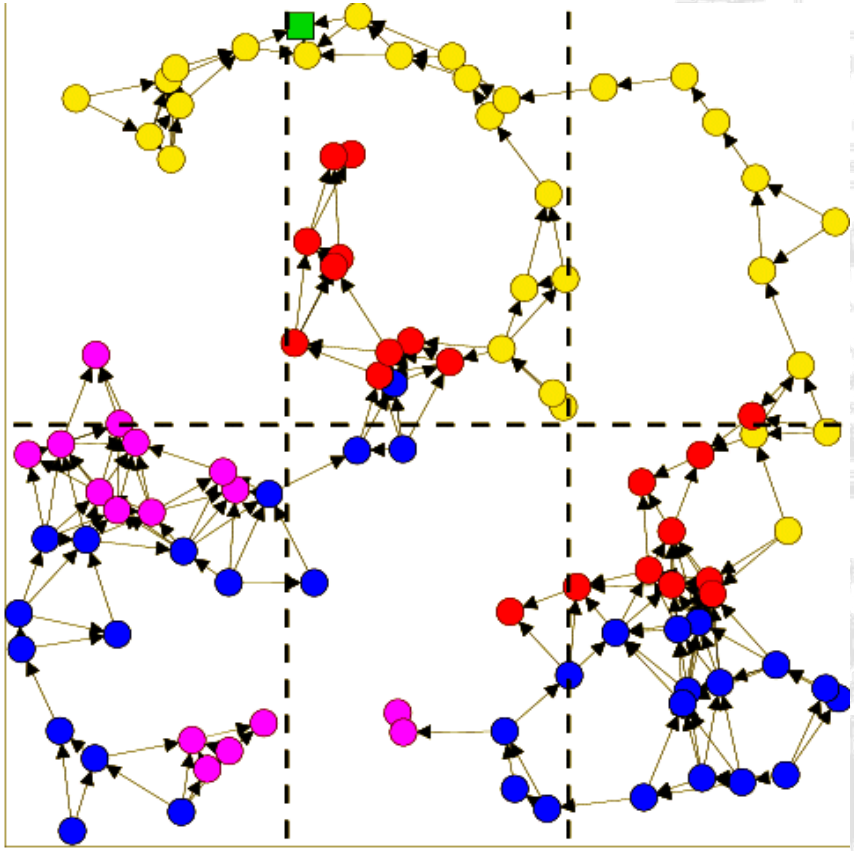
Rainbow: Wrap up

- **Concepts**
  - Nodes progressively realize to be dead end and automatically adapt to this condition
    - ✓ No abrupt changes in the color of a node (relays might be present but just *unavailable* for the moment)
  - More colors mean more nodes can successfully deliver packets
- **Pros**
  - Effectively routes around dead ends
  - Completely blind and distributed
  - Does not require planarization
  - The load-balancing features of ALBA are seamlessly used throughout
- **Cons**
  - The network requires some training for nodes to achieve the correct color

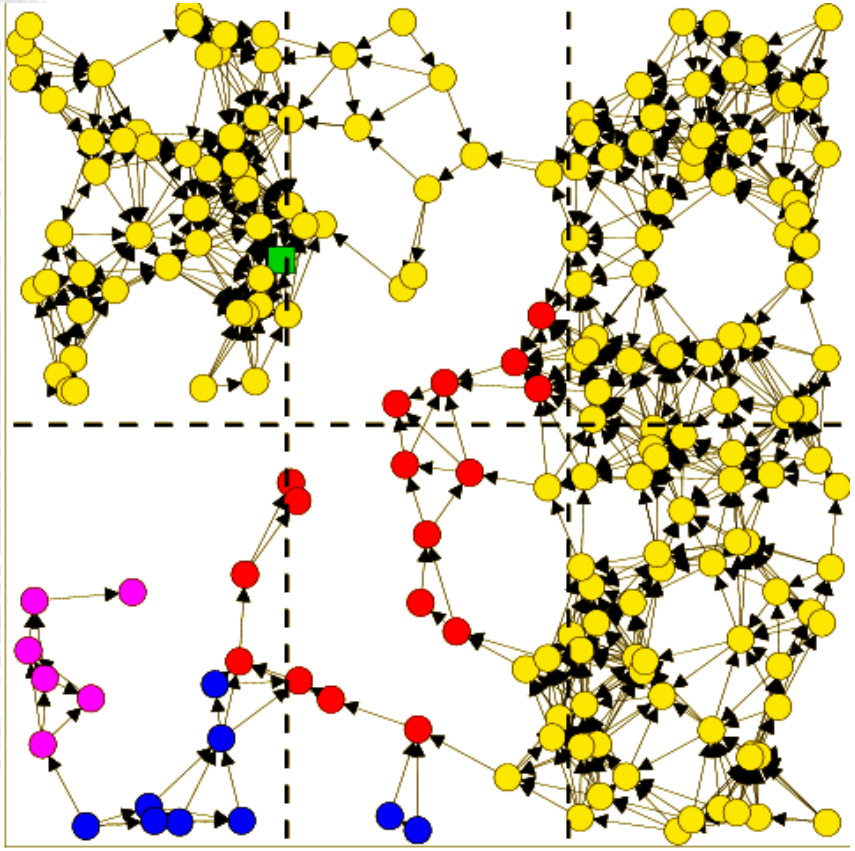
Results: Simulation Setting

- Simulation area: 320 m x 320 m
  - Random and uniform deployment
  - Non-uniform deployment
    - ✓ A more general case than uniform deployment
    - ✓ The area is divided in 3 high-density and 3 low-density zones
    - ✓ 75% of the nodes are randomly placed in high-density zones, the remaining 25% in low-density zones
- First set of results → Comparison
  - ALBA-R vs. GeRaF and MACRO
- Second set of results → High node densities
  - Show that Rainbow does not decrease performance if not used
  - N = 300, 600, 800, 1000 nodes
- Third set of results → Low node densities and different number of colors used in Rainbow
  - Used to show the effectiveness of Rainbow in rerouting packets

Sample non-Uniform Deployments



100 nodes



200 nodes

Localization in sensor networks

Thanks to Prof. Mani Srivastava

Some of these slides come from his tutorial on WSNs

SAPIENZA
UNIVERSITÀ DI ROMA

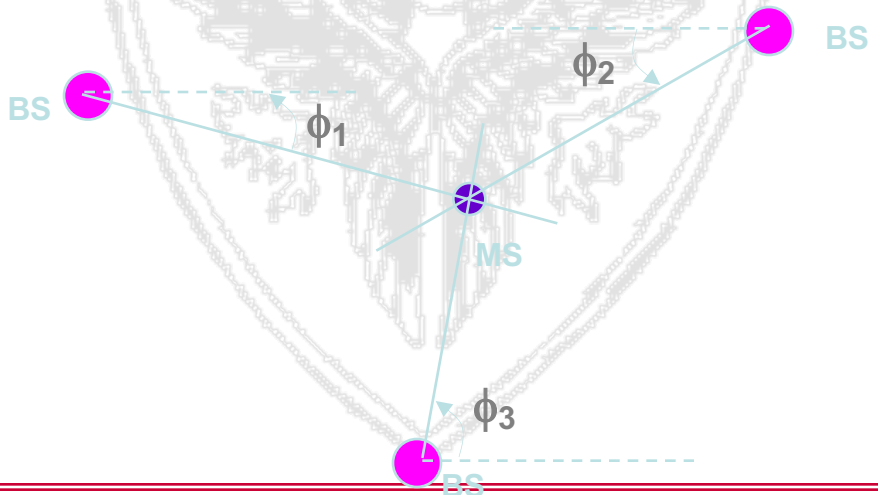
Localization

- 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

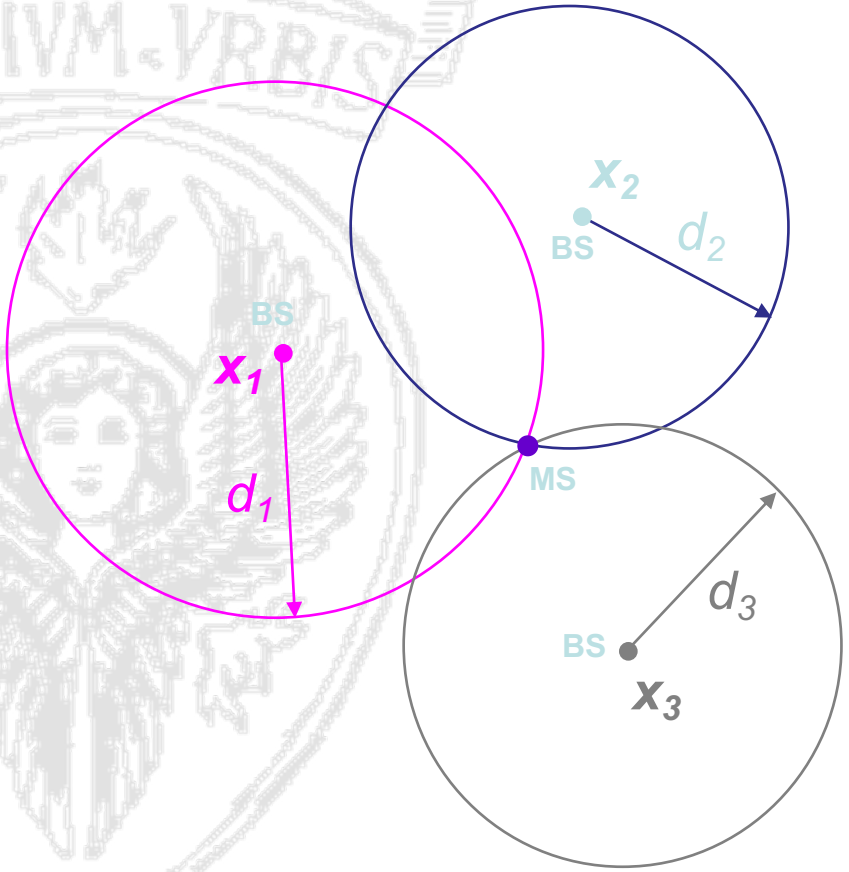
- Basic step is to evaluate distance between two nodes (ranging). Different techniques depending on the available HW:
  - AoA (e.g. directional antennas)
  - RSS
  - ToA
- Range free approaches (number of hops between nodes used to estimate the distance between them without using any extra HW)

- 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)

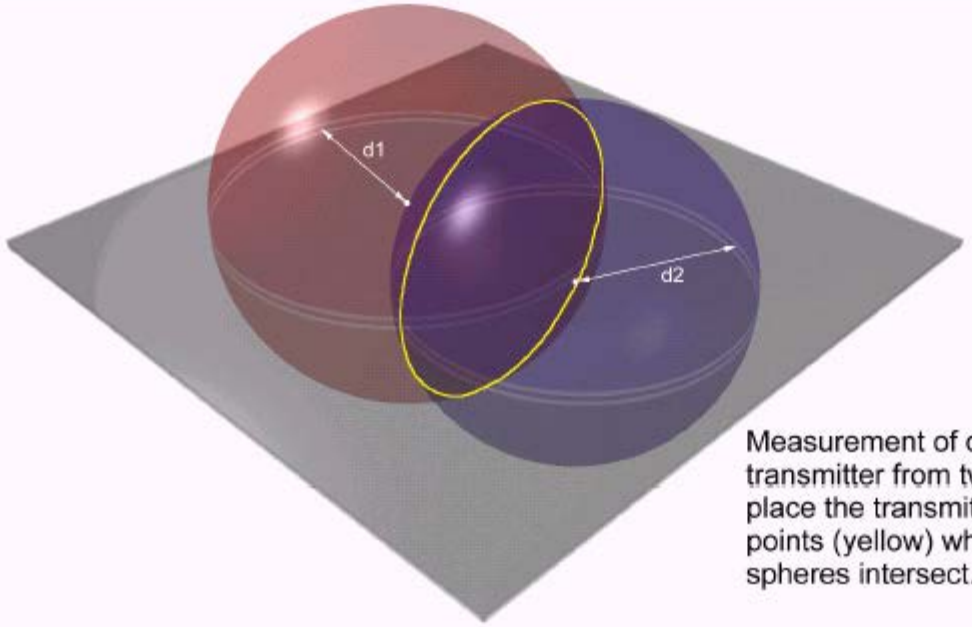
- Measure distance to landmarks, or Ranging
  - e.g. Radiolocation using signal-strength or time-of-flight
    - ✓ also done with optical and acoustic signals
  - Distance via received signal strength
    - ✓ use a mathematical model that describes the path loss attenuation with distance
      - each 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.)?

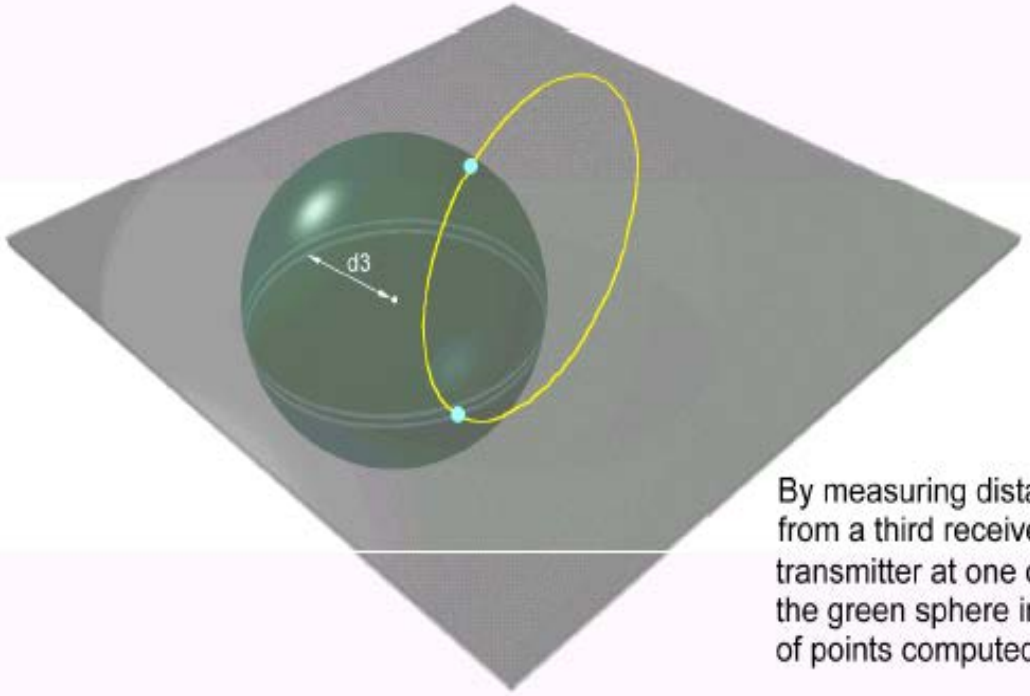
→will have to identify the best 'guess' given errors

Location in 3D



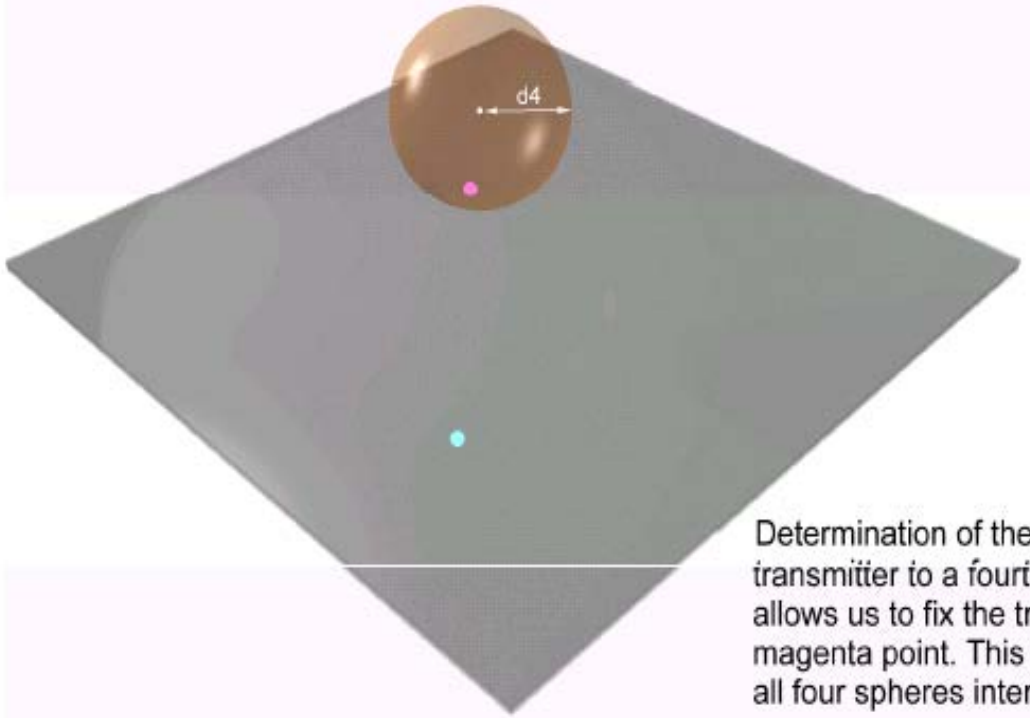
Measurement of distances d1 & d2 to the transmitter from two receivers allows us to place the transmitter on a circular locus of points (yellow) where the red and blue spheres intersect.

Location in 3D



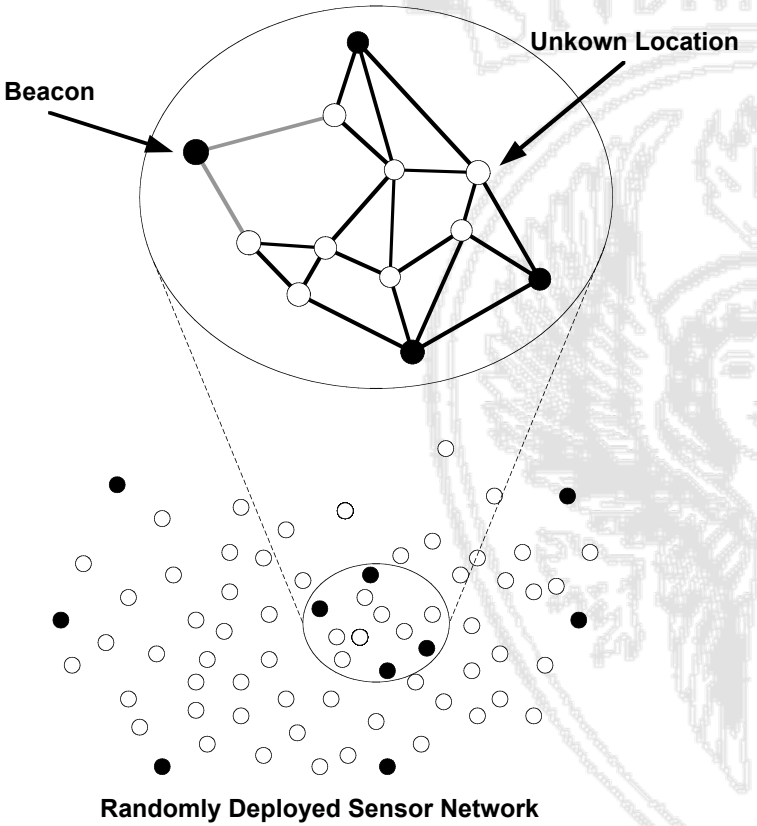
By measuring distance d3 to the transmitter from a third receiver, we can place the transmitter at one of two points (cyan) where the green sphere intersects the circular locus of points computed previously.

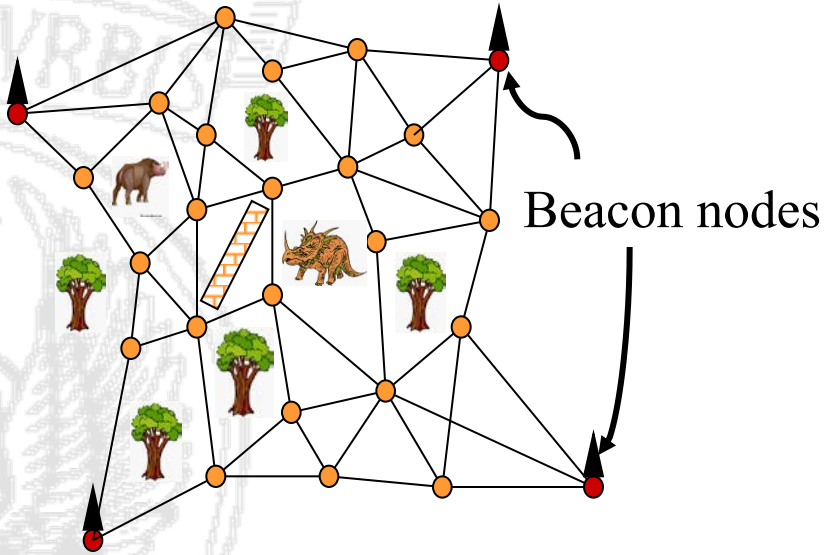
Location in 3D



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.

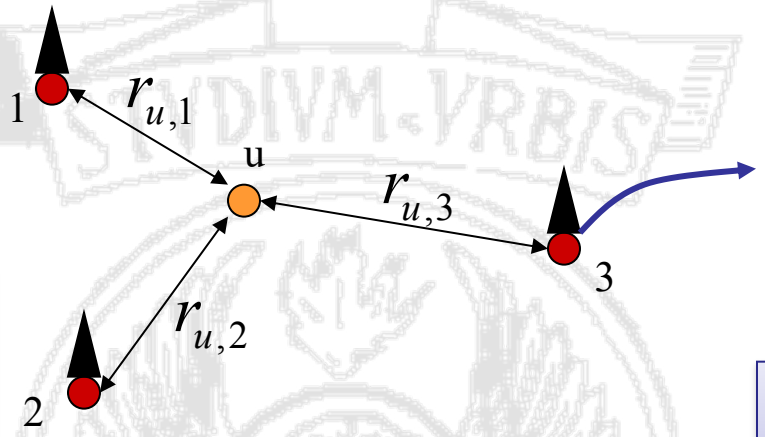
A possible solution: Absolute Localization





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

Atomic Multilateration



Beacon node with known location

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

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

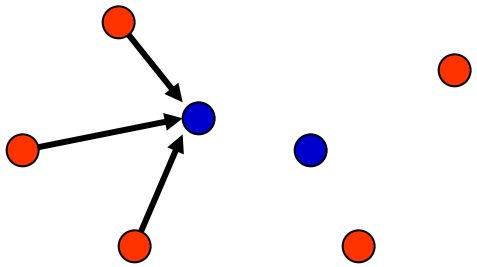
$(\hat{x}_u, \hat{y}_u)$ - initial position estimate for node u

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.



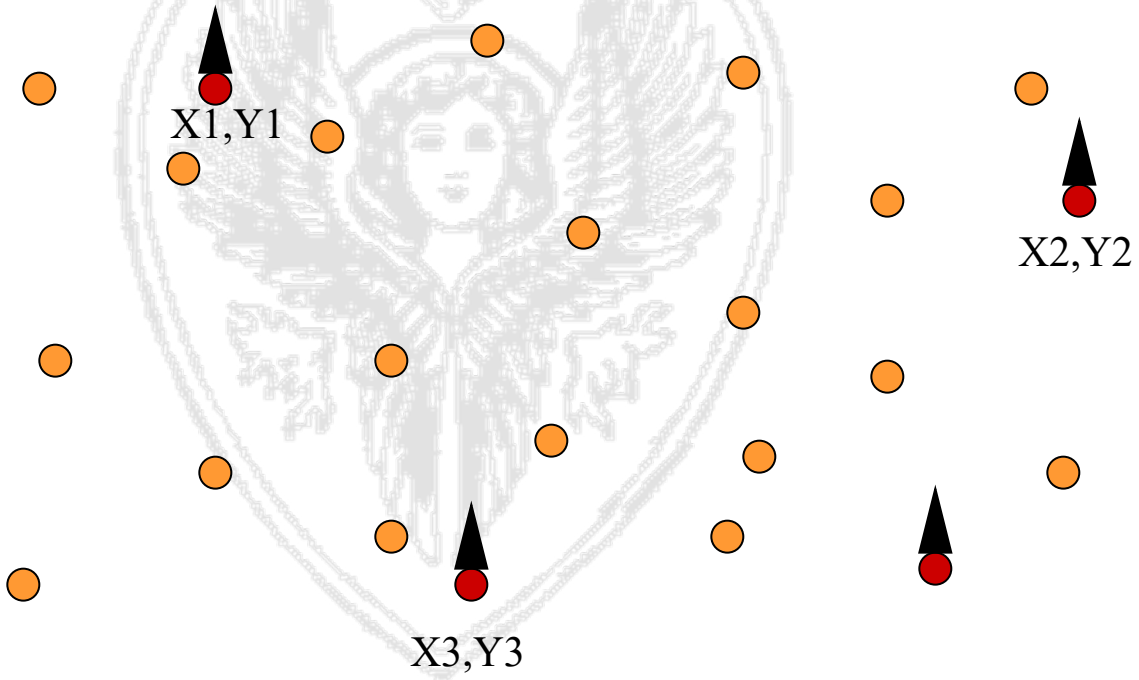
- Some observations:
  - Can work for small networks, if ranging is accurate
  - Energy efficient
  - Still requires quite a lot of initial beacons
  - Suffers from error accumulation
  - **Bad geometry yields bad results => unpredictable performance**
  - Still a useful primitive for Distributed Collaborative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

Range free localization

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

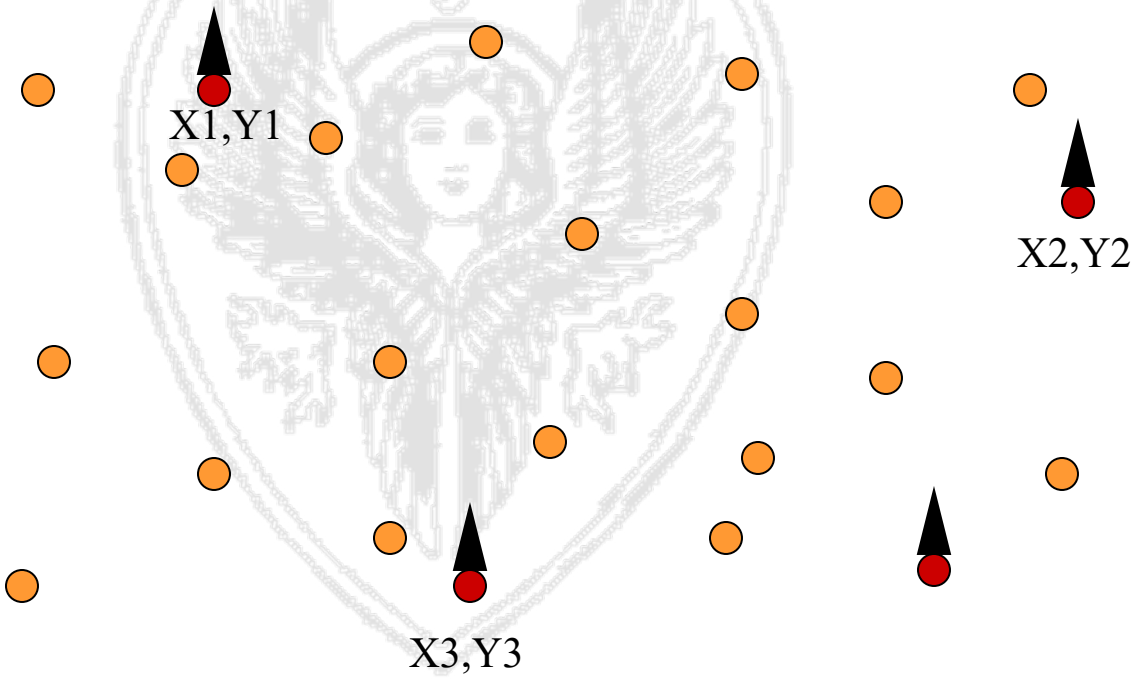
How does it work?...

- 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