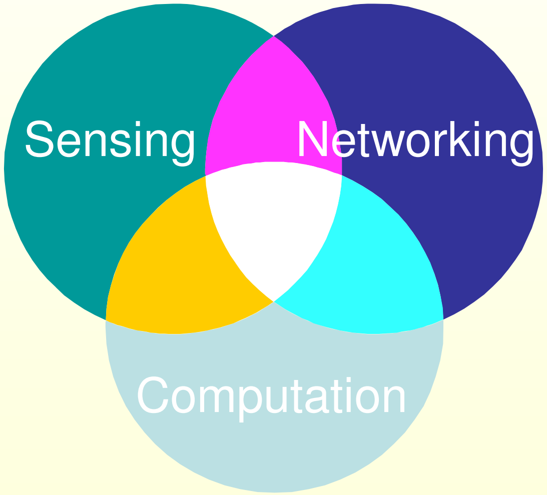
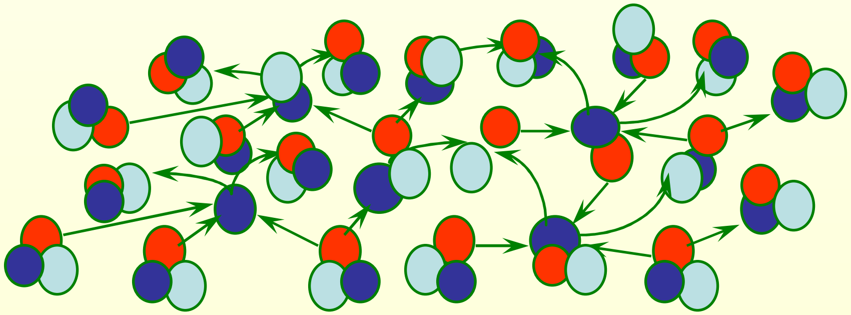
Infrastructure Establishment



Leonidas Guibas
Stanford University



CS428

Infrastructure Establishment in a Sensor Network

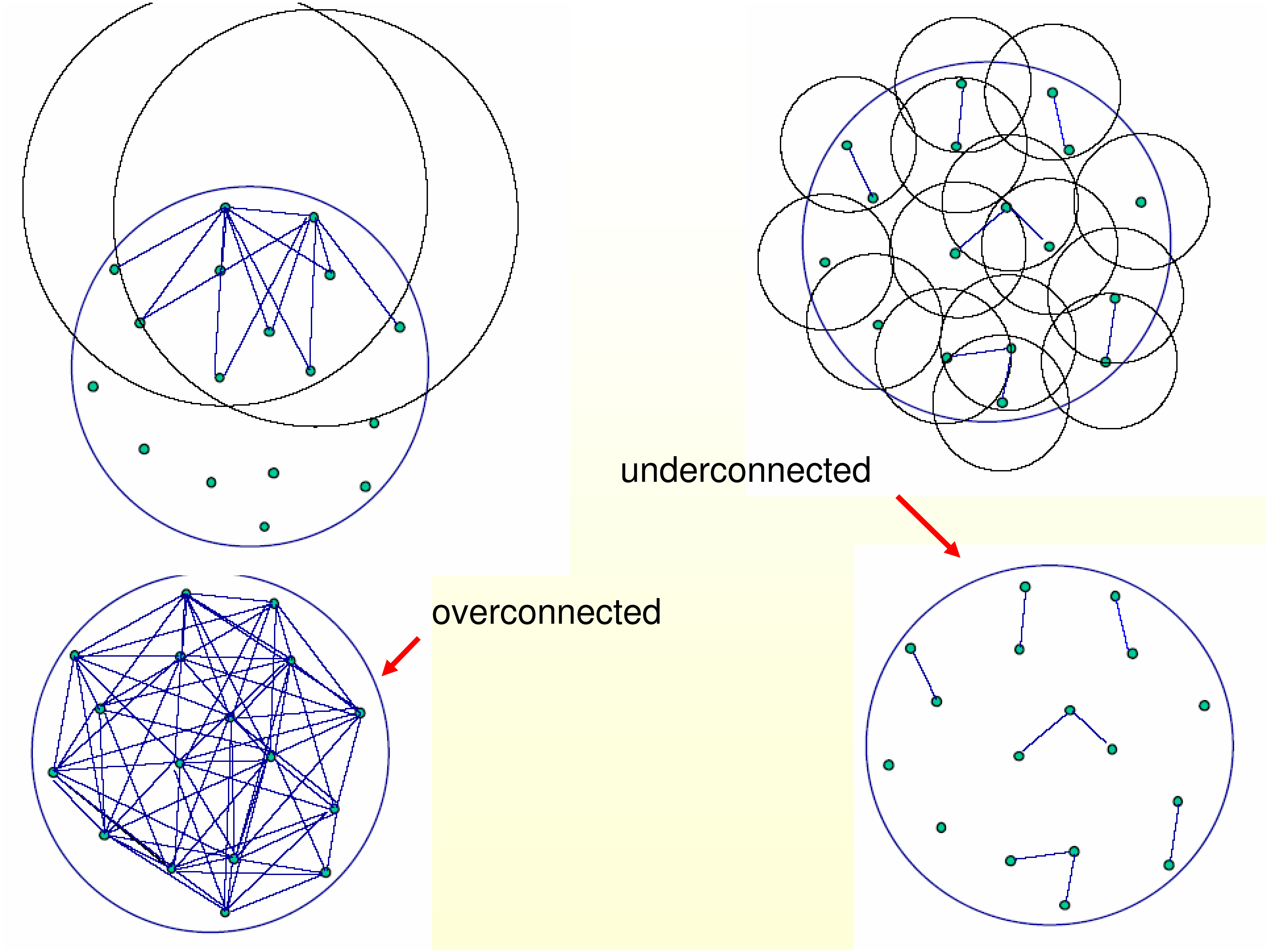
For the sensor network to function as a system, the individual nodes must be brought into a common framework and establish necessary infrastructure

- Network topology discovery and control
- Node clustering and hierarchy formation
- Clock synchronization
- Localization
- Location and other network-wide services

Topology Discovery and Control

Topology Discovery and Control

- Each node must discover which other nodes it can talk to directly
- Depends on the radio power setting – a node may be able to vary that setting according to local conditions
- These elementary connections establish the topology of the network
- We always want radio power settings so that a network that is connected
- But ranges that are too long waste power and cause interference
- Assume for now that node locations are known

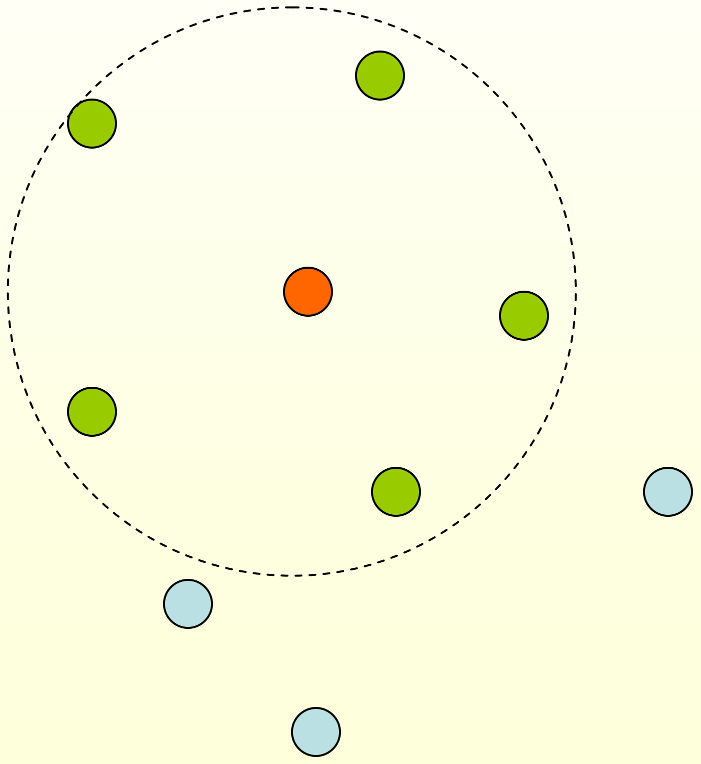


underconnected

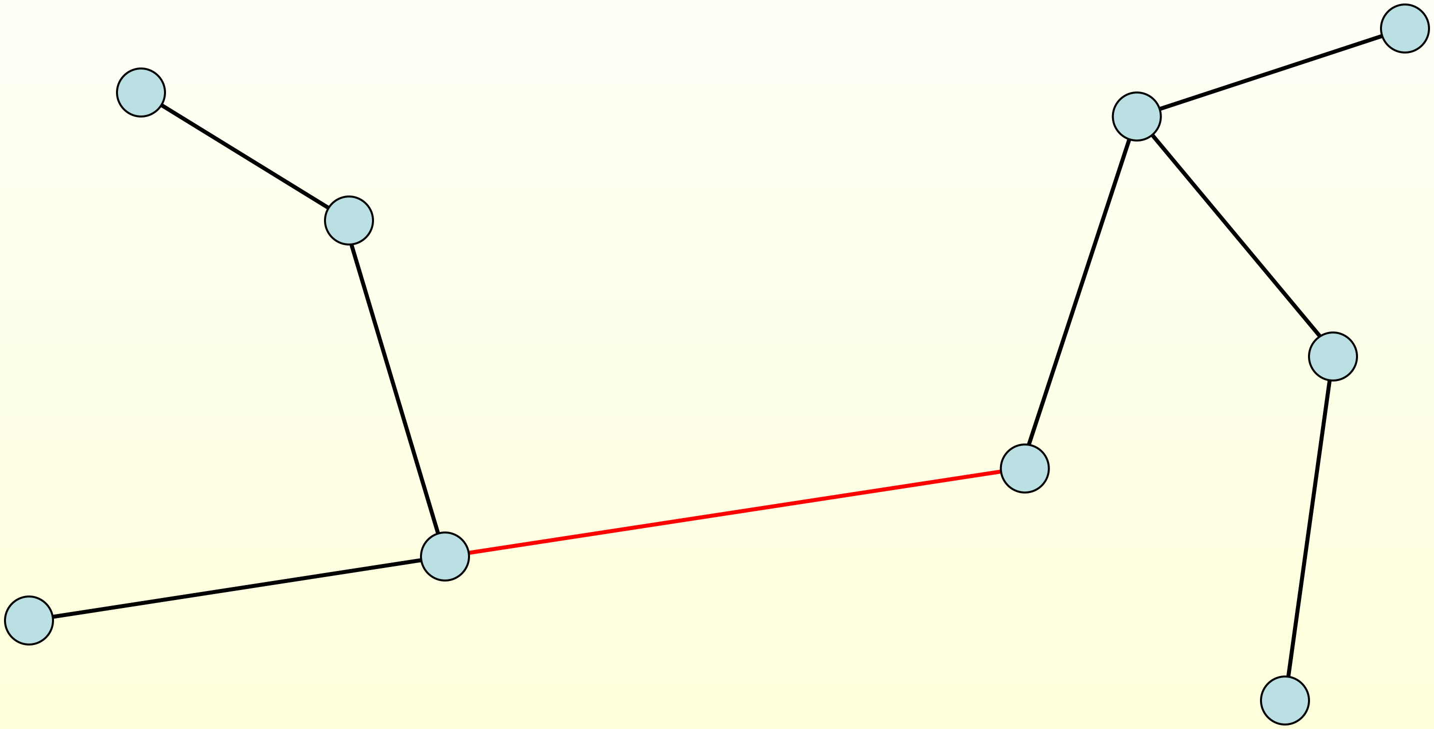
overconnected

The Critical Transmitting Range (CTR) Problem

- Assume all nodes must use exactly the same radio range
- How can we compute the minimum radio range that is guaranteed to just connect all the nodes?
- Theorem: It is the length of the longest edge of an MST connecting the nodes
- The MST can be computed in a distributed fashion [Gallager, Humblet, Spira]



Why The MST Solves the CTR Problem



A Probabilistic Variant

- Say *n* points are dropped in the unit square randomly and uniformly. What can we say about the CTR *r* ?
- With high probability it will be:

$$r = c\sqrt{\frac{\log n}{n}}$$

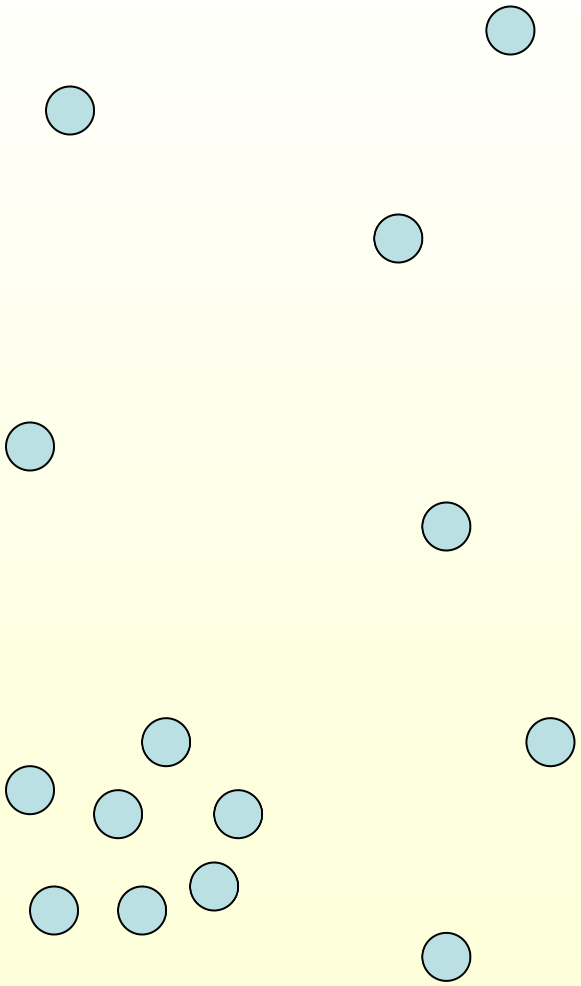
- Result from *Geometric Random Graph Theory* – there is a critical constant *c* for a threshold effect

Variable Transmitting Ranges

- If the node density is highly variable, then we should choose short ranges when the density is high, and long when it's low
- The goal is to minimize

$$\sum_{i=1}^{n} r_i^{\alpha},$$

while still connecting the network



The Range Assignment Problem

- The previous minimization problem is know as the range assignment problem
- Unfortunately, it is NP-complete ...
- The MST of the nodes provides a factor 2 approximation
  - define graph weights by $w(i,j) = \delta^{\alpha}(i,j)$
  - Solve the MST problem
  - set the range of each node so as to reach all of its MST neighbors

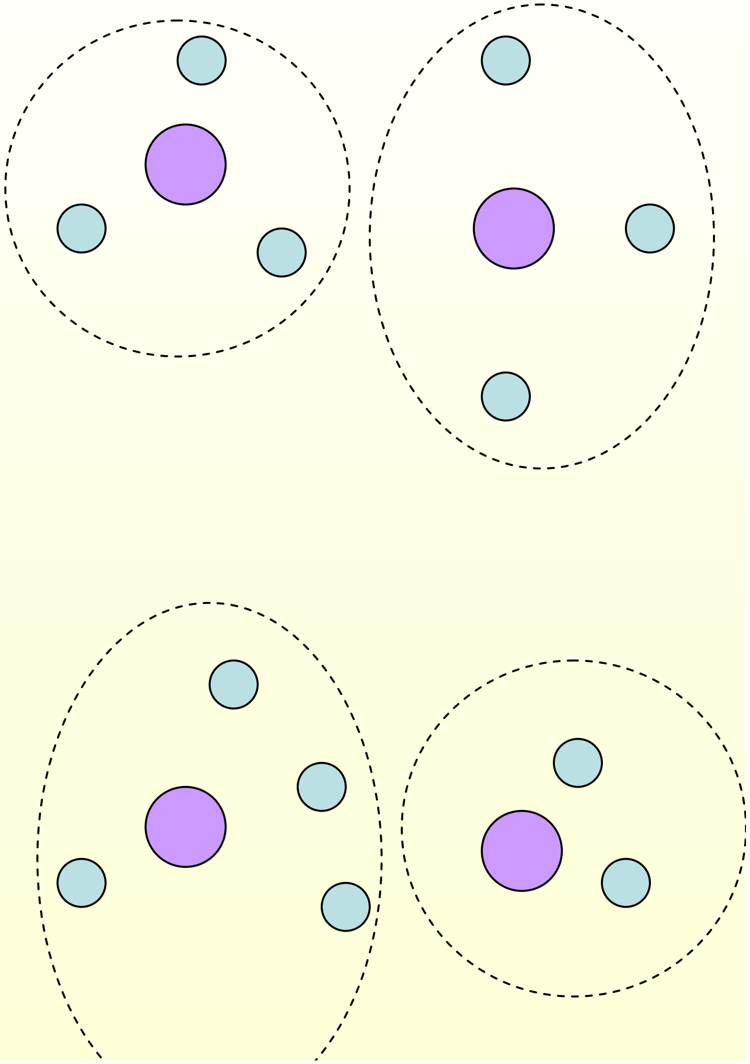
The COMPOW Protocol
[Narayanaswamy, *et. al*, ’03]

- In practice, we use greedy methods
- The COMPOW (COMmon POWer) protocol computes routing tables for each node at different power levels
- A node selects the minimum transmitting power so that its routing table has paths to all the other nodes

Clustering Nodes

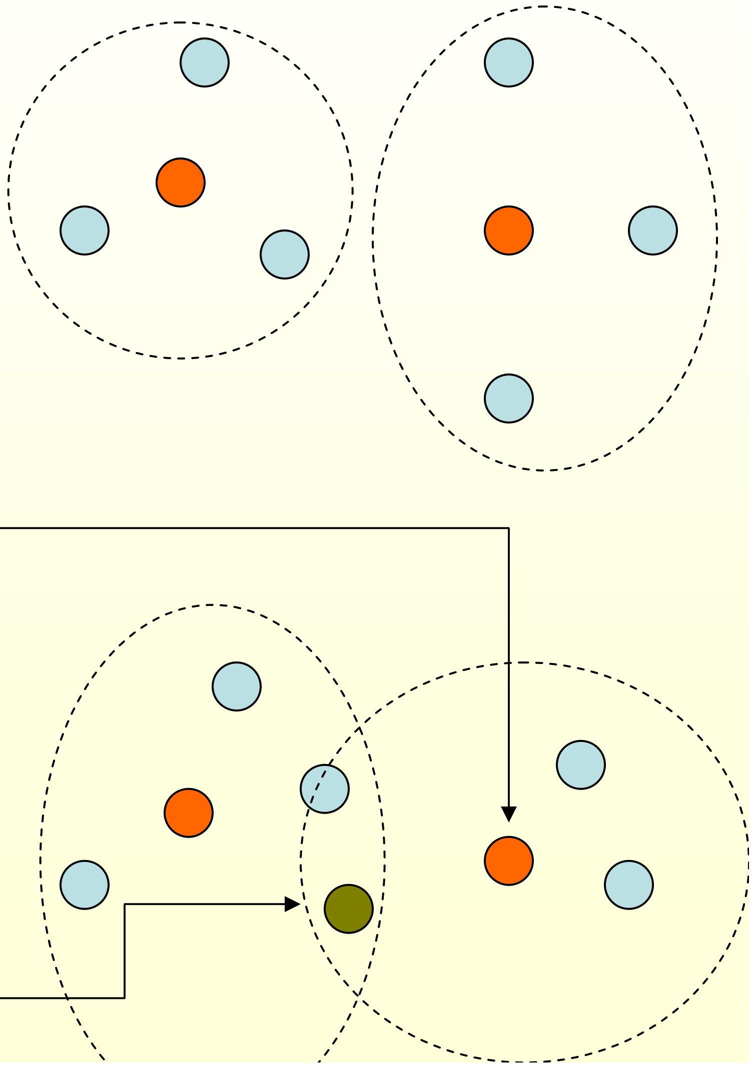
Clusters and Other Hierarchies

- Node clustering is extremely common in sensor networks
- It is natural in settings where nodes of different capabilities are available



Clustering is Useful Even in Homogeneous Networks

- Clusters are usually of size comparable with the node communication range
- Clusters allow better resource utilization
- Each cluster elects a node as its clusterhead
- Nodes belonging to multiple clusters can function as gateways



Clusterhead Election

- Assume each node has a unique ID
- Each node nominates the highest ID node it can hear to become a clusterhead
- All nominated nodes become clusterheads -- and form a cluster with their nominators

A Two-Level Communication Network

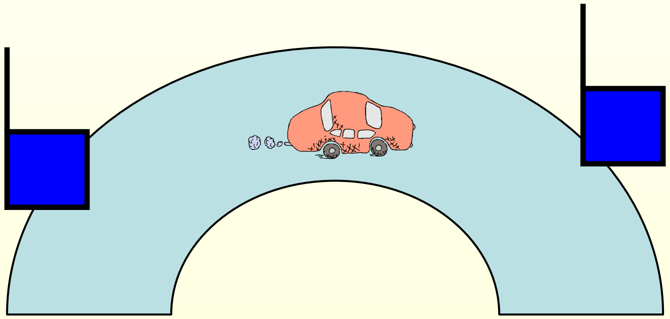
- Local traffic: within a cluster, directly or via the clusterhead
- Long-distance traffic: via clusteheads and gateways
- Clustering can even out node density in a network

Time Synchronization

Time Synchronization in Sensor Networks

Physical time needed to relate events in the physical world

- Time sync is critical at *many* layers
  - Beam-forming, localization, (sound) tracking
  - Data fusion, aggregation, caching
  - fine-grained radio scheduling
- High precision sometimes required
  - order of 1 microsecond (e.g., sleep cycles)
- Low precision sometimes sufficient
  - order of 10 milliseconds (e.g., temperature readings)



Clock Synch in Wired Networks

- Clock synchronization problem
  - bound differences between reading of two clocks
  - very well studied in computer networks

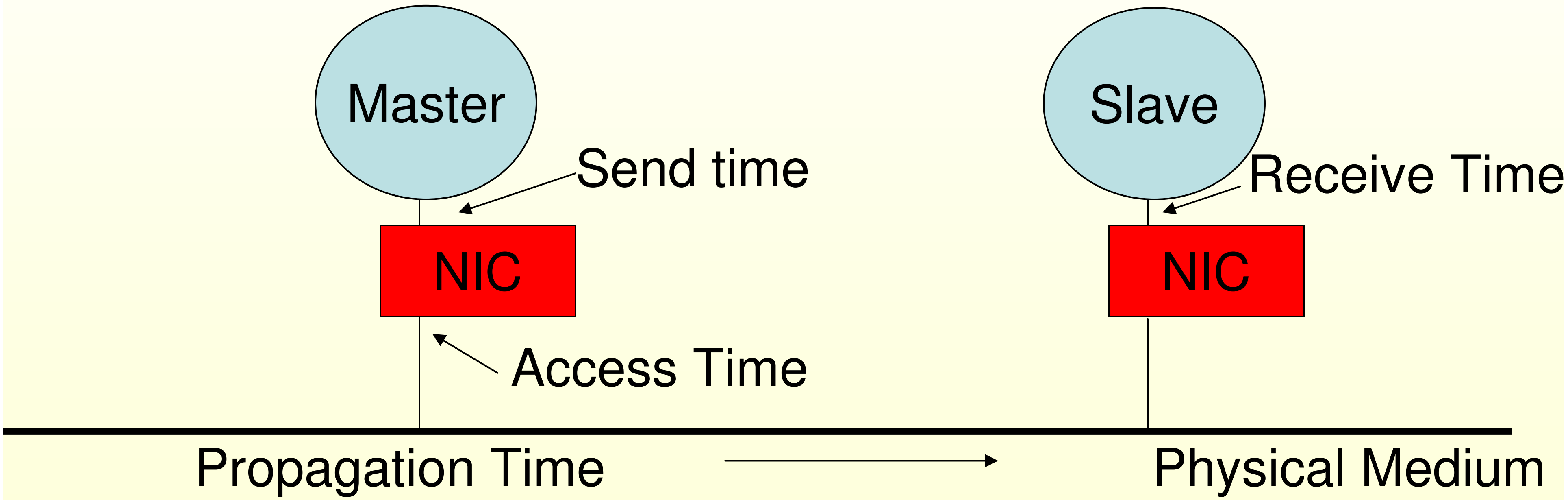
- NTP (Network Time Protocol)
  - Ubiquitous in the Internet
- 802.11 synchronization
  - Precise clock sync within a cluster
- GPS, WWVB, other radio time services
  - High precision anywhere
- High-stability oscillators (Rubidium, Cesium)

Synchronization Challenges

- Time synchronization is well-studied in computer networks
- But in sensor networks we have
  - Fewer resources
    - energy, network bandwidth constraints
  - Less infrastructure available
    - no accurate master clocks
    - no stable connections with reliable delays
    - no master NTP server
  - Sensors may be located on hostile environments
    - no GPS signal
  - Cost and size factor
    - $50 GPS receiver or $500 oscillator on a $5 mote?
  - High precision sometimes required

Traditional Local Clock Sync

- Slave sends a message to master
- Master replies with current time
- Slave estimates delay, updates its local clock

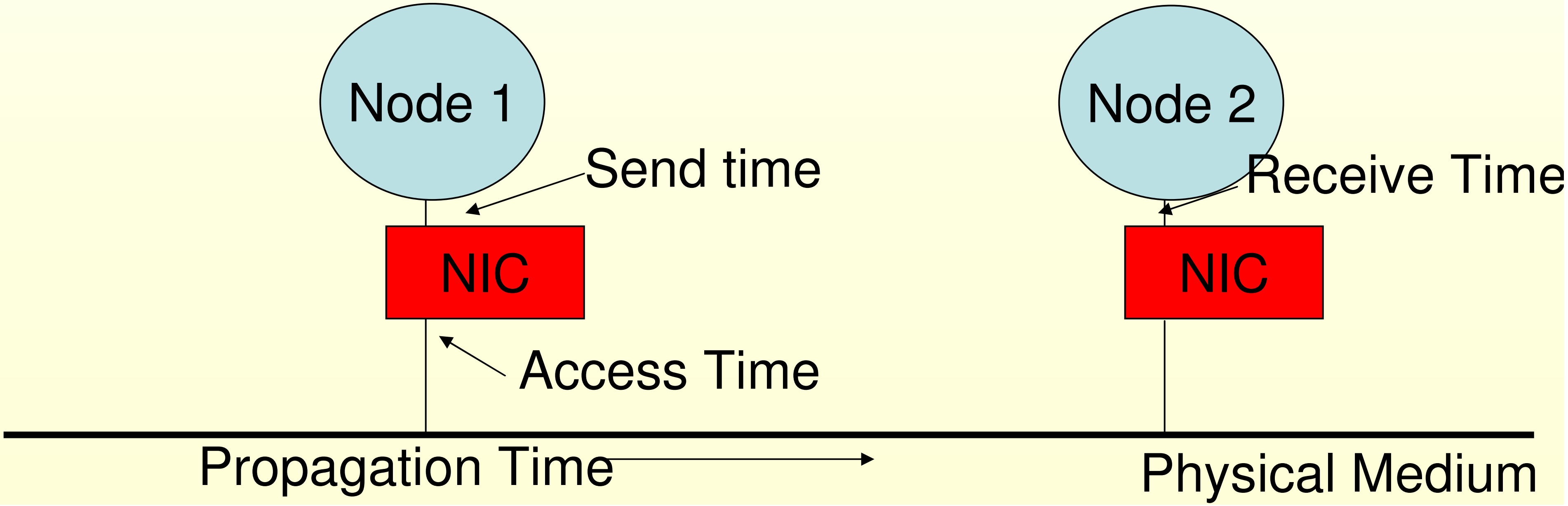


- **Problem:** many sources of unknown, nondeterministic latency between timestamp and its reception

Communication Delays

Communication delays comprise four parts:

- send time (preparing the packet)
- access time (getting medium access)
- propagation time (in the medium)
- receive time (receiving and decoding)



Clock Mappings

- It may be had to get all sensor node clocks to agree
- A less demanding requirement is to provide mappings between the clock readings of nodes that need to talk to each other

Clocks and Their Differences

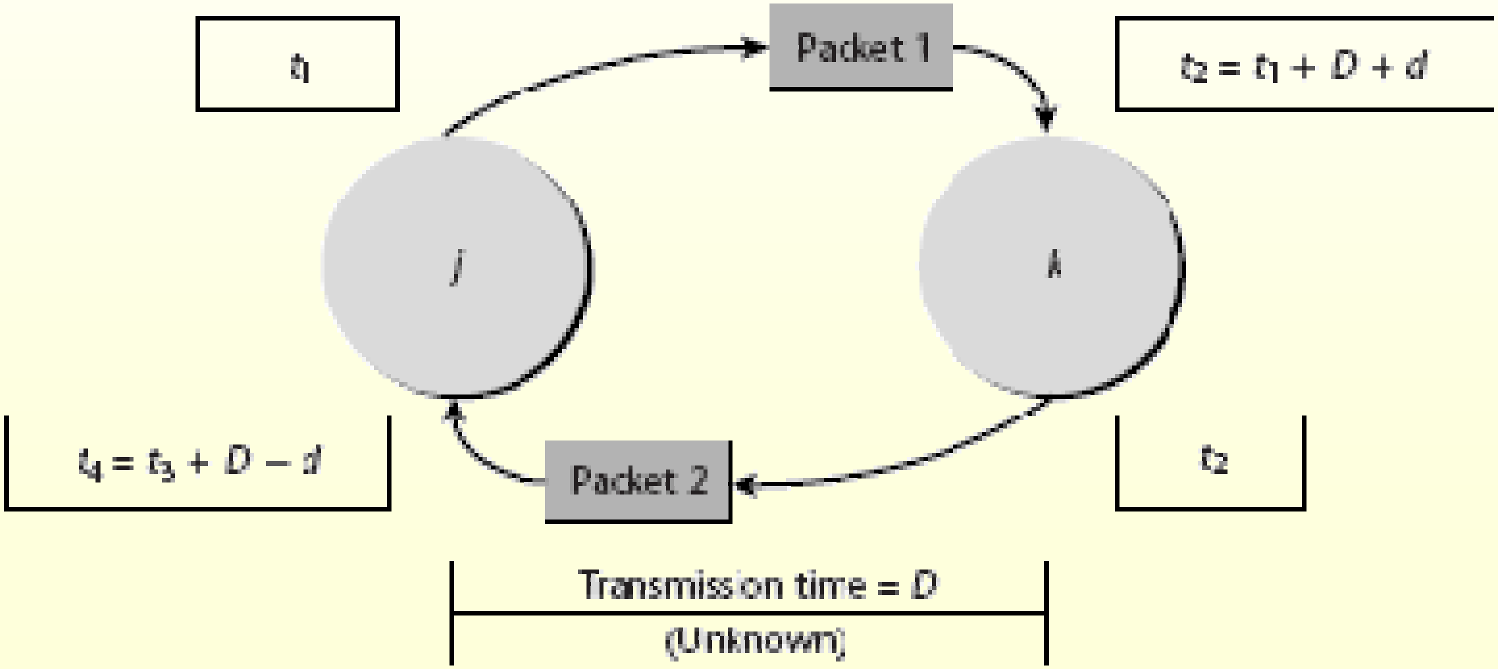
- Computer clocks are based on hardware oscillators
- The clock of different nodes may not agree because of
  - clock skew (or drift)

$$1-\rho \leq \frac{dC(t)}{dt} \leq 1+\rho$$

  - clock phase (or bias)

Symmetric Delay Estimation

In the absence of skew, the transmission delay *D* can be estimated as follows (*d* denotes the unknown phase difference)



Delay Estimation, II

- Node *j* can compute

$$d = (t_2 - t_1 - t_4 + t_3)/2$$

- -- and send that to node *k*
- Now node *k* can compute *D*

Interval Methods

- In temporal reasoning, often the ordering of events matters more than the exact times when the events occurred
- The goal is to map timestamps of events in one node to time intervals in other nodes, and thus perform temporal comparisons

Mapping Durations

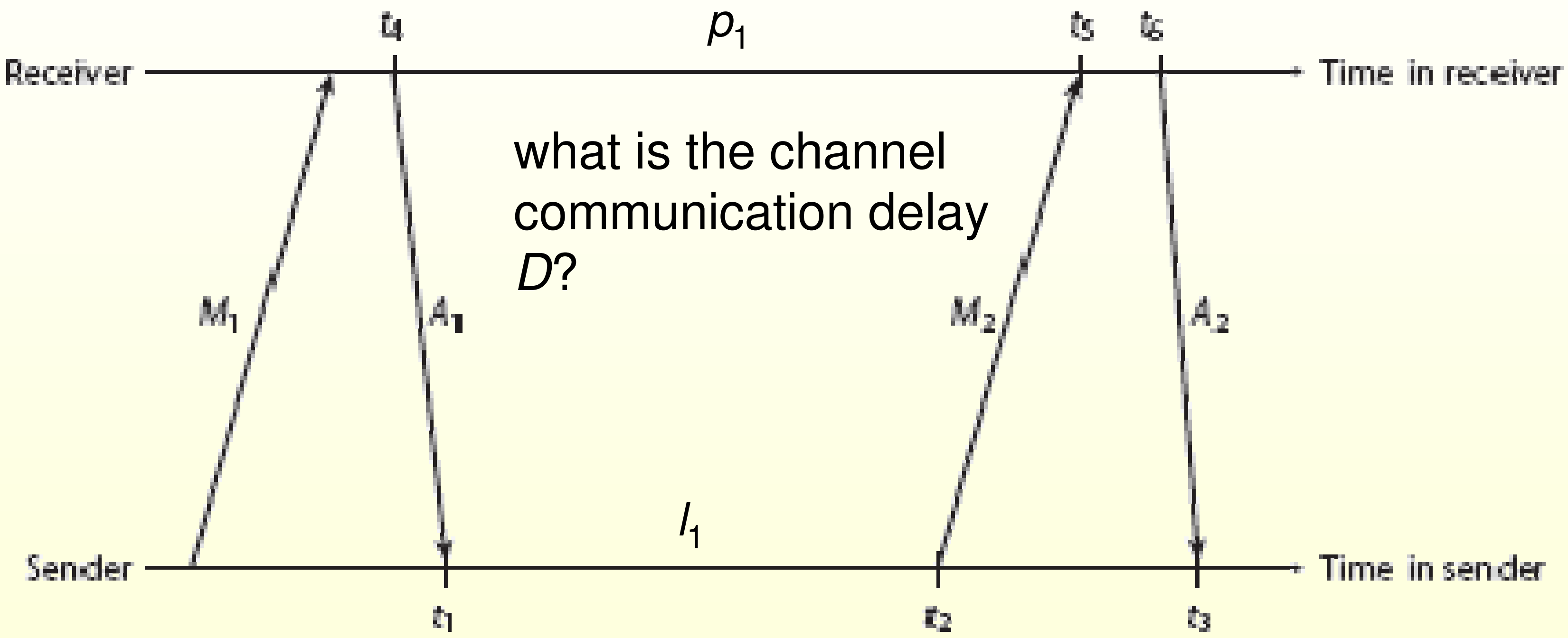
- In general we work we time intervals we call durations
- Node 1 with max. clock skew ρ_1 wishes to transform a local duration ΔC_1 into the time framework of node 2 with maximum clock skew ρ_2 . We must have:

$$1-\rho_i \leq \frac{\Delta C_i}{\Delta t} \leq 1+\rho_i \qquad \text{i = 1,2}$$

$$\Delta C_2 \subseteq \left[\Delta C_1 \frac{1-\rho_2}{1+\rho_1}, \Delta C_1 \frac{1+\rho_2}{1-\rho_1}\right]$$

Estimating Communication Delays

- Node 1 detects event E and time stamps with $r_1 = S_1(E)$



what is the channel communication delay D?

$$0 \leq D \leq p_1 - \ell_1 \frac{1-\rho_2}{1+\rho_1}$$

Propagation of Time Stamps

- Time notation:

r_i	s_i	l_i	p_i
receive	send	idle	round-trip

$$[r_1, r_1] = [S_1(E), S_1(E)]$$

node 1

$$\left[r_2 - (s_1 - r_1)\frac{1+\rho_2}{1-\rho_1} - (p_1 - \ell_1\frac{1-\rho_2}{1+\rho_1}),\ r_2 - (s_1 - r_1)\frac{1-\rho_2}{1+\rho_1}\right]$$

node 2

Propagation, Continued ...

$$\left[r_n - (1+\rho_n)\sum_{i=1}^{n-1}\frac{s_i - r_i + p_i}{1-\rho_i} - p_{i-1} + (1-\rho_n)\sum_{i=1}^{n-1}\frac{\ell_i}{1+\rho_i},\ r_n - ((1-\rho_n)\sum_{i=1}^{n-1}\frac{s_i - r_i}{1+\rho_i}\right]$$

node *i*

- intervals can get large fast, and then they become useless
- interval size increases with the number of hops
- interval size increases with holding times

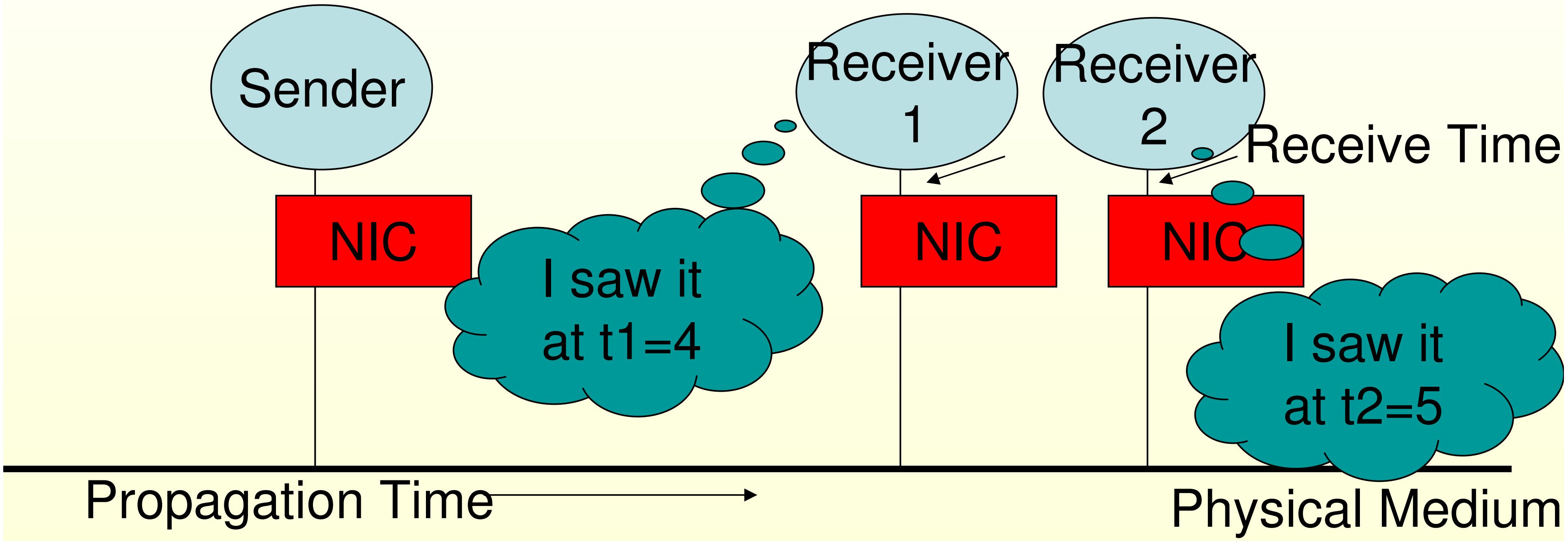
- many possible paths for node 1 to node *i*
- how do we choose the best?

Reference Broadcasts

- Sometimes we do need to look a real values, not just time comparisons (localization, for example)
- The reference broadcast system (RBS) exploits the broadcast nature of the wireless medium by synchronizing two receivers with each other, as opposed to a sender and a receiver

Reference Broadcasts
[Elson, Girod, Estrin ’02]

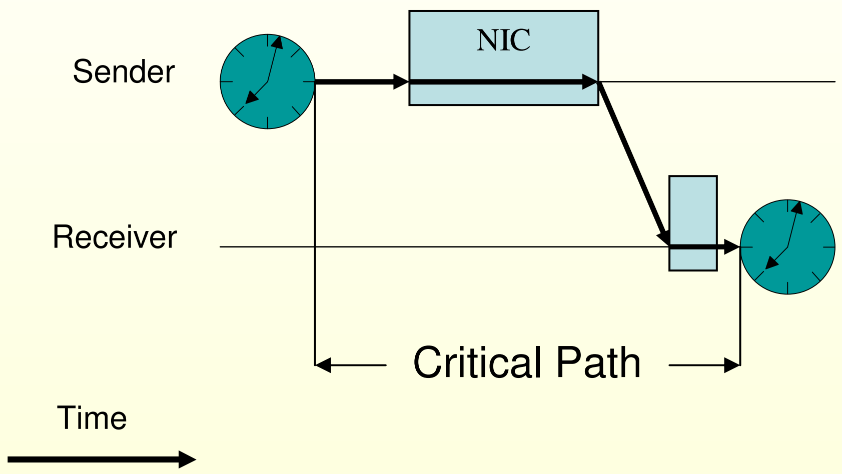
- Sender sends a broadcast reference packet
- Receivers record time of arrival
- *Receivers* exchange observations (and update clocks)



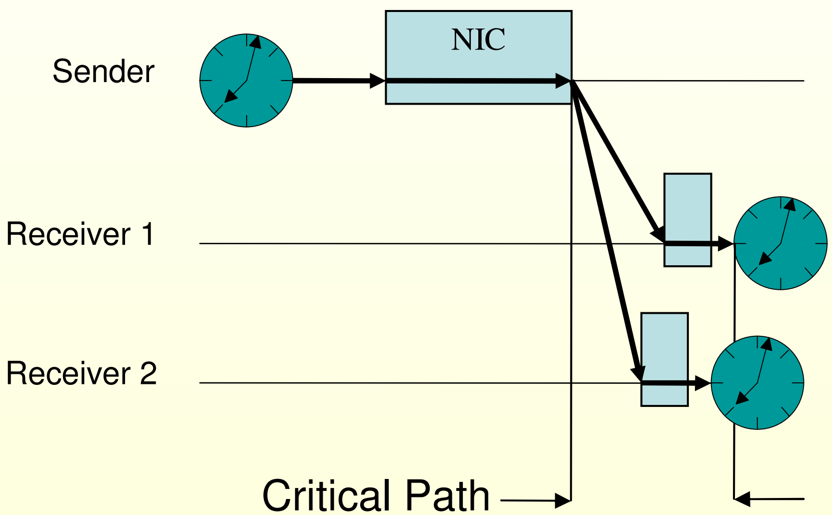
- Syncs two receivers *with each other*, NOT sender with receiver

Reference Broadcasts

- RBS reduces error by removing much uncertainty from critical path



Traditional critical path:
From the time the sender
reads its clock, to when the receiver
reads its clock



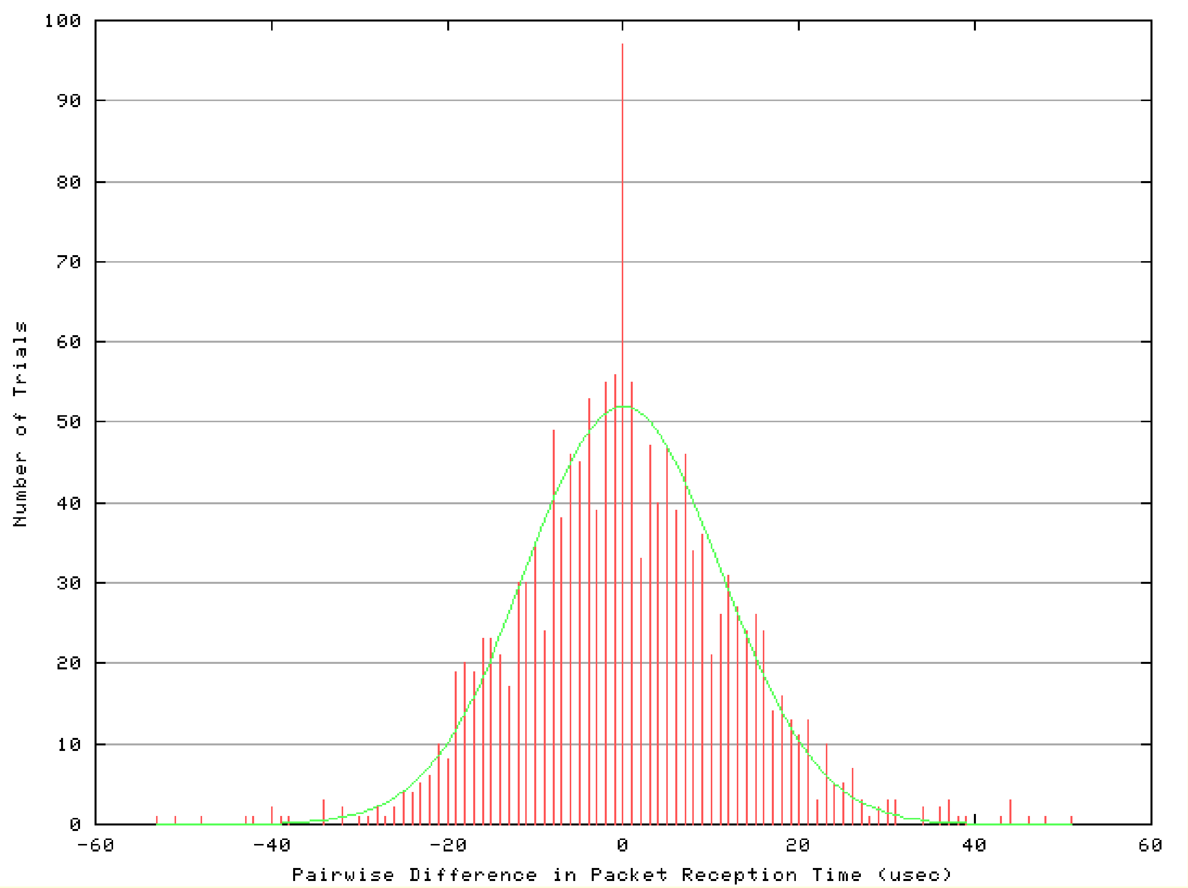
RBS: Only sensitive to the differences in receive time and propagation delay

Variations in Critical Path

- Differences in time-of-flight of packet
  - geographical distances
  - usually negligible
- Delays in recording time of packet arrival
  - read local system clock within NIC driver
  - quite deterministic
- Differences between recording time is small
  - order of transmission time of a single bit
  - can be accounted for

Experiments with Receive Time

- Obtain exact packet arrival time (using external global clock)
- Compute differences
- Bin using 1 microsecond
- 1 bit TX time is 52 microseconds
- Error can be modeled using Gaussian distribution



Removing Receive Time Differences

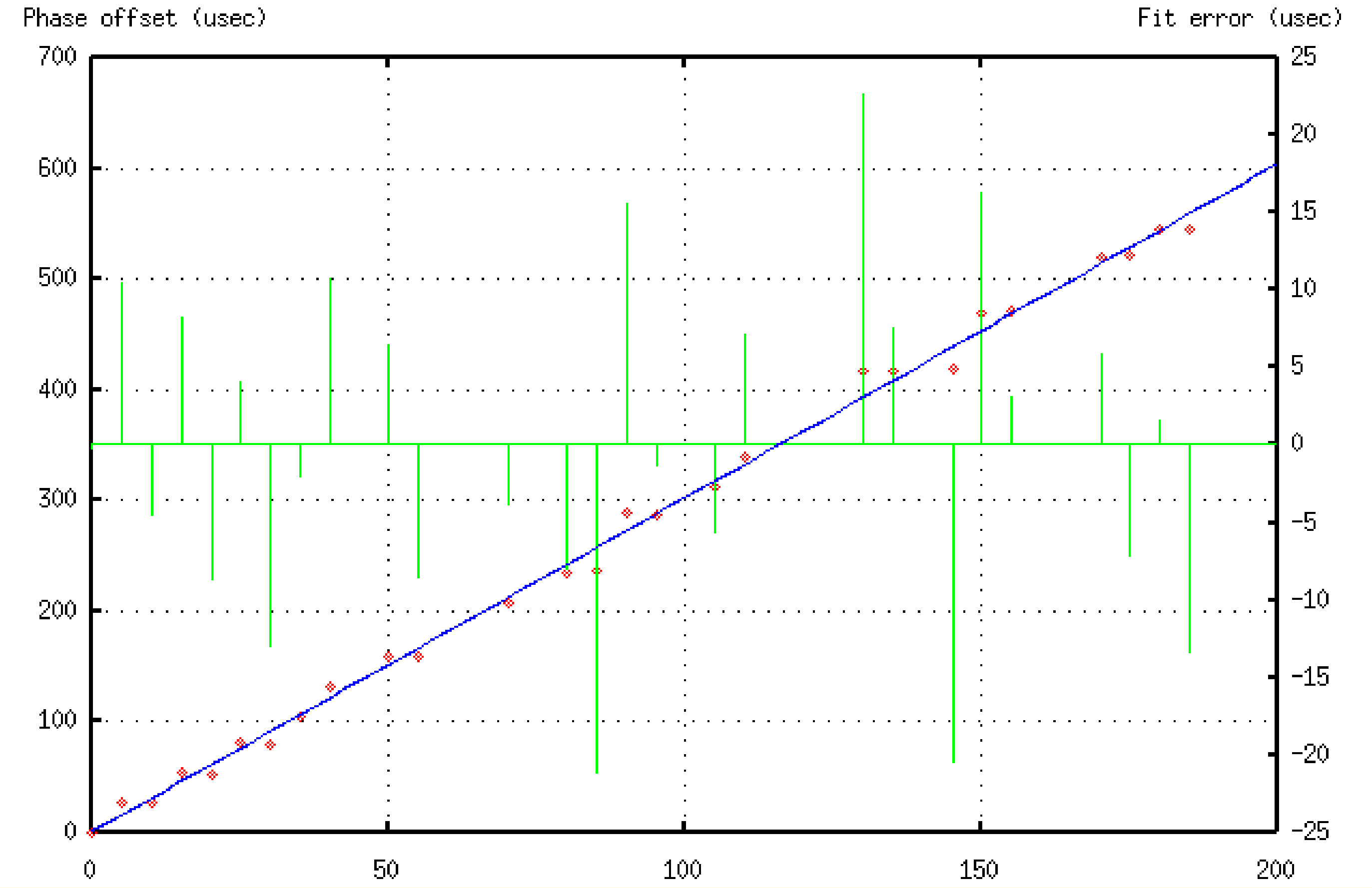
- Receive time differences are at most around transmission time of 1 bit (52 microseconds)
- Reduce this potential error by *averaging*
- Server broadcasts *m* reference packets
- Each of receiver records local time of each of thr *m* reference packets
- Receiver *i* and *j* exchange all *m* observations
- Compute offset[i,j] = $1/m \, \Sigma \, (T_{j,k} - T_{i,k})$

Clock Skew Problem

- It takes time to send multiple reference packets
- Clocks do not have identical heartbeats
  - differences in frequency make them drift
- After collecting *m* reference packets, clocks will have drifted
- Direct averaging the differences will not work
- Solution:
  - Fit data to a line to estimate clock skew and offset

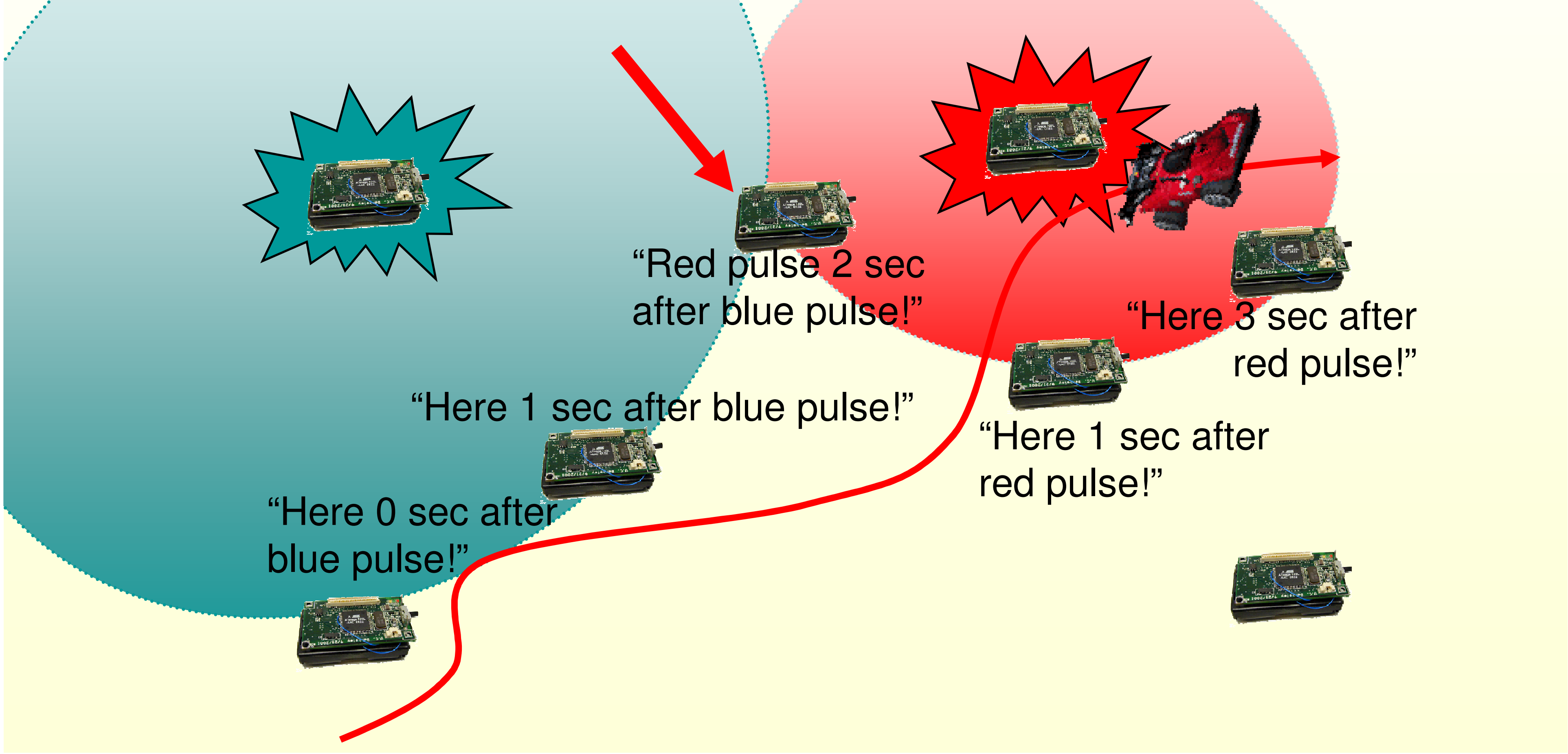
Measuring Clock Skew

- Each point is difference of arrival times of reference packet between nodes i and j
- Clock skew is the slope, y intercept is the offset



Multi-Hop RBS

- Some nodes broadcast RF synchronization pulses
- Receivers in a neighborhood are synced by using the pulse as a time reference. (The pulse *senders* are *not* synced.)
- Nodes that hear both can relate the time bases to each other



"Red pulse 2 sec after blue pulse!"

"Here 3 sec after red pulse!"

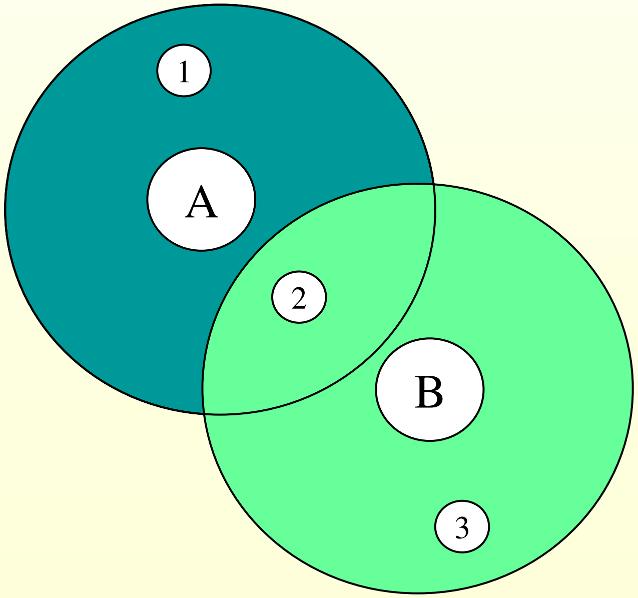
"Here 1 sec after blue pulse!"

"Here 1 sec after red pulse!"

"Here 0 sec after blue pulse!"

Multi-hop RBS

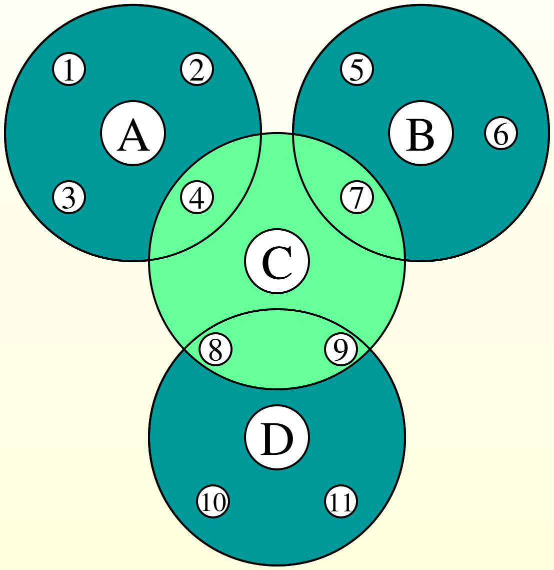
- Some nodes broadcast reference packets
- Receivers within transmission range are synced using RBS
- Nodes that hear both reference packets can relate to both time bases

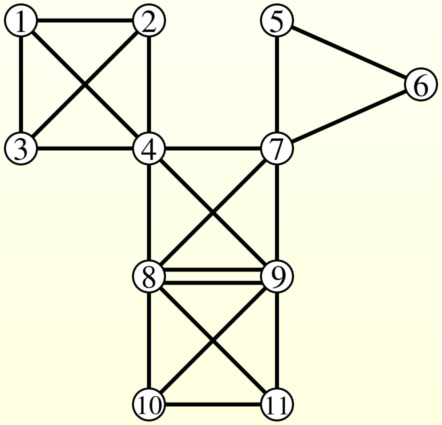


- Event e1 occurred in node 1 at local time t1
  - convert t1 to corresponding time in local clock of node 2 (t2)
  - convert t2 to corresponding time in local clock of node 3 (t3)

Multi-hop RBS

- Physical topology easily converts into logical topology
  - links represent possible clock conversions

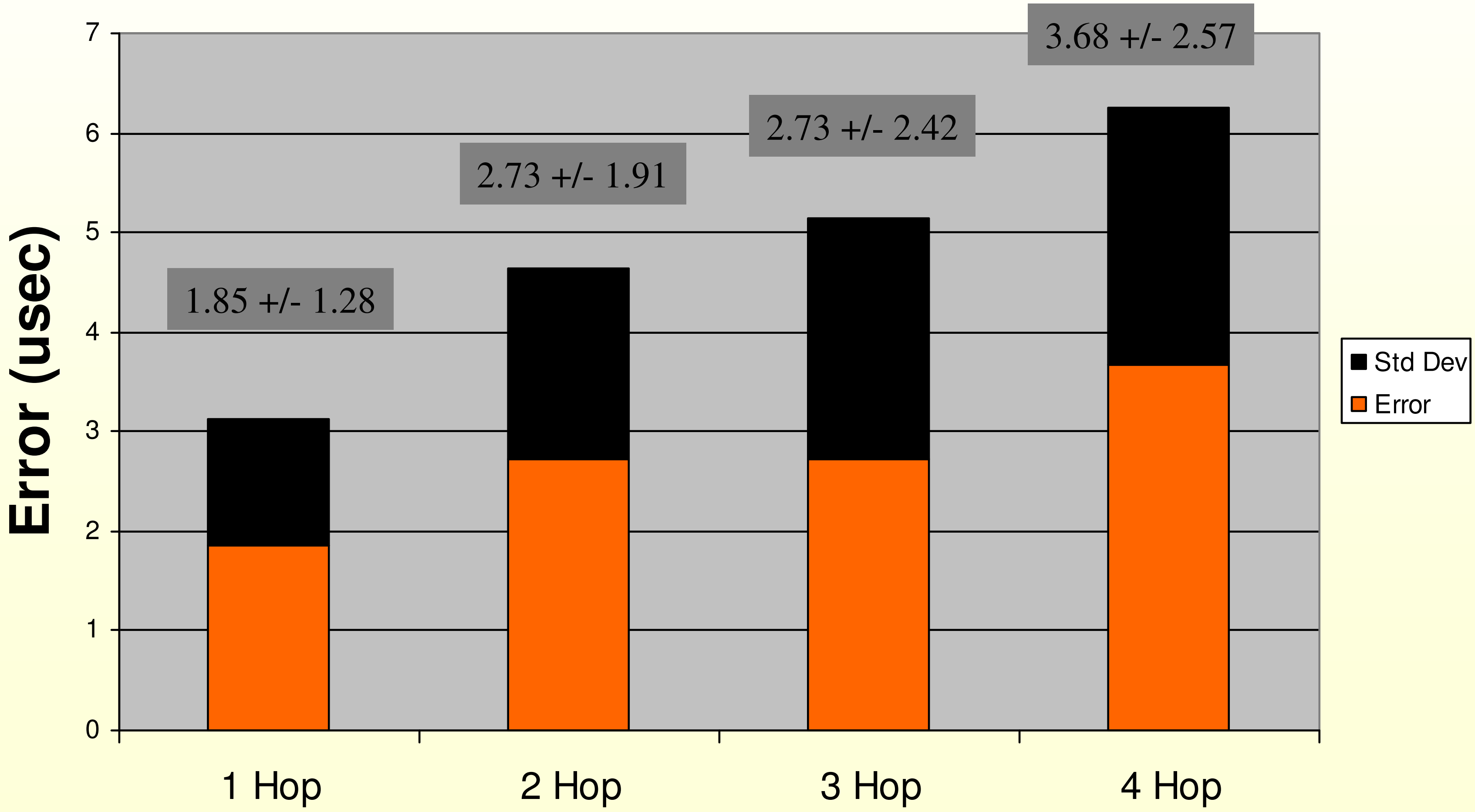




- Use *shortest path* search to find a “time route”
- Edges can be weighted by error estimates

Multi-Hop RBS

Error (and std dev) over multiple hops, in μsec



Optimal and Global Clock Sync

- Line fitting in RBS provides an estimate for
  - skew
  - offset
- But clock synchronization is between nodes pairwise

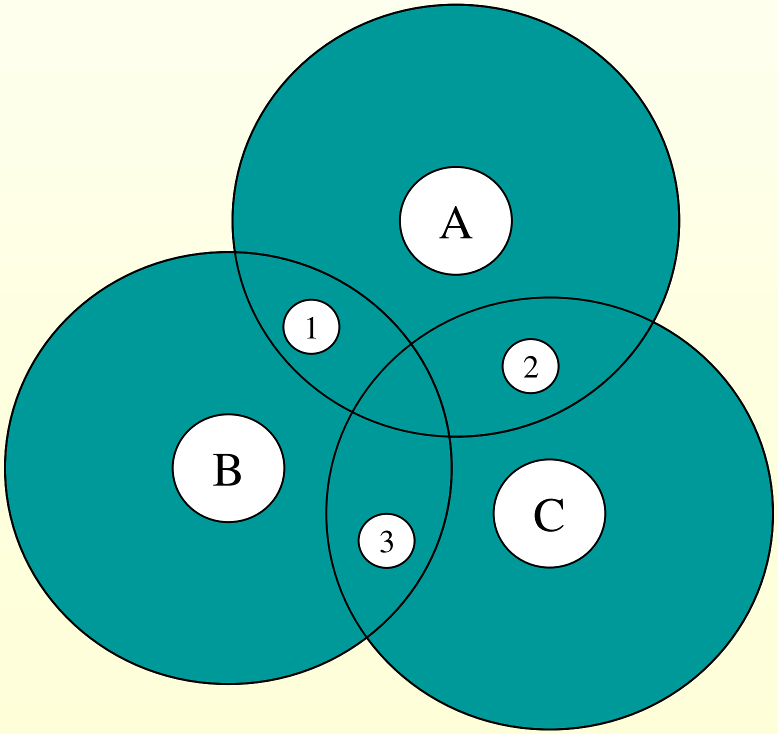
Two problems:

- Synchronization is not globally consistent
- Synchronization is not optimally precise

Global Consistency

- Event e1 occurs at node 1 at local time t1
- Convert this time to node's 2 clock
  - directly via skew/offset relative to 2
  - indirectly via skew/offset relative to 3, then via skew/offset relative to 2

- These 2 times represented in node 2's clock may be different!
- In large networks, several conversion paths exist



Localization

Location Discovery (LD) Service

- Very fundamental component for many other services
  - Enables *ad hoc* node deployment
  - GPS does not work everywhere, nor is it economical
- Necessary for many network operations
  - Geographic routing and coverage problems
  - People and asset tracking
  - Need spatial reference when monitoring spatial phenomena
  - Smart systems – devices need to know where they are

It is Worth Understanding LD

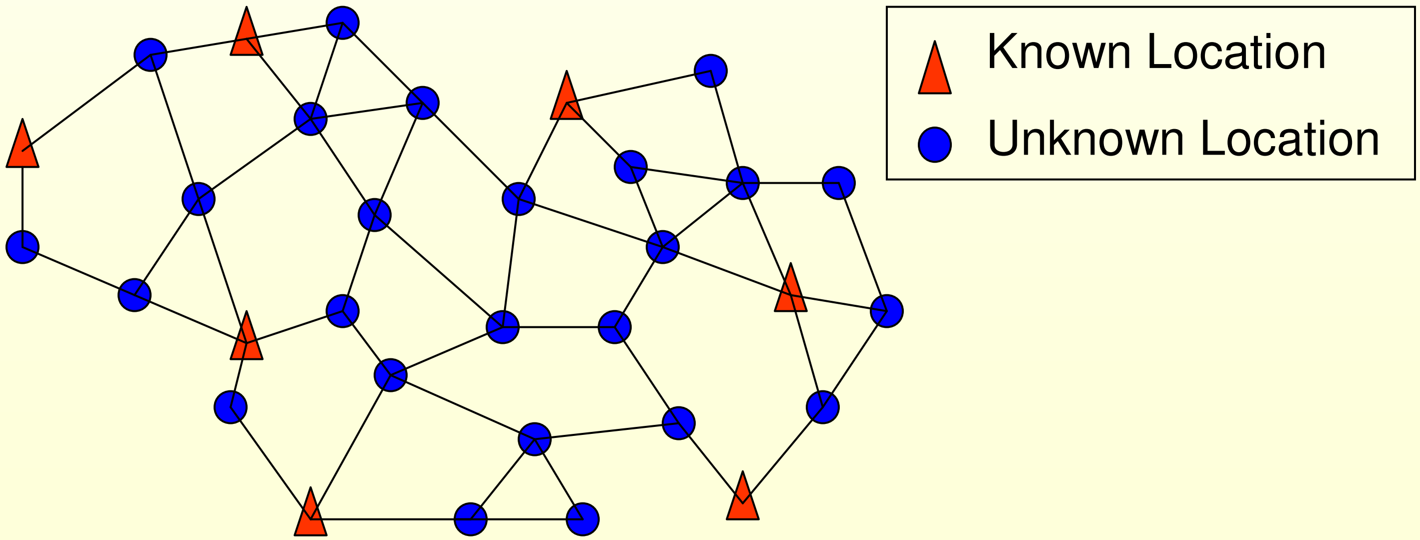
- LD captures multiple aspects of sensor networks:
  - **Physical layer imposes measurement challenges**
    - Multipath, shadowing, sensor imperfections, changes in propagation properties and more
  - **Extensive computation aspects**
    - Many formulations of localization problems -- how do you solve the corresponding optimization problem?
    - How do you solve the problem in a distributed manner?
      - You may have to solve the problem on a memory constrained processor…
  - **Networking and coordination issues**
    - Nodes have to collaborate and communicate to solve the problem
    - If you are using locations for routing, these are not yet available! How do you do it?
  - **System Integration issues**
    - How do you build a whole system for localization?
    - How do you integrate location services with other applications?
    - Different implementation for each setup, sensor, integration issue

Ranging Techniques

- *Ranging* refers to measuring distances between nodes
  - *Received Signal Strength* (RSS) measurements
    - Can be used with RF, but have to deal with fading, shadowing, multipath, and other channel effects
    - Also possible with ultrasound
  - *Time of Arrival* (ToA), or *Time Difference of Arrival* (TDoA) measurements
    - medium propagation speed must be estimated
    - requires clock synchronization

LD from Ranging

- Assume that initially a small number of nodes know their positions (base stations, with GPS, etc.) and can act as landmarks. We call these nodes *beacons*.
- Other nodes will localize themselves my measuring their distances to these, and then can become beacons themselves, and so on ...

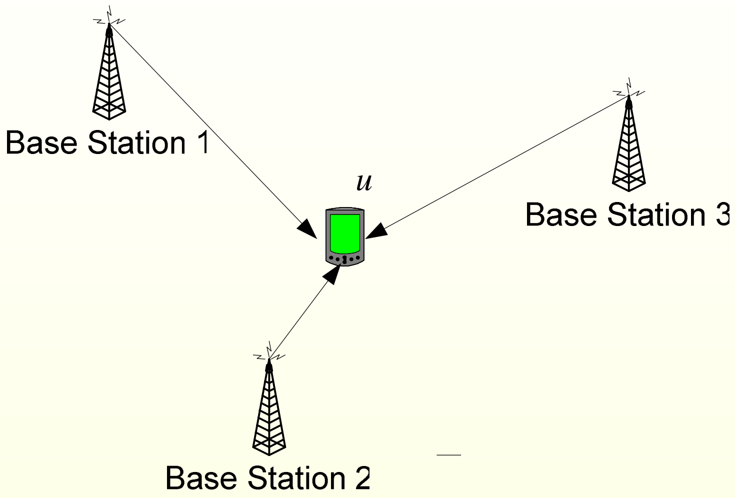


Two Phase Protocols

- Location discovery approaches consist of two phases : Ranging phase, Estimation phase
- Ranging phase (distance estimation)
  - Each node estimate its distance from its neighbors
- Location estimation phase (distance combining)
  - Nodes use ranging information and beacon node locations to estimate their positions

Using Distances
to
Immediate Neighbors

Atomic Multilateration



- Base stations advertise their coordinates & transmit a reference signal
- Node *u* uses these reference signals to estimate distances to each of the base stations
- <u>Note:</u> Distance measurements can be noisy!

Problem Formulation

- Need to minimize the sum of squares of the distance residuals for node *u*

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

measured distance to node *i*

- The objective error function to be minimized is

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

- This a non-linear optimization problem
  - Many ways to solve it (e.g. force formulation, gradient descent methods, etc.)

System Linearization

- We saw exactly the same equations in the localization of a source using acoustic distance measurements
- That solution was obtained by subtracting equations pairwise, to remove quadratic terms in the unknown location. Then least squares was used to solve the over-constrained system

Solution for an Embedded Processor

- Linearize the measurement equations using Taylor expansions

$$\Delta f_{u,i}^{2} \approx \Delta x_i \delta_x + \Delta y_i \delta_y + O(\Delta^2)$$

where

$$\Delta x_i = \frac{x_i - \hat{x}_u}{r_i}, \quad \Delta y_i = \frac{y_i - \hat{y}_u}{r_i}$$

$$r_i = \sqrt{(x_i - \hat{x}_u)^2 + (y_i - \hat{y}_u)^2}$$

Now this is in linear form $A\delta = z$

Incremental Least Squares Estimation

The linearized equations in matrix form become

$$\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}, \quad A = \begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \Delta x_3 & \Delta y_3 \end{bmatrix}, \quad z = \begin{bmatrix} f_1^{(u)} \\ f_2^{(u)} \\ f_3^{(u)} \end{bmatrix}$$

Now we can use the least squares equation to compute a *correction* to our initial estimate

$$\delta = (A^T A)^{-1} A^T z$$

Update the current position estimate

$$\hat{x}_u = \hat{x}_u + \delta_x \text{ and } \hat{y}_u = \hat{y}_u + \delta_y$$

Repeat the same process until δ comes very close to 0

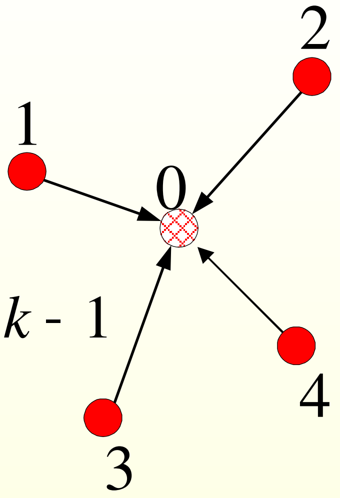
Some Issues

- Check several conditions
  - Landmark nodes must not be collinear
  - Assumes measurement error follows a Gaussian distribution
- Create a system of equations
  - Exactly how would you solve this in an distributed embedded system?
  - In ToA, TDoA settings, how do you solve for the speed of the medium?

Estimate Also Medium Speed

Minimize over all

$$f(x_i, x_0, s) = st_{i0} - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$



$$i = 1,2 \cdots k - 1$$

This can be linearized to the form

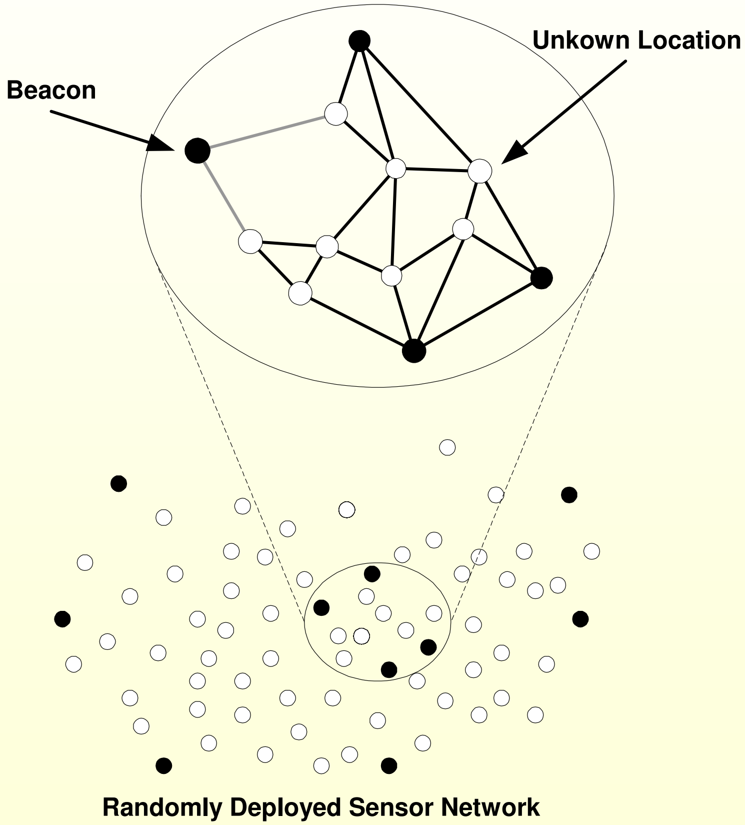
$$y = Xb$$

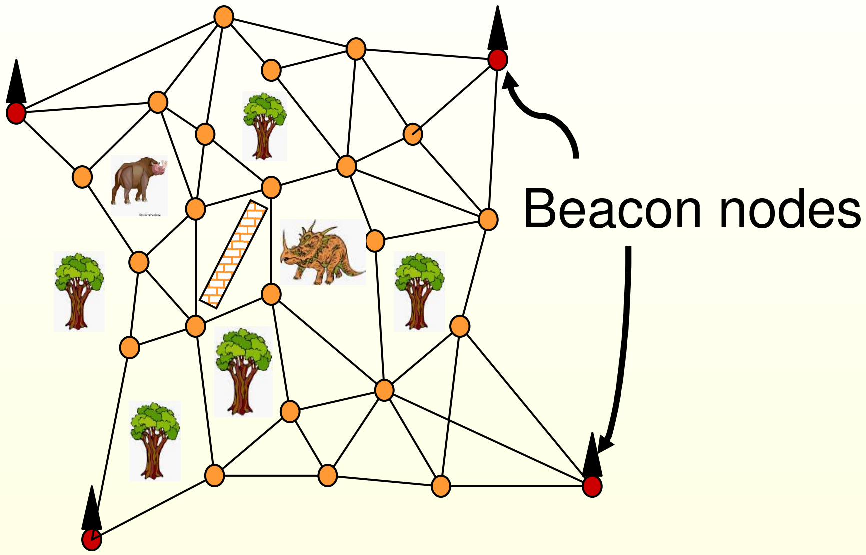
where

$$y = \begin{bmatrix} -x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ -x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \vdots \\ -x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix} \quad X = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) & t_{k0}^2 - t_{10}^2 \\ 2(x_k - x_2) & 2(y_k - y_2) & t_{k0}^2 - t_{20}^2 \\ \vdots & \vdots & \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) & t_{k0}^2 - t_{(k-1)0}^2 \end{bmatrix} \quad b = \begin{bmatrix} x_0 \\ y_0 \\ s^2 \end{bmatrix}$$

MMSE Solution: $b = (X^T X)^{-1} X^T y$

The Node Localization Problem

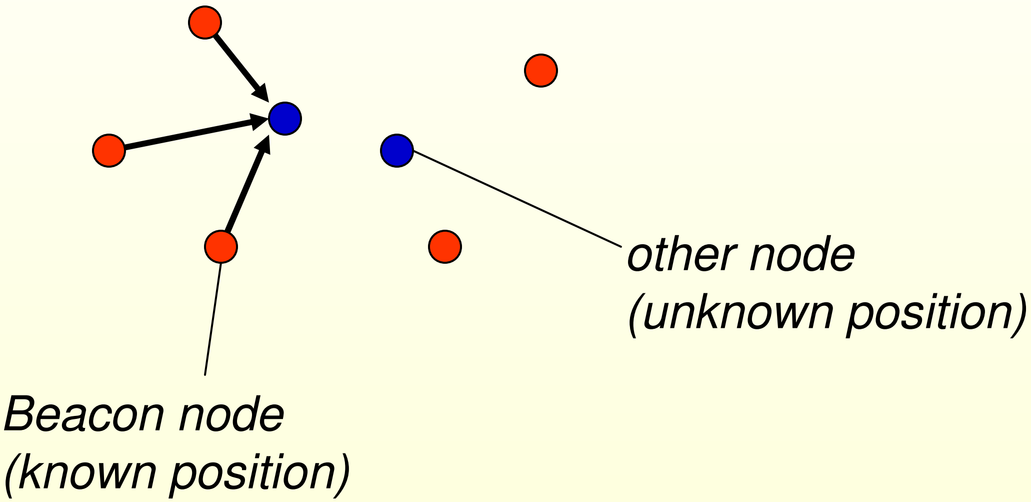




- Localize nodes in an ad-hoc *multihop* network
- Based on a set of inter-node distance measurements

Solving over multiple hops

- Iterative Multilateration

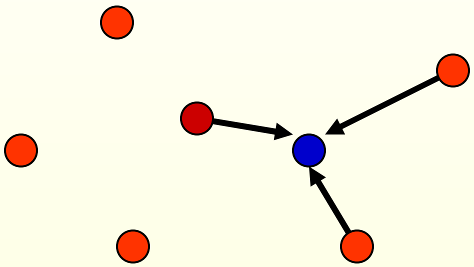


other node
(unknown position)

Beacon node
(known position)

Iterative Multilateration

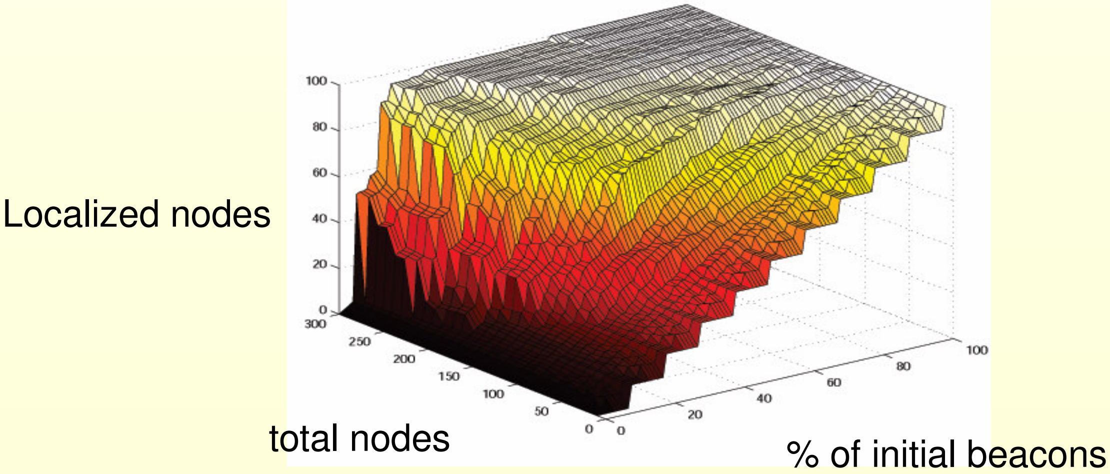
Iterative (Sequential) Multilateration



Problems

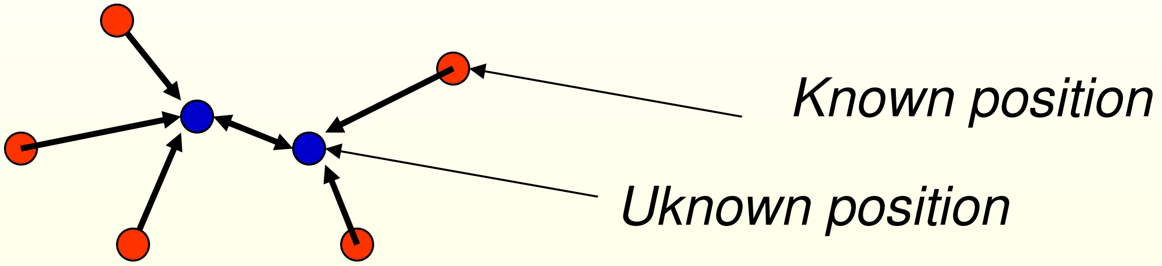
Error accumulation

May get stuck!!!



Collaborative Mutlilateration (Savvides *et. al*., ‘03*)*

- All available measurements are used as constraints



- Solve for the positions of multiple unknowns simultaneously
- Catch: This is a non-linear optimization problem!
- How do we handle this?

Problem Formulation

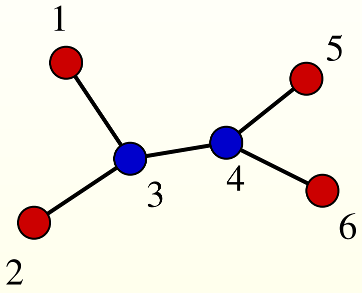
$$f_{2,3} = R_{2,3} - \sqrt{(x_2 - \hat{x}_3)^2 + (y_2 - \hat{y}_3)^2}$$

$$f_{3,5} = R_{3,5} - \sqrt{(\hat{x}_3 - x_5)^2 + (\hat{y}_3 - y_5)^2}$$

$$f_{4,3} = R_{4,3} - \sqrt{(\hat{x}_4 - \hat{x}_3)^2 + (\hat{y}_4 - \hat{y}_3)^2}$$

$$f_{4,5} = R_{4,5} - \sqrt{(\hat{x}_4 - x_5)^2 + (\hat{y}_4 - y_5)^2}$$

$$f_{4,1} = R_{4,1} - \sqrt{(\hat{x}_4 - x_1)^2 + (\hat{y}_4 - y_1)^2}$$



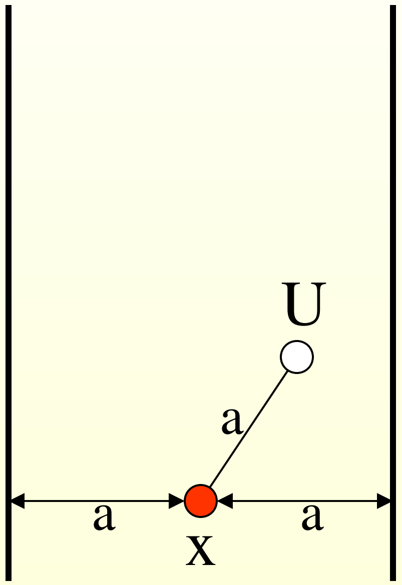
The objective function is

$$F(\hat{x}_3, \hat{y}_3, \hat{x}_4, \hat{y}_4) = \min \sum f_{i,j}^2$$

Need some decent initial estimates, then iterate using a Kalman Filter

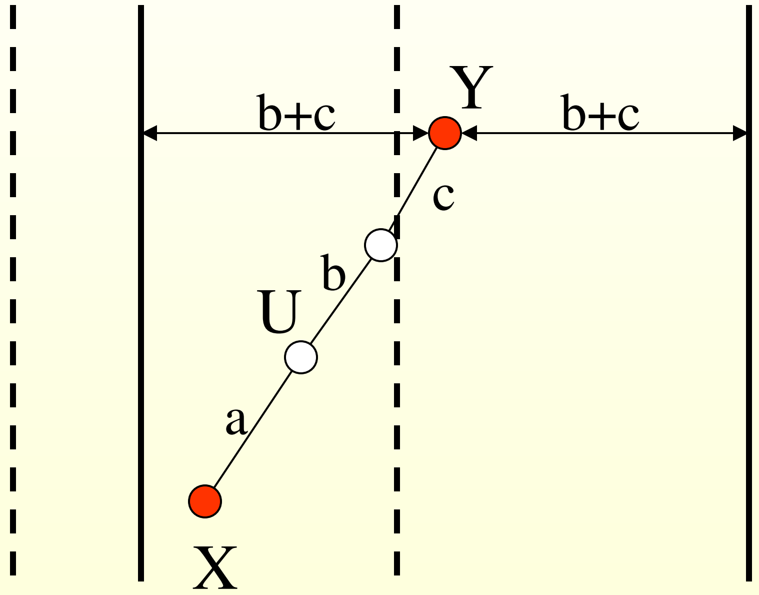
Initial Estimates

- Use the accurate distance measurements to impose constraints in the x and y coordinates – bounding box
- **Use the distance to a beacon as bounds on the x and y coordinates**



Initial Estimates, Con’t

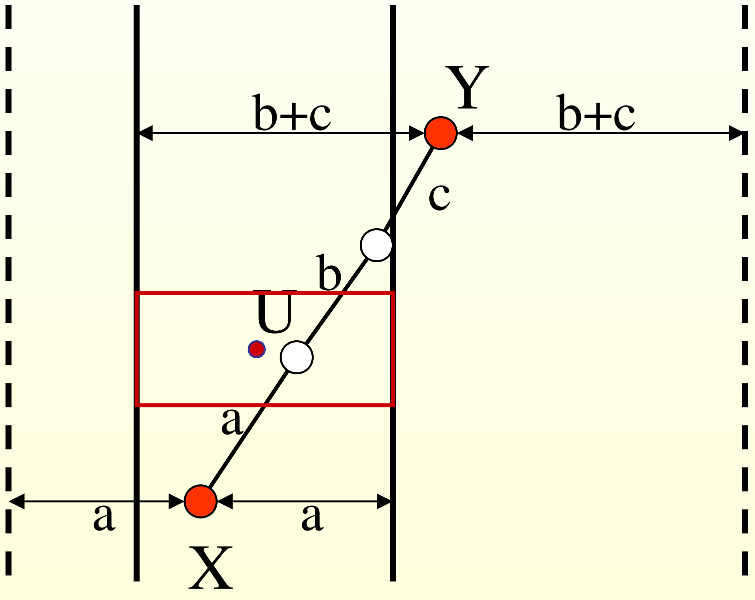
- Use the accurate distance measurements to impose constraints in the x and y coordinates – bounding box
- Use the distance to a beacon as bounds on the x and y coordinates
- **Do the same for beacons that are multiple hops away**
- **Select the most constraining bounds**



U is between [Y-(b+c)] and [X+a]

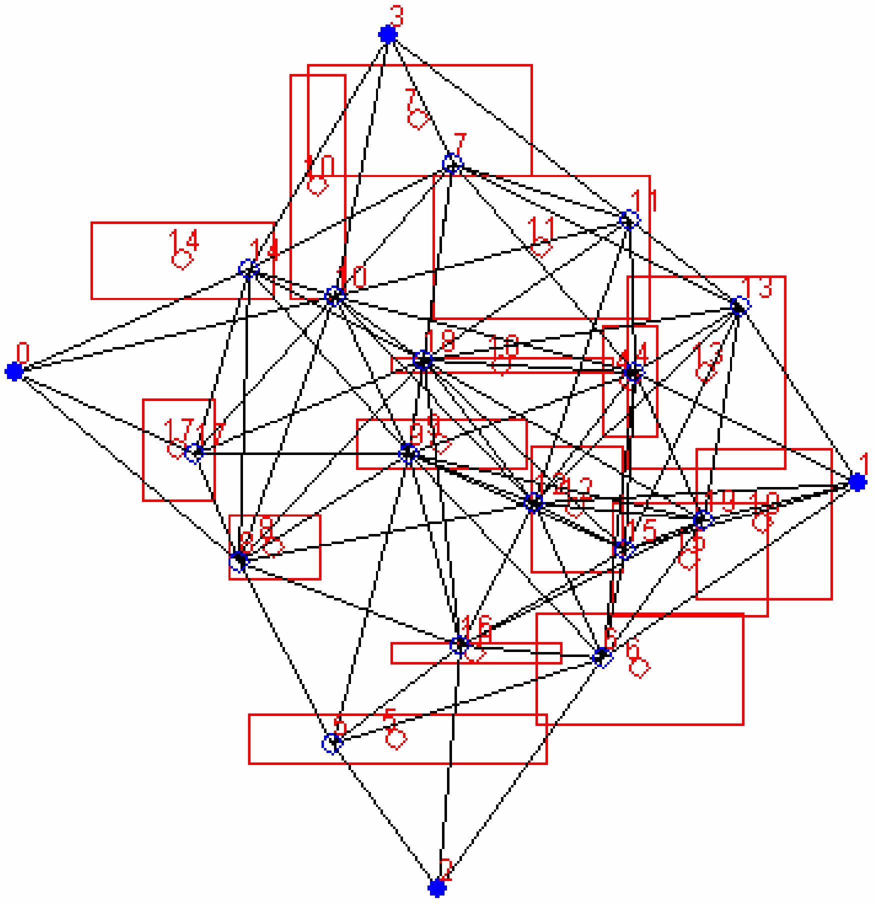
Initial Estimates, Cont'd

- Use the accurate distance measurements to impose constraints in the x and y coordinates – bounding box
- Use the distance to a beacon as bounds on the x and y coordinates
- Do the same for beacons that are multiple hops away
- Select the most constraining bounds
- **Set the center of the bounding box as the initial estimate**



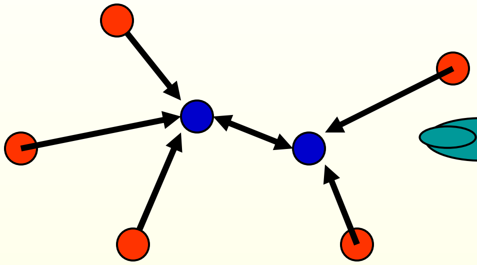
Initial Estimates, Cont'd

- Example:
  - 4 beacons
  - 16 unknowns
- To get good initial estimates, beacons should be placed on the perimeter of the network
- **Observation:** If the unknown nodes are outside the beacon perimeter then initial estimates are on or very close to the convex hull of the beacons



Overview: Collaborative Multilateration

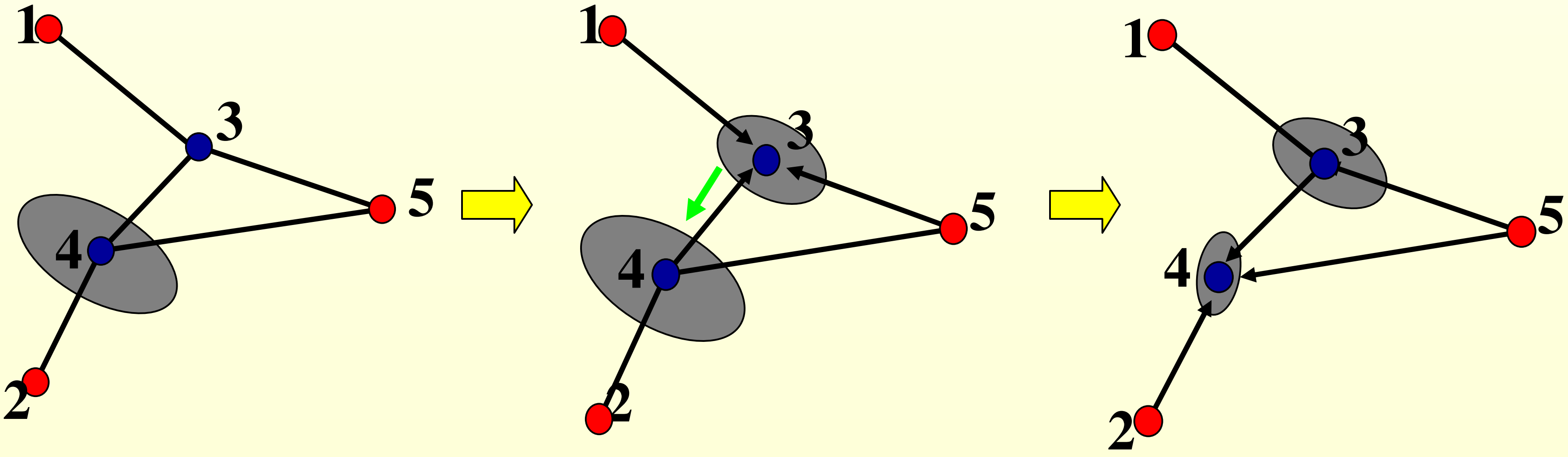
Collaborative Multilateration



Challenges

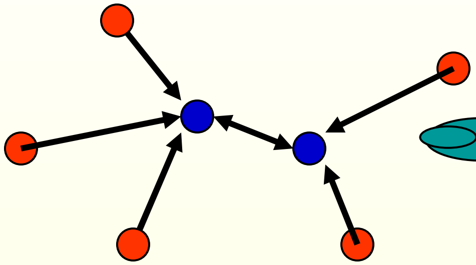
Computation constraints

Communication cost



Overview: Collaborative Multilateration

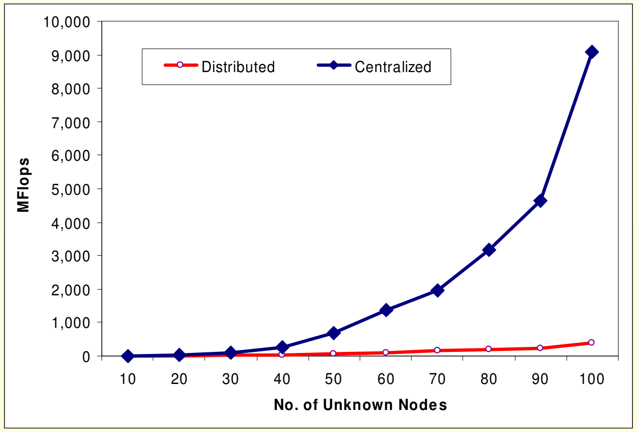
Collaborative Multilateration

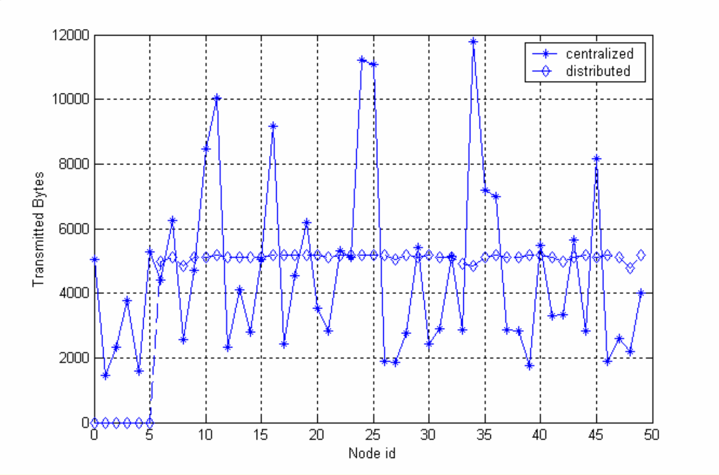


Challenges

Computation constraints

Communication cost

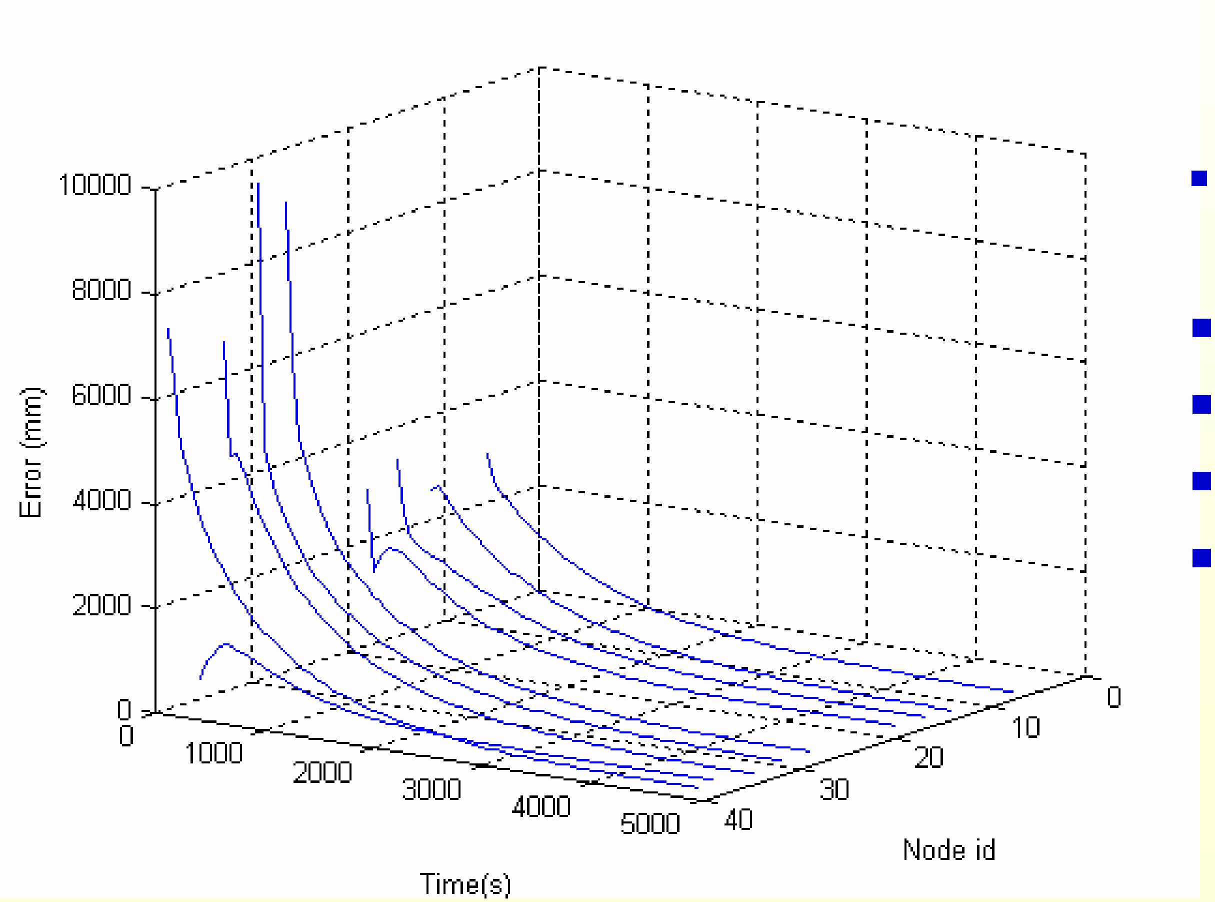




Distributed has reduced cost

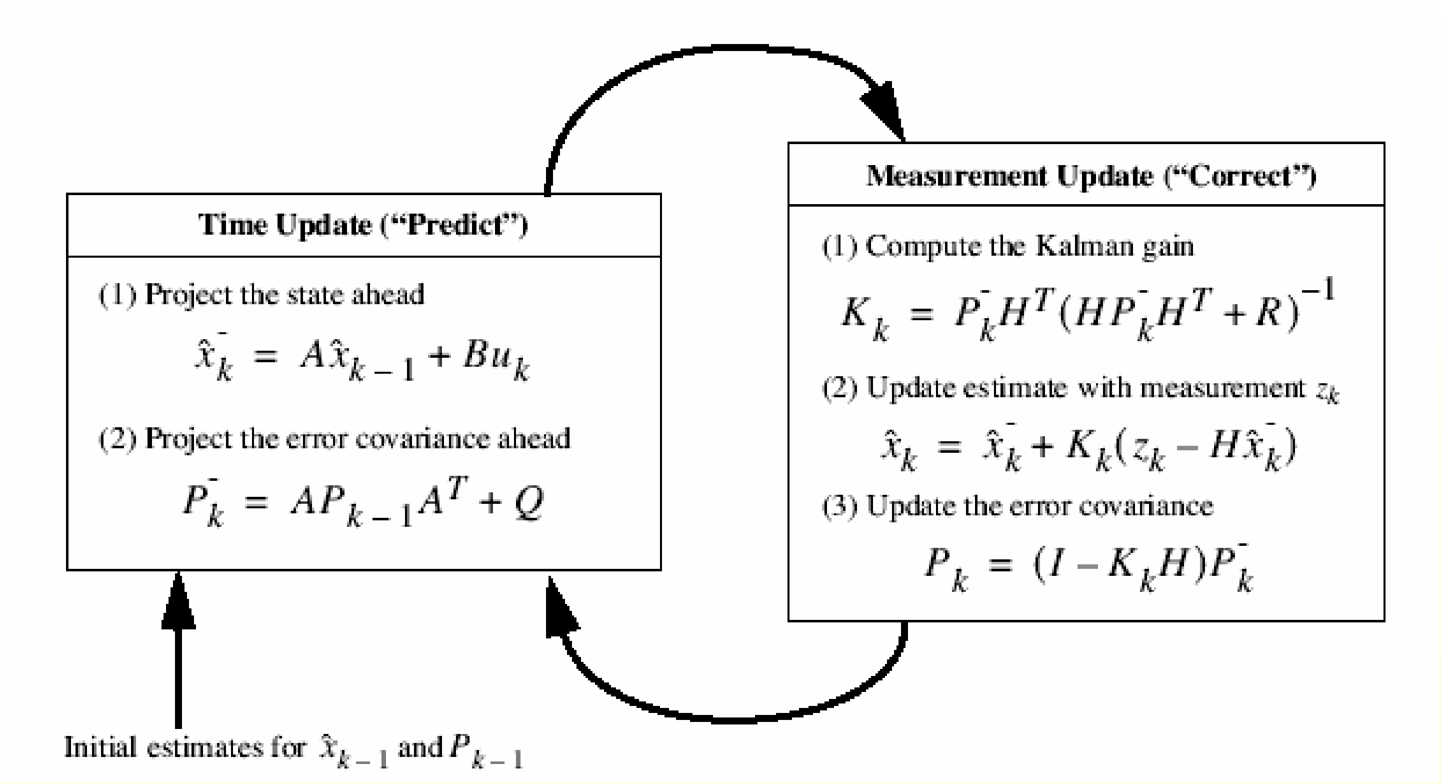
Even sharing of communication cost

Satisfy Global Constraints with Local Computation



- From SensorSim simulation
- 40 nodes, 4 beacons
- IEEE 802.11 MAC
- 10Kbps radio
- Average 6 neighbors per node

Kalman Filter



Time Update (“Predict”)

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update (“Correct”)

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

(3) Update the error covariance

$$P_k = (I - K_k H)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

- We only use measurement update since the nodes are static
- We know *R* (ranging noise distribution)
- Artificial notion of time: sequentially introduce distance constraints

Global Kalman Filter

$$\hat{z}_k^{\ T} = \begin{bmatrix} \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ \sqrt{(ex_3 - ex_4)^2 + (ey_3 - ey_4)^2} \\ \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \\ \sqrt{(ex_4 - x_5)^2 + (ey_4 - y_5)^2} \end{bmatrix}$$

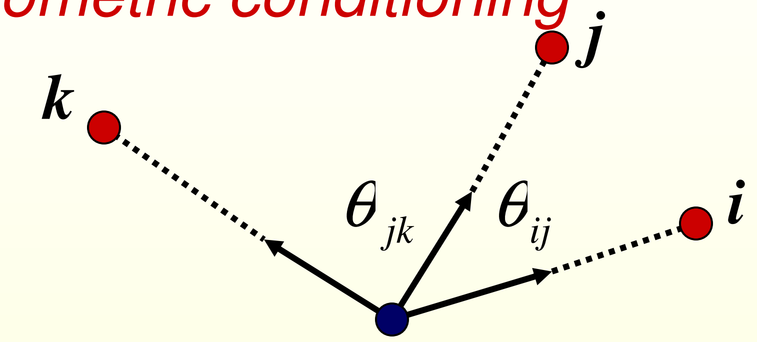
$$H = \left. \underbrace{\begin{bmatrix} 0 & 0 & \frac{x_2 - ex_3}{\hat{z}_k(1)} & \frac{y_2 - ey_3}{\hat{z}_k(1)} \\ \frac{ex_3 - x_5}{\hat{z}_k(2)} & \frac{ey_3 - y_5}{\hat{z}_k(2)} & 0 & 0 \\ \frac{ex_3 - ex_4}{\hat{z}_k(3)} & \frac{ey_3 - ey_4}{\hat{z}_k(3)} & \frac{ex_4 - ex_3}{\hat{z}_k(3)} & \frac{ey_4 - ey_3}{\hat{z}_k(3)} \\ \frac{ex_4 - x}{\hat{z}_k(4)} & \frac{ey_4 - y}{\hat{z}_k(4)} & 0 & 0 \\ \frac{ex_4 - x_5}{\hat{z}_k(5)} & \frac{ey_4 - y_5}{\hat{z}_k(5)} & 0 & 0 \end{bmatrix}}_{\text{\# of nodes to be located x 2}} \right\} \text{\# of edges}$$

- Matrices grow with density and number of nodes → so does computation cost
- Computation is not feasible on small processors with limited computation and memory

Beware of Geometry Effects!

Known as Geometric Dilution of Precision(GDOP)

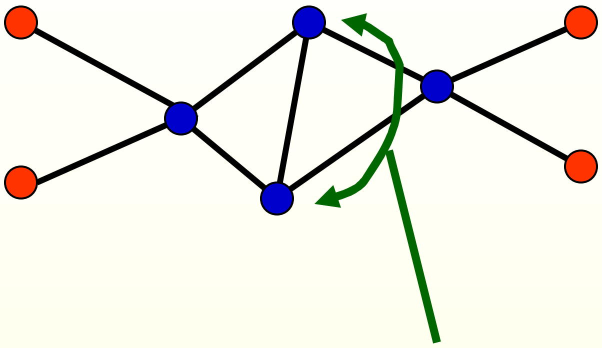
Position accuracy depends on *measurement accuracy* and *geometric conditioning*



From pseudoinverse equation $\hat{x} = (A^T Q_b^{-1} A)^{-1} A^T Q_b^{-1} b$

$$\boxed{GDOP = GDOP(N, \theta) = \sqrt{\frac{N}{\sum_i \sum_{j,\, j>i} |\sin \theta_{ij}|^2}}}$$

Beware of Uniqueness Requirements



Nodes can be exchanged without violating the measurement constraints

- In a 2D scenario a network is uniquely localizable if:
  1. It belongs to a subgraph that is redundantly rigid
  2. The subgraph is 3-connected
  3. It contains at least 3 beacons

Using Distances
to
Distant Neighbors

Three-phase approach

1. Determine distance to beacon nodes

   (communication)

2. Establish position estimates

   (computation)

3. Iteratively refine positions using additional range measurements

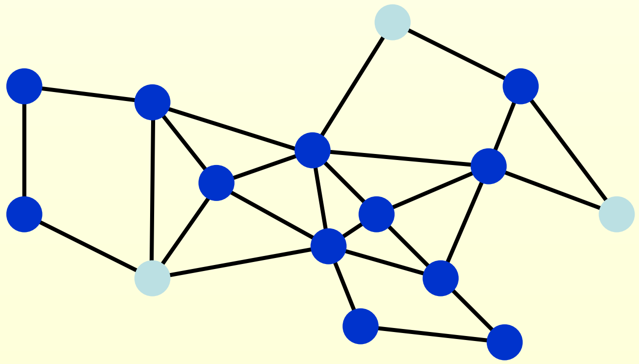
   (both)

Phase 1: Distance to Beacons

- Three algorithms

Sum-dist	[Savvides et al.]
DV-Hop	[Niculescu et al., Savarese et al.]
Euclidean	[Niculescu et al.]

- beacons flood network with their known positions



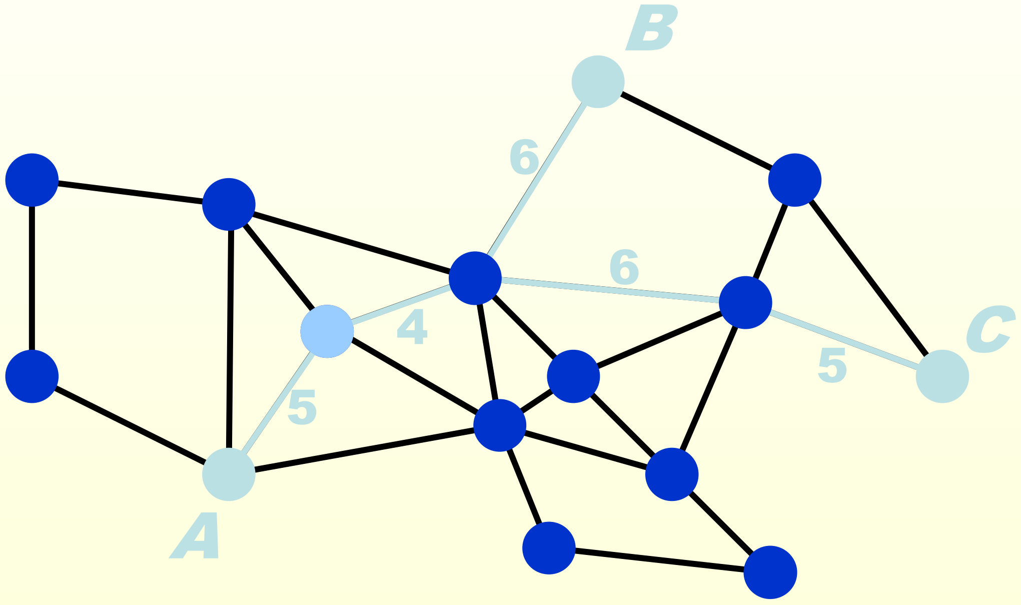
Phase 1:
Sum-dist

Anchors

- flood network with known position

Nodes

- add hop distances
- requires range measurement



A: 5

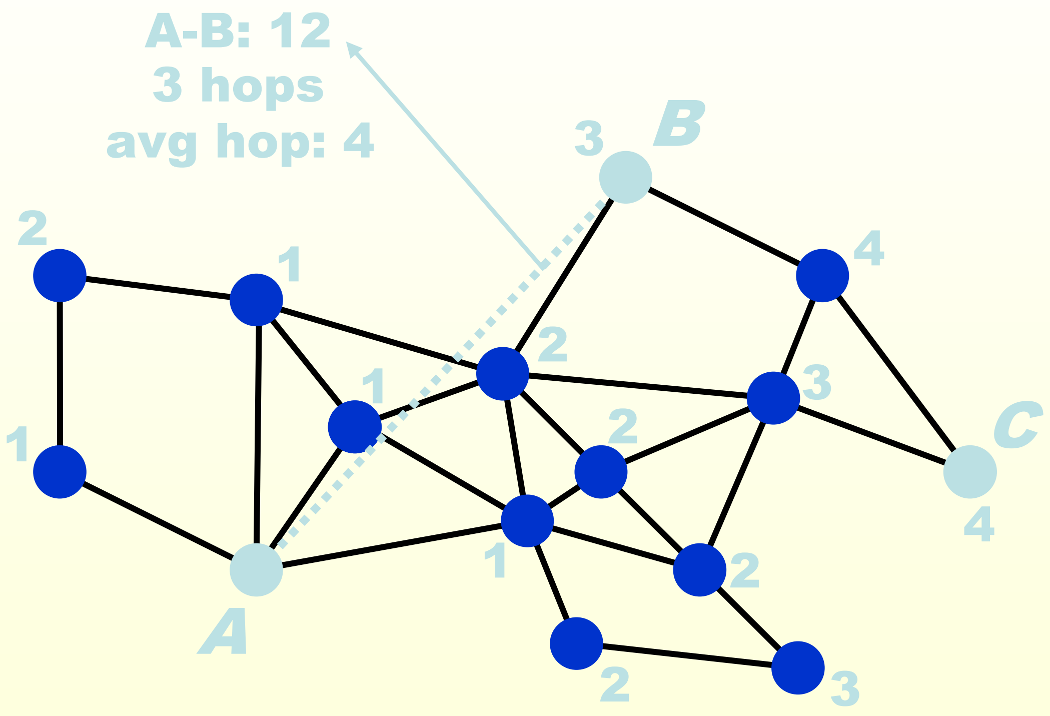
B: 6+4 = 10

C: 5+6+4 = 15

Phase 1: DV-hop

Anchors

- flood network with known position
- flood network with avg hop distance



Nodes

- count # of hops to anchors
- multiply with avg hop distance

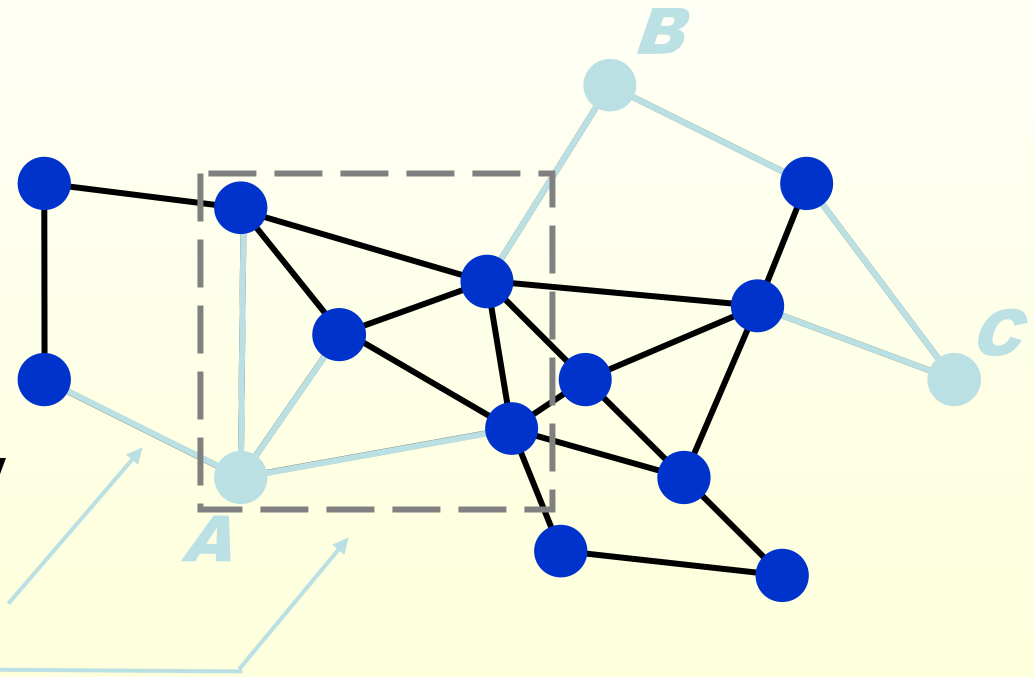
Phase 1: Euclidean

Anchors

- flood network with known positions

Nodes

- determine distance by
  1. range measurement
  2. geometric calculation
- require range measurement



Phase 1: Euclidean (2)

- Wanted:

  Distance A-G

Using AEG*F*:

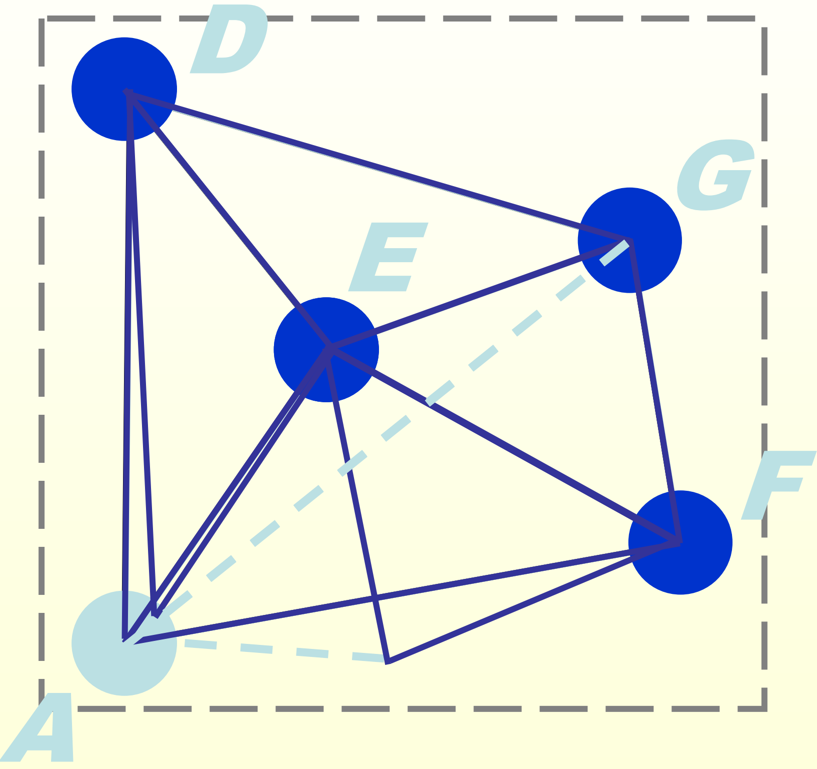
A-G = 8 ...or 3

Using AEG*D*:

A-G = 8 ...or 0.5

↓

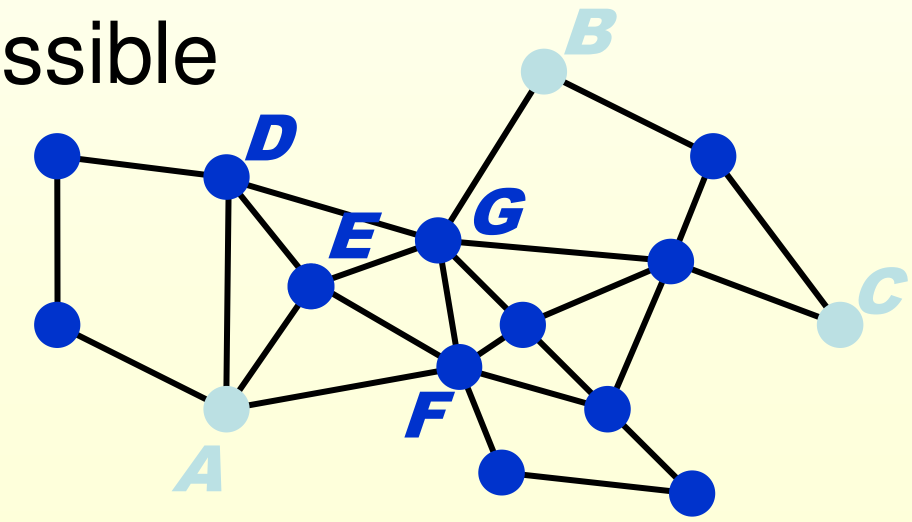
A-G = 8



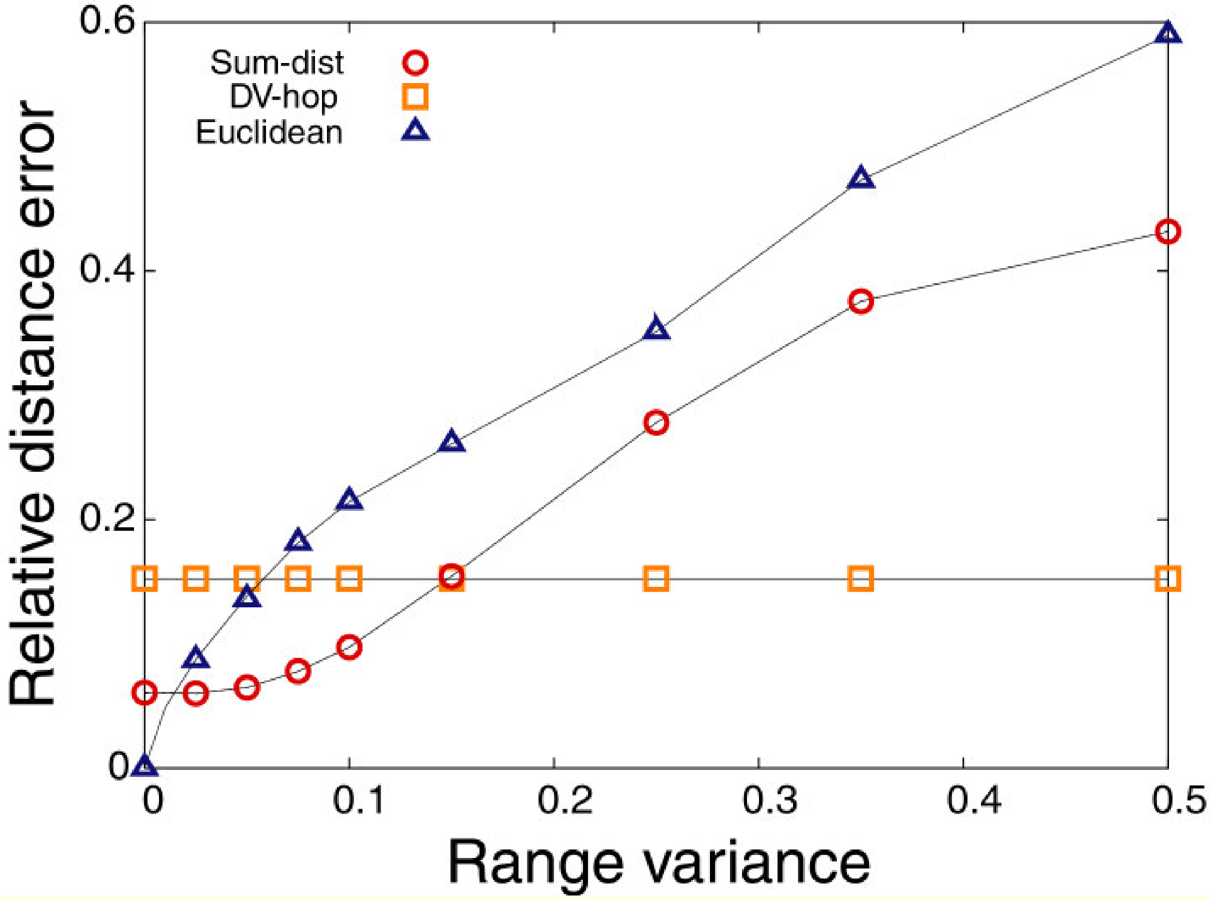
Phase 1: Euclidean (3)

- Needs high connectivity
- Error prone (selecting wrong distance)

- Perfect accuracy possible



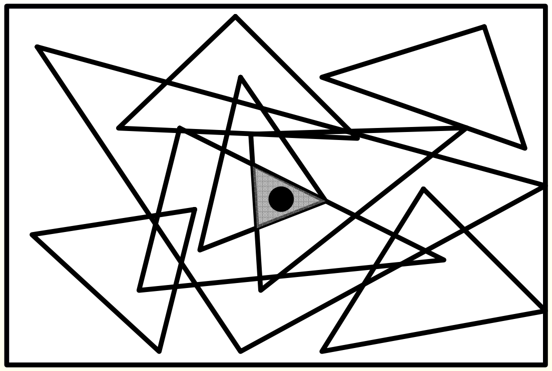
Phase 1: Comparison



- Range measurement
  - Very accurate: Euclidean
  - Reasonable: Sum-dist
  - None / very bad: DV-hop

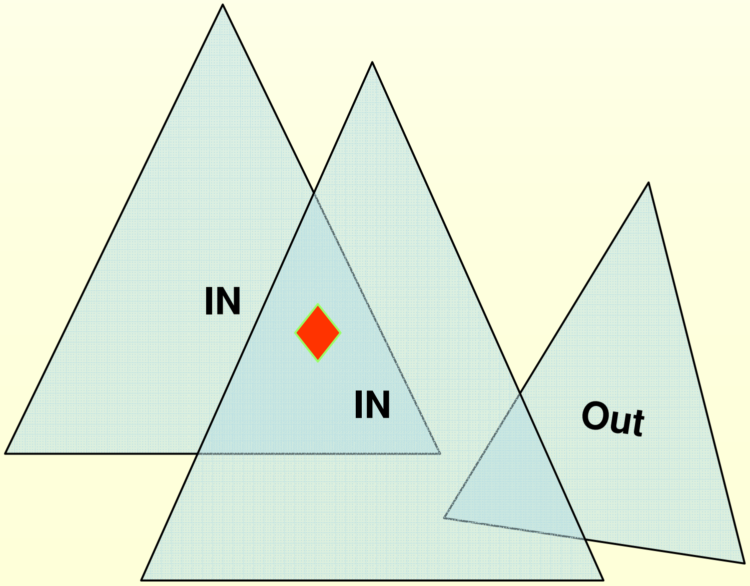
APIT: a Method Using Only Distance Comparisons

- APIT employs a novel *area-based* approach. Beacons divide the field into triangular regions



- A node’s presence inside or outside of these triangular regions allows a node to narrow the area in which it can potentially reside.

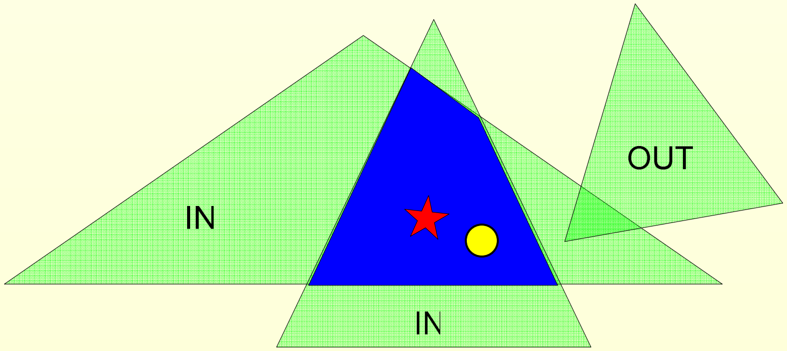
- The method to do so is called Approximate Point In Triangle Test (APIT).



APIT Main Algorithm

For each node

- Get Beacon Locations
- Individual APIT Test
- Triangle Aggregation
- Center of Gravity Estim.

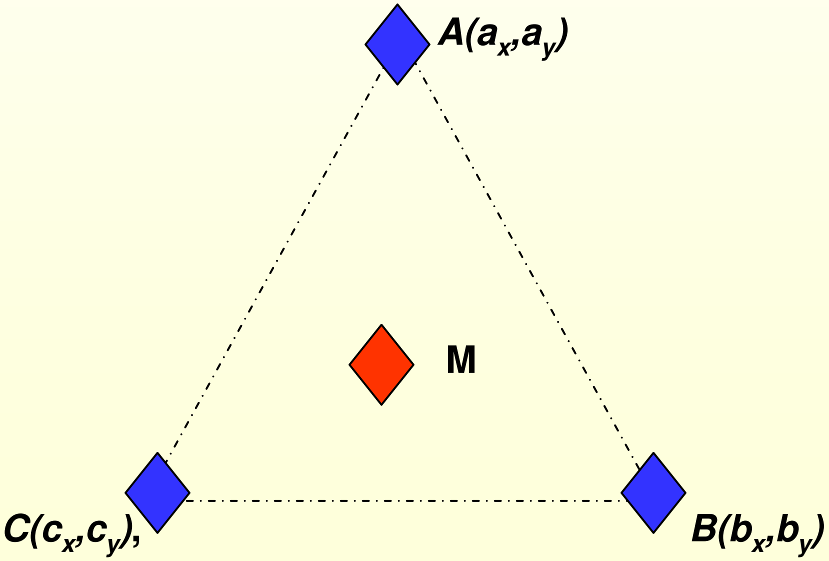


Pseudo Code:

```
Receive locations $(X_i,Y_i)$ from $N$ beacons
$N$ beacons form $\binom{N}{3}$ triangles.
For ( each triangle $T_i$ Є $\binom{N}{3}$ ){
    InsideSet ← Point-In-Triangle-Test ($T_i$)
}
Position = $CoG(\cap T_i \in InsideSet)$;
```

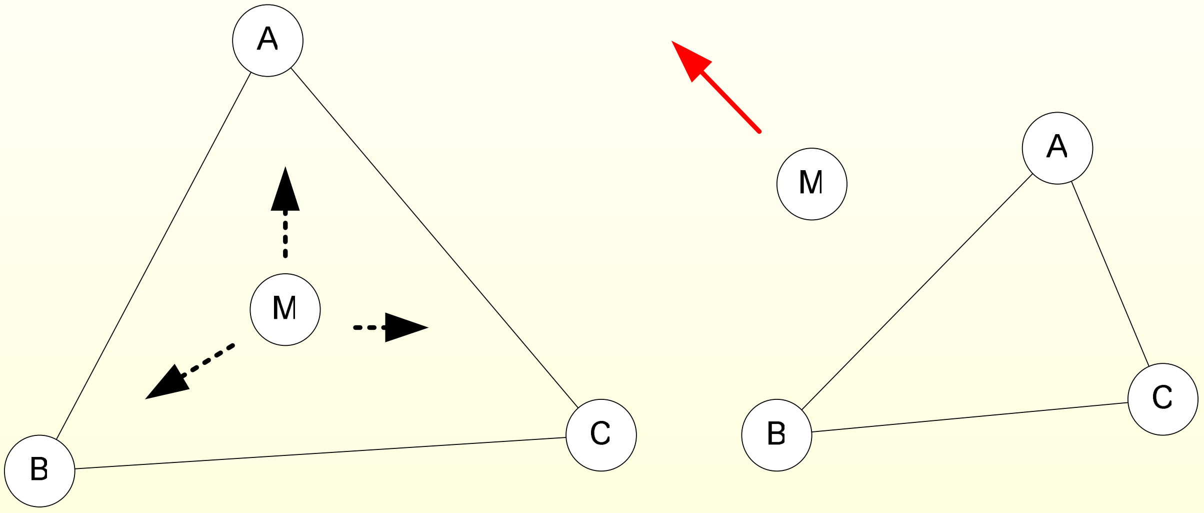
Point-In-Triangle-Test

- *For three beacons with known positions: $A(a_x,a_y)$, $B(b_x,b_y)$, $C(c_x,c_y)$, determine whether a point M with an unknown position is inside triangle ΔABC or not.*



Perfect P.I.T Theory

- *If there exists a direction in which M gets further from points A, B, and C simultaneously, then M is outside of ΔABC. Otherwise, M is inside ΔABC.*



Inside Case

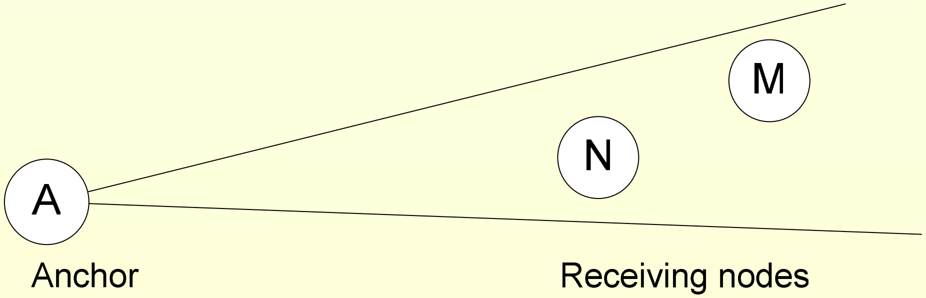
Outside Case

- Require approximation for practical use
  - Nodes cannot move, how to recognize direction of departure (moving away)
  - Exhaustive test on all directions is impractical

Distance Test

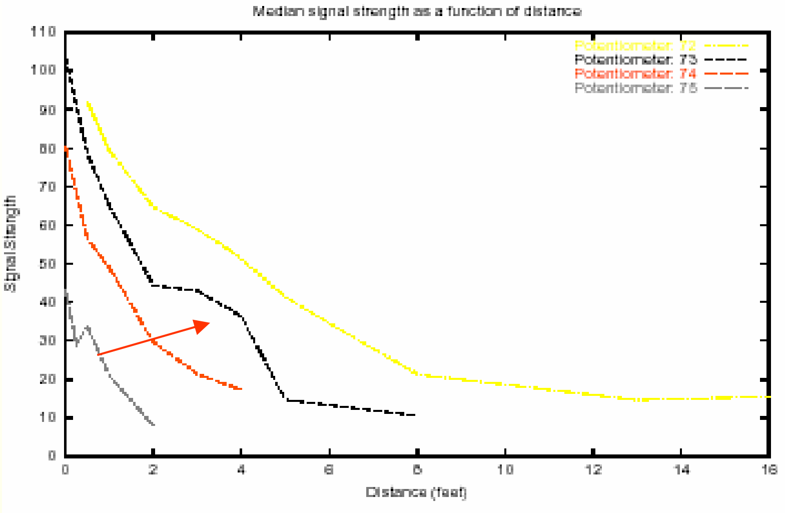
Recognize directions of departure (moving away) via neighbor exchange

1. Receiving Power Comparison
   Smoothed Hop Distance Comparison

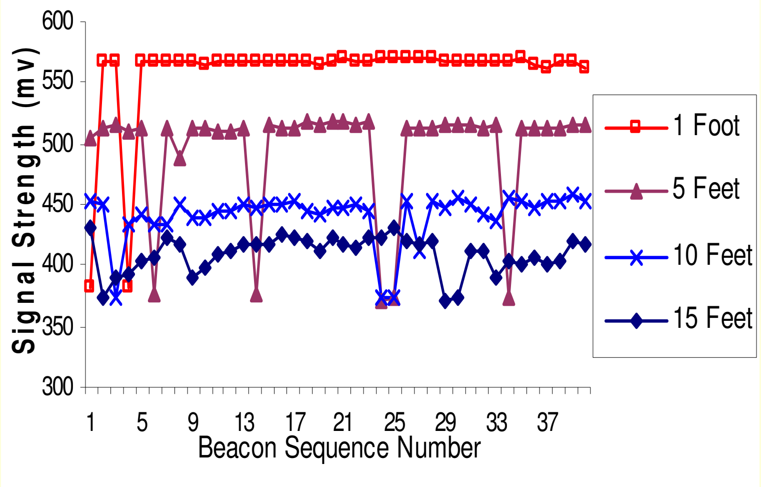


Anchor

Receiving nodes



Experiment Result from Berkeley

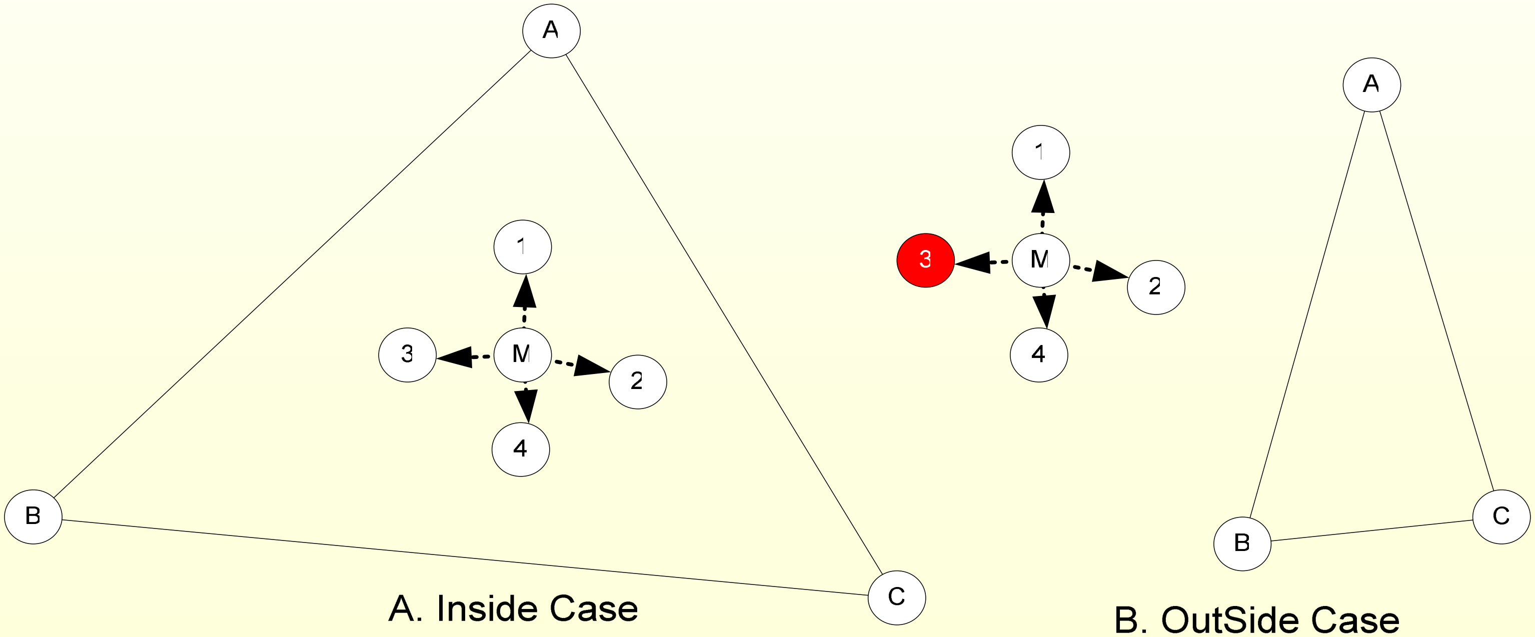


Experiment Result from UVA

A.P.I.T. Test

Approximation: Test only directions towards neighbors

- Error in individual test exists , however is relatively small and can be masked by APIT aggregation.



A. Inside Case

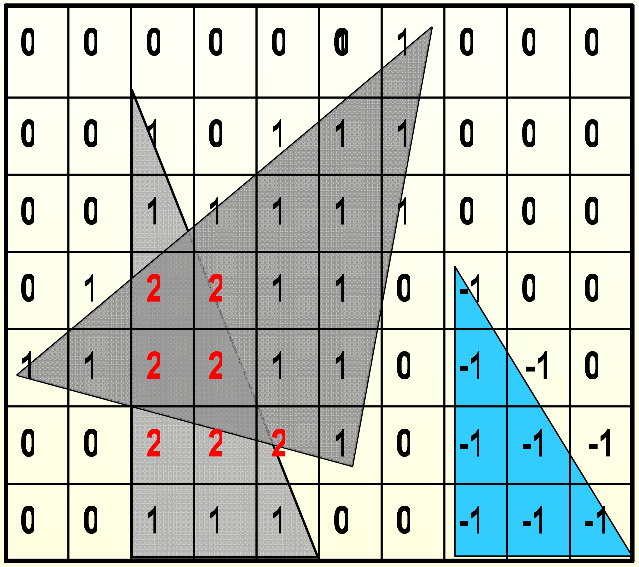
B. OutSide Case

APIT(A,B,C,M) = IN

APIT(A,B,C,M) = OUT

APIT Aggregation

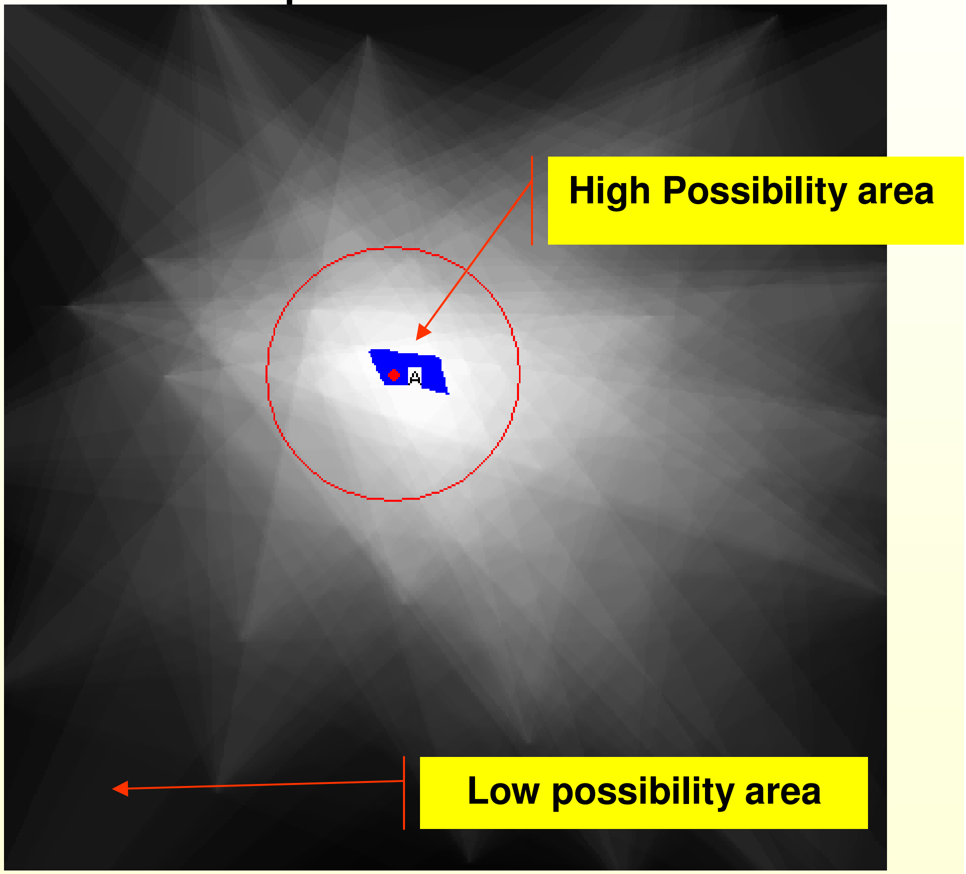
- Aggregation provides a good accuracy, even results by individual tests are coarse and error prone.



0	0	0	0	0	0	1	0	0	0
0	0	1	0	1	1	1	0	0	0
0	0	1	1	1	1	1	0	0	0
0	1	2	2	1	1	0	-1	0	0
1	1	2	2	1	1	0	-1	-1	0
0	0	2	2	2	1	0	-1	-1	-1
0	0	1	1	1	0	0	-1	-1	-1

Grid-Based Aggregation

With a density 10 nodes/circle,
Average 92% A.P.I.T Test is correct
Average 8% A.P.I.T Test is wrong



Localization Simulation example

Does All This Solve the LD Problem?

- No! Several other challenges
- Solution depends on
  - Problem setup
    - Infrastructure assisted (beacons), fully ad-hoc & beaconless, hybrid
  - Measurement technology
    - Distances vs. angles, acoustic vs. rf, connectivity based, proximity based
    - The underlying measurement error distribution changes with each technology
- The algorithm will also change
  - Fully distributed computation or centralized
  - How big is the network and what networking support do you have to solve the problem?
  - Mobile vs. static scenarios
  - Many other possibilities and many different approaches

Location Services

Location Services Motivation

- Even after nodes localize themselves and learn their locations, issues remain ...
- It is not reasonable to assume that all nodes know the locations of all other nodes
- Operations like geographic routing require that we know the location of our destination
- We need to find distributed ways to map node IDs or other attributes to node locations
- This is where *location services* come in

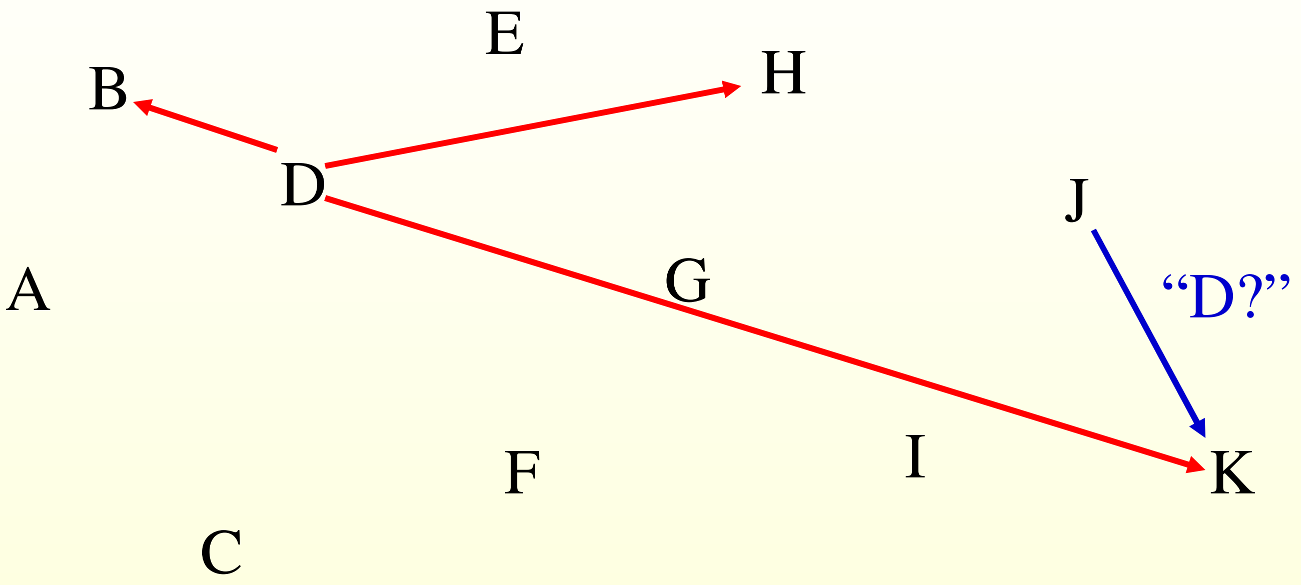
Possible Designs for a Location Service

- Flood to get a node’s location
  - excessive flooding messages
- Central static location server
  - not fault tolerant
  - too much load on central server and nearby nodes
  - the server might be far away for nearby nodes or inaccessible due to network partition.
- Every node acts as server for a few others
  - good for spreading load and tolerating failures.

Desirable Properties of a Distributed Location Service

- Spread load evenly over all nodes.
- Degrade gracefully as nodes fail.
- Queries for nearby nodes stay local.
- Per-node storage and communication costs grow slowly as the network size grows.

Grid Location Service (GLS) Overview



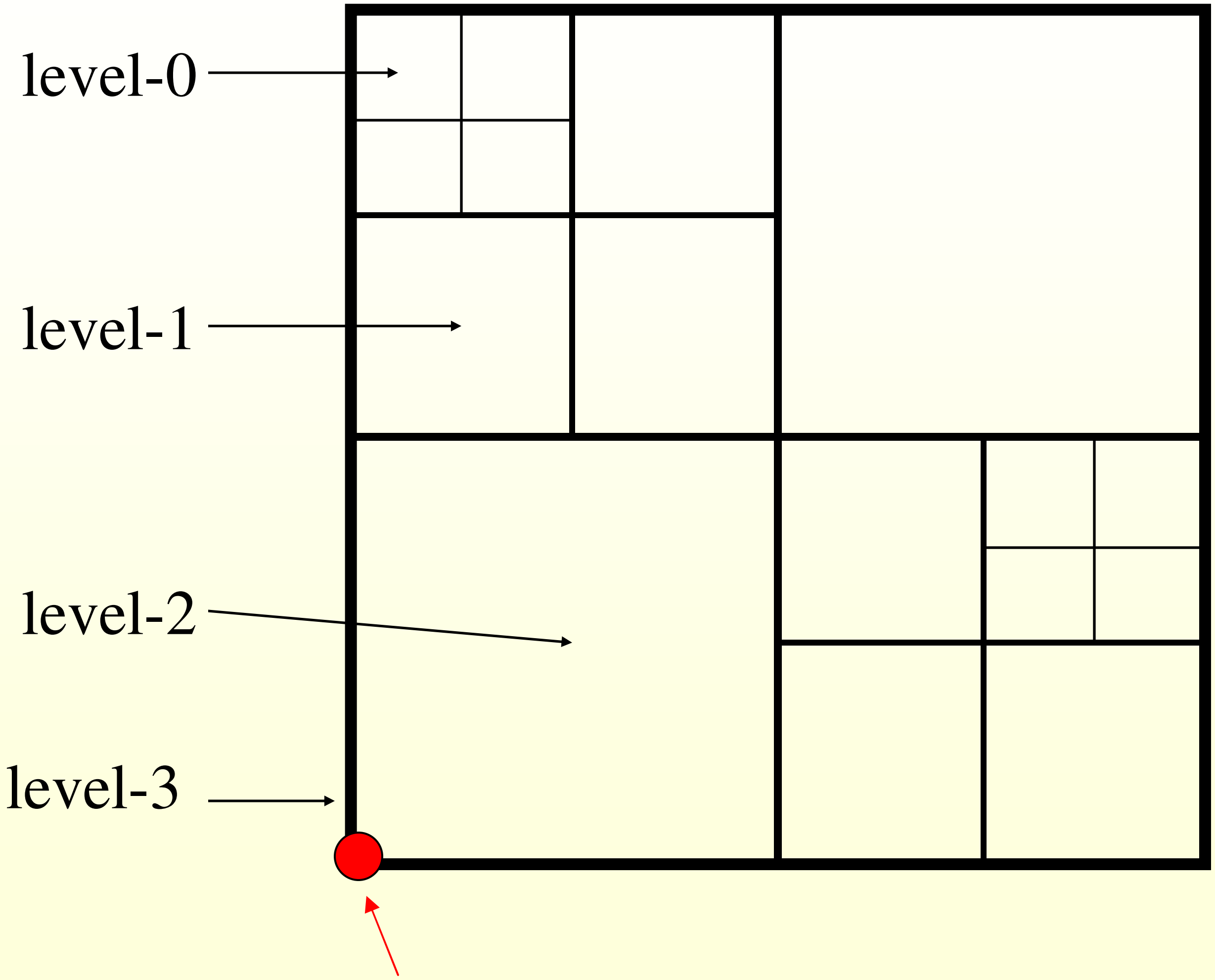
Each node has a few servers that know its location.

1. Node D sends location updates to its servers (B, H, K).
2. Node J sends a query for D to one of D's close servers.

Grid Location Service (GLS)

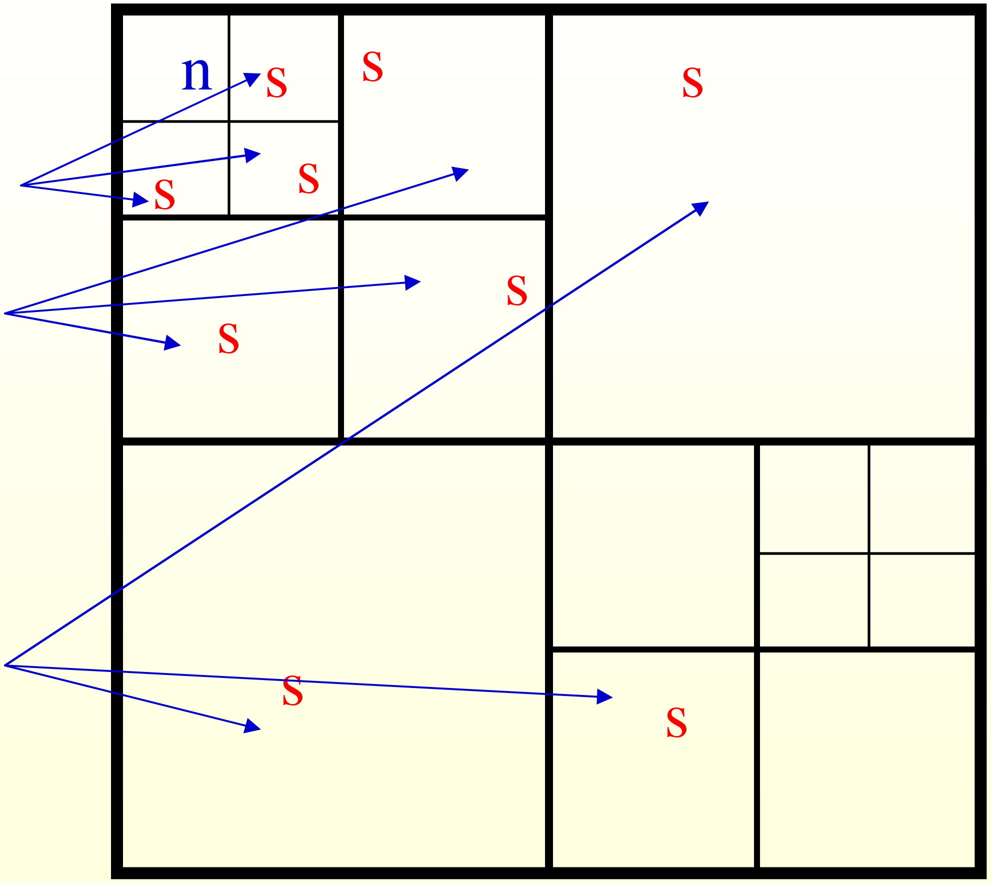
- Each Grid node has a unique identifier.
  - Identifiers are numbers.
  - Perhaps a hash of the node’s ID or network address.
- Identifier X is the “successor” of Y if X is the smallest identifier greater than Y.

GLS's Spatial Hierarchy



All nodes agree on the global origin of the grid hierarchy

Three Servers Per Node Per Level



sibling level-0 squares

sibling level-1 squares

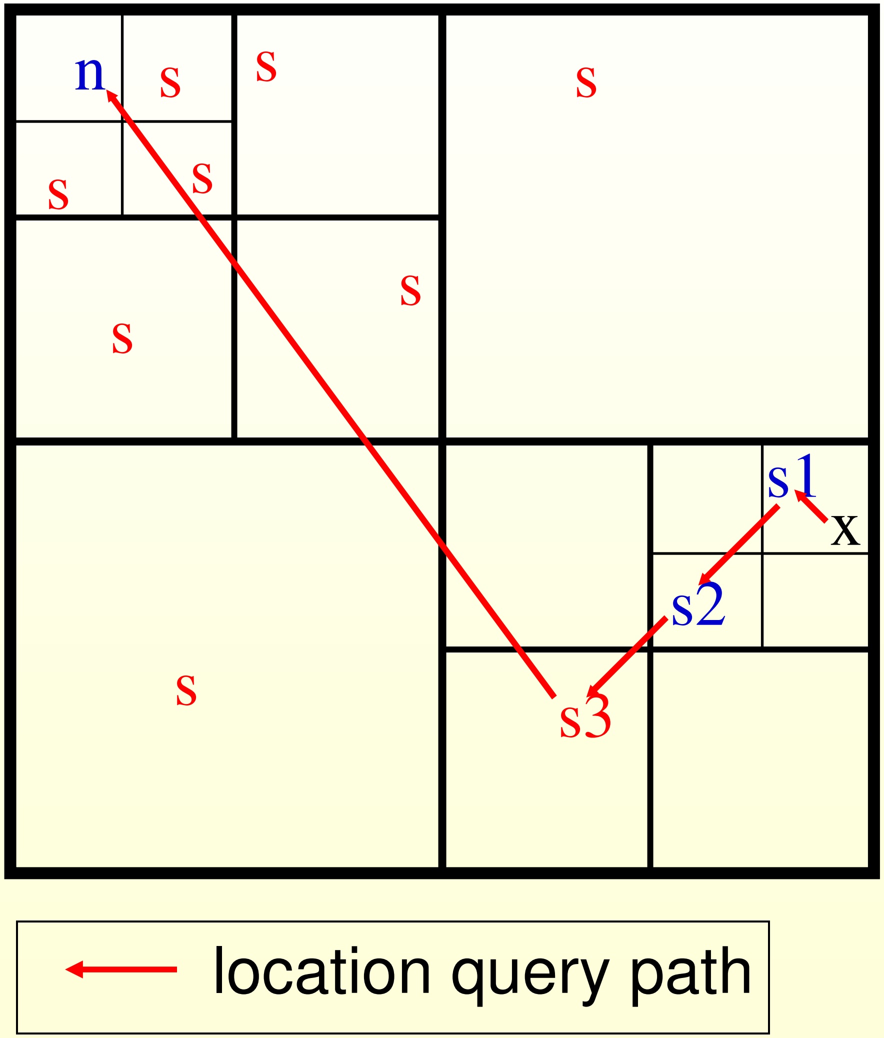
sibling level-2 squares

- *s* is *n*’s successor in that square.

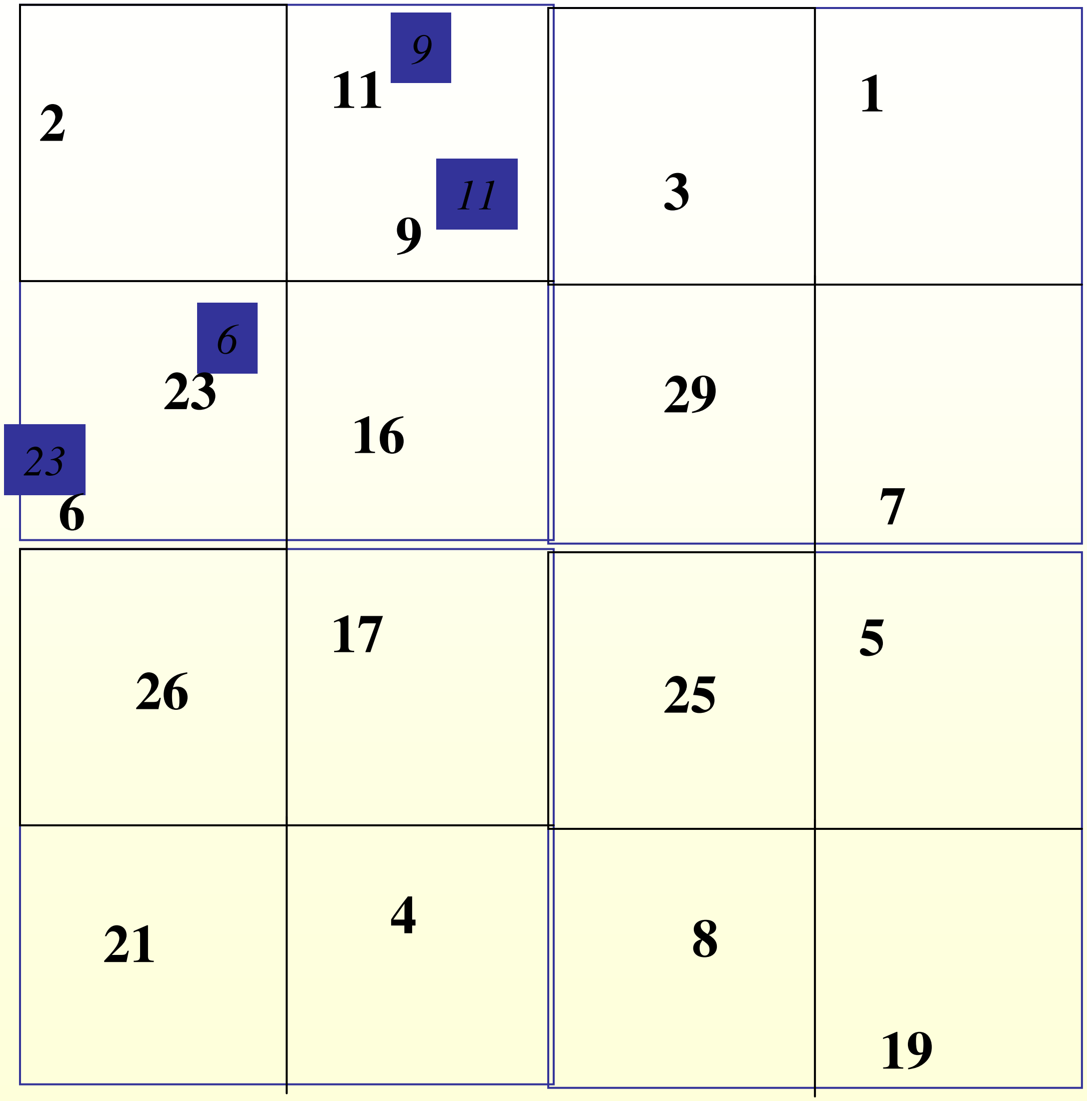
(Successor is the node with “least ID greater than” *n*)

Queries Search for Destination's Successors

Each query step: visit *n*'s successor at surrounding level.



GLS Update (level 0)

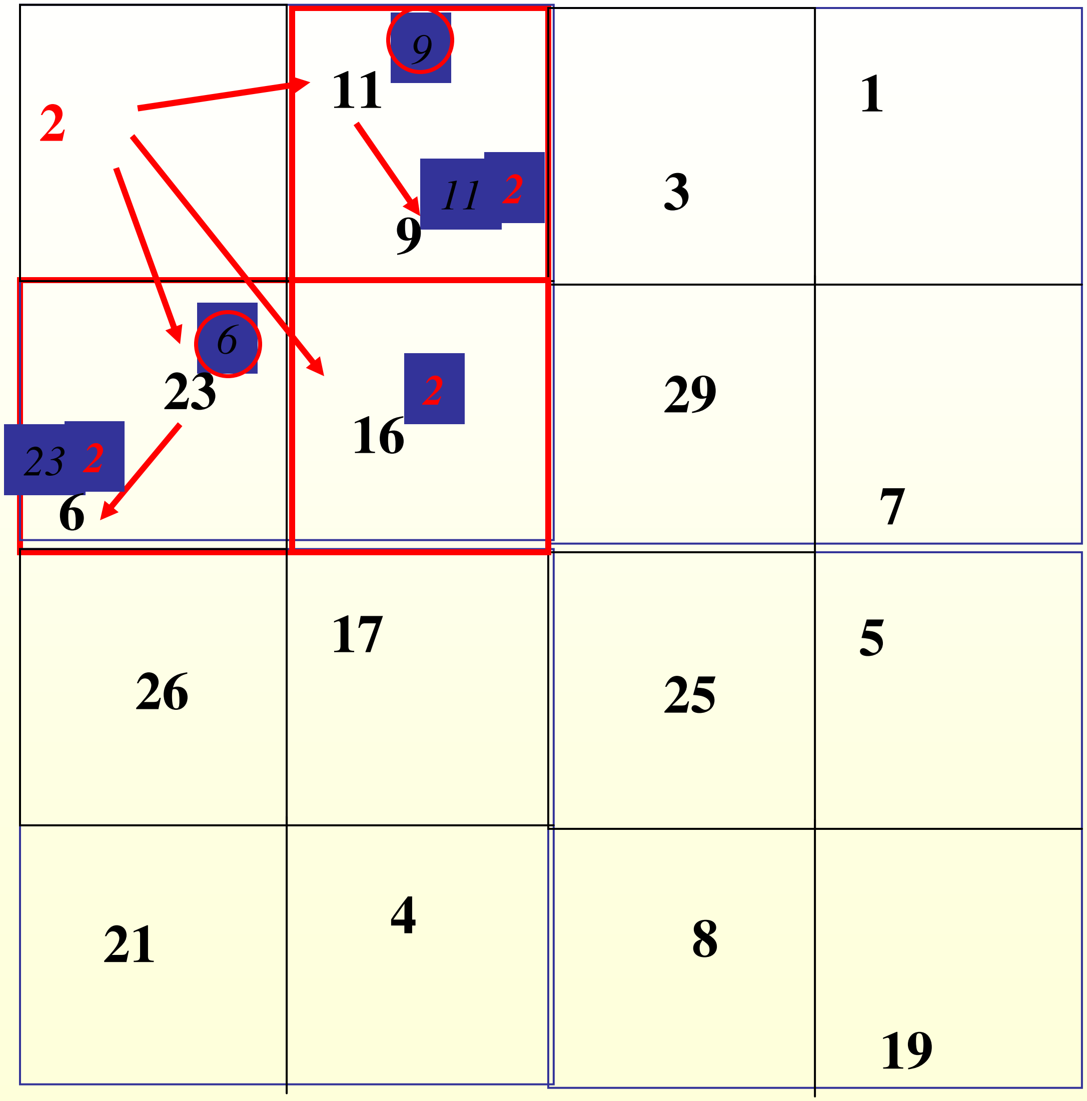


Invariant (for all levels):
For node *n* in a square, *n*'s successor in each sibling square "knows" about *n*.

Base case:
Each node in a level-0 square "knows" about all other nodes in the same square.

■ location table content

GLS Update (level 1)

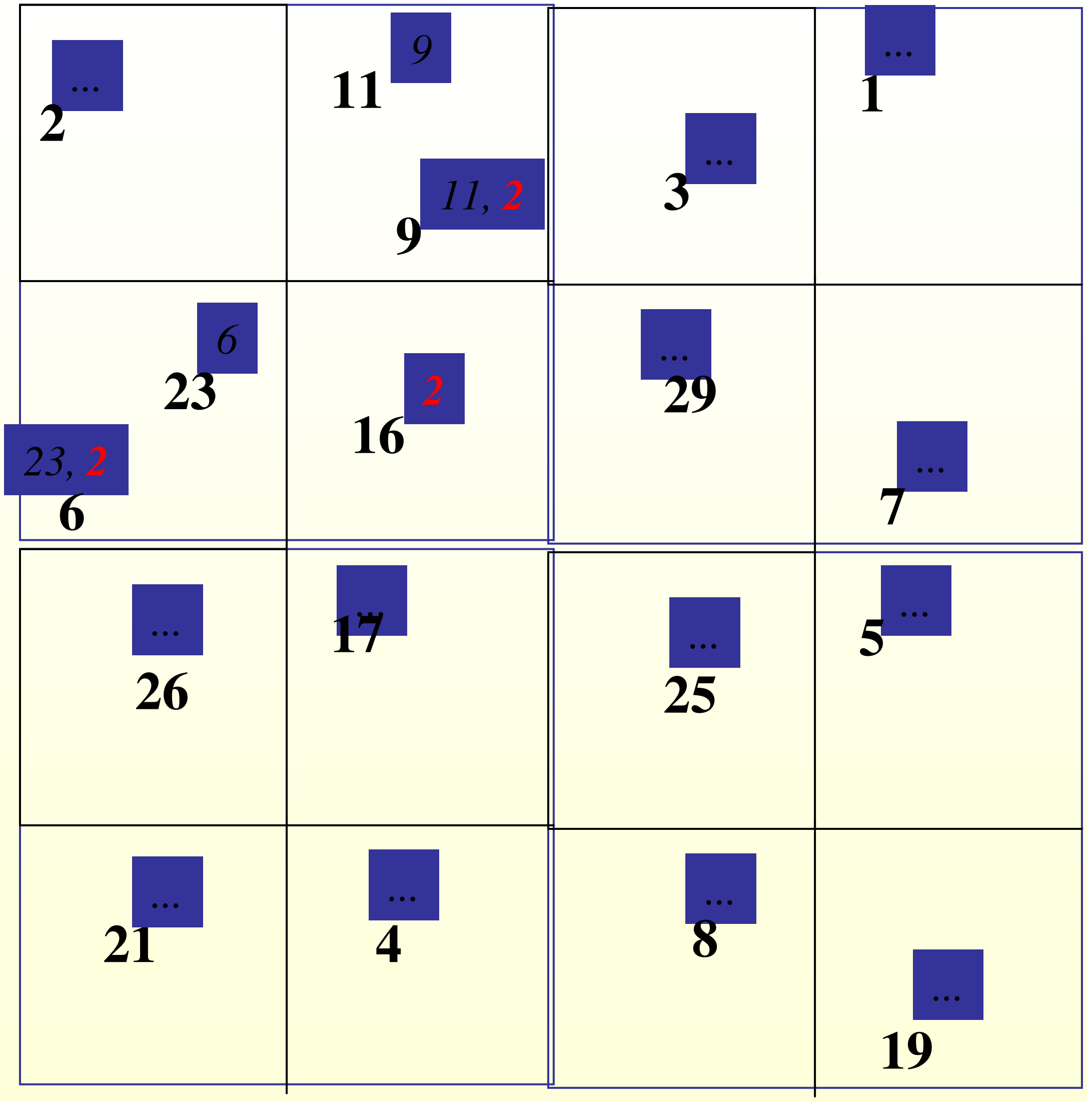


Invariant (for all levels):
For node *n* in a square, *n*'s successor in each sibling square "knows" about *n.*

■ location table content

→ location update

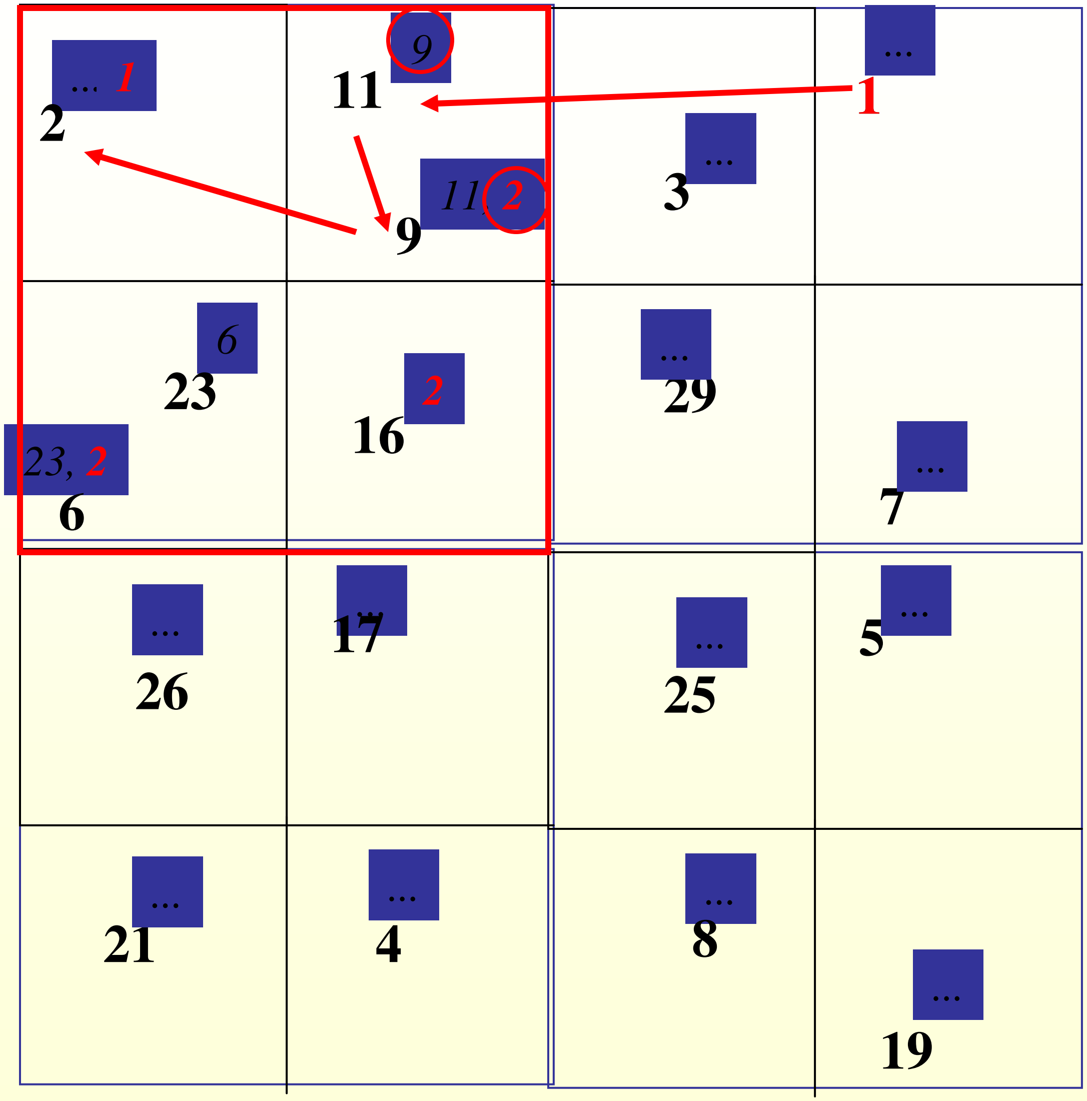
GLS Update (level 1)



Invariant (for all levels):
For node *n* in a square, *n*’s successor in each sibling square “knows” about *n.*

■ location table content

GLS Update (level 2)

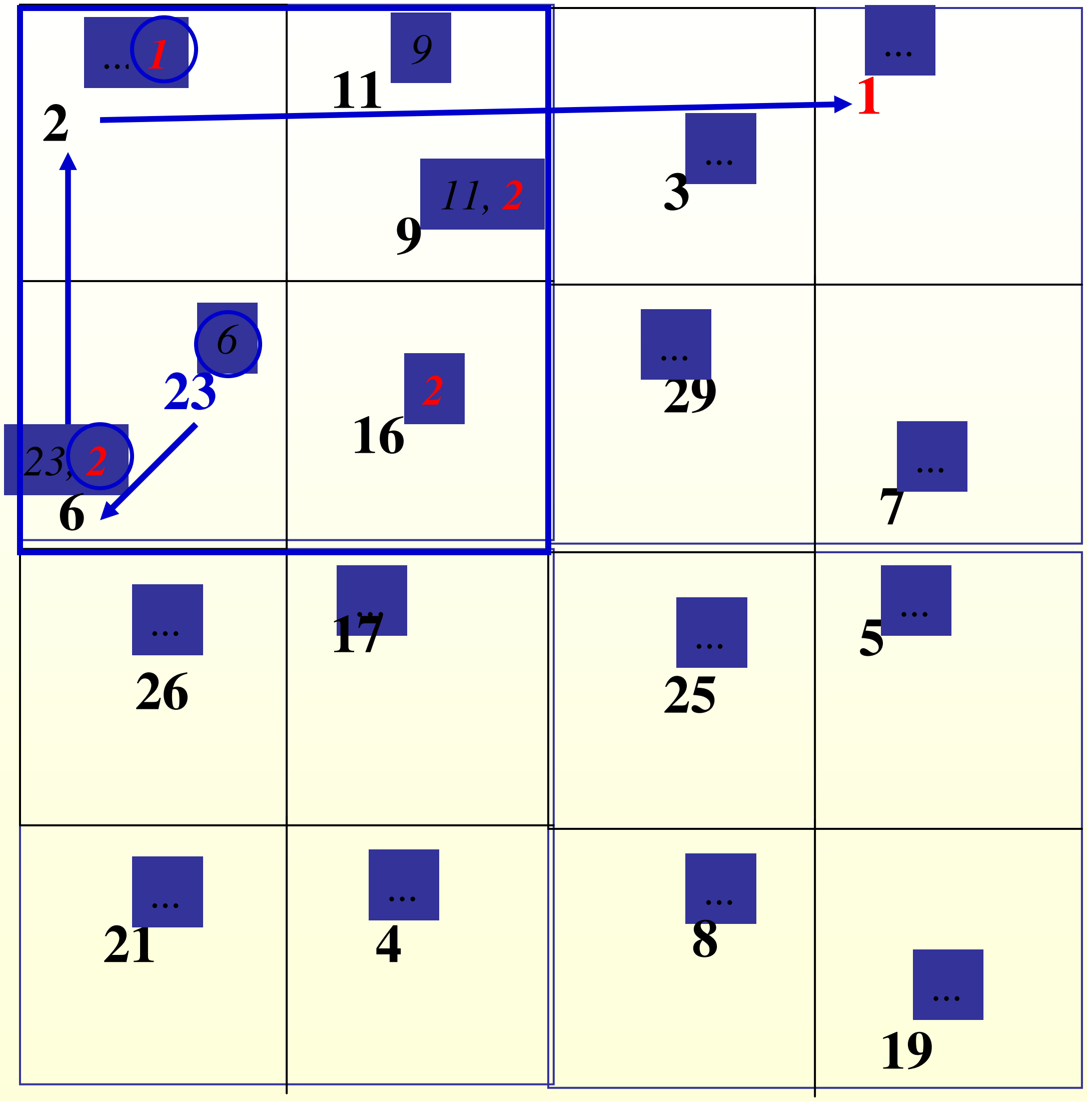


Invariant (for all levels): For node *n* in a square, *n*’s successor in each sibling square “knows” about *n.*

■ location table content

→ location update

GLS Query



▇ location table content

→ query from *23* for *1*

Challenges for GLS in a Mobile Network

- Even in a static sensor network, we may have deal with locating mobile processes hopping from node to node
- Slow updates risk out-of-date information.
  - Packets dropped because we can’t find the destination.
- Aggressive updates risk congestion.
  - Update packets leave no bandwidth for data.
- Large mobile ad-hoc nets usually suffer from one or the other.

Summary

- Location discovery is a central but difficult problem in sensor networks
- Most localization methods are based on ranging (distance estimates)
- Localization algorithms can be demanding for small nodes
- Localization errors need to be taken into account
- Location services are needed so that location information becomes globally available

The End