Location management in ad hoc networks

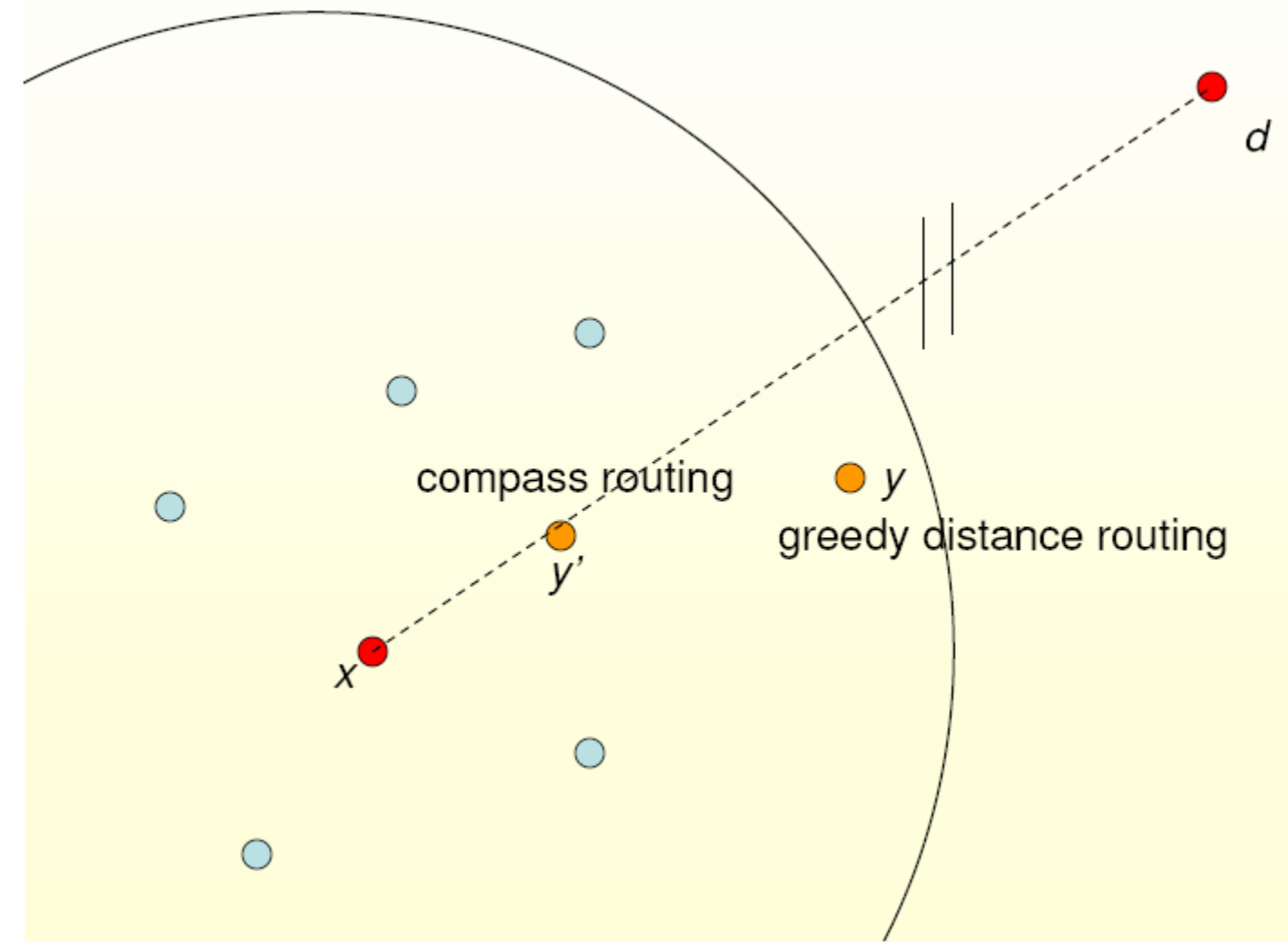
3/17

Location management in cellular networks

- Base stations keep the location information.
- Two primitives: paging, update.
- The challenge is to minimize the cost of location management.
- Relatively easy problem. Infrastructure helps.

- In an ad hoc network, the problem is challenging
- Where is the node?
- Who keeps this information?
- How is query implemented?

Location helps network functionality



Location-based applications

- E.g., sensor networks.
- Target tracking.
- Geographical multicast.
- Distributed sense-and-respond system.

Distributed location service

- Geographical routing requires obtaining the location of the destination.
- What if the sensors move? How to update the location information?
- Location service: a distributed service that maps IDs to locations and answers the location query for any node.

Location service

- The network must somehow know where a node is
  - a node has to **publish** its location
- A node must be able to find out another node's location, given its network identifier
  - a node wants to **lookup** another node's location
- A **lookup service** offers **publish** and **lookup** primitives

A publish-driven approach

- When a node comes up or changes its position, it simply floods the network with its new location
- Each node has to hold information about every other node in the network
  - memory...
  - traffic...

A lookup-driven approach

- When a node wants to know another node's position, it simply floods the network with a query
- The corresponding node answers with its position
  - no memory needed
  - even more traffic
  - what is the point of geographical routing?

A centralized location server

- Centralized 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.

Design principle

- No node should be a bottleneck.
- Failure of a node should not affect the reachability of many other nodes.
- Queries of nearby hosts should be answered with correspondingly local communication.
- Per-node storage and communication cost should grow moderately slow.
- Every node serve as location servers for some other nodes.
- No hierarchy, since the node on top of the hierarchy tends to be overloaded.

Challenge

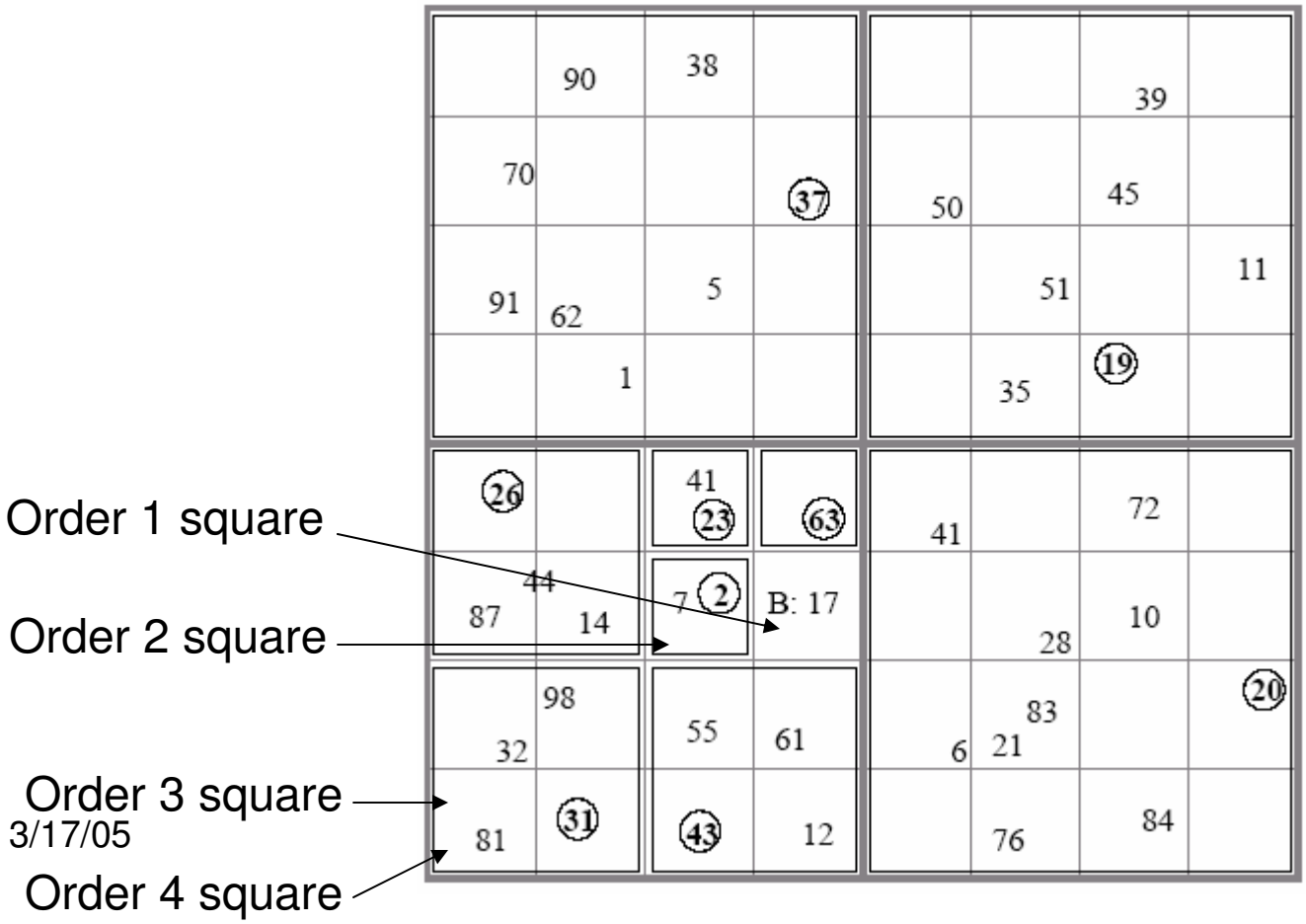
- Each node has several other nodes as location servers and also acts as a location server on behalf of some other nodes
- Problem 1: how to get to the location server?
  - We need a routing algorithm, say geographical routing.
- Problem 2: geographical routing requires the knowledge of destinations.
  - How to get the location of the location server?
  - Every node can be moving.
- Chicken and egg problem?

Grid location service

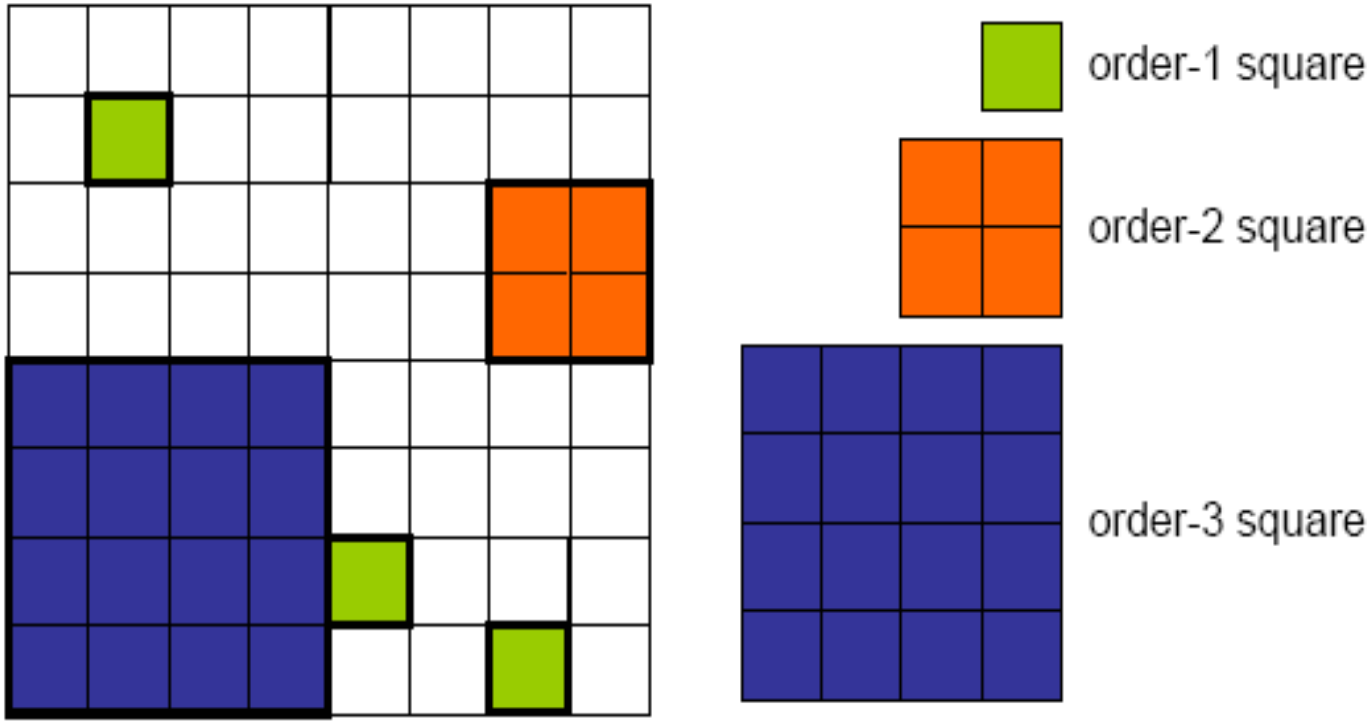
- Each node is assigned a random ID: computed by a strong hash function on physical name, e.g., MAC address.
- Each node stores/updates its location information at a set of location servers, more at nearby regions, fewer at far away regions.
- Location query uses nothing beyond the ID.

Recursive partitioning

- Quad-tree partition: each node is inside a unique square on each level.



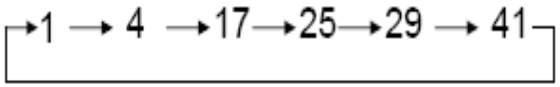
Partitioning the world

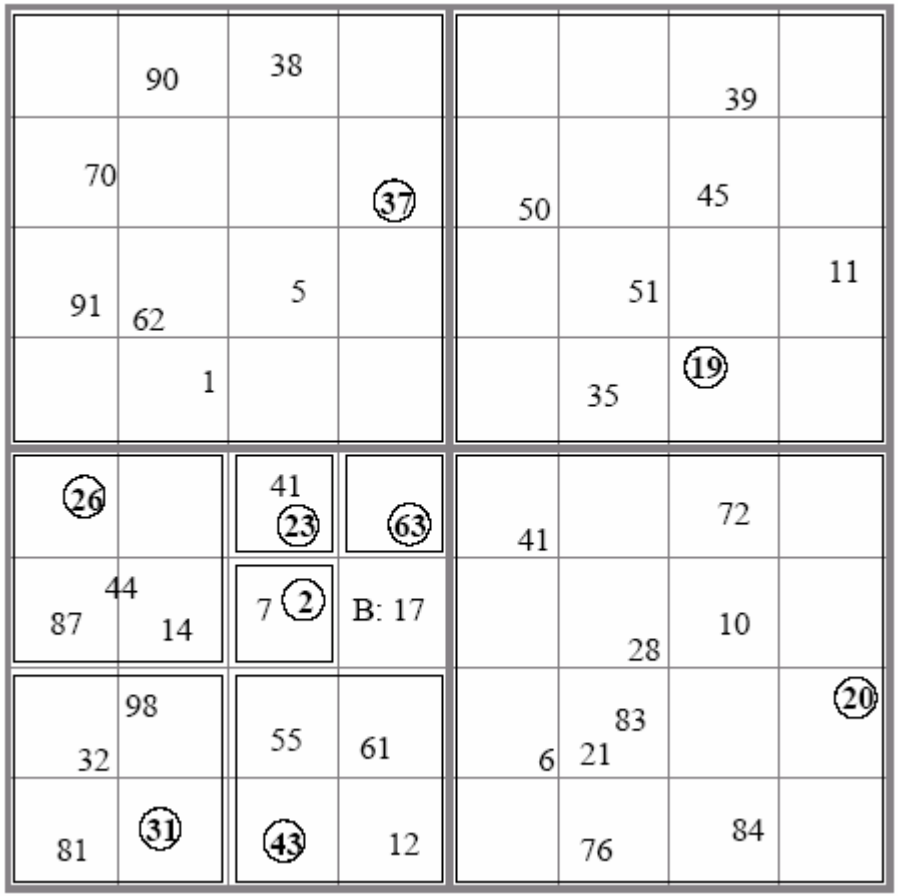


Invariant: a node is located in exactly one square of each size (no overlapping)
An order-x square contains always 4 order-(x-1) squares

Location servers

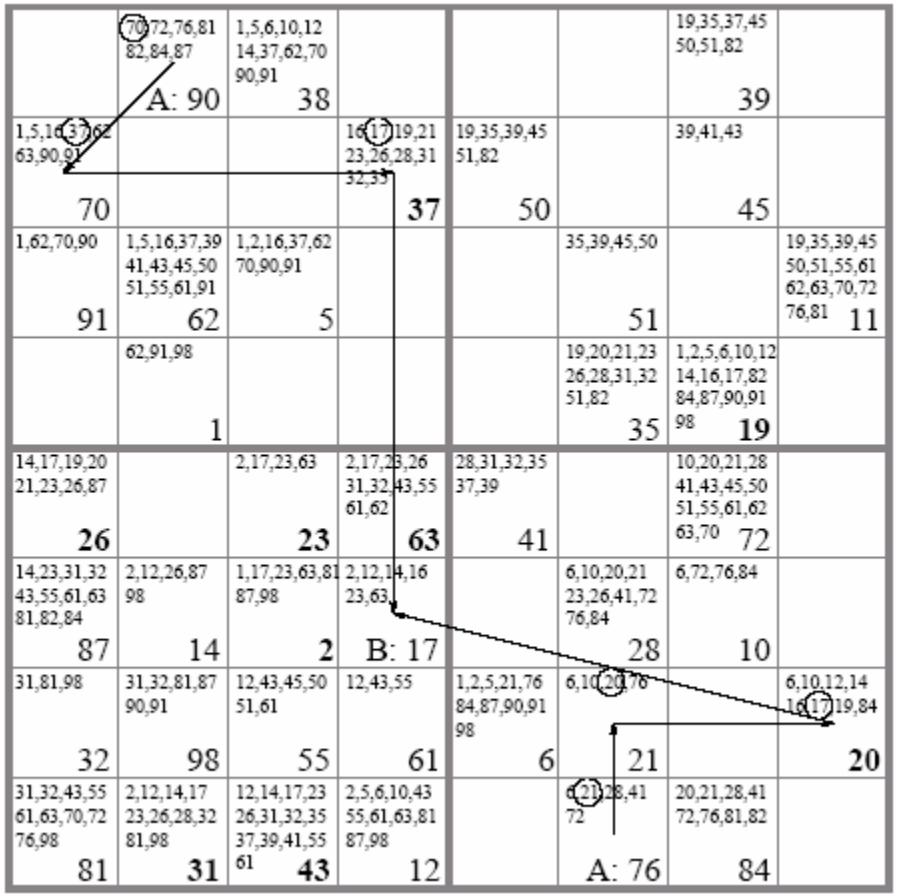
- Node B's location servers: Inside each sibling square on each level, choose B's closest node.
- ***Def.:*** Node **closest** to B in ID space: node with least ID greater than B
- Circular ID space: 2 is closer to 17 than 7 is.





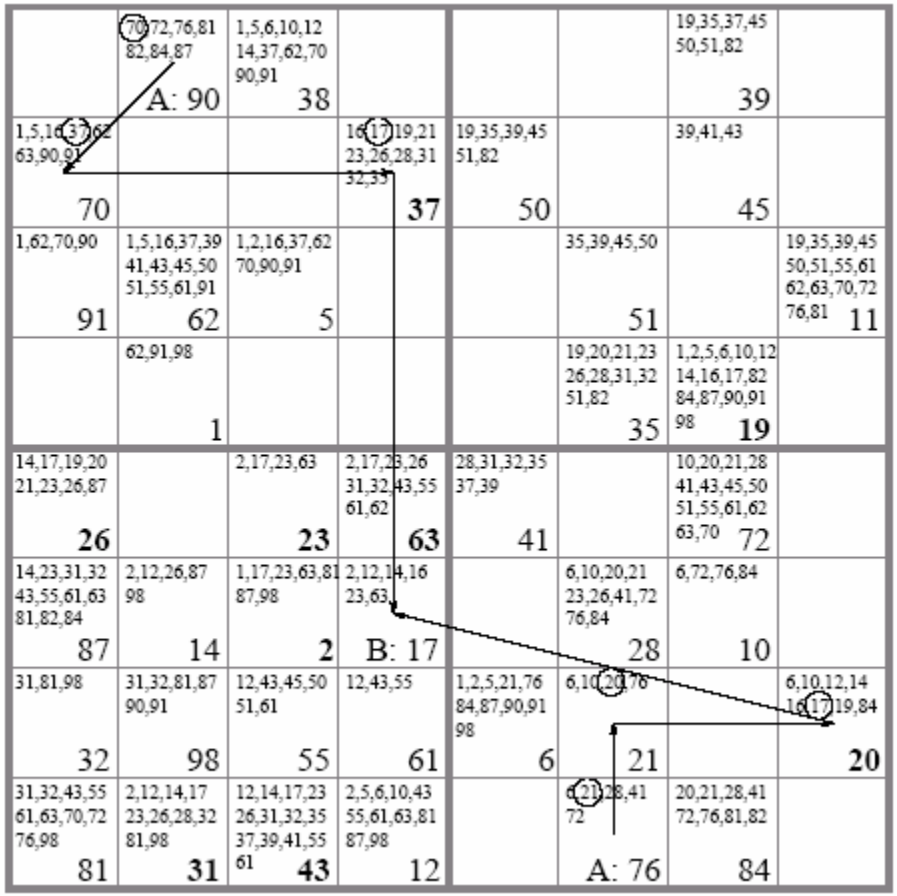
Location queries

- A queries the location of B:
- A's only information about B is the ID of B.
- A does not know who are B's location servers.
- B even doesn't know its location servers.
- How to implement location query?



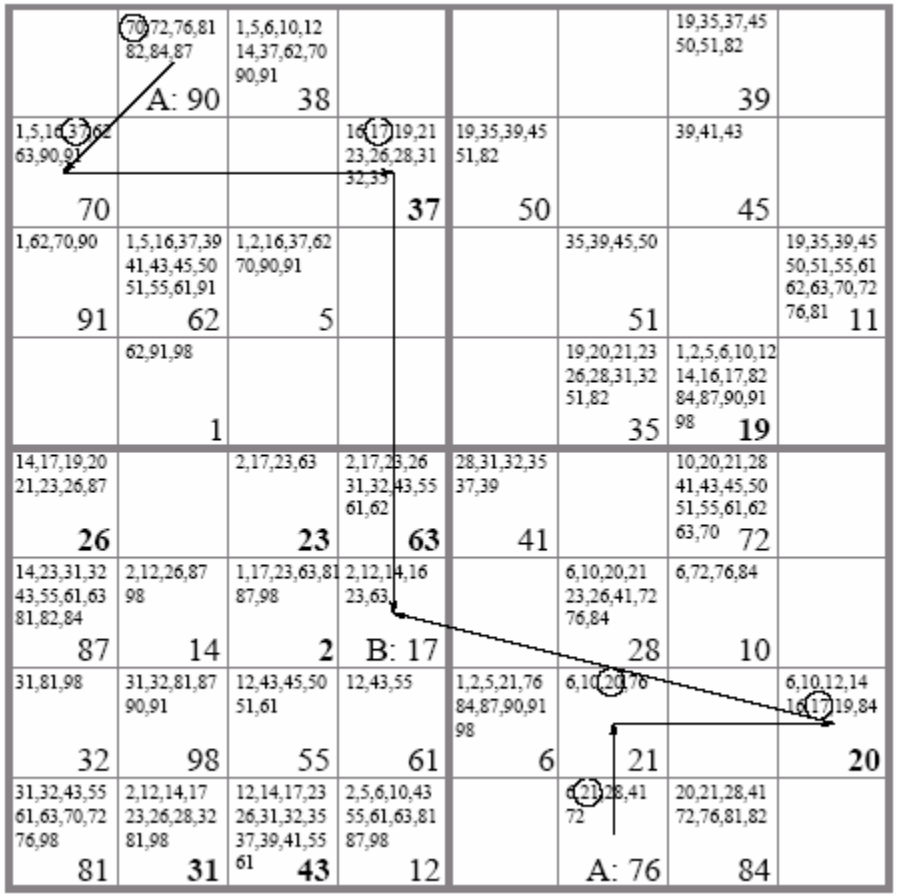
Location queries

- A queries location of B:
- A stores location information for some other nodes.
- A send the request to the one that is closest to B, among those about which A has location information.
- Continue until hit one of B's location servers.
- This works! Why?



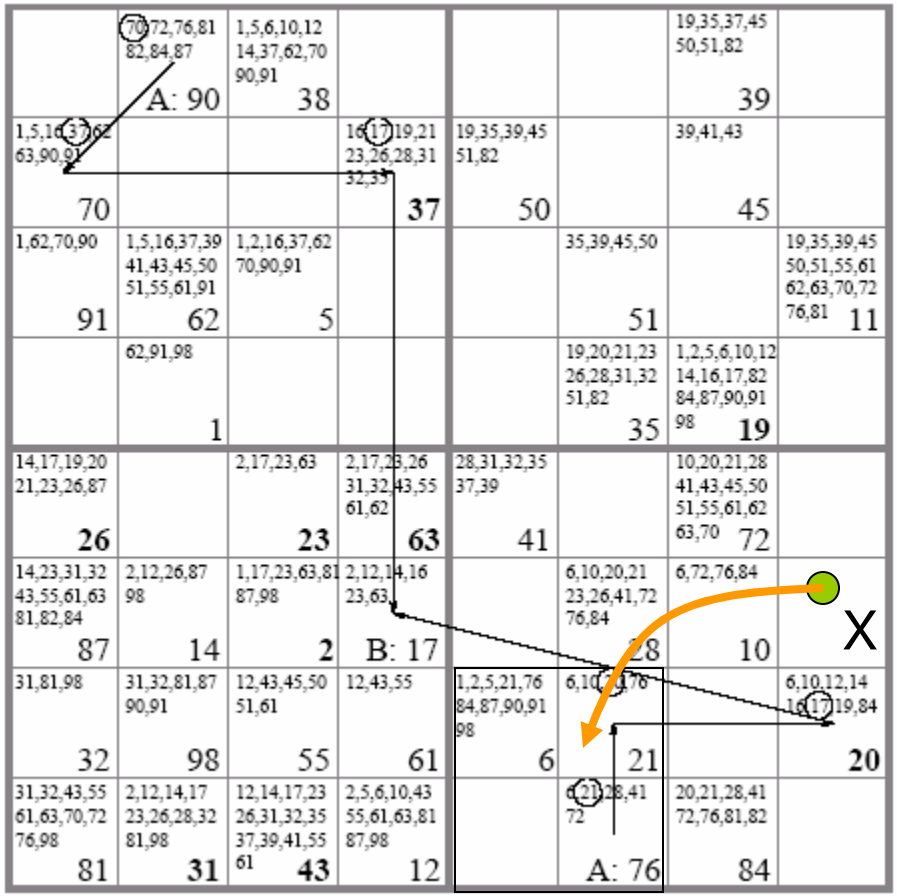
Location queries

- Claim: the query visits the node closest to B in A's order-i square.
- The query always goes to B's closest node, as the covering scope increases.
- The correctness of the alg: when A's order-i square contains B, the closest node is B itself.
- Proof by induction. It's obvious for order-0 and order-1 square.



Location queries

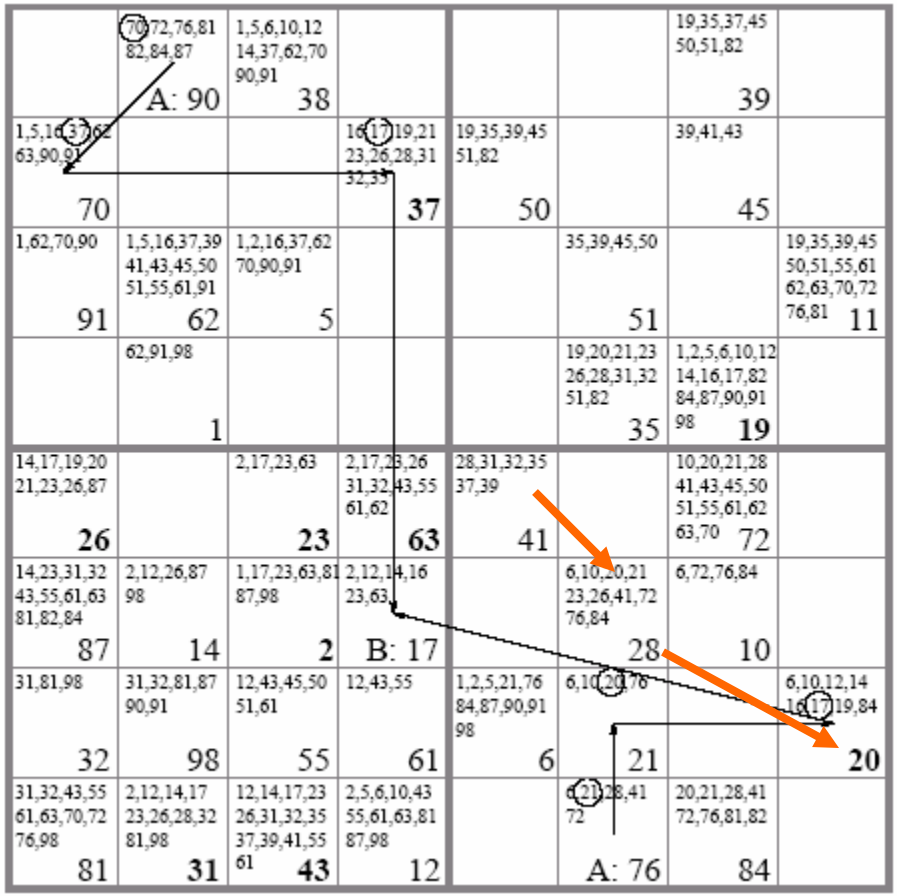
- 21 is B's closest node in order-1 square ➔ no node is between 17 and 21 in order-1 square.
- Suppose a node X in A's order-2 sibling square is between 17 and 21. By the replication rule, X picks 21 as its location server.
- 21 stores the location of all the nodes between 17 and 21 in order-2 square, obviously the one closest to 17.



Inform/update location servers

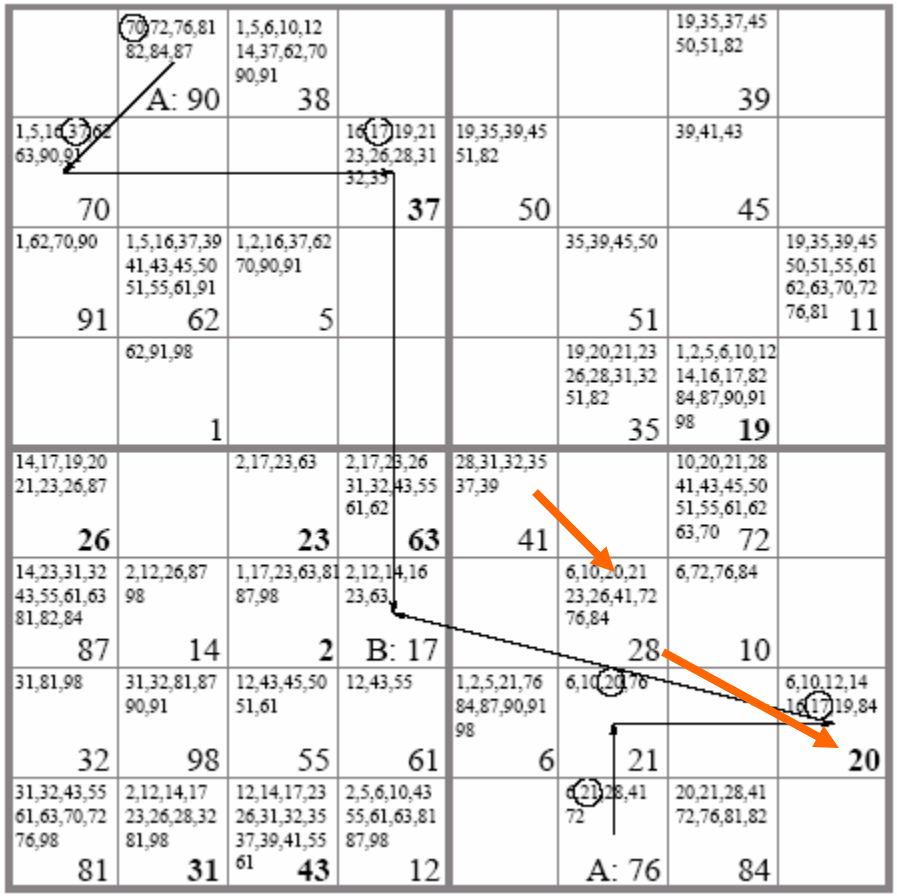
- A can update its location server inside a square S without knowing its identify.
- A routes to a square with geographical routing.
- The first node in the square S performs a location query of A.
- The query ends up at a node closest to A, who is A's location server!

Hidden assumption: the nodes in S have distributed their locations inside S!



The bootstrapping

- When the entire system is turned on, order-1 squares exchange their information with local protocol, then nodes recruit their order-2 location servers and so on.
- No flooding needed. The location service is constructed by geographical unicast routing only.



Location service

- It solves location service problem by using geographical routing.
- More locality sensitive: a node acquires the location from a nearby server.
- Load balancing: location servers are spatially distributed.
- Simple rule, simple construction and maintenance.
- Worst-case query behavior is not bounded, however. ☹

Challenges

- 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.
- Delicate balancing is needed.

Summary

- Location service in ad hoc networks
- Distributed location servers.
- Publish / lookup
- Hierarchical management.