



A Scalable Location Service for Geographic Ad Hoc Routing

GLS

Relation to previous protocols

- DSR (layer 2)
 - On demand and supports unidirectional links
 - Amount of cached state can approach LS
- DSDV (layer 3)
 - Adds sequence numbers to DV to remove loops
 - Proactively broadcasts network updates

GLS

Why do we care?

- Previous ad hoc solutions simply cannot scale to meet real scalability requirements for network size and endpoint movement.
- We need something that can scale! (and requires no administration.)

GLS

Geographic Routing

- Forwards packets solely based on the router's immediate neighbors.
- Makes the local optimal choice; its greedy.
- Minimal state and computation required.
- Its SCALES!!!
- Needs a location resolution service.

Previous Location Servers

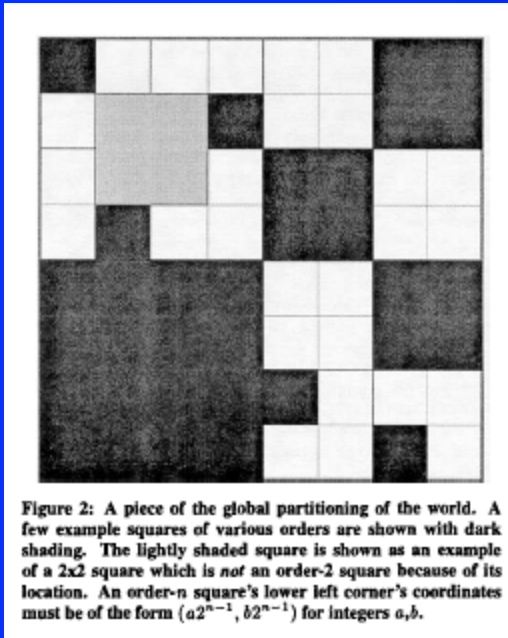
- Naive static/predefined location server
 - Single point of failure
 - Scaling problems
 - No benefits from locality
- Landmark system
 - Address hierarchy => root server => administration
 - Better performance, but still has the same fundamental problems

Scaling Requirements

- No single node should become a bottleneck.
- Failure of one node should not affect the reach ability of other nodes.
- Queries for nearby hosts should be resolved by correspondingly local communication.
- Cost of storage and communication should grow slower than the number of total nodes.

GLS

The Grid/ i^{th} Order Squares



- Nodes create their own cooperative infrastructure or grid.
- The smallest square is an order-1 square.
- A node is located in exactly one square of each size.
- An order-n square is part of only one order- $(n+1)$ square.
- Four order-n squares make up an order- $(n+1)$ square.

Mobile Nodes

- Hash unique name (MAC/IP/hostname) to get a random/uniform distribution of addresses/IDs.
- Use geographic routing to forward packets.
- Use location servers to resolve geographic locations for routing from addresses.

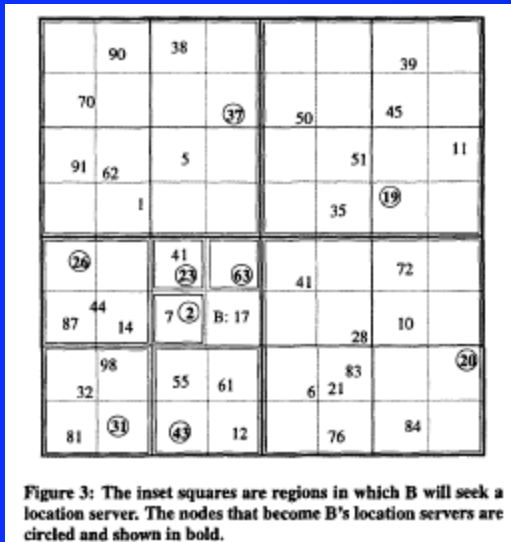
GLS

Location Servers

- Allows nodes to resolve locations with no prior knowledge beyond the node's ID.
- Each node may act as a location server for other nodes.
- Location of location servers are relatively denser closer to the node. Query paths become proportional to data path lengths.

GLS

Selecting Location Servers



- A node recruits location servers with IDs “close” to its own. (the least value greater than the node’s.)
- Registers with three location servers at every order in the grid hierarchy.
- Update/Choose Location Server:
 - First routes an an update query to a node in the appropriate square.
 - Then that node will do a location update process that mimics a query for the original nodes location, passing along the original node’s location.
 - Stops when there are no known nodes with “closer” IDs in that square.

GLS

Location Resolution/Queries

- To route from A to B:
 - A forwards the query to the “closest” node to B in the order-1 square.
 - At each next step, a query is routed to the closest (known) node at the next hierarchy square.
- It is efficient: No more than n location server hops are required to resolve a destination that is collocated in an order n square with the source.
 - The order- $(n+1)$ square’s “closest” node (the next hop) will have chosen the current node in the order- n square as its location server (the current node must be the “closest” node in the order- n square.) Therefore it will know its location and can route to it.

	1.5, 1.12.12 14.37.12.70 90.1		19.35.37.45 50.51.81
A: 90	38		39
1.3.16.38 63.90.32		18.0.18.21 21.26.28.31 72.37	19.35.39.45 21.26.28.31 31.82
70		37	50
1.1.20.90	1.5.16.37.39 41.43.45.50 51.27.41.91	1.2.16.37.81 78.90.91	35.38.45.50
91	62	5	45
62.91.98			19.35.39.45 30.31.50.61 52.63.78.72 78.81
1			18.0.18.21 21.26.28.31 31.82
			35
14.17.19.20 21.23.28.37	1.17.21.81	1.17.21.28 31.31.41.55 81.81	28.31.31.35 37.38
26	23	63	41
14.21.31.31 43.55.61.63 81.81.84	1.17.21.28.87 87.88	1.17.21.81.81 21.81	6.30.20.21 21.30.41.72 35.84
87	14	2	28
10.0.0.8	1.17.21.81.87 90.91	18.41.45.50 31.81	1.5.5.78 84.87.90.91 98
32	98	55	61
10.32.43.55 61.63.78.72 78.98	1.17.21.81.87 87.88	1.2.16.37.81 78.90.91	6.30.20.21 21.30.41.72 35.84
81	31	43	12
			6.30.20.21 21.30.41.72 35.84
			6.30.20.21 21.30.41.72 35.84
			20.21.28.41 30.31.50.61 72.78.81.82
			20
			A: 76
			84

Figure 4: An entire network's location server organization. Each node is shown with the list of nodes for which it has up to date location information; B's location servers are shown in bold. Two possible queries by A for B's location are shown.

Mobility Enhancements

- A mobile node updates its location server proportionally to its distance traveled.
 - Updates order- i location servers after each movement of $2^{i-2}d$ (d is a specified threshold.)
- Registrations will eventually timeout in location servers.
- Piggyback location information on data packets to allow the nodes to know how to locate each other even when both are moving.
- Nodes leave a forwarding pointer in an order- i square when they leave it.

Performance Analysis

- All the results were supporting/positive towards this service (of course.)
- It used a uniform distribution of nodes with 100 nodes per km².
- Order-1 square was small enough for all nodes in it to be within each others range.
- Random waypoint w/o pause.
- No entry/exit of nodes?

Questions

- Are mobile nodes in reality uniformly distributed or even close? (If nodes are not uniformly distributed, a node in a certain square may have to end up routing/tracking location information for a large number of nodes in sibling squares.)
- How does a node know what order squares it belongs to (is the global partitioning of the world feasible?)
- Is it acceptable to assume that all nodes can determine its own geographic location (through GPS?)
- What about nodes that do not need to or are not willing to accept connections (don't register with location servers?)
- The number of location server hops to resolve a location is not great, but is the number of intermediate node hops excessive?