CarNet/Grid: Scalable Ad-Hoc Geographic Routing

Robert Morris
MIT / LCS

Acknowledgments to Robert Morris for slides
Ad-Hoc Nets: The Dream

Nodes forward each others’ packets.
No infrastructure; easy to deploy; fault tolerant.
Short hops are good for power and spectrum.
Can it be made to work?
Flooding-based on-demand routing works best in small nets. Can we route without global topology knowledge?
Geographic Forwarding Scales Well

- Assume each node knows its geographic location.

- A addresses a packet to G’s latitude, longitude
- C needs to know its immediate neighbors to forward to G
- **Geographic forwarding needs a location service!**
Possible Designs for a Location Service

• Flood to get a node’s location (LAR, DREAM).
  • 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 / partitioned

• 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.
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 Node Identifiers

- Each Grid node has a unique identifier.
  - Identifiers are numbers.
  - Perhaps a hash of the node’s IP 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
3 Servers Per Node Per Level

- $s$ 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.

location query path
GLS Update (level 0)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>11</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>6</td>
<td>23</td>
<td>16</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>19</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>17</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>19</td>
</tr>
</tbody>
</table>

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
Invariant (for all levels):
For node \( n \) in a square, \( n \)'s successor in each sibling square “knows” about \( n \).
GLS Update (level 1)

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>11</th>
<th>9</th>
<th>3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11, 2</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>6</td>
<td>16</td>
<td>29</td>
<td>...</td>
<td>7</td>
</tr>
<tr>
<td>23, 2</td>
<td>6</td>
<td>16</td>
<td>29</td>
<td>...</td>
<td>7</td>
</tr>
</tbody>
</table>
|      | 26 | 17 | 25 | 5 | ...
|      | 21 | 4  | 8  | 19 |...|

Invariant (for all levels): For node \( n \) in a square, \( n \)'s successor in each sibling square “knows” about \( n \).
GLS Update (level 2)

Invariant (for all levels): For node \( n \) in a square, \( n \)'s successor in each sibling square “knows” about \( n \).
GLS Query

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>11, 2</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23, 2</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**location table content**

query from 23 for 1
Challenges for GLS in a Mobile Network

• Slow updates risk out-of-date information.
  • Packets dropped because we can’t find dest.

• Aggressive updates risk congestion.
  • Update packets leave no bandwidth for data.
Performance Analysis

• How well does GLS cope with mobility?
• How scalable is GLS?
Simulation Environment

• Simulations using ns with CMU’s wireless extension (IEEE 802.11)

• Mobility Model:
  • random way-point with speed 0-10 m/s (22 mph)

• Area of square universe grows with the number of nodes in the network.
  • Achieve spatial reuse of the spectrum
• GLS level-0 square is 250m x 250m
• 300 seconds per simulation
GLS Finds Nodes in Big Mobile Networks

Failed queries are not retransmitted in this simulation

Queries fail because of out-of-date information for destination nodes or intermediate servers

Biggest network simulated: 600 nodes, 2900x2900m (4-level grid hierarchy)
GLS Protocol Overhead Grows Slowly

- Protocol packets include: GLS update, GLS query/reply
• Geographic forwarding is less fragile than source routing.
• DSR queries use too much b/w with > 300 nodes.
Grid Deployment

• Deployed 16 nodes with partial software.
  • 12 fixed relays.
  • 4 Compaq iPaq handhelds.
• Linux, 802.11 radios.
• Aiming for campus-wide deployment.
• Most nodes “sleep,” waking every second to poll for packets.
• “W” nodes stay awake to form low-delay backbone.
• Works well with high node densities.
• Algorithm: wake only to connect two neighbors.
In Progress: Geographic “Hole” Avoidance

- Node C has no neighbor closer than it to H.
  - There’s a “hole” between C and H.
- Use right-hand rule to traverse perimeter?
- Pick a random intermediate point?
Grid Summary

- Grid routing protocols are
  - Self-configuring.
  - Easy to deploy.
  - Scalable.

http://www.pdos.lcs.mit.edu/grid
Rumor Routing in Sensor Networks

David Braginsky and Deborah Estrin

LECS – UCLA

Acknowledgments to Sugata Hazarika for the slides
Sensor Data Routing

• Several sensor based applications
  • Some require frequent querying
  • Or unsure where events occur
    • Pull mechanism (google map sensors)

• Some require frequent event broadcast
  • Push mechanism (earthquake monitoring)
Issues

- Query flooding
  - Expensive for high query/event ratio
  - Allows for optimal reverse path setup
  - Gossiping scheme possible to reduce overhead

- Event Flooding
  - Expensive for low query/event ratio

- Both provide shortest delay paths
  - But can have high energy cost
Rumor Routing

- Designed for query/event ratios between query and event flooding

  - Motivation
    - Sometimes a non-optimal route is satisfactory

  - Advantages
    - Tunable best effort delivery
    - Tunable for a range of query/event ratios

  - Disadvantages
    - Optimal parameters depend heavily on topology
    - Does not guarantee delivery
Rumor Routing
Basis for Algorithm

- Observation: Two lines in a bounded rectangle have a 69% chance of intersecting.

- Create a set of straight line gradients from event, then send query along a random straight line from source.

- Thought: Can this bound be proved for a random walk. What is this bound if the line is not really straight?
Creating Paths

- Nodes having observed an event send out agents which leave routing info to the event as state in nodes
- Agents attempt to travel in a straight line
- If an agent crosses a path to another event, it begins to build the path to both
- Agent also optimizes paths if they find shorter ones.
Algorithm Basics

• All nodes maintain a neighbor list.
• Nodes also maintain a event table
  • When it observes event:
    • event is added with distance 0.

• Agents
  • Packets that carry event info across network.
  • Aggregate events as they go.
Agents

Figure 5

Figure 6
Agent Path

• Agent tries to travel “somewhat” straight
  • Maintains a list of recently seen nodes.

• When it arrives at a node adds the node’s neighbors to the list.

• For the next tries to find a node not in the recently seen list.

• Avoids loops
Forwarding Queries

• Forwarding Rules:
  • If a node has seen the query before, it is sent to a random neighbor
  • If a node has a route to the event, forward to neighbor along the route
  • Otherwise, forward to random neighbor using straightening algorithm
Simulation Results

- Assume that undelivered queries are flooded
- Wide range of parameters allow for energy saving over either of the naïve alternatives
- Optimal parameters depend on network topology, query/event distribution and frequency
- Algorithm was very sensitive to event distribution

![Graph showing simulation results for 10 Events, 4000 Nodes with Query Flooding and Event Flooding lines, and labels A=28, Lq=500, Lq=1000 and A=52, Lq=100, Lq=2000.]
Fault Tolerance

- After agents propagated paths to events, some nodes were disabled.
- Delivery probability degraded linearly up to 20% node failure, then dropped sharply.
- Both random and clustered failure were simulated with similar results.
Some Thoughts

- The effect of event distribution on the results is not clear.
- The straightening algorithm used is essentially only a random walk … can something better be done.
- The tuning of parameters for different network sizes and different node densities is not clear.
- There are no clear guidelines for parameter tuning, only simulation results in a particular environment.
Questions?
Let’s Talk about Projects

Email me description
Deadline: Feb 22, Thursday.
Groups

- Brian, Ashwin, Roman: senor network on maps
- Pradeep, Thilee: model based suppression
- Ola, Soji, Tom: DTN
- Michael, Kunal: ??
- TingYu, Gary, Yuanchi: intrusion detection in SN
- Ian, William: DTN gossip
- Wayne, Tray: beam overlap not harmful
- Deepak, Karthik, Boyeum: ??
- Tong: DTN ??
- Shawn, Simrat: ??
Thanks!
Can we analyze

• The inherent concept looks powerful.
• Even though not presented in this way … this algorithm is just an example of gossip routing.
• There are two types of gossip, gossip of events and gossip of queries.
• With the same gossip probability = 1/number of neighbors. (change this, would that help)
• It maybe possible to find the probability of intersection of these two.
• That might lead to a set of techniques for parameter estimation, or an optimal setting.
Other similar algos.

- Content based pub/sub.
  - Both the subscription and notification meet inside the network.
  - Can we borrow some ideas from wired networks
- DHT
  - DHTs can also be used to locate events.
  - Underlying routing is the problem. DHT over DSR or AODV may not be suitable.
Future Work

- Network dynamics
- Realistic environment
- Non-localized Events
- Asynchronous Events
- Self-tuning algorithm dynamics