GPSR

Brad Karp and H. T. Kung

GPSR: Greedy Perimeter Stateless Routing for Wireless Networks.


-Shaan Mahbubani
Objective

- Routing algorithm
- Network:
  - Dynamic
  - Multi hop
  - Wireless
- Scalable
Why a new routing algorithm?

- **General problems**
  - Others (e.g. link state) require entire network topology
  - Small inaccuracies result in loops/disconnections

- **Scalability:**
  - Shortest path (DV, LS): requires state proportional to number of reachable destinations
  - On demand: requires state proportional to number of destinations sent to
Scalability

- Dominant factors
  - Rate of topology change
  - Number of routers

- Current solutions
  - Hierarchy: AS, inter domain, intra domain
  - Caching: DSR, AODV

- Proposed solution
  - Geography
Geographic routing

- **Minimal state:**
  - Only need to know about neighbors a single hop away

- **Minimal information propagation**
  - Only need to update locations of neighbors a single hop away

- **Assumptions (problems with these?):**
  - All nodes know their own location (GPS)
  - All nodes know their destination locations
Greedy Forwarding

- Destination's geographical location known
- Repeatedly forward to locally greedy optimal next hop
- Beaconing:
  - Active traffic to realize neighbor's positions
  - Overhead?

*Figure 1: Greedy forwarding example. y is x’s closest neighbor to D.*
Greedy Forwarding Problem

- Next hop may be geographically further from destination
- Need to augment greedy forwarding with a way of routing through known neighbors to get to destination

Figure 2: Greedy forwarding failure. \( x \) is a local maximum in its geographic proximity to \( D \); \( w \) and \( y \) are farther from \( D \).
Perimeters

- Void: empty intersection of x's and D's neighbors

- Navigate around the void via perimeter

- Right hand rule: next edge to traverse is sequentially counter clockwise
Find a perimeter

- Eliminate edges to find a perimeter
- Construct planar graph (no two edges cross)
- Relative Neighborhood Graph (RNG), subset of:
- Gabriel Graph (GG)

*Figure 5: The RNG graph. For edge \((u,v)\) to be included, the shaded lune must contain no witness \(w\).*
GPSR

- Try to forward using greedy

- If no closer neighbor, send via perimeter

- Return to using greedy as soon as possible (perimeter used to recover from greedy failure)

- If forwarding via perimeter and get a loop, destination unreachable
Figure 8: Perimeter Forwarding Example. $D$ is the destination; $x$ is the node where the packet enters perimeter mode; forwarding hops are solid arrows; the line $xD$ is dashed.
Simulation

- Emulates “protocol comparison” paper:
  - ns-2 from DSR
  - 6 random motion patterns using Random Waypoint Model, 20 m/s
  - 50, 112, 200 nodes
  - 0, 30, 60, 120 second pause times
  - 30 constant bit rate flows, 22 originating nodes

- Did not include a distribution location database
  - Sources already know destination positions!
Results: Packet Delivery Success Rate

- Only connected destinations
  - At time of sending, or time of attempted delivery?

- What about latency per packet?

- Why the dip?

Figure 9: Packet Delivery Success Rate. GPSR with varying beacon intervals, B, compared with DSR. 50 nodes.
Results: Routing Protocol Overhead

- GPSR beacons are pro-active: constant overhead
- DSR routing overhead is reactive: traffic increases with mobility

Figure 10: Routing Protocol Overhead. Total routing protocol packets sent network-wide during the simulation for GPSR with varying beacon intervals, B, compared with DSR. 50 nodes.
Results: Path Length

- # of hops beyond ideal (0 is ideal)
- How did they measure shortest path length?
- What about latency per path?

Figure 11: Path length beyond optimal for GPSR’s and DSR’s successfully delivered packets. 50 nodes.
Results: Effect of Network Diameter

- Evaluate scaling
- GPSR: linear in node count
- DSR: route cache becomes full
  - what if they increased it?
  - Better delivery rate? More overhead?
Location Database Overhead

- One time lookup, at beginning of connection
  - Thereafter, stamped on data packets
- Like a DNS lookup?
- Only originating nodes must query, and only destination nodes must update
- Some networks (e.g. sensor), some nodes may have a known location
- More applicable to specific types of network?
Evaluation

• Seems like a great idea: we know exactly where to send a node, but not how; so send it and the network will greedily get it there.

• How do we know where a node is?
  – Overhead/latency of GPS lookup?
  – Accuracy in a high mobility network?

• They mention minimizing per node state: what about the feasibility of each node knowing it's own GPS location?