

## 1 Papers

- Johnson, D., and Maltz, D., Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile Computing, Vol. 353, Kluwer Academic Publishers, 1996.
- Hu, Y., and Johnson, D., Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000), pp:231-242, Boston, MA, USA, 2000

These papers address the problem of routing in a wireless network. The first paper proposes a base routing algorithm with numerous enhancements and variations possible. The second paper explores how varying the route cache effects performance.

## 2 Previous methods and drawbacks

### Distance Vector Link State Routing

Previous methods have relied on nodes periodically broadcasting their state or change of state. The periodic state broadcast allow for failed nodes, failed links, and newly created links to be detected by all nodes in the network. The later two can also trigger broadcasts to keep the network as up to date as possible. Typically nodes store the information and use it to make local packet routing decisions.

The above has a couple of drawbacks in terms of resources used. Nodes must periodically send out update messages even when there is no traffic on the network. Energy and bandwidth are used even for changes that could be very short lived.

To address this Johnson and Maltz proposed Dynamic Source Routing. Each node maintained its own view of the network. When it wishes to transmit, it gives the entire path for the packet to take in the header. Only when a node has packets for another node with an unknown location does any route information updating occur.

Assumptions about the network made in this paper are:

- The network has a low diameter
- The host in the network are slow moving. This means the topology does change very quickly.

- Node in the network can run their receivers in a promiscuous mode.
- Links need not be bidirectional.

The routing can be divided into two parts.

- Route Discovery
- Route Maintenance

Route discovery works by a node sending out a Route Request Packet which has fields identifying the node and giving a request ID. The packet floods the network in order to find the destination. If a second copy of a route discovery packet reaches a node, it is discarded. When the destination node receives a Route Request Packet it sends a Route Reply Packet to the source listing the path taken by the route discovery packet. If the destination does not know the location of the source, it sends out a Route Request Packet to find the path to the original node. The Route Reply Packet is piggy backed onto the second Route Request Packet to avoid a circular loop.

Route Maintenance is carried out by detecting and reporting link failures along a path so new ones can be established. There are a number of ways to detect link faults. If the MAC protocol in use requires acknowledgements, this suffices to detect a broken link. It may also be possible to do passive acknowledgment, that is each node listens to see if it can overhear the next hop node retransmitting the message. As neither of these will work when the link is asymmetric, higher level protocol mechanisms can be used. An explicit acknowledgment could be requested at the link granularity or application level. Such an acknowledgement would be above the link layer and hence not require symmetric links for an acknowledgement to succeed.

A number of optimizations are given. A node can learn new nodes by snooping on packets it relays. Examining the packet header a node can learn paths to all nodes between itself and the nodes that will be forwarding or receiving this message. Nodes that know a path to the destination node can reply to a route request packet using information in their caches. If several nodes are in this condition, random delay plus snooping can lower blocking probability and keep length as low as possible. A cache could also store negative information so a node can ignore route reply packets with stale data.

Class discussion: An additional optimization was suggested by the class. If an error report packet is forwarded by a node with a cache showing how to reach the destination without using the failed link, it could send this information along with the error report.

The second paper examines the issue of what type of data structure to use for the cache.

The trade offs involved in these structures are storage required, computation required to find a route, and accuracy in representing network state. If information is removed in the cache to make room for new information, it is possible that the removed information may need to be rediscovered by sending out route request packets. On the other hand, the newly found route maybe a shorter route formed after the last route request for the destination. Another issue is that the longer information stays in the cache, the chances that it is no longer accurate increase. Thus using old information may result in increase overhead due to route error packets.

The second paper examines a number of cache strategies and movement patterns for the network nodes. The caching strategies fall into two categories, those that store links and those that store paths. Link caching strategies included links never expire, links expire after 5 seconds, links expire time is dependent on the stability of the nodes its incident on, links expire based on stability and path calculated uses the route with the longest lifetime. Node stability is calculated at runtime based and is based on the number of times a link is used, and the number of times its observed to fail. Also included with the link caches was a cache where nodes always had perfect information about the network state provided by the simulator. Path caches included a cache with infinite capacity, a couple caches that used a FIFO replacement policy, a couple that had the cache divided into a pair of FIFOs one of which held routes that had been user or the discovered directly while the second FIFO held all other routes.

Movement patterns examined included Brownian motion, where a random angle and a velocity in a fixed range were select and changed every time interval. Column motion, where nodes could move only along one direction on each axis. Random Gauss-Markov Motion, where the velocity randomly changes and the changes are governed by expressions given. Random Waypoint Motion, where a node selects a destination and speed, travels there and after a pause repeats the process. Pursue Motion, where nodes are paired with one node in a pair using random waypoint motion and the other guessing the position at the end of the time interval and moving there.

Simulations evaluated the performance of the caching strategies under each type of motion. Evaluations were based on the packet delivery ratio, overhead, latency, and path optimality.