*A Survey on Position-Based Routing in*

*Mobile Ad Hoc Networks*

In recent years the widespread availability of wireless communication and handheld devices has stimulated research on self-organizing networks that do not require a pre-established infrastructure. These ad hoc networks, as they are commonly called, consist of autonomous nodes that collaborate in order to transport information. Usually, these nodes act as end systems and routers at the same time.

Ad hoc networks can be subdivided into two classes: Static and Mobile

1. Static:

the network doesn’t change example rooftop network.

1. Mobile:

the systems may move arbitrarily without prior notice, example handheld device.

In Mobile there are two different approaches: Topology-based and Position-based routing

1. Topology-based:

It uses the information about the links that exist in the network to perform packet forwarding.

They can be divided into proactive, reactive, and hybrid approaches

1. Proactive:

It employs classical routing strategies such as distance-vector or link-state routing.

They maintain routing information about the available paths in the network even if theses paths are not currently use.

The main drawback of these approaches is that the maintenance of unused paths may occupy a significant part of the available bandwidth if the topology changes frequently.

1. Reactive:

It maintains only the routes that are currently in use.

But they still have some inherent limitations

🡪 It is typically required to perform a route discovery before packets aca be exchanged between communication peers.

This lead to a delay for the first packet to be transmitted.

🡪 It may still generate a significant amount of network traffic when the topology of the network change frequently.

🡪 Packets en route to the destination are likely to be lost if the route to the destination changes.

1. Hybrid:

It combines local proactive routing and global reactive routing in order to achieve a higher of efficiency and scalability.

However, even a combination of both strategies still needs to maintain at least those network paths that are currently in use, limiting the amount of topological changes that can be tolerated within a given amount of time.

1. Position-based routing

Position-based routing algorithms eliminate some of the limitations of topology-based routing by using additional information.

They require that information about the physical position of the participating nodes be available. Commonly, each node determines its own position through the use of GPS or some other type of positioning service.

A location service is used by the sender of a packet to determine the position of the destination and to include it in the packet’s destination address.

The routing decision at each node is then based on the destination’s position contained in the packet and the position of the forwarding node’s neighbors.

Position-based routing thus does not require the establishment or maintenance of routes.

The nodes have neither to store routing tables nor to transmit messages to keep routing tables up to date. As a further advantage, position-based routing supports the delivery of packets to all nodes in a given geographic region in a natural way. This type of service is called geocasting.

In this article we present a survey of position-based routing for mobile ad hoc networks. We outline the main problems that have to be solved for this class of routing protocols and present the solutions currently available.

The remainder of this article is structured as follows:

We present the basic idea of position-based addressing and routing, and give criteria for a taxonomy of the various proposals.

We cover techniques for location services and outline position based forwarding strategies. A later section contains a qualitative comparison of the location services, and forwarding strategies.

We point out open issues and possible directions of future research, and then conclude the article.

*Basic Principles and Problems:*

Before a packet can be sent, it is necessary to determine the position of its destination.

Typically, a location service is responsible for this task. Existing location services can be classified

according to how many nodes host the service. This can be either some specific nodes or all nodes of the network.

Furthermore, each location server may maintain the position of some specific or all nodes in the network. We abbreviate the four possible combinations as some-for-some, some-for-all, all-forsome, and all-for-all in the discussion of location services.

In position-based routing, the forwarding decision by a node is primarily based on the position of a packet’s destination and the position of the node’s immediate one-hop neighbors.

The position of the destination is contained in the header of the packet. If a node happens to know a more accurate position of the destination, it may choose to update the position in the packet before forwarding it.

The position of the neighbors is typically learned through one-hop broadcasts.

These beacons are sent periodically by all nodes and contain the position of the sending node.

We can distinguish three main packet forwarding strategies for position-based routing:

greedy forwarding, restricted directional flooding, and hierarchical approaches.

For the first two, a node forwards a given packet to one (greedy forwarding) or more (restricted directional flooding) one-hop neighbors that are located closer to the destination than the forwarding node itself.

The selection of the neighbor in the greedy case depends on the optimization criteria of the

algorithm. We will present the diverse strategies that existing algorithms use to make this selection.

It is fairly obvious that both forwarding strategies may fail if there is no one-hop neighbor that is closer to the destination than the forwarding node itself. Recovery strategies that cope

with this kind of failure are also discussed in a later section.

The third forwarding strategy is to form a hierarchy in order to scale to a large number of mobile nodes. In this article we investigate two representatives of hierarchical routing that use

greedy forwarding for wide area routing and non-position based approaches for local area routing. Figure 1 illustrates the two building blocks

— location service and forwarding strategy

— required for position-based routing, together with classification criteria for the various existing approaches.



*Location Services:*

In order to learn the current position of a specific node, the help of a location service is needed.

Mobile nodes register their current position with the service.

When a node does not know the position of a desired communication partner, it contacts

the location service and requests that information. In classic cellular networks, there are dedicated position servers (with well-known addresses) that maintain position information about the nodes in the network.

With respect to the classification, this is a some-for-all approach since the servers are some specific nodes, each maintaining position information about all mobile nodes.

In mobile ad hoc networks, such a centralized approach is viable only as an external service that can be reached via non ad-hoc means. There are two main reasons for this. First, it would be difficult to obtain the position of a position server if the server were part of the ad hoc network itself. This would represent a chicken-and-egg problem: without a position server, it is not possible to get position information, but without the position information the server cannot be reached. Second, since an ad hoc network is dynamic, it might be difficult to guarantee that at least one position server will be present in a given ad hoc network.

In the following we concentrate on decentralized location services that are part of the ad hoc network.

*Distance Routing Effect Algorithm for Mobility*

Within the Distance Routing Effect Algorithm for Mobility (DREAM) framework [16], each node maintains a position database that stores position information about each other node that is part of the network. It can therefore be classified as an all-for-all approach.

An entry in the position database includes a node identifier, the direction of and distance to the node, as well as a time value that indicates when this information was generated. Of course, the accuracy of such an entry depends on its age.

Each node regularly floods packets to update the position information maintained by the other nodes. A node can control the accuracy of its position information available to other nodes by:

• The frequency at which it sends position updates (temporal resolution)

• Indicating how far a position update may travel before it is discarded (spatial resolution)

The temporal resolution of sending updates is coupled with the mobility rate of a node (i.e., the higher the speed, the more frequent the updates). The spatial resolution is used to provide accurate position information in the direct neighborhood of a node and less accurate information at nodes farther away. The costs associated with accurate position information at very remote nodes can be reduced since, as the authors argue, “the greater the distance separating two nodes, the slower they appear to be moving with respect to each other” (termed the distance effect [17]). An example of this “distance effect” is given in Fig. 2. Assume that in this example node A is not moving, while nodes B and C are moving in the same direction at the same speed. From node A’s perspective, the change in direction will be greater for node B than for node C. The distance effect allows low spatial resolution in areas far away from the target node, provided that intermediate hops are able to update the position information contained in the packet.



*Quorum-Based Location Service*

The concept of quorum systems is well known from information replication in databases and distributed systems. Information updates (write operations) are sent to a subset (quorum) of available nodes, and information requests (read operations) are referred to a potentially different subset.

When these subsets are designed such that their intersection is nonempty, it is ensured that an up-to-date version of the sought-after information can always be found.

In [18], this scheme is used to develop a location service for ad hoc networks. We will discuss it by means of the simple sample network shown in Fig. 3. A subset of all mobile nodes is chosen to host position databases; in the example, these are nodes 1–6.

A virtual backbone is constructed between the nodes of the subset, using a non-position-based ad hoc routing mechanism.

A mobile node sends position update messages to the nearest backbone node, which then chooses a quorum of backbone nodes to host the position information. Thus, node D sends its updates to node 6, which might then select quorum A with the nodes 1, 2, and 6 to host the information. When a node S wants to obtain the position information, it sends a query to the nearest backbone node, which in turn contacts the nodes of a (usually different) quorum. Node 4 might, for example, choose quorum B, consisting of nodes 4, 5, and 6, for the query. Since by definition the intersection of two quorums is non-empty, the querying node is guaranteed to obtain at least one response with the desired position information.



It is important to timestamp position updates, since some nodes in the queried quorum might have been in the quorum of previous updates and would then report outdated position information. If several responses are received, the one representing the most current position update is chosen.

An important aspect of quorum-based position services is the following trade-off: the larger the quorum sets, the higher the cost for position updates and queries, but also the larger the number of nodes in the intersection of two quorums, which improves resilience against unreachable backbone nodes.

In [18] several methods on how to generate quorum systems with the desired properties are discussed. In the article, the authors also show that the size of the quorum can be kept independent of the number of nodes by dividing the nodes into sub-sets of a constant size. An individual virtual backbone is constructed for each of these subsets.

The quorum-based position service can be configured to operate as all-for-all, all-for-some, or some-for-some approach, depending on how the size of the backbone and the quorum is chosen. However, it will typically work as a some-for-some scheme with the backbone being a small subset of all available nodes and a quorum being a small subset of the backbone nodes.

Other work based on quorums is presented in [19]. Here, position information for the nodes is propagated in a north south direction. Whenever a node whose position is unknown has to be contacted, position information is searched in east west direction until the information is found. While the algorithm described is still at an early stage, it is an interesting idea worth further study.

*Grid Location Service*

The Grid Location Service (GLS) [20, 21] is part of the Grid project [22]. It divides the area that contains the ad hoc network into a hierarchy of squares. In this hierarchy, n-order squares contain exactly four (n– 1) order squares, forming a so called quad-tree. Each node maintains a table of all other nodes within the local first-order square. The table is constructed with the help of periodic position broadcasts scoped to the area of the first-order square.



Again, we demonstrate the mechanism by means of a simple example (Fig. 4). To determine where to store position information, GLS establishes a notion of near node IDs, defined as the least ID greater than a node’s own ID. When node 10 in the figure wants to distribute its position information, it sends position updates to the respective node with the nearest ID in each of the three surrounding first-order squares. Thus, the position information is available at the nodes 15, 18, 73 and at all nodes that are in the same first-order square as 10 itself. In the surrounding three second-order squares, again the nodes with the nearest ID are chosen to host the node’s position; in the example these are nodes 14, 25, and 29. This process is repeated until the area of the adhoc network has been covered. The density of position information for a given node thus decreases logarithmically with the distance from that node. Assume now that node 78 wants to obtain the position of node 10. It should therefore locate a nearby node that knows about the position of node 10. In the example this is node 29.

While node 78 does not know that node 29 holds the required position, it is able to discover this information. To see how this process works, it is useful to take a look at the position servers for node 29. Its position is stored in the three surrounding first-order squares at nodes 36, 43, and 64. Note that each of these nodes, as well as node 29, are automatically also the ones in their respective first-order square with the ID nearest to 10. Thus, there exists a “trail” of descending node IDs from each of the squares of all orders to the correct position server. Position queries for a node can now be directed to the node with the nearest ID of which the querying node knows. In our example this would be node 36. The node with the nearest ID does not necessarily know the node sought, but will know a node with a nearer node ID (node 29, which is already the sought-after position server). The process continues until a node that has the position information available is found.

Note that a node need not know the IDs of its position servers, which makes a bootstrapping mechanism to discover a node’s position servers unnecessary. Position information is forwarded to a certain position (e.g., the lower left corner) of each element in the quad tree. After reaching a node close to this position, the position information is forwarded progressively to nodes with closer IDs in a process resembling position queries. This ensures that the position information reaches the correct node, where it is then stored. Since GLS requires that all nodes store the information on-some other nodes, it can be classified as an all-for-some approach.