Chapter 1

ROUTING IN MOBILE AD HOC NETWORKS

Mahesh K. Marina and Samir R. Das
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

Efficient, dynamic routing is one of the key challenges in mobile ad hoc networks. In the recent past, this problem was addressed by many research efforts, resulting in a large body of literature. We survey various proposed approaches for routing in mobile ad hoc networks such as flooding, proactive, on-demand and geographic routing, and review representative protocols from each of these categories. We further conduct qualitative comparisons across various approaches. We also point out future research issues in the context of individual routing approaches as well as from the overall system perspective.

Keywords: Unicast routing, mobile ad hoc networks, multihop wireless networks, packet radio networks, flooding, proactive routing, on-demand routing, geographic routing.

1. Introduction

Developing support for routing is one of the most significant challenge in ad hoc networks and is critical for the basic network operations. Certain unique combinations of characteristics make routing in ad hoc networks interesting. First, nodes in an ad hoc network are allowed to move in an uncontrolled manner. Such node mobility results in a highly dynamic network with rapid topological changes causing frequent route failures. A good routing protocol for this network environment has to dynamically adapt to the changing network topology. Second, the underlying wireless channel provides much lower and more variable bandwidth than wired networks. The wireless channel working as a shared medium makes available bandwidth per node even lower. So routing protocols should be bandwidth-efficient by expending a minimal overhead for computing routes so that much of the remaining bandwidth is available for the actual data communication. Third, nodes run on batteries which have limited energy supply. In order for nodes to stay and communicate for longer periods, it is desirable that a routing protocol be energy-efficient as well. This also provides also another reason why overheads must be kept low. Thus, routing protocols must meet the conflicting goals of dynamic adaptation and low overhead to deliver good overall performance.

Routing protocols developed for wired networks such as the wired Internet are inadequate here as they not only assume mostly fixed topology but also have high overheads[1]. This has lead to several routing proposals specifically targeted for ad hoc networks. While some of these proposals are optimized variants of protocols originally designed for wired networks, the rest adopt new paradigms such as on-demand routing, where routes are maintained "reactively" only when needed. This is in contrast with the traditional, proactive Internet-based protocols. Other new paradigms also have emerged – for example, exploiting location information fro routing, and energy-efficient routing.

All our discussions here implicitly assume that underlying network topology can be viewed as an undirected graph. In practice, this assumption may not hold, since unidirectional links may be present. This commonly occurs when there is a difference in the transmit powers in the nodes of the network. Even in a perfectly homogeneous network, interference at the wireless channel can be spatially diverse, causing unidirectionality of links. However, there is both empirical [38] and theoretical [3] evidence showing that using unidirectional links for routing may not yield any substantial benefit. On the contrary, using such links is complex and may increase overheads. On the other hand, ignoring unidirectional links, when indeed present, is straightforward. It can be realized via simple two-way message exchanges between neighboring nodes. Many routing protocols ordinarily exchange such messages (often called "beacons" or "hello" messages) for the purpose of finding the neighbor node set (neighbor discovery).

Routing research in ad hoc networks is quite broad. In this chapter, we limit ourselves to unicast routing and associated techniques, and do not discuss multi-destination routing such as multicast or geocast.

[1]Routing protocols for satellite networks are also inadequate here as the topology in satellite networks is completely deterministic at any time even though nodes are moving.

Fundamental routing issues can be understood quite well by developing a good background on unicast routing issues and techniques. The rest of this chapter is organized as follows. In the next Section, we discuss flooding and a few efficient variants of basic flooding — flooding is not only a legitimate candidate for unicast routing in extremely mobile scenarios, but also is an integral part of several other routing protocols. In Section 3, we will review optimized variants of traditional distance vector and link state protocols tailored for ad hoc networks. Section 4 describes three prominent on-demand protocols along with some generic optimizations. In Section 5, we compare proactive and reactive approaches and also discuss hybrid approaches that attempt to combine the benefits of these two approaches. In Section 6, we discuss routing using geographic location information. We finally conclude in Section 7.

2. Flooding

Flooding (or network-wide broadcasting) is the simplest way to deliver data from a node to any other node in the network. In flooding, the source simply broadcasts the data packet to its neighboring nodes via a MAC layer broadcast mechanism. Each node hearing the broadcast *for the first time* re-broadcasts it. Thus, the broadcast propagates in "layers" outwards from the source, eventually terminating when every node has heard the packet and transmitted it once. The rule "every node transmit only once" guarantees termination of the procedure and also avoids looping. This can be achieved using unique identifiers on all packets being flooded. The flooding technique delivers the data to every node in the connected component of the network.

With flooding, no topological information needs to be maintained or known in advance. In network scenarios where node mobility is so high that a given unicast routing protocol may fail to keep up with the rate of topology changes, flooding may become the only alternative for routing data reasonably. However, in other scenarios where node mobility is trackable by a routing protocol, flooding can be a very inefficient option. This is because the total number of transmissions to deliver a single message to a destination with flooding is in the order of network size, as opposed to the network diameter with a unicast routing protocol (assuming that a route is already found).

Although flooding is not usually attractive for efficiently delivering data, it is still very useful in carrying out certain routing tasks such as route discovery and topology dissemination, and as a bootstrapping mechanism when nothing is known a priori about the network topology. Therefore, flooding appears as a key component in many routing protocols (OSPF [40] is a classic example).

In the simple flooding protocol as described above (also called *pure flooding*), each node transmits (broadcasts) the data once. As a result, a node may receive the same packet from several neighbors. Thus, depending on the network density, simple flooding may take far more transmissions than necessary for the flood to reach every node. Such redundancy can be eliminated to achieve less contention and collisions at the radio link layer, thus increasing network utilization. Several efficient alternatives have been proposed that use only a small subset of nodes to transmit the data packet during a flood, however ensure that all nodes in the network receive the packet.

Efficient Flooding Techniques

Efficient flooding essentially attempts to eliminate the redundant broadcasts, but still ensures that all nodes in the network receives the packet. In the simplest of all techniques, every node (other than the source) rebroadcasts the data packet only with a certain probability p [42, 17]. Clearly, the correct choice of the probability p determines the effectiveness of this technique — a very small value prevents the flood from reaching every node ("flood dying out" problem), while a very large value results in many redundant broadcasts. The right value of p depends on several factors, including the average node degree. Haas et. al. [17] evaluate several variants of this basic technique and show that with appropriate choice of p, that changes as the flood propagates away from the source, significant savings are possible without affecting the coverage of the flood (i.e., number of nodes receiving the flooded packet). Ideally, the flood should cover all nodes in the network. Determining the right value of p remains a hard problem.

Other techniques are also possible. For example, when a node hears the broadcast packet, it does not transmit it immediately, but waits for a brief period to see whether it hears the same packet again. If it does hear it multiple times (say, k times) within this period, it assumes that all its neighbors must have heard this packet, and refrains from transmitting it [42]. As before, determining suitable values for k and the waiting period becomes complex. This technique can be improved by incorporating neighborhood knowledge. For example, if each node knows its neighbor set and this set is included in each broadcast, then it is easy for a node to completely determine whether all its neighbors have heard this packet by computing the union of the neighbor sets transmitted in the packets it has heard. Still, how long a node should wait to hear all the broadcasts from its neighbors remains a question.

This problem of eliminating redundant broadcasts can be solved via a more algorithmic approach. The objective is to determine a small subset of nodes for broadcasting data such that every node in the network receives it. Often this subset is called the *forwarding set.* This problem is equivalent to finding a *Minimum Connected Dominating Set (MCDS)*[2]. In a Dominating Set (DS), a node is either designated as a dominator or is a neighbor of at least one dominator node. A Connected Dominating Set (CDS) is a DS such that the subgraph formed by considering only the dominator nodes (and edges among them) is connected. MCDS is a CDS of the smallest size. Even with full topology information, the MCDS problem is difficult and shown to be NP-hard [14]. Therefore, research efforts have focused on developing efficient centralized approximation algorithms [16] and distributed heuristics using only partial and local topology information [32, 54, 51, 46, 59].

Distributed heuristics for efficient flooding (also termed *neighborhood-knowledge techniques*) seek to find a small forwarding set without incurring too much overhead in the process. Some heuristics explicitly find a CDS [32, 54], while others do it implicitly [51, 46]. The implicit heuristics can be further classified into two categories: neighbor designating methods (e.g., [51]) and self-pruning (e.g., [46]). In neighbor designating nodes, the status of whether a node should be in the forward set is determined by its neighbors. On the other hand, each node determines its own status in self-pruning methods. These heuristics also differ in the time the forwarding set is computed. Some heuristics dynamically compute the forward node set (e.g., [46]) depending on the source of the flood and neighboring nodes that have already rebroadcasted, while other heuristics determine the forward node set statically (e.g., [51]) independent to any specific source.

Williams and Camp [58] have compared the performance of several efficient flooding techniques including the probabilistic flooding technique described earlier. They found that neighborhood-knowledge techniques in general perform better than probabilistic technique, especially in low and moderate mobility scenarios. The effectiveness of the neighborhood-knowledge methods reduces at high mobility because of inaccuracy in neighborhood information used in finding the forward node set. In fact, some amount of controlled redundancy in the forward node set is beneficial in coping with mobility and unreliability of broadcasts in some

[2]This problem is also closely related to maximum leaf spanning tree problem and a special case of minimum set cover problem

MAC protocols such as IEEE 802.11 DCF [24]. Recently, there is also some work on unifying different neighborhood-knowledge techniques into a single generic broadcast scheme by recognizing that they share similar features [59]. We will discuss just one scheme, called *Multipoint Relaying*, in some detail here to give a flavor of the available techniques. We have chosen this particular scheme since it is the key component of a routing protocol to be discussed later.

In Multipoint Relaying [51], the main idea is that each node selects a small subset of its neighbors as Multipoint Relays (MPRs) sufficient to cover its 2-hop neighborhood (Figure 1.1). When a node floods a packet, only the MPRs of the node rebroadcast the packet and their MPRs rebroadcast and so on. Nodes exchanges their list of neighbors via periodic "hello" packets. As a result, each node knows its 2-hop neighborhood information. Each node then locally computes its MPR set using the following heuristic because finding the minimum size MPR set is NP-hard. The node includes a neighbor in its MPR set if it is the only neighbor to reach a 2-hop neighbor. After including all such neighbors, the node picks a neighbor not already in the MPR set which can cover the most number of nodes that are uncovered by the current MPR set. The node repeats this last step until all 2-hop neighbors are covered by the MPR set. The node then informs the neighbors selected as MPRs via hello packets and it becomes the MPR-Selector for those neighbors. Every node is also responsible for updating its MPR set and notifying the corresponding neighbors whenever the neighborhood changes. It is also shown that MPR set computed by the above heuristic is within a $\log(n)$ bound of the optimal size set, where n is the network size.

3. Proactive Routing

Proactive protocols maintain unicast routes between all pairs of nodes regardless of whether all routes are actually used. Therefore, when the need arises (i.e., when a traffic source begins a session with a remote destination), the traffic source has a route readily available and does not have to incur any delay for route discovery. These protocols also can find optimal routes (shortest paths) given a model of link costs.

Routing protocols on the Internet (i.e, distance vector-based RIP [19] and link state-based OSPF [40]) fall under this category. However, these protocols are not directly suitable for resource-poor and mobile ad hoc networks because of their high overheads and/or somewhat poor convergence behavior. Therefore, several optimized variations of these protocols have been proposed for use in ad hoc networks. These protocols

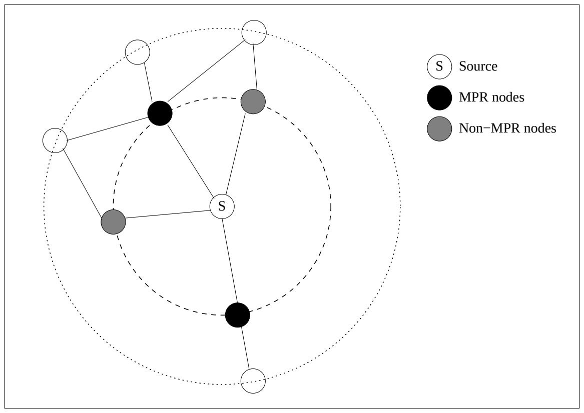


Figure 1.1. Multipoint Relay concept. Two dotted circles around the source S represent its logical 1-hop and 2-hop neighborhood respectively.

are broadly classified into the two traditional categories: distance vector and link state. In distance vector protocols, a node exchanges with its neighbors a vector containing the current distance information to all known destinations; the distance information propagates across the network transitively and routes are computed in a distributed manner at each node. On the other hand, in link state protocols, each node disseminates the status of each of its outgoing links throughout the network (typically via flooding) in the form of link state updates. Each node locally computes routes in a decentralized manner using the complete topology information. In the rest of this section, we describe two protocols from each of these categories that have received wide attention.

Distance Vector Protocols

Destination-Sequenced Distance-Vector (DSDV) [48] was one of the earliest protocols developed for ad hoc networks. Primarily design goal of DSDV was to develop a protocol that preserves the simplicity of RIP, while guaranteeing loop freedom. It is well known that Distributed Bellman-Ford (DBF) [2], the basic distance vector protocol, suffers from both short-term and long-term routing loops (the *counting-to-infinity* problem) and thus exhibits poor convergence in the presence of link failures. Note that RIP is DBF with the addition of two ad hoc techniques (split-horizon and poisoned-reverse) to prevent two hop loops. The variants of DBF proposed to prevent loops (Merlin-Segall [39], Jaffe-Moss [25], and DUAL [13]), however, involve complex inter-nodal coordination. Because of inter-nodal coordination, the overheads of these proposals are much higher than basic DBF and match that of link-state protocols using flooding to disseminate link-state updates; so, these protocols are effective only when topology changes are rare.

The main idea in DSDV is the use of destination sequence numbers to achieve loop freedom without any inter-nodal coordination. Every node maintains a monotonically increasing sequence number for itself. It also maintains the highest known sequence number for each destination in the routing table (called "destination sequence numbers"). The distance/metric information for every destination, typically exchanged via routing updates among neighbors in distance-vector protocols, is tagged with the corresponding destination sequence number. These sequence numbers are used to determine the relative freshness of distance information generated by two nodes for the same destination (the node with a higher destination sequence number has the more recent information). Routing loops are prevented by maintaining an invariant that destination sequence numbers along any valid route monotonically increase toward the destination.

DSDV also uses triggered incremental routing updates between periodic full updates to quickly propagate information about route changes. In DSDV, like in DBF, a node may receive a route with a longer hop count earlier than the one with the smallest hop count. Therefore, always propagating distance information immediately upon change can trigger many updates that will ripple through the network, resulting in a huge overhead. So, DSDV estimates route settling time (time it takes to get the route with the shortest distance after getting the route with a higher distance) based on past history and uses it to avoid propagating every improvement in distance information.

Wireless Routing Protocol (WRP) [41] is another distance vector protocol optimized for ad hoc networks. WRP belongs to a class of distance vector protocols called *path finding algorithms.* The algorithms of this class use the next hop and second-to-last hop information to overcome the counting-to-infinity problem; this information is sufficient to locally determine the shortest path spanning tree at each node. In these algorithms, every node is updated with the shortest path spanning tree of each of its neighbors. Each node uses the cost of its adjacent links along with shortest path trees reported by neighbors to update its own shortest path tree; the node reports changes to its own shortest path tree to all the neighbors in the form of updates containing distance and second-to-last hop information to each destination.

Path finding algorithms originally proposed for the Internet (e.g., [8]) suffer from temporary routing loops even though they prevent the counting-to-infinity problem. This happens because these algorithms fail to recognize that updates received from different neighbors may not agree on the second-to-last hop to a destination. WRP improves on the earlier algorithms by verifying the consistency of second-to-last hop reported by all neighbors. With this mechanism, WRP reduces the possibility of temporary routing loops, which in turn results in faster convergence time. One major drawback of WRP is its requirement for reliable and ordered delivery of routing messages.

Link State Protocols

Optimized Link State Routing (OLSR) [9] is an optimized version of traditional link state protocol such as OSPF. It uses the concept of Multipoint Relays (MPRs), discussed in the previous section, to efficiently disseminate link state updates across the network. Only the nodes selected as MPRs by some node are allowed to generate link state updates. Moreover, link state updates contain only the links between MPR nodes and their MPR-Selectors in order to keep the update size small. Thus, only partial topology information is made available at each node. However, this information is sufficient for each to locally compute shortest hop path to every other node because at least one such path consists of only MPR nodes.

OLSR uses only periodic updates for link state dissemination. Since the total overhead is then determined by the product of number of nodes generating the updates, number of nodes forwarding each update and the size of each update, OLSR reduces the overhead compared to a base link state protocol when the network is dense. For a sparse network, OLSR degenerates to traditional link state protocol. Finally, using only periodic updates makes the choice of update interval critical in reacting to topology changes.

Topology Broadcast based on Reverse-Path Forwarding (TBRPF) [43] is a partial topology link state protocol where each node has only partial view of the whole network topology, but sufficient to compute a shortest path source spanning tree rooted at the node. When a node obtains source trees maintained at neighboring nodes, it can update its own shortest path tree. This idea is somewhat similar to that in path finding algorithms such as WRP discussed above. TBRPF exploits an additional fact that shortest path trees reported by neighbors can have a large overlap. A node can still compute its shortest path tree even if it receives partial trees from each of its neighbors as long as they minimally overlap. Thus, every node reports only a part of its source tree (called Reported Tree (RT)) to all neighbors in an attempt to reduce the size of topology updates. A node uses periodic topology updates to inform its complete RT to all neighbors at longer intervals, while it uses differential updates to inform them about the changes to its RT more frequently.

In order to compute RT, a node X first determines a Reported Node (RN) set. RN contains itself (node X) and each neighbor Y for which X is on the shortest path to Y from another neighbor. RN so computed contains X and a subset (possibly empty) of its neighbors. For each neighbor Y included in RN, X acts as a forwarding node for data destined to Y. Finally, X also includes in RN all nodes which can be reached by a shortest path via one of its neighbors already in RN. Once X completes computing RN as stated above, the set of all links (u, v) such that $u \in$ RN, constitute the RT of X. Note that RT only specifies the minimum amount of topology that a node must report to its neighbors. To obtain some redundancy in the topology maintained at each node (e.g., a subgraph more connected than a tree), nodes can report more topology than RT.

TBRPF also employs an efficient neighbor discovery mechanism using differential hellos for nodes to determine their bidirectional neighbors. This mechanism reduces the size of hello messages by avoiding the need to include every neighbor in each hello message.

Performance of Proactive Protocols

Among the proactive protocols we have discussed, DSDV seems to suffer from poor responsiveness to topology changes and slow convergence to optimal paths. This is mainly because of the transitive nature of topology updates in distance vector protocols. Simulation results [5, 26] also confirm this behavior. Although reducing the update intervals appears to improve its responsiveness, it might also proportionately increase the overhead leading to congestion. WRP, the other distance vector protocol we have discussed, assumes reliable and in-order delivery of routing control packets which is an unreasonable requirement in error-prone wireless networks. The performance of the protocol when this assumption does not hold is unclear. As far as the two link state protocols — OLSR and TBRPF — are concerned, both of them share some features such as being partial topology protocols. However, the details of the protocols are quite different. Whereas OLSR is more like a traditional link state protocol with optimizations to reduce overhead in ad hoc networks, TBRPF is a link state variant based on tree sharing concept. TBRPF also has one desirable feature of using frequent incremental updates in addition to periodic, less frequent full updates. This feature will likely improve responsiveness to topology changes. OLSR, on the other hand, relies solely on periodic full updates. Although in our knowledge there is no comprehensive study focusing on relative performance of OLSR and TBRPF, they expected to show comparable performance (and likely better than their distance vector counterparts).

4. On-demand Routing

On-demand (reactive) routing presents an interesting and significant departure from the traditional proactive approach. Main idea in on-demand routing is to find and maintain *only needed* routes. Recall that proactive routing protocols maintain all routes without regard to their ultimate use. The obvious advantage with discovering routes on-demand is to avoid incurring the cost of maintaining routes that are not used. This approach is attractive when the network traffic is sporadic, bursty and directed mostly toward a small subset of nodes. However, since routes are created when the need arises, data packets experience queuing delays at the source while the route is being found at session initiation and when route is being repaired later on after a failure. Another, not so obvious consequence of on-demand routing is that routes may become suboptimal, as time progresses since with a pure on-demand protocol a route is used until it fails. In the rest of this Section, we describe three well-known on-demand protocols and follow them up with some generic set of optimizations that can benefit any on-demand protocol.

Protocols for On-Demand Routing

Dynamic Source Routing (DSR) [27, 28] is characterized by the use of *source routing.* That is, the sender knows the complete hop-by-hop route to the destination. These routes are stored in a *route cache.* The data packets carry the source route in the packet header.

When a node in the ad hoc network attempts to send a data packet to a destination for which it does not already know the route, it uses a *route discovery* process to dynamically determine such a route. Route discovery works by *flooding* the network with *route request* (also called *query*) packets. Each node receiving a request, rebroadcasts it, unless it is the destination or it has a route to the destination in its route cache. Such a node replies to the request with a *route reply* packet that is routed back to the original source. Route request and reply packets are also source routed. The request builds up the path traversed so far. The reply routes itself back to the source by traversing this path backward. The route carried back by the reply packet is cached at the source for future use. If any link on a source route is broken (detected by the failure of an attempted data transmission over a link, for example), a *route error* packet is generated. Route error is sent back toward the source which erases all entries in the route caches along the path that contains the broken link. A new route discovery must be initiated by the source, if this route is still needed and no alternate route is found in the cache.

DSR makes aggressive use of source routing and route caching. With source routing, complete path information is available and routing loops can be easily detected and eliminated without requiring any special mechanism. Because route requests and replies are both source routed, the source and destination, in addition to learning routes to each other, can also learn and cache routes to all intermediate nodes. Also, any forwarding node caches any source route in a packet it forwards for possible future use. DSR employs several optimizations including promiscuous listening which allows nodes that are not participating in forwarding to overhear on-going data transmissions nearby to learn different routes free of cost. To take full advantage of route caching, DSR replies to *all* requests reaching a destination from a single request cycle. Thus the source learns many alternate routes to the destination, which will be useful in the case that the primary (shortest) route fails. Having access to many alternate routes saves route discovery floods, which is often a performance bottleneck. This may, however, result in route reply flood unless care is taken.

However, aggressive use of route caching comes with a penalty. Basic DSR protocol lacks effective mechanisms to purge stale routes. Use of stale routes not only wastes precious network bandwidth for packets that are eventually dropped, but also causes cache pollution at other nodes when they forward/overhear stale routes. Several performance studies [20, 50] have shown that stale caches can significantly hurt performance especially at high mobility and/or high loads. These results have motivated subsequent work on improved caching strategies for DSR [21, 37, 23]. Besides stale cache problems, the use of source routes in data packets increases the byte overhead of DSR. This limitation was addressed in a later work by the DSR designers [22].

Ad hoc On-demand Distance Vector (AODV) [49, 47] shares DSR's on-demand characteristics in that it also discovers routes on an "*as needed*" basis via a similar route discovery process. However, AODV adopts a very different mechanism to maintain routing information. It uses traditional routing tables, one entry per destination. This is in contrast to DSR, which can maintain multiple route cache entries for each destination. Without source routing, AODV relies on routing table entries to propagate a RREP back to the source and, subsequently, to route data packets to the destination. AODV uses destination sequence numbers as in DSDV [48] (Section 3) to prevent routing loops and to determine freshness of routing information. These sequence numbers are carried by all routing packets.

The absence of source routing and promiscuous listening allows AODV to gather only a very limited amount of routing information with each route discovery. Besides, AODV is conservative in dealing with stale routes. It uses the sequence numbers to infer the freshness of routing information and nodes maintain only the route information for a destination corresponding to the latest known sequence number; routes with older sequence numbers are discarded even though they may still be valid. AODV also uses a timer-based route expiry mechanism to promptly purge stale routes. Again if a low value is chosen for the time-out, valid routes may be needlessly discarded.

In AODV, each node maintains at most one route per destination and as a result, the destination replies only once to the first arriving request during a route discovery. Being a single path protocol, it has to invoke a new route discovery whenever the only path from the source to the destination fails. When topology changes frequently, route discovery needs to be initiated often which can be very inefficient since route discovery flood is associated with significant latency and overhead. To overcome this limitation, we have proposed a multipath extension to AODV called Ad hoc On-demand Multipath Distance Vector (AOMDV) [36]. AOMDV discovers multiple paths between source and destination in a single route discovery. As a result, a new route discovery is necessary only when each of the multiple paths fail. AOMDV, like AODV, ensures loop freedom and at the same time finds disjoint paths which are less likely to fail simultaneously. By exploiting already available alternate path routing information as much as possible, AOMDV computes alternate paths with minimal additional overhead over AODV.

Temporally Ordered Routing Algorithm (TORA) [44] is another on-demand protocol. TORA's route discovery procedure computes multiple loop-free routes to the destination which constitute a *destination-oriented directed acyclic graph* (DAG).

While the ad hoc network is looked upon as an undirected graph, TORA imposes a logical directionality on the links. TORA employs a route maintenance procedure requiring strong inter-nodal coordination based on a *link reversal* concept proposed in a seminal work by Gafni and Bertsekas [12] for localized recovery from route failures. The basic idea behind link reversal algorithms is as follows. Whenever a link failure at a node causes the node to lose all downstream links to reach the destination (and thus no longer in a destination-oriented state), a series of link reversals starting at that node can revert the DAG back to a destination-oriented state.

There are two types of link reversal algorithms namely full reversal and partial reversal differing in the way links incident on a node reverse their direction during the link reversal process. TORA specifically uses a modified version of partial link reversal technique. This modified version allows TORA to detect network partitions, a useful feature absent in many ad hoc networking protocols.

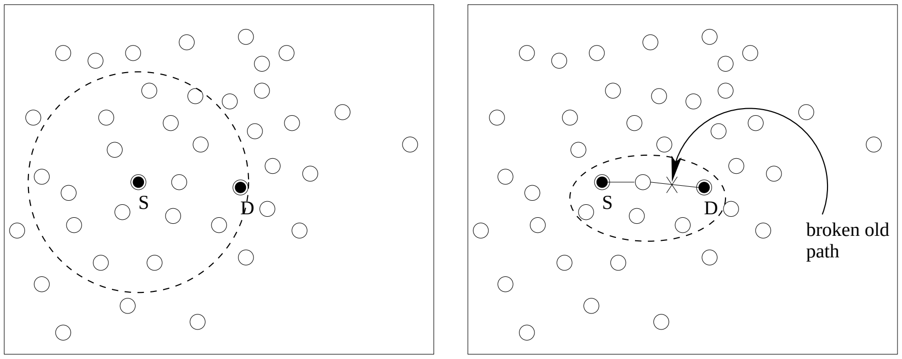
By virtue of finding multiple paths and using the link reversals for recovering from route failures, TORA can avoid a fresh route discovery until all paths connecting the source and the destination break (which is similar to AOMDV [36]). But TORA requires reliable and in-order delivery of routing control packets. Also, the nature of link reversal based algorithm makes it hard to keep track of path costs. Some performance studies [5, 10] have shown that these requirements hurt the performance of TORA so much so that they undermine the advantage of having multiple paths. Also, the link reversal in TORA by its nature leads to short-term routing loops. However, TORA remains an attractive option when a large number of nodes must maintain paths directed to a chosen destination.

Optimizations for On-demand Routing

Several general purpose optimizations have been proposed for on-demand routing that are largely independent of any specific protocol. These optimizations can be classified into three categories: flooding optimizations, stable route selection, and route maintenance optimizations. We will give a brief overview of techniques in each of these categories below.

In describing various protocols in the previous section, we have assumed that simple flooding is used for route discovery. However, efficient flooding techniques discussed in Section 2 can be used to reduce route discovery overhead. But when neighborhood-knowledge techniques are employed, the overall benefit depends on the relationship between the frequency of route discovery operations, network density, and the overhead incurred in maintaining up-to-date neighborhood information at each node.

Recognizing that route discovery flood is intended to search only the destination offers more room for optimization since flood need not reach every node. *Expanding ring search* [47] and *query localization* [7] are two representative examples which exploit this fact by performing a restricted flood within a small region (relative to network size) containing



(a) Expanding ring search (b) Query localization

Figure 1.2. Comparison of search regions using expanding ring search and query localization. Dotted circles in each figure indicate the search regions.

both source and destination. In expanding ring search, source estimates the distance (in hops) to the destination and uses this estimated distance (ring size) in the form of TTL to do a limited flood around the source; when the route search fails, ring size is increased iteratively until the whole network is searched or a route is found. Simplest mechanism for distance estimation is to use the last known hop count to destination; more sophisticated procedures have also been studied [56]. Query localization, on the other hand, is based on the notion of spatial and temporal locality of paths. It makes the assumption that new path and broken old path will only differ in a few nodes and therefore, the query is restricted within a few hops around the old path; when route discovery fails, the search region is expanded in subsequent tries. Figure 1.2 illustrates the difference in search regions used by these two techniques.

All three protocols described in the previous section use hop counts as a metric for path selection. However, it is possible that the quality of the links on a shortest hop path is not be strong. The likelihood for this is not negligible because two neighboring nodes in a shortest hop path can be separated by physical distance almost equal to their transmission range. This not only makes the signal strength on the link weak, but also increases likelihood of path failure when either of them moves slightly away from the other node. This observation led to the work on better metrics for path selection. *Associativity-based Routing (ABR)* [57] and *Signal Stability-Based Adaptive Routing (SSR)* [11] are among the earliest protocols with the goal of long-lived route selection. ABR uses a link metric called degree of association stability which is calculated as the number of successful beacon exchanges between neighbors sharing a link in some interval; more beacon exchanges indicate a stable link and such links are preferred during route selection. In contrast, SSR uses signal strength information to determine link stability. In general, alternative metrics other than hop counts to determine path costs are possible. Suitable choice of metrics can serve other purposes, such as balancing load or energy usage in the network.

Local route repair (e.g., [47]) and *preemptive routing* [15] are key examples of route maintenance optimizations proposed for on-demand routing. In the local repair mechanism, the basic idea is to have an intermediate node repair a broken route locally. Using local repair, an intermediate node can find an alternate (possibly longer) route quickly and efficiently as compared to the source performing a new route discovery. The effectiveness of local repair depends on how far away the destination is from the intermediate node. Local repair can be done either reactively after a route failure or proactively. Preemptive routing, on the other hand, proactively repairs routes by monitoring the likelihood of a path break by means of signal strength information and informing the source which will initiate an early route discovery. Using this mechanism, applications will not experience the latency involved in discovering a route after the route breaks.

Performance of On-Demand Routing

Performance of on-demand protocols is quite well-understood. Some empirical performance results in literature have found TORA to be the worst performer among the three protocols we have discussed [5, 10]. TORA's link reversal technique, though elegant, requires strong internodal coordination and thus has very high overhead. Besides, reliable and in-order delivery requirement imposes even greater demand in terms of bandwidth. As a result, later performance studies focused solely on the relative performance of DSR and AODV [50]. According to these studies, DSR with the help of caching is more effective at low mobility and low loads. AODV performs well in more stressful scenarios of high mobility and high loads. These relative performance differentials are attributed to DSR's lack of effective mechanisms to purge stale routes and AODV's need for resorting to route discovery often because of its single path nature. However, DSR with improved caching strategies, and AODV with the ability to maintain multiple paths are expected to have similar performance.

5. Proactive Versus On-demand Debate

As research on routing for ad hoc networks have matured, the superiority of one approach over the other has been debated. This question has motivated several simulation-based performance comparison studies [5, 10, 26, 4] and some theoretical studies (e.g., [52]). No clear winner emerged, although on-demand approach usually provides better or similar efficiency relatively for most common scenarios. Here we qualitatively compare the relative merits of the two approaches independently of any specific protocol.

Aggregate throughput and end-to-end delay are key measures of interest when assessing protocol performance. Throughput is directly related to the packet drops. Packet drops typically happen because of network congestion (e.g., buffer overflows) or for lack of a route. Since most dynamic protocols (proactive or reactive) try to keep the latter type (no route) of drops low by being responsive to topology changes, network congestion drops become the dominant factor when judging relative throughput performance. For the same data traffic load, routing protocol efficiency (in terms of control overhead in bytes or packets) determines the relative level of network congestion because both routing control packets and data packets share the same channel bandwidth and buffers.

End-to-end delay of a packet depends on route discovery latency, additional delays at each hop (comprising of queuing, channel access and transmission delays), and the number of hops. At low loads, queuing and channel access delays do not contribute much to the overall delay. In this regime, proactive protocols, by virtue of finding optimal routes between all node pairs, are likely to have better delay performance. However, at moderate to high loads, queuing and channel access delays become significant enough to exceed route discovery latency. So, like in the case of throughput, routing protocol overhead again becomes key factor in determining relative delay performance.

The efficiency (in terms of control overhead) of one approach over the other depends to a large extent on the relative node mobility and traffic diversity. Note that individual node speeds are irrelevant unless they affect the relative node speeds because path stability is primarily determined by relative node mobility; relative node mobility can be low even when nodes individually move at high speeds as with group movement scenarios. Traffic diversity measures the traffic distribution among nodes. A low traffic diversity indicates that majority of the traffic is directed toward a small subset of nodes. This can happen when there are fewer source-destination pairs communicating or when most of the nodes communicate with a few set of nodes. A realistic example of the latter case is when an ad hoc network is attached to the Internet and mobile nodes spend most time accessing the Internet via a few gateway nodes. High traffic diversity, on the other hand, means that traffic is more uniformly distributed across all nodes (e.g., when every node communicates with every other node).

On-demand routing is naturally adaptive to traffic diversity and therefore its overhead proportionately increases with increase in traffic diversity. On the other hand, for proactive routing overhead is independent of the traffic diversity. So when the traffic diversity is low, on-demand routing is relatively very efficient in terms of the control overhead regardless of relative node mobility. When the majority of traffic is destined to only few nodes, a proactive protocol maintaining routes to every possible destination incurs a lot of unnecessary overhead. Mobility does not alter this advantage of on-demand routing. This is because an on-demand protocol reacts only to link failures that break a currently used path, whereas proactive protocol reacts to every link failure without regard to whether the link is on a used path. On-demand routing can also significantly benefit by caching multiple paths when node mobility is low.

With high traffic diversity, the routing overhead for on-demand routing could approach that of proactive routing. The overhead alone is not the whole picture. Path optimality also plays a role in determining the overall overhead — using a suboptimal path results in excess transmissions which contribute to overhead. Using suboptimal routes also increases the end-to-end delay. Recall that pure proactive protocols aim to always provide shortest paths. Whereas with pure on-demand protocols, a path is used until it becomes invalid even though the path may become suboptimal due to node mobility. The issue of path suboptimality becomes more significant at low node mobility because each path is usable for a longer period. Thus, accounting suboptimal path overhead increases the total overhead with on-demand approach.

The discussion so far implicitly assumed that traffic sessions are long-lived. However, when traffic sessions are short-lived, i.e., come and go quickly, the overhead required to handle each session becomes expensive with on-demand routing. Also, initial route discovery latency inherent to on-demand routing may also be unacceptable in this case.

Hybrid Approaches

It is not hard to hypothesize that a combination of proactive and on-demand approaches is perhaps better than either approach in isolation. As an example, consider augmenting a primarily reactive protocol such as AODV with some proactive functionality by making each active destination periodically refresh routes to itself as in DSDV. The advantage of such a protocol is two-fold: (i) the overhead and delay due to suboptimal routes can be limited to the refresh interval; (ii) such destination-initiated refresh mechanism also offers routes in advance to nodes that might route traffic to the destination later, making this mechanism proactive. The overhead created by this proactive mechanism is determined by the number of active destinations and the refresh interval. Carefully choosing the refresh interval can improve the overall performance compared to the pure reactive mechanism. A variant of the above idea has been suggested in [44] and evaluated in [31]. There have been several other efforts based on this theme of combining proactive and reactive approaches. Below, we will review two representative protocols from this category. These two protocols, though mainly aim toward scalable routing for large networks, still demonstrate the benefit attainable by the combined proactive/reactive approach.

Zone Routing Protocol (ZRP) [18, 45] is a *hybrid* protocol with distinct proactive and reactive components working in cohesion. ZRP defines a zone for each node X which includes all nodes that are within a certain distance in hops, called zone radius, around the node X. Nodes that are exactly zone radius distance away from node X are called border nodes of X's zone. A proactive link state protocol is used to keep every node aware of the complete topology within its zone. When a node X needs to obtain a route to another node Y not in its zone, it reactively initiates a route discovery which works similar to flooding except that it involves only X's border nodes and their border nodes and so on. Route query accumulates the traversed route on its way outward from X (like in source routing) and when the query finally reaches a border node which is in destination Y's zone, that border node sends back a reply using the accumulated route from the query. Depending on the choice of zone radius, ZRP can behave as a pure proactive protocol, a pure reactive protocol, or somewhere in between. While this is an attractive feature to adapt to network conditions by tuning a single parameter, zone radius, it is not straightforward to choose the zone radius dynamically.

Hazy Sighted Link State (HSLS) protocol [53] is a link state protocol based on limited dissemination. Though HSLS does not *per se* have any reactive component as in ZRP, it partially exhibits behavior typical of reactive protocols, specifically use of suboptimal routes. Main idea here is to control the link state dissemination scope in space and time — closer nodes are sent link state updates more frequently compared to far away nodes. This idea is based on the observation that two nodes move slowly with respect to each other as the distance between them increases (also referred in the literature as the *distance effect*). So distant nodes through infrequent updates are only provided "hints" to route a packet closer toward the destination. As the packet approaches the destination, it takes advantage of progressively recent routing information that improve its chances of reaching the destination. A consequence of this limited dissemination strategy is that a packet may take suboptimal routes initially, but eventually arrives at the destination via an optimal route. Thus some amount of suboptimal routing is allowed to reduce the overall control overhead.

One important shortcoming of both ZRP and HSLS is that their design assumes a uniform traffic distribution and then optimizes the overall overhead. When the traffic is non-uniform, these protocols may not actually be efficient. A better strategy, perhaps, is to have the protocol also adapt to the traffic diversity.

6. Location-based Routing

Proactive, reactive or hybrid approaches we looked at in the previous sections share one common feature. In all these approaches, nodes discover (partial or full) topology information by exchanging routing messages and use this information to guide future routing decisions. We will now look at a completely different routing approach that utilizes *geographic location* of nodes.

Location-based (also called geographic) routing assumes that each node knows its own location by using the global positioning system (GPS) or some other indirect, localization technique. Besides, every node learns locations of its immediate neighbors by exchanging hello messages. The location of potential destination nodes is assumed to be available via a *location service.* When a source wants to send a packet to a destination, it uses the destination's location to find a neighbor that is closest in geographic distance to the destination, and closer than itself, and forwards the packet to that neighbor. That neighbor repeats the same procedure and until the packet makes it to the destination. This idea is often referred to as *greedy forwarding* in the literature. Note that greedy forwarding may fail to make progress, but we will postpone this discussion until later in this section.

Observe that geographic routing does not need any explicit route discovery or route maintenance mechanisms unlike other approaches. Except for gathering knowledge of node locations, nodes need not maintain any other routing state nor do they have to exchange any routing messages. As a result, geographic routing, in comparison with other ap-

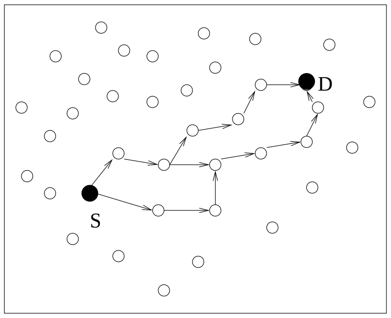


Figure 1.3. Restricted directional flooding in LAR and DREAM.

proaches, can potentially be more efficient when topology changes quite frequently and can scale better with network size.

Location-based Routing Protocols

Location-Aided Routing (LAR) [30] is an optimization for reactive protocols to reduce flooding overhead. LAR uses an estimate of destination's location to restrict the flood to a small region (called *request zone*) relative to the whole network region. The idea is somewhat similar to query localization discussed in Section 4, although LAR was proposed earlier and it additionally demonstrates how location information can be exploited to benefit topology-based routing protocols.

LAR assumes that each node knows its own location, but does not employ any special location service to obtain location of other nodes. Destination location information obtained from a prior route discovery is used as an estimate of destination's location for limiting the flooding region in a subsequent route discovery. Two different LAR schemes with different heuristics to choose the request zone have been proposed. In the scheme that is shown to perform well, source floods a route request by including its estimate of destination's location and its estimated distance to destination in the request. Neighboring nodes receiving this request calculate their distance to destination using the destination's location in the request. If they are closer to the destination than the source, then they forward the request further by replacing the source's distance to destination with their own distance. A similar procedure is repeated at other nodes resulting in a directed flood toward the destination (Figure 1.3). Note that LAR uses location information only for finding routes and not for geographic forwarding of data packets.

Distance Routing Effect Algorithm for Mobility (DREAM) [1] is an early example of a routing protocol which is completely location-based. The location service is also part of the same protocol. With DREAM's location service, every node proactively updates every other node about its location. The overhead of such location updates is reduced in two ways. First, distance effect (nodes move slowly with respect to each other as their distance of separation increases) is exploited by sending location updates to distant nodes less frequently than closer nodes (This is similar to HSLS (Section 5) which uses the distance effect for limited dissemination of link state updates). Second, each node generates updates about its location depending on its mobility rate — fast moving nodes update more often whereas slow moving nodes generate updates less often.

DREAM geographically forwards data packets in the form of a directional flood (similar to LAR's scheme for route request flood). Such directional flooding increases the likelihood of correct data delivery by compensating for inaccuracy in destination location information. At the same time, it can be very inefficient too. One simple mechanism to avoid this inefficiency is to follow the LAR strategy, except that here the first data packet acts as a route request; subsequent data packets will not be flooded, but they take the path found by the first packet.

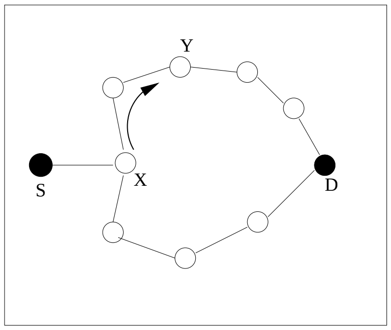


Figure 1.4. Illustration of greedy forwarding failure and perimeter routing in GPSR. In this figure, S is the source and D is the destination. By greedy forwarding, S sends the packet to node X. But all neighbors of X are farther to D than itself, so greedy forwarding fails at X. X then switches to perimeter mode and routes the packet along the perimeter until it reaches Y (closer to D than itself). From Y, greedy forwarding is used again until the packet reaches D. For simplicity, in this example we have assumed that actual network graph is planar.

Greedy Perimeter Stateless Routing (GPSR) [29] is a protocol that specifies only the geographic forwarding strategy, unlike DREAM, and assumes the existence of a location service. So any location service, either DREAM's location service or other schemes mentioned later in this section, could be used. GPSR's data forwarding algorithm comprises of two components: greedy forwarding and perimeter routing. Greedy forwarding is the same idea described at the beginning of this section. GPSR uses it as the default forwarding mechanism. But when greedy forwarding is not possible, perimeter routing is used. Greedy forwarding becomes impossible when the packet reaches a node which does not have any neighbor closer to the destination than itself, i.e., packet reaches a dead end or void. Figure 1.4 shows an example. The basic idea in perimeter routing is to begin at the node X where greedy forwarding failed and walk around the void until a node Y which is closer to the destination than X is reached. From then on (i.e., Y onward), greedy forwarding is resumed until the packet reaches the destination or another void is encountered. One might wonder at this point why greedy forwarding is at all used if it can sometimes fail and why not simply use only perimeter routing always? The answer is as follows. Greedy forwarding is not only simple, but also optimal when it succeeds. On the other hand, perimeter routing always guarantees data delivery, it is seldom optimal.

In the simple description of perimeter routing above, we have omitted several key details. In order to apply perimeter routing, a planarized version of the actual graph has to be constructed first by removing all crossing edges. In simple terms, planar graph can be seen as collection of closed polygons stitched together. Local algorithms using only neighbors and their locations are available for constructing different kinds of planar graphs (e.g., restricted neighborhood graph and gabriel graph). Once a planar graph is constructed in a distributed manner, perimeter routing of a packet starting at a node X destined for node D reduces to moving across the successively closer faces to the destination which intersect the line segment joining X and D by traversing some edges in each face. Since perimeter routing moves across faces, it is also called *face routing.* Figure 1.4 illustrates the perimeter routing idea. Note that GPSR uses the planarized graph only for perimeter routing and the actual graph for greedy forwarding.

In a recent paper [33], it was observed that perimeter routing used in GPSR to come out of a void can result in much longer paths than needed. A new routing algorithm called *Greedy Other Adaptive Face Routing (GOAFR)* was also proposed in the same paper which avoids long paths by using a bounding region and a slightly different variant of perimeter routing. More importantly, GOAFR is shown to be worst-case optimal and also on the average significantly outperforms GPSR. It is also worth mentioning here that most geographic routing protocols in the literature can be classified into greedy forwarding, face routing, or a combination of both.

Location Service Protocols

Location service providing destination's location is the key component of any system that does geographic routing. We already looked at one location service mechanism in the context of DREAM. Recall that in DREAM's mechanism every node maintains location information for every other node via a flooding-like (though optimized) location dissemination. So as the network size becomes large, the overhead of this mechanism grows very fast. Predictive mechanisms, such as *dead reckoning*, however, can be used to contain such overheads [34]. Here, infrequent dissemination may be sufficient if a movement model of the mobile nodes can be constructed.

At the opposite end, location service protocols can take a database approach. Location database systems typically rely on one or more nodes in the network that work as location servers. The servers may be dynamically elected. They are updated proactively by moving nodes. These systems are inefficient when locations are frequently queried, as this increases the query-reply load. *Grid Location Service (GLS)* [35] and *HomeZone* [55]) fall under this category.

In summary, geographic routing is undoubtedly a promising approach for large and dynamic networks, provided every node has the ability to find its own location and the availability of an efficient location service. Efficient location service in a mobile network is the key to the success of geographic routing because the location service amounts to a major fraction of the overhead. It is also important to recognize that geographic routing is still an evolving area with not sufficient evidence for substantial gains in overhead compared to traditional approaches. For instance, a recent performance comparison between DREAM and DSR has shown that DSR outperforms DREAM in both performance and efficiency in some scenarios [6]. Finally, a combination of geographic routing and local topology routing is worth investigating as a way to increase the likelihood of packet delivery in situations where the overhead of providing accurate location is high.

7. Concluding Remarks

In this chapter, we have described unicast routing protocols for mobile ad hoc networks – focusing on proactive, reactive and geographic approaches. In this process, we also reviewed efficient flooding techniques since most routing protocols employ some form of flooding. It is fair to conclude that no single routing approach is clearly superior to others in every possible scenario. Their relative merits are heavily dependent on the application context. Reactive protocols typically use a lower routing overhead when traffic diversity is low. But they incur a high route discovery latency and also may use suboptimal routes. So they may not be effective for delay-sensitive applications and short-lived flows. Hybrid protocols can potentially combine the benefits of both proactive and reactive approaches and avoid their drawbacks. However, more work is needed before this potential can be fully realized. Geographic routing has the potential to scale to large and dynamic networks. But to be effective, it needs some way to gather accurate location information for the current node, the destination node and also the neighboring nodes. The impact of inaccuracy in location information is not fully understood yet.

For the most part, we focused only on shortest-hop routing. Using load-aware metrics and protocols, instead will help better utilize the network resources by spreading the traffic uniformly across the network, especially in low mobility scenarios. Designing load-aware algorithms is, however, quite challenging because it not only requires an good understanding and modeling of wireless channel interference behavior, but also routing decisions and ensuing interference are intertwined.

A closely related issue is that of cross-layer interactions. Several performance studies have shown that cross-layer interactions can play a big part in determining overall network performance almost to the same extent as protocol mechanisms at each layer. This calls for a recognition of the sensitivity of a protocol's performance at one layer on the specific higher and lower layer protocols — choosing a different set of higher and lower protocols may give different performance results.

Finally in current research, routing protocols are designed in isolation by considering the wireless link as an uncontrollable entity, and abstracting out a "graph" view of the network and developing routing protocol for this graph. But the reality is different. Wireless link can indeed be tuned by varying certain transmission parameters. For example, increasing or decreasing transmission power can make or break a link. Likewise, varying the modulation and coding properties can change the quality of the link. Not exploiting this controllability of the wireless link limits the extent to which performance of a routing protocol can be optimized.

References

[1] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of IEEE/ACM MobiCom*, pages 76–84, 1998.

[2] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[3] D. M. Blough et al. On the Symmetric Range Assignment Problem in Wireless Ad Hoc Networks. In *Proceedings of IFIP Conference on Theoretical Computer Science*, pages 71–82, 2002.

[4] R. V. Boppana, M. K. Marina, and S. P. Konduru. An Analysis of Routing Techniques for Mobile and Ad Hoc Networks. In *Proceedings of International Conference on High Performance Computing (HiPC)*, pages 239–245, 1999.

[5] J. Broch, D. Maltz, D. Johnson, Y-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of IEEE/ACM MobiCom*, pages 85–97, 1998.

[6] T. Camp et al. Performance Comparison of Two Location Based Routing Protocols for Ad Hoc Networks. In *Proceedings of IEEE Infocom*, pages 1678–1687, 2002.

[7] R. Castaneda, S. R. Das, and M. K. Marina. Query Localization Techniques for On-demand Protocols in Ad Hoc Networks. *ACM/Kluwer Wireless Networks*, 8(2/3):137–151, 2002.

[8] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. A Loop-free Extended Bellman-Ford Routing Protocol Without Bouncing Effect. In *Proceedings of ACM SIGCOMM*, pages 224–236, 1989.

[9] T. Clausen et al. Optimized Link State Routing Protocol. http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-11.txt, July 2003. IETF Internet Draft (work in progress).

[10] S. R. Das, R. Castaneda, and J. Yan. Simulation-based Performance Evaluation of Routing Protocols for Mobile Ad hoc Networks. *ACM/Baltzer Mobile Networks and Applications (MONET)*, 5(3):179–189, 2000.

[11] R. Dube et al. Signal Stability-based Adaptive Routing Routing (SSA) for Ad Hoc Mobile Networks. *IEEE Personal Communications*, 4(1):36–45, 1997.

[12] E. Gafni and D. Bertsekas. Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, 29(1):11–18, 1981.

[13] J. J. Garcia-Luna-Aceves. Loop-Free Routing Using Diffusing Computations. *IEEE/ACM Transactions on Networking*, 1(1):130–141, 1993.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[15] T. Goff, N. Abu-Ghazaleh, D. Phatak, and R. Kahvecioglu. Preemptive Routing in Ad Hoc Networks. In *Proceedings of IEEE/ACM MobiCom*, pages 43–52, 2001.

[16] S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. *Algorithmica*, 20:374–387, 1998.

[17] Z. Haas, J. Halpern, and L. Li. Gossip-based Ad Hoc Routing. In *Proceedings of IEEE Infocom*, pages 1707–1716, 2002.

[18] Z. Haas and M. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. *IEEE/ACM Transactions on Networking*, 9(4):427–438, 2001.

[19] C. Hedrick. Routing Information Protocol. RFC 1058, 1988.

[20] G. Holland and N. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. *ACM/Kluwer Wireless Networks*, 8(2/3):275–288, 2002.

[21] Y-C. Hu and D. Johnson. Caching strategies in On-demand Routing Protocols for Wireless Ad Hoc Networks. In *Proceedings of IEEE/ACM MobiCom*, pages 231–242, 2000.

[22] Y-C. Hu and D. Johnson. Implicit Source Routes for On-demand Ad Hoc Network Routing. In *Proceedings of ACM MOBIHOC*, pages 1–10, 2001.

[23] Y-C. Hu and D. Johnson. Ensuring Cache Freshness in On-demand Ad Hoc Routing Protocols. In *Proceedings of Int'l Workshop on Principles of Mobile Computing (POMC)*, pages 25–30, 2002.

[24] IEEE Standards Department. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11–1997, 1997.

[25] J. M. Jaffe and F. H. Moss. A Responsive Distributed Routing Algorithm for Computer Networks. *IEEE Transactions on Communications*, 30(7):1758–1762, 1982.

[26] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-Hoc Networks. In *Proceedings of IEEE/ACM MobiCom*, pages 195–206, 1999.

[27] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile computing*, chapter 5. Kluwer Academic, 1996.

[28] D. B. Johnson, D. A. Maltz, Y. Hu, and J. G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, Feb 2002. IETF Internet Draft (work in progress).

[29] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of IEEE/ACM MobiCom*, pages 243–254, 2000.

[30] Y. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of IEEE/ACM MobiCom*, pages 66–75, 1998.

[31] S. P. Konduru and R. V. Boppana. On Reducing Packet Latencies in Ad Hoc Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1482–1487, 2000.

[32] U. C. Kozat, G. Kondylis, B. Ryu, and M. K. Marina. Virtual Dynamic Backbone for Mobile Ad Hoc Networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, pages 250–255, 2001.

[33] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-Case Optimal and Average-case Efficient Geometric Ad-hoc Routing. In *Proceedings of ACM MobiHoc*, pages 267–278, 2003.

[34] V. Kumar and S. R. Das. Performance of Dead Reckoning-Based Location Service for Mobile Ad Hoc Networks. *Wireless Communications and Mobile Computing*, 2003. To appear.

[35] J. Li, J. Jannoti, D. DeCouto, D. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proceedings of IEEE/ACM MobiCom*, pages 120–130, 2000.

[36] M. K. Marina and S. R. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.

[37] M. K. Marina and S. R. Das. Performance of Route Caching Strategies in Dynamic Source Routing. In *Proceedings of the Int'l Workshop on Wireless Networks and Mobile Computing (WNMC) in conjunction with Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 425–432, 2001.

[38] M. K. Marina and S. R. Das. Routing Performance in the Presence of Unidirectional Links in Multihop Wireless Networks. In *Proceedings of ACM MobiHoc*, pages 12–23, 2002.

[39] P. M. Merlin and A. Segall. A Failsafe Distributed Routing Protocol. *IEEE Transactions on Communications*, 27(9):1280–1287, 1979.

[40] J. Moy. OSPF version 2. RFC 2328, 1998.

[41] S. Murthy and J. J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM/Baltzer Mobile Networks and Applications*, 1(2):183–197, 1996.

[42] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proceedings of IEEE/ACM MobiCom*, pages 151–162, 1999.

[43] R. Ogier, M. Lewis, and F. Templin. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). http://www.ietf.org/internet-drafts/draft-ietf-manet-tbrpf-09.txt, June 2003. IETF Internet Draft (work in progress).

[44] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE Infocom*, pages 1405–1413, 1997.

[45] M. Pearlman and Z. Haas. Determining the Optimal Configuration for the Zone Routing Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1395–1414, 1999.

[46] W. Peng and X. Lu. On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks. In *Proceedings of ACM MobiHoc*, pages 129–130, 2000. Poster.

[47] C. E. Perkins, E. Belding-Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. http://www.ietf.org/rfc/rfc3561.txt, July 2003. RFC 3561.

[48] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of ACM SIGCOMM*, pages 234–244, 1994.

[49] C. E. Perkins and E. M. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

[50] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. *IEEE Personal Communications*, 8(1):16–28, 2001.

[51] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3898–3907, 2002.

[52] C. Santivanez, B. McDonald, I. Stavrakakis, and R. Ramanathan. On the Scalability of Ad Hoc Routing Protocols. In *Proceedings of IEEE Infocom*, pages 1688–1697, 2002.

[53] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making Link-State Routing Scale for Ad Hoc Networks. In *Proceedings of ACM MobiHoc*, pages 22–32, 2001.

[54] R. Sivakumar, B. Das, and V. Bharghavan. Spine Routing in Ad Hoc Networks. *Cluster Computing*, 1:237–248, 1998.

[55] I. Stojmenovic. Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks. Technical Report, Computer Science, University of Ottawa, September 1999.

[56] J. Sucec and I. Marsic. An Application of Parameter Estimation to Route Discovery By On-Demand Routing Protocols. In *Proceedings of IEEE ICDCS*, pages 207–216, 2001.

[57] C-K. Toh. Associativity-Based Routing for Ad-Hoc Mobile Networks. *Wireless Personal Communications*, 4(2):1–36, 1997.

[58] B. Williams and T. Camp. Comparision of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of ACM MobiHoc*, pages 194–205, 2002.

[59] J. Wu and F. Dai. A Generic Distributed Broadcast Scheme in Ad Hoc Wireless Networks. In *Proceedings of IEEE ICDCS*, pages 460–467, 2003.