Chapter 1

Routing Scalability in MANETs

With today's rapidly improving link-layer technology, and the widespread adoption of wireless networking, the creation of large-scale ad hoc networks could be construed as all but inevitable. However, for routing in a network to be feasible, there is a pressing need for a scalable ad hoc routing protocol. Applications for large-scale ad hoc networking include consumer owned networks, tactical military networks, natural disaster recovery services and vehicular networks.

Ad hoc routing protocols used experimentally today, such as DSDV, OLSR, AODV and DSR, only scale reasonably well to dozens or sometimes hundreds of nodes. In order to support networks one or several orders of magnitude larger, there is a need for routing protocols designed specifically to scale to large networks. Under certain limiting assumptions, geographical location information can be used to help the routing layer scale to support very large networks. However, in this chapter, we will focus on the more generally viable approach of multi-level clustering, which to some extent is what has made made the Internet scale as well as it does.

We will study various aspects of routing protocol scalability. First, we will take a look at the analytical results thus far, with regards to ad hoc network scalability. These results assess the theoretical limits for ad hoc network scalability in terms of the capacity achievable per

node in the network. To set the stage for the scalable routing techniques, and to introduce the reader to the issues that impact scalability, we briefly discuss several techniques used for ad hoc routing. These include flat proactive routing, pure reactive routing, geographic routing and zone-based hybrid protocols. We will then take a more detailed look at routing based on clustering, in its single-level and multi-level forms. Finally, we will spend the last third of the chapter describing a recent promising scalable routing technique based on multi-level clustering, called Dynamic Address Routing.

Defining Scalability

The scalability of a network protocol could potentially be defined in many different ways, and at several different levels. In this chapter, we will use the following high-level definition of scalability.

Scalability *is the ability of a routing protocol to perform efficiently as one or more inherent parameters of the network grow to be large in value.*

Typical parameters that are studied for ad hoc networks are the number of **nodes (N)**, and the average rate of **mobility(M)** in m/s under various mobility models. Other parameters that have an impact on scalability include **node density (D)**, number of **links (L)**, the **frequency of connection establishment (F)** and the average number of **concurrent connections (C)**. Measuring performance can also be done in several ways. Typical metrics that are used to evaluate routing protocols are overall message or byte overhead, amount of per-node state to be maintained, latency and total network throughput. In this chapter, we will primarily discuss the overhead aspect. However, we will discuss the other metrics briefly in the sections to follow.

In the rest of this chapter, we will be using notation commonly used in asymptotic analysis to describe various scalability characteristics. In particular, we will be using the $\Omega(X)$ to denote a lower asymptotic bound, $O(X)$ to denote an upper asymptotic bound, and $\Theta(X)$ to denote a simultaneous upper *and* lower asymptotic bound. By asymptotic bound, we

refer to the scaling behavior of the protocol with respect to a given variable. For example, if a protocol is said to have an overhead of $O(N)$, this means that there exists a constant c such that the amount of overhead incurred in a network of N nodes is at most cN, where N can take on any finite value. Except where explicitly stated, node identifiers are taken to be 48-bit MAC addresses. It is reasonable to assume that a 48-bit identifier space will not be exhausted within the foreseeable future ($2^{48} = 281474976710656$ or about 281 trillion unique identifiers).

1.1 Analytical Results on Ad Hoc Network Scalability

The analytical study of scalability relationships in ad hoc networks can provide us with valuable insights into the proper design of ad hoc routing protocols and possibly related mechanisms at other layers. So far, the study of scalability in ad hoc networks has been mostly limited to simulation. However, a few significant analytical results have emerged fairly recently, and we will introduce them in this section.

Link Layer

Even without considering the effects of routing overhead on the performance of ad hoc networks, there are several concerns regarding the scalability of current wireless networking link-layer technology.

It is easily seen that the popular 802.11 link layer, when deployed with omnidirectional antennas, does not scale with respect to node density, D. Clearly, as D grows, each node will receive only a proportional share of the channel capacity. The upper limit on the average link layer capacity made available to each node decreases as $1/D$. A well-known solution to this problem is to reduce the transmission range of each node, thereby reducing D. The effect achieved is called *spatial reuse*, where several transmissions can take place on the same frequency band simultaneously, due to the limited spatial overlap of the transmitters involved.

However, as a direct effect of reducing the transmission range, packets will in some cases have to be forwarded over an increased number of wireless links in order to reach their respective destinations. Increasing the number of hops is likely to lead to longer end-to-end delays, lower packet delivery ratios, and in some cases, increased traffic congestion.

A fundamental result in multi-hop ad hoc networking was shown by Gupta and Kumar in [11]. A simplified argument for their result follows. In a network of nodes with omnidirectional antennas, and with a constant node density, we can expect the average path length to be $\Theta(\sqrt{N})$, where N is the number of nodes in the network. Therefore, for every packet a node generates, it will see on average $\Theta(\sqrt{N})$ packets originated by other nodes. Thus, with a channel capacity of C the capacity available for a node's own packets will be

$$O(\frac{C}{\sqrt{N}}), \tag{1.1}$$

where C is the total channel capacity, i.e. the maximum throughput achievable by a single link when there are no other links competing for the channel. The unfortunate conclusion is that under certain reasonable assumptions, purely omnidirectional ad hoc networks cannot grow beyond certain fairly restrictive limits. However, we would like to point out that all hope is not lost. As link layer technologies evolve, the channel capacity, C, will continue to increase. And for every increase in channel capacity, the feasible network size grows by the square of this increase, as per Eq 1.1. Since the publication of the paper mentioned above, channel capacity has grown by approximately 100 times. Our conclusion is that whatever the feasible network size was at the time of publication of [11] (1999), the upper limit today is 10,000 times higher. Clearly, this shows that link layer capacity by itself is not the limiting factor in multi-hop ad hoc networks. Note that this highly theoretical result does not take into account any routing layer overhead, the scalability of which is the topic of this chapter.

In addition, there is the prospect of using directional antennas [16] and adaptive beamforming antennas [26]. These could be employed to have nodes dynamically direct a narrow transmission beam toward the neighbor it wishes to communicate with, greatly improving both transmission range and spatial reuse.[1]

[1] To see how both range and spatial reuse can be improved simultaneously, consider the extreme case of directional trans-

Grossglauser and Tse published a somewhat controversial result in [10]. Here, the authors show that if nodes are mobile, then each node could potentially achieve a throughput that *does not* decrease with the size of the network. By relaying each packet only once, to a random one-hop neighbor, a source can achieve a stationary uniform distribution of its packets throughout the network. Subsequently, as the destination moves around, each of its neighbors will always have packets to deliver to it. As each packet only traverses two hops, the throughput of the node can be expected to remain the same, regardless of the size of the network.

This result relies on strong assumptions with regard to the mobility patterns of the nodes, and even given those assumptions, the expected delay is of the order of node mobility, in the sense that nodes have to move considerable distances before a packet can be delivered. In our opinion, although this result holds in theory, it is unclear as of yet if it will have much practical relevance.

Hierarchical Routing

Hierarchical routing protocols, such as those based on multilevel clustering, consist of a number of different components, such as clustering, routing and location management. Here, clustering is the process by which nearby nodes form groups, called *clusters*. For the purpose of routing, clusters can be treated as a single destination, thereby reducing the amount of routing state that needs to be maintained at each node. Location management is any technique by which a source can determine the current address or *location* of an intended destination node, given its identifier.

When studying the scalability of such protocols, the scaling properties of each of these components need to be considered. In [30, 29], the authors study the theoretical scalability aspects of multi-level hierarchical routing in ad hoc networks. In the general scheme they analyze, nodes are organized in clusters, which are then grouped in higher level clusters. The number of levels is logarithmic in the network size. The location management technique they analyze is a distributed location server, where each node stores the current address

mission, a point to point laser link.

of $\Theta(N)$ other nodes, where N is the number of nodes in the network. A similar location management scheme is discussed in section 1.8. Specifically, their analysis focuses on the number of routing-related control datagrams that a node needs to transmit per unit of time, on average, given a wide variety of parameters.

Their main result is that routing overhead is polylogarithmic in the size of the network. More specifically, the channel capacity required for routing control messages sent by each node, on average, is

$$\Omega(log^3 N).$$

Interestingly, they show that the dominating factor in the overhead calculation is not routing updates, but the retransmission of location information due to changes in the clustering hierarchy, called *location management handoff*. Other potentially valuable results of the same paper include the overhead incurred by cluster formation and maintenance, which is computed to be

$$O(log\ N)$$

packet transmissions per node per second, and the overhead for location management handoff, which is shown to be

$$\Theta(log^2 N)$$

packet transmissions per node per second, where the size of every control packet is $\Theta(log\ N)$. Note that this study is targeted at a particular group of clustering schemes (Max-Min D-hop clustering [1]). These are based on finding the node with the maximum node identifier in a D-hop neighborhood, and assigning that node to be the cluster head. Other types of clustering, such as that described in section 1.8, could potentially have different scaling behaviors. It is also geared toward a scalable hierarchical location management scheme similar to that used in [7, 8], described in section 1.8. Again, other types of location management will exhibit different scaling behavior. Nevertheless, these results offer valuable insights into the scalability of hierarchical, multi-level clustering ad hoc routing protocols. To our knowledge, [30] is the first paper with comprehensive theoretical results on the overhead of multi-level hierarchical routing protocols.

1.2 Flat Proactive Routing

Flat proactive routing scales very well with respect to the frequency of connection establishment (F) and the number of concurrent connections (C). However, the number of control packet transmissions per node is $\Theta(N)$.

In proactive routing, the routing protocol periodically disseminates routing information throughout the network. With flat proactive routing, every node keeps routing information for every other node: there is no abstraction for nodes far away. This strategy generally leads to close to optimal paths, but this is achieved at the cost of lacking scalability. The flat proactive routing protocols proposed so far can be roughly divided into two subcategories; link-state (LS) and distributed Bellman-Ford (DBF) algorithms.

In LS algorithms such as Fish-eye State Routing [22], Global State Routing [4] and Optimized Link-State Routing [6], each node has complete, although not always accurate, knowledge of the state of every link in the network. Using this information, it can calculate the entire path to the destination on its own accord. This has many advantages. In particular, recovery from link failure is typically very quick in LS protocols. With large N or D, the number of links in the network, and thus the routing table size, may be prohibitive. Fisheye State Routing (FSR) [22] tries to reduce the overall overhead by limiting the rate of link-state updates far away from the source of the update. The idea in FSR is that link changes far away generally have a small effect on local routing decisions.

In DBF algorithms, such as Destination Sequenced Distance Vector routing [24] and Wireless Routing Protocol [19], each node has much less information about the network. For every destination, a node maintains a routing table consisting of the distance to the destination, and the next hop neighbor on the shortest route toward the destination. Typically, after a link failure, there is an interval of time where faulty routes may exist, until the protocol has settled on a new route.

Common for all of these protocols is that the necessary amount of state kept at each node scales at least linearly with N. In a mobile network, this state will have to be updated frequently, resulting in protocol overhead on the order of $O(N)$ [30]. For this reason, flat

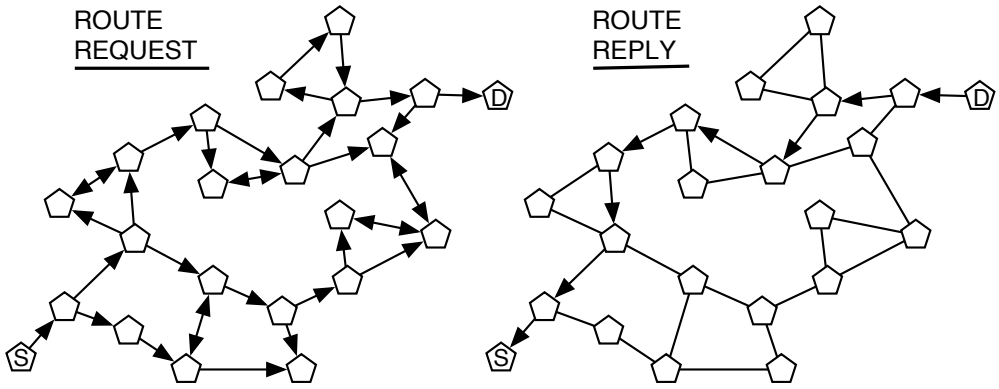

Figure 1.1: Reactive routing. A route request is flooded throughout the network. Once the request reaches the intended destination, a route reply is sent back along a discovered path.

proactive routing protocols are only feasible for small networks.

1.3 Pure Reactive Routing

> *Reactive routing is scalable with respect to most parameters, as long as the frequency of connection establishment (F), and the average number of concurrent connections (C), remain low. Control packet transmissions per node grow as* $O(F + C)$*, which is* $\Omega(1)$*, but* $O(N^2)$*.*

In an effort to address the problem of maintaining state for all nodes in the network, reactive protocols such as Ad hoc On-demand Distance Vector routing [25], Dynamic Source Routing [13], Associativity Based Routing [31] and Labeled Distance Routing [9], defer the expenditure of routing overhead until the time of connection establishment. With this technique, nodes keep completely quiet as long as there is no data to transmit. If a connection is to be established, the source node, S, needs to flood the network with a route request, as shown in Figure 1.1. When the intended destination, D, receives the route request, it responds to the source with a route response, using one of the routes discovered during the route request phase. In networks where a large majority of nodes have nothing to send, and where connections involve more than just a few packets, this strategy pays off in terms of reducing the overall routing overhead.

Reactive routing protocols have seen much popularity in ad hoc networks research. This is due to several good reasons, including the battery savings achieved by not transmitting anything during idle periods. Other important reasons are the good performance and the straight-forward design principles of AODV and DSR, the two most well-known reactive ad hoc routing protocols.

However, by deferring the routing overhead, these protocols lose many potential aggregation benefits made possible by proactively distributing routing information. In contrast with flat proactive routing, every connection establishment sets off a reactive route request with an asymptotic cost of $O(N)$, as a non-negligible constant fraction of all nodes will rebroadcast the request packet. In addition, in a mobile network, established connections will fail regularly due to link breakages caused by node motion, thereby initiating additional route requests. This gives us an overhead complexity of $\Theta(F + C)$ for the number of route requests per second. Putting the two terms together, the expected number of control packet transmissions is $O(N(F + C))$. With N nodes to share the burden, the average per-node cost is $O(F + C)$. Note that if a constant fraction of the nodes may be expected to start or maintain a connection every second, this reverts back to the $O(N)$ per node cost of flat proactive routing. In the worst case, where a constant fraction of the nodes can be expected to set up k connections, and k grows linearly with N, the overhead incurred will be $O(N^2)$.

A performance optimization used aggressively in DSR [13] is route caching, where intermediate nodes are allowed to send a route response, if they have recently observed a route to the desired destination. This can result in greatly improved performance, but there is also a high risk of *route poisoning*, where intermediate nodes unwittingly return routes that are no longer accurate. In general, reactive routing has been shown through simulation to scale better than flat proactive routing in most considered scenarios.

As we will see below, proactive routing has an advantage which a purely reactive protocol lacks: the ability to cluster nodes and aggregate routes. As we shall see in the following sections, clustering and address aggregation have the potential to drastically reduce the protocol overhead of a proactive routing protocol, as network size increases. The relationship between the overhead of reactive and proactive routing under different traffic scenarios is

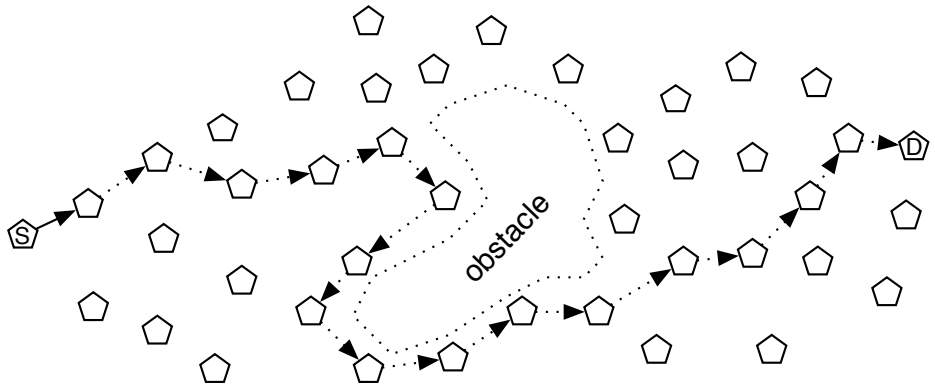

Figure 1.2: Geographical routing. The next hop is selected on greedily, until there is no neighbor that is closer to the destination than the current node. When this happens, routing switches to the *right-hand rule* until the obstacle has been successfully routed around.

discussed in more detail in [8].

1.4 Geographical Routing

> *The control overhead of geographical routing is typically $O(1)$, not counting location management. However, geographical routing relies heavily on two assumptions: a) that each node knows its position, and b) that the geographical distance between nodes corresponds well to the distance between these nodes in the network topology. In many situations, these assumptions are unacceptable.*

Geographical routing protocols make use of the geographical location of a node to make routing decisions. Such location information would generally be acquired either from GPS satellites, or from location interpolation given the positions of neighboring nodes.

In addition to knowing it's own geographical location, a node also need to know the locations of its neighbors, as well as the location of its intended destination. Dream (Distance Routing Effect Algorithm for Mobility) [2] and Grid Location Service [17] are mechanisms for finding out the location of any given node in the network. In Dream, nodes periodically an flood their location information throughout the network. However, as the flood travels away from the source, the speed with which updates are propagated is decreased, thereby

drastically reducing the overall overhead of the protocol. This is similar to the technique used in Fisheye routing [22] to reduce the cost of disseminating link state updates. In GLS, the location for a given node is stored at an *anchor node*. An anchor node is defined as the node positioned closest to a geographic location which is determined by hashing the node identifier. Every node is responsible for keeping its anchor node up-to-date on its current location. This method of distributing responsibility for storing location information is highly scalable and efficient, given that some characteristics of the network are known, such as the extent of the network in geographical terms.

Geographic routing protocols include Greedy Perimeter State-less Routing (GPSR) [14] and Location Aided Routing (LAR) [15]. GPSR greedily routes packets to the one-hop neighbor that is closest to the destination. Should an obstacle appear between source and destination, GPSR uses a planarized version of the network graph, and follows the "right hand rule" to route around the obstacle. The technique is illustrated in Figure 1.2, where a packet destined for node D is originated at S. When the large obstacle in the middle of the network is encountered, the *right hand rule* is triggered, routing the packet around it. The use of the *right hand rule* for routing around obstacles can result in paths of length $O(N)$, making greedy routing a risky proposition unless the characteristics of the topology is known in advance. LAR uses the geographic location of the destination to guide a reactive route lookup. By limiting the route request flood to neighbors in the approximate direction of the destination, the cost of route setup is reduced.

Any geographical routing protocol relies on the assumption that the geographical distance between two nodes corresponds well to their distance in the network topology. In scenarios where this is not the case, such as sparse or heterogeneous networks, or networks with directional or wired links, geographical routing is unlikely to achieve acceptable performance.

1.5 Zone-Based Routing

Zone-Based Routing combines the merits of flat proactive and pure reactive routing. However, while these hybrid protocols are more efficient than the component protocols they are made

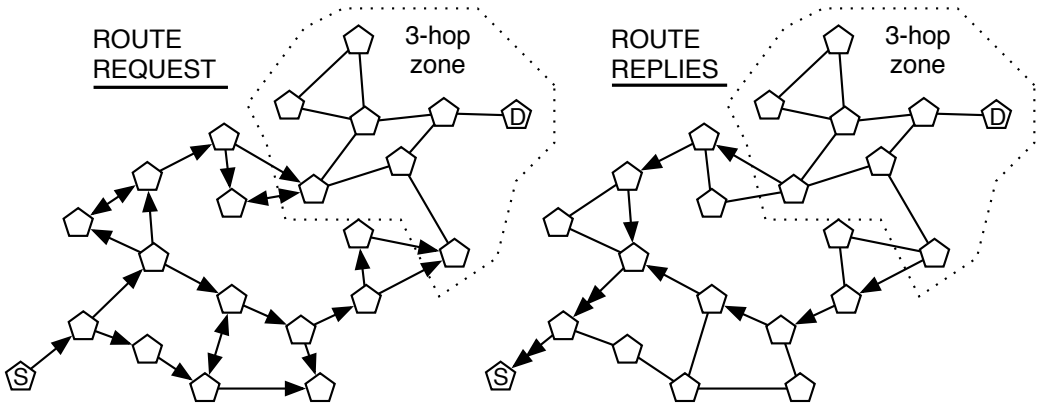

Figure 1.3: Hybrid proactive-reactive routing with SHARP. Route requests are flooded until they reach a node within the destination's proactive zone.

up out of, the asymptotic scalability of zone based routing is the same as that of other flat routing protocols.

In Zone Routing Protocol (ZRP) [12] and Sharp Hybrid Adaptive Routing Protocol (SHARP) [28], the merits of proactive and reactive routing are combined to form two hybrid proactive-reactive protocols. Both protocols follow a similar architecture. Around every node, a *zone* of d hops is maintained in which proactive routing is performed. For all destinations outside the zone, reactive route requests are used to establish a route. As soon as a route request reaches a node in the zone of the intended destination, this node replies with a route response.

In ZRP, the size of the zones can be varied depending on the mobility and traffic characteristics of the network [20]. Thanks to the proactive routing information available within the zone, the damaging effect of the flood is limited, as route requests can be efficiently routed to the edges of the zone, using a technique called *bordercasting*. Several other techniques are introduced to minimize the duplication of effort that could otherwise happen due to zone overlap.

While ZRP introduces its own routing components, such as *bordercasting*, SHARP is a straightforward combination of proactive and reactive routing. In SHARP, every node individually adapts the size of its zone, i.e. the distance (in hops) up to which its proactive routing updates should be forwarded. Reactive routing is done according to whatever re-

active protocol is used, with the modification that intermediate nodes that have proactive routing information for the desired destination node are allowed to reply to the route request. SHARP trades off the constant overhead of proactive routing against the high incremental cost of reactive routing by adaptively tuning the zone size of a node to correspond to the popularity or usage profile of the node. In addition to improving performance for popular nodes, the same trade-off is used to achieve desired packet delivery ratio and delay characteristics.

However, if properly done, route caching in reactive routing protocols can likely achieve a constant-term savings in terms of protocol overhead. Moreover, neither route caching, nor the hybrid approaches taken in ZRP and SHARP, can efficiently handle the case where there are frequent connection establishments, unless traffic is concentrated to a small number of nodes. In order for SHARP to achieve a successful trade-off between flat proactive and reactive routing, it is necessary for a few nodes to receive the majority of the network traffic.

Compared to flat proactive routing, or pure reactive routing, this middle ground between reactive and proactive routing can be expected to achieve lower overhead and delay under many traffic scenarios. However, although hybrid methods can be expected to reduce routing overhead, they only do so by a constant factor.

1.6 Single-Level Clustering

> *Single-level clustering improves scalability with respect to the network size (N) if the size of each cluster can be set to* $\sqrt{(N)}$*. In this case the control packet overhead is* $\Theta(\sqrt{N})$*. With constant size clusters, overhead remains at* $\Theta(N)$*. Certain node mobility patterns (M) can have a larger detrimental effect on the performance of clustering protocols than they have on flat protocols.*

Several protocols propose to use clustering to improve routing protocol scalability. *Clustering* is a process by which neighboring nodes form connected subsets, with one node elected as the cluster-head. Depending on the clustering technique used, clusters can be of radius of one or more hops from the cluster head. The cluster-heads may have responsibilities in addition

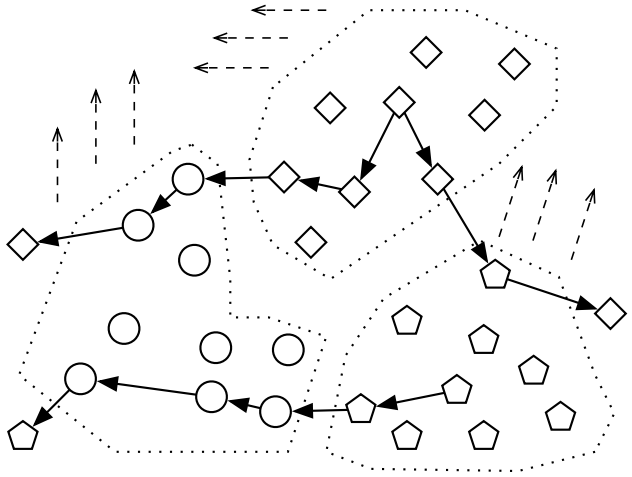

Figure 1.4: Mobile groups and stray nodes in LANMAR. Nodes move together in groups, stray nodes are handled with separate distance vector entries.

to that of a regular node, such as inter-cluster routing and intra-cluster coordination.

In hierarchical protocols, such as LANMAR and CGSR [21, 5], routes are aggregated by cluster. Inside a cluster, every node has complete routing information for every other node in the cluster. Externally, however, only a route to the cluster as a whole is published. Packets are first routed toward the cluster head of the destination. Once the cluster head, or simply any node within the destination cluster, has been reached the packet is routed directly toward its final destination within the cluster. Through this technique, a smaller amount of routing state is necessary on each node, and intra-cluster changes in the topology do not affect external routes. Note that all routing schemes that use clustering for routing will incur a cost in terms of increased path length. However, this cost is usually negligible compared to the savings achieved by reducing the amount of routing overhead incurred.

In contrast to the hybrid schemes mentioned earlier, which rely on flooding, hierarchical schemes also need to keep track of which cluster a node belongs to. This is sometimes referred to as *location management*. Depending on the assumptions used, location management can be a crucial factor in the performance of a clustering-based routing protocol.

In LANMAR, it is assumed that most nodes will remain in the same cluster throughout their lifetimes, and group membership is determined at network initialization. The authors

use a *group mobility* model, which applies mostly to military scenarios. LANMAR builds on ideas from Landmark Routing [32] and Fisheye State Routing (FSR) [22]. Nodes within a cluster exchange link state information using FSR. In addition to this link state information, each node keeps a distance vector table for a specific node in each cluster. This node is referred to as the Landmark. Any stray nodes, i.e. nodes that are not directly connected to their home cluster, are handled as special cases: a separate distance vector routing entry is kept by every node on the shortest path between the Landmark node and the stray node. Assuming that only a constant number of nodes stray from their home clusters, asymptotic control packet overhead is $\Theta(\sqrt{N})$.

In CGSR [5], one-hop clustering is performed, and is mainly used for transmission scheduling. A technique is also proposed in which each node globally advertises its cluster membership, and routing entries are kept only for clusterheads. Since the cluster radius is limited to a single hop, a cluster will contain only a constant number of nodes, leading to, at best, a constant improvement in the overhead incurred.

Both of these protocols rely on nodes staying within their original clusters throughout their lifetimes, or overhead will grow quickly. More flexible and scalable location management methods have been developed, and these will be discussed in the upcoming sections.

As with the hybrid scheme above, these single-level clustering protocols only reduce overhead to at best $O(\sqrt{N})$, depending on the cluster size. In the next section, we will discuss how to extend the idea of cluster based routing to reduce the size of the routing tables from $O(\sqrt{N})$, to $O(logN)$.

1.7 Multi-Level Clustering

Multi-level clustering protocols scale well with network size (N), frequency of connection establishment (F), and the number of concurrent connections (C). The number of control packet transmissions per node is $\Omega(\log^2 N)$. *For large networks, the address size in bits is* $\Omega(\log^2 N)$, *which in practice could easily grow beyond the limit of feasibility.*

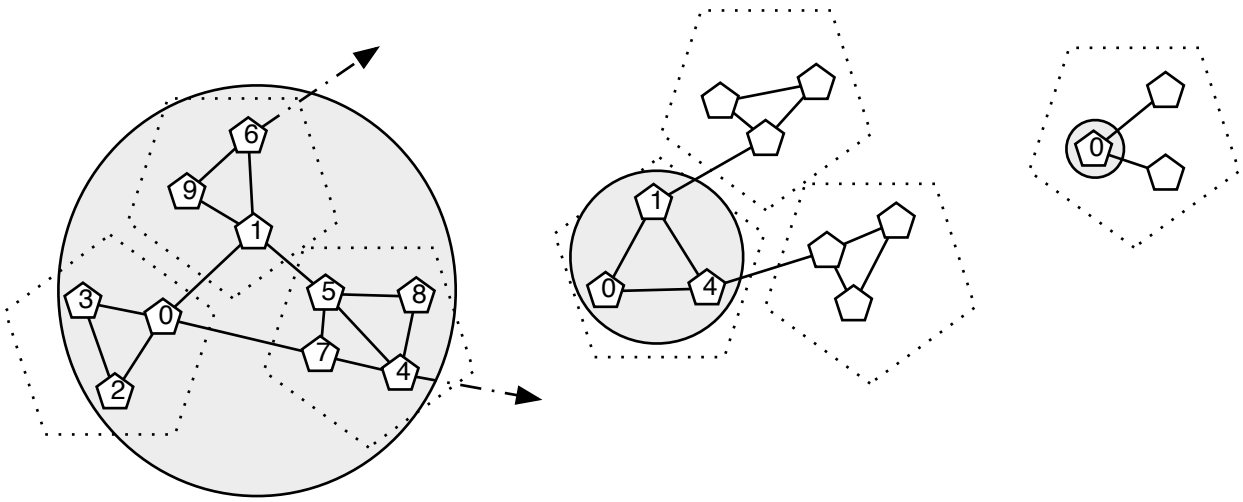

Figure 1.5: An example multilevel cluster hierarchy. At the left, individual nodes with their respective node IDs, partial view. In the middle, level-1 clusters forming level-2 clusters, and to the right, a level-2 cluster view of the entire network.

The ability to achieve true routing scalability with respect to network size (N), under most common scenarios, has so far only been demonstrated through the use of multi-level clustering. In these protocols, physical nodes cluster first into level-1 clusters. Then, up to d level-1 clusters, are further clustered into level-2 clusters, and so on. In general, with a *clustering degree* of d, the size of the routing table will be on the order of $O(d \log_d N)$.

In addition to reducing the size of the routing table, multi-level clustering will also make the network appear much less dynamic, as link state changes within a given cluster generally are not propagated to nodes that are not part of the cluster. This will reduce the overall control packet overhead under node mobility.

Examples of multi-level clustering are Hierarchical State Routing (HSR) [23], and MMWN [27]. These are link-state protocols, and they use the clustering abstraction to define *virtual links* between clusters. Instead of keeping track of all links in the network, a node now only needs to maintain entries for the virtual links going to or from a cluster in which the node is a member, a much smaller number.

Initially, one-hop physical-level clusters are formed as in the previous section, by electing a cluster head and having nodes in the k-hop neighborhood of that node join the cluster head

to form a cluster. To build the next higher clustering level, the cluster heads of neighboring clusters elect a higher-level cluster head from among themselves. Once the cluster hierarchy is formed, each node creates its own hierarchical identifier (HID), by concatenating the identifiers of all the clusterheads from the root of the hierarchy to the node in question. In theory, the size of a node identifier is $\Theta(\log N)$ bits, which results in an asymptotic HID size of $\Theta(log^2 N)$. However, in practice node identifiers are typically 48-bit MAC addresses.[2]

Data packet headers contain their destination HID. Routing is performed one level at a time: First, a packet is routed directly to the lowest-level cluster head in the HID which exists in the current node's routing table. Once this cluster head has been reached, the routing proceeds to the next lower level, as indicated in the HID. Eventually, the intended destination is reached, through the recursive application of this procedure.

Another example is Landmark routing, which is similar to HSR and MMWN, but uses a Distributed Bellman-Ford-like routing scheme, and does not concentrate traffic to clusterheads to the same extent. First described in [32], Landmark routing establishes a set of self-elected Landmark nodes in multiple levels. The main difference in Landmark routing is how the hierarchy is formed. Here, each Landmark periodically broadcasts an advertisement, announcing its presence. Depending on the level, k, of the Landmark, the advertisement will travel r_k hops. A new node initially assigns itself level 0, and sends an advertisement. If it can hear the advertisement of a level 1 Landmark, it may remain at level 0 and select that Landmark as its parent. If it does not, it cooperates with its Level-0 neighbors to elect a new Level-1 node. This process is repeated until a small subset of the nodes in the network are Level-d Landmarks, where r_d is larger than the diameter of the network. At this point, the Landmark hierarchy is complete.

An interesting difference between Landmark routing other multilevel clustering schemes in that several Landmarks can cover a single node, giving the node several valid addresses. Landmark routing was the basis for LANMAR mentioned in the previous section, and was later extended in L+ [3], and Safari routing [18].

[2]Beside the obvious practical reasons for using MAC addresses, it is worth noting that if a node identifier is to be constant throughout the lifetime of a node, it must be unique not only in the current network, but in every network it could conceivably be part of. The only way to feasibly assign identifiers to ensure this, is to assign every node a globally unique ID, which is the sole purpose of the current 48-bit MAC addresses used by network interface cards.

Safari routing [18] is similar in many respects to Landmark routing. Landmark nodes, here called *drums*, self-elect and form a multi-level Landmark hierarchy. One major difference is that the Safari hierarchy does not extend all the way to the physical (node) level. Instead, it extends down to the level of a *fundamental cell*, consisting of approximately 10 to 100 nodes. Inside a fundamental cell, routing is done by Dynamic Source Routing (DSR) [13]. Note that this is the opposite of Zone-based routing, where the local scope is handled by proactive routing, and distant nodes are served through reactive route requests. In Safari, the local scope is handled by DSR, and proactive routing is used for computing routes to more distant nodes, to avoid the high cost of long-range reactive route requests. If the size of the fundamental cell is kept constant, the size of the routing table in Safari is $O(log\ N)$.

Routing based on multi-level clustering, just like single-level clustering and geographical routing, needs a mechanism through which a node can acquire the current location (HID, or Landmark address) of its intended destination. This has been addressed in a variety of ways, including assumptions of group mobility [21] (which makes the problem go away by assuming nodes stay with their original clusters), flooding [5], Mobile IP-style home agents [23], and distributed location servers [30, 27, 3, 18, 7, 8]. The distributed location server is the most versatile and scalable of these options. Here, the responsibility for storing the current location of a given node is distributed across the network. MMWN uses a combination of a hierarchical organization of location servers together with *paging*, essentially a restricted flood, to find the current location of a node. In [30, 3, 18], a different method is used, similar to the *anchor node* idea in GLS [17]. To find the anchor node of a node i, a function $hash(ID_i)$ is computed. For every level, the cluster with the identifier most similar to $hash(ID_i)$ is selected as the cluster to which the anchor node should belong, until eventually a level-0 cluster (a single node) has been reached. This is the *anchor node* which is responsible for storing the current location of node i. In [7, 8], a similar $hash(ID_i)$ is computed, and the node with the *routing address* most similar to $hash(ID_i)$ is the anchor node for node i. This will be discussed in more detail in section 1.8. In several of these location management schemes, multiple anchor nodes are selected, such that there are many anchor nodes close to the node, and fewer anchor nodes far away from it. This improves the scalability of the distributed location server, as local changes and requests only have an effect on local network resources.

Multi-level clustering protocols that depend on hierarchical identifiers (including Landmark Addresses) are highly sensitive to changes in the clustering hierarchy: whenever a clusterhead gets disconnected or otherwise leaves the cluster, a new clusterhead must be elected, and all the nodes within the affected cluster need to update their hierarchical identifiers. In addition, the election of a new clusterhead will change the *anchor node* relationships, causing a necessity for a location-handoff mechanism. This has been identified in [30] as the dominating component of the total routing overhead of multi-level hierarchical routing protocols. This takes the total number of control packet transmissions per node in routing protocols based on multi-level clustering to $\Omega(\log^2 N)$, where every packet is of length $\Omega(log\ N)$ bits.

1.8 Dynamic Address Routing

Dynamic address routing is similar to routing based on multi-level clustering, but the address size is reduced to $\Omega(\log N)$ *from the* $\Omega(\log^2 N)$ *address size required with the earlier multi-level clustering protocols. Dynamic address routing is also less sensitive to node movement than previous multi-level clustering approaches, since its routing addresses are not built up from individual node identifiers.*

Dynamic address routing, described in [7, 8] takes the idea of multi-level clustering one step further. Whereas previous multi-level clustering schemes use cluster identifiers to form addresses (HID's), dynamic address routing dynamically assigns a considerably shorter *index* to each cluster. The *routing address* of a node is formed by concatenating the *indexes* of the cluster that the node belongs to at every level. In more detail, with a clustering degree of d, each of the $1 \ldots d$ clusters belonging to the same higher level cluster get an index in the range $0 \ldots d-1$. The more lengthy cluster identifiers are used only to ensure that there is a single, unique cluster per index.

As shown in Table 1.1, the difference in size between hierarchical identifiers and routing addresses is dramatic. With a low clustering degree d, the size of the hierarchical identifiers can be quite daunting. In contrast, the routing addresses used in dynamic address routing are roughly constant with respect to d.

d	Routing Table Size	Routing Address Size	Hierarchical ID Size
2	10	10	480
4	15	10	240
16	45	12	144
64	126	12	96
1024	1023	10	48

Table 1.1: Routing Table and Address Sizes for a 1024 node network Varying d. Address and ID sizes in bits. Changing routing address size is due to rounding to the nearest $log_2 d$ bit word.

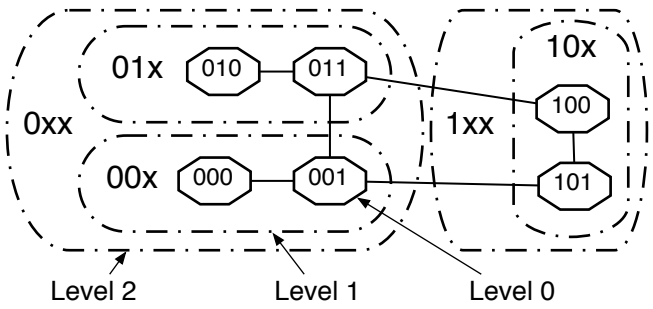

Figure 1.6: A network topology and 3-level clustering. Nodes have 3-bit routing addresses, with each bit selecting one out of two possible clusters at a given level in the hierarchy.

Moreover, the size of the routing table in a multi-level clustering hierarchy is equal to

$$(d-1)log_d N;$$

there are log_d levels, and $d-1$ routing entries per level. As shown in Table 1.1, selecting $d = 2$ minimizes the size of the routing table. Clearly, $d = 2$ is not feasible with previous multi-level clustering protocols, as the size of the hierarchical identifier is unacceptably large. Instead, these protocols are forced to use higher clustering degrees. Regardless of the choice of d, the address size in a regular multilevel clustering protocol is likely to exceed that of a dynamic address routing protocol by at least an order of magnitude, together with a marked increase in routing table size. In the rest of this section, we will assume $d = 2$ for dynamic address routing, since this choice of d minimizes the size of the routing table.

Conveniently, with $d = 2$ the *index* can be represented using a single bit. Figure 1.6 shows an example address allocation for a 6-node network. Since $log_2\ 6 < 3$, 3 bits of address is sufficient for this small network. The most significant bit of the address selects the top-level

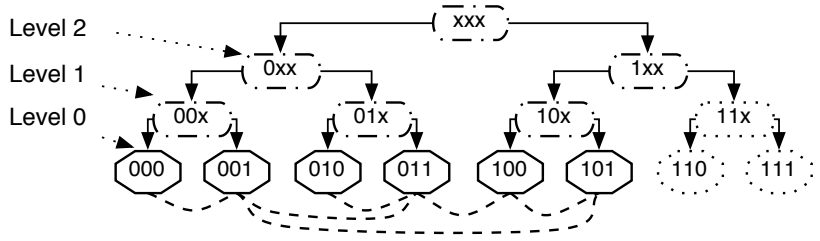

Figure 1.7: The address tree of a 3-bit binary address space. Leaves represent actual addresses, whereas inner nodes represent groups of addresses with a common prefix. Dashed lines show physical connectivity between nodes, corresponding to Figure 1.6.

cluster.

Since $d = 2$, we can also think of the cluster hierarchy as a binary tree, as shown in Figure 1.7. The root of the tree represents the entire network. The leaves of the tree represent nodes, and the internal nodes of the tree represent clusters. Each node (leaf) has one routing entry for every level of the tree. This routing entry indicates the path to the other subtree (cluster) at any given level. For example, the node with address [000] would have routing entries for subtrees [001], [01x] and [1xx]. If it wanted to route a packet to the node with address [100], it would lookup the routing entry for the subtree [1xx]. After an additional routing step, the packet reaches the node with address [101]. This node has routing entries for subtrees [100], [11x], and [0xx], and is able to forward the packet to its final destination.

One definition of a cluster in the routing context is that the nodes in a cluster form a connected subgraph in the network topology. Since address prefixes uniquely identify clusters in dynamic address routing, nodes with a common address prefix need to have the same property, which is called the *prefix subgraph* constraint. Ensuring that this constraint is satisfied is the primary objective of dynamic address allocation. Next, we will describe how this is handled in DART, the dynamic address routing protocol described in [8].

Address Allocation

Dynamic address allocation has many things in common with clustering in multi-level hierarchical networks. However, since dynamic address routing does not rely on concatenating

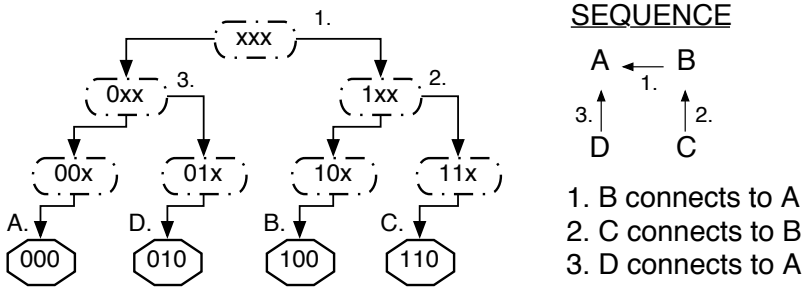

Figure 1.8: Address tree for a small network topology. The numbers 1-3 show the order in which nodes were added to the network.

unique identifiers to form its *routing address*, a major concern is to ensure the uniqueness of the addresses allocated.

When a node joins an existing network, it uses the periodic routing updates of its neighbors to identify and select an unoccupied and legitimate address. In more detail, every null entry in a neighbor's routing update indicates an empty subtree. This subtree represents a block of free, and valid routing addresses. By definition, the prefix constraint is satisfied if the two subtrees under a given parent are connected, and any empty subtree in a neighbor's routing update by definition shares a parent with the neighbor's subtree at the same level.

Let us see an example of address allocation. Figure 1.8 illustrates the address allocation procedure for a 3-bit address space. Node A starts out alone with address [000]. When node B joins the network, it observes that A has a null routing entry corresponding to the subtree [1xx], and picks the address [100]. Similarly when C joins the network by connecting to B, C picks the address [110]. Finally, when D joins via A, A's [1xx] routing entry is now occupied. However, the entry corresponding to sibling [01x] is still empty, and so, D takes the address [010].

To handle cluster merging and splitting, each cluster, or subtree in this case, is loosely associated with the lowest of all the identifiers of the nodes that belong to that subtree. This is called the **subtree identifier**. With node mobility, subtree identifiers may need to be updated, but these updates are piggybacked on the periodic routing updates at little extra cost. When the node with the lowest identifier within any subtree leaves the subtree, the identifier of that subtree will have to be recomputed. However, this is generally a

non-disruptive process, since the route updates from the new lowest identifier node in the subtree will propagate, and eventually reach all the concerned nodes without forcing any address changes in the process. Note that because of this, the routing address of a node does not depend directly on the identifiers of a set of clusterheads. Therefore, if the node with the lowest identifier gets disconnected, we can expect to see a smaller effect on the cluster hierarchy.

Due to node mobility, clusters will sometimes be partitioned into two or more parts. When this happens, the prefix subgraph constraint does not hold, and the clustering is thus invalid. The solution is to have one of the two partitions acquire new addresses as soon as the partitioning event is detected. The remaining problem is to detect such an event. As described above, subtree identifiers are assigned to be the minimum identifier in the cluster. If the cluster partitions, one of the two partitions will quickly compute a new identifier, as routing updates propagate through the cluster. However, a mere change of the cluster identifier is not enough to accurately diagnose a partitioning event. It could simply be that the node with the lowest identifier went out of range, or ran out of battery power. Instead, all route advertisements are made to contain the identifier of the destination subtree. The idea is that in the event that a node receives two routing updates for the same subtree, but with different identifiers, only the update with the lower identifier prevails and gets forwarded further. In addition, when a node perceives a route to its own address subtree, but with a lower identifier, it must acquire a new address. This solution also solves the problem of network merging: if two networks merge, this event will be detected as one or more cluster partitionings, causing some or all of the nodes in one of the two networks to immediately acquire new, valid addresses.

Distributed Location Server

Like in several other types of routing protocols, dynamic address routing protocols need a distributed location server.

The main problem in designing any distributed location server is to find an effective

method to select the *anchor node* of any given identifier. The solution proposed for dynamic address routing protocols is similar to that used in multi-level clustering protocols. However, the methods do not depend on any node identifier, except that of the destination node. A global and a priori known function $hash(ID_i)$, which takes a node identifier ID_i, and returns a bit string with the same length as the *routing address* is defined. Second, the $hash(ID_i)$ for the desired destination node, i, is calculated. If there exists a node that occupies this address, then that node is the **anchor node**. If there is no node with that address, then the node with the most similar address[3] is the **anchor node**.

For example, using figure 1.7 for reference, let us consider a node with identifier ID_1, that has a current routing address of [010]. This node will periodically send an updated entry to the lookup table, namely $<ID_1, 010>$. To figure out where to send the entry, the node uses the hash function to calculate an address: $hash(ID_1)$. If the return value of the hash function is [100], the packet will simply be routed to the node with that address. However, if the returned bit string was instead [111], the packet could not be routed to the node with address [111] because there is no such node. In such a situation, the packet gets routed to the node with the most similar address to [111], which in this case would be [101].

To improve the scalability of the distributed location server, each lookup entry is stored in several locations, at increasing distances from the destination node. By starting with a small, local lookup and gradually going to further away locations, nodes can avoid sending lookup requests across long distances to find a node that is nearby. Similarly, when a node makes a small address change, it need only contact nearby location servers with the location update, as the records at distant location servers will still be sufficiently accurate to guide the packets to the correct neighborhood, where more recent information is readily available.

Coping with Temporary Route Failures. On occasion, due to link or node failure, a node will not have a completely accurate routing table. This could potentially lead to lookup packets, both updates and requests, terminating at the wrong node. The end result of this is that requests cannot be promptly served. In an effort to reduce the effect of such intermittent errors, a node can periodically check the lookup entries it stores to see if a route

[3]The metric used here for similarity between addresses can be described as the integer value of the XOR result of the two addresses.

to a more suitable host has been found. If this should be the case, the entry is forwarded in the direction of this more suitable host.

Requests are handled in a similar manner: if the request could not be responded to with an address, it is kept in a buffer awaiting either the arrival of the requested information, or the appearance of a route to a node which more closely matches the *hash* of the identifier the request was in regard to. This way, even if a request packet arrives at the **anchor node** before the update has the anchor, the request will be buffered and served as soon as the update information is available.

Dynamic Address routing has size $O(logN)$ routing tables and an $O(logN)$ address size. The $\Omega(log^2N)$ result for location management handoff shown in [30] has not yet been shown for dynamic address routing. However, there are considerable structural similarities between the distributed location server described in that paper, and the one described for dynamic address routing. Moreover, dynamic address routing has a decreased reliance on node identifiers for clustering and addressing. These observations lead us to conjecture that the lower bound on per-node channel utilization for control packets is at most $\Omega(log^3N)$.

1.9 Conclusions

In this chapter, we have discussed variety of aspects of ad hoc routing scalability. We deliberated the various routing protocols that have been proposed over the past decade in order to understand how these scale with respect to various parameters. In order to achieve true scalability, it is the belief of the authors that multi-level clustering is the only viable option. While geographic routing is an attractive alternative for certain niche applications, multi-level clustering applies well to all scenarios, except for those with extremely high mobility.

From a scalability perspective, dynamic address routing represents the current state of the art in scalable ad hoc routing. The use of dynamic address routing, a variation on multi-level clustering, results in addresses of length $\Omega(log\ N)$. This is considerably shorter

than the hierarchical identifiers used in previous multi-level clustering protocols, which are of size $\Omega(\log^2 N)$. Dynamic address routing achieves a similar average routing table size of $\Theta(log\ N)$ and offers a reduced dependence on node identifiers for ensuring the stability of clustering and location management.

In summary, scalable ad hoc routing continues to be a focal point of interest in terms of making the deployment of large scale ad hoc networks a reality.

References

[1] Alan D. Amis, Ravi Prakash, Dung Huynh, and Thai Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *INFOCOM (1)*, pages 32–41, 2000.

[2] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *ACM/IEEE MOBICOM*, 1998.

[3] Benjie Chen and Robert Morris. L+: Scalable landmark routing and address lookup for multi-hop wireless networks, 2002.

[4] T. Chen and M. Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proc. of IEEE ICC*, 1998.

[5] C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks. In *The IEEE Singapore International Conference on Networks*, 1997.

[6] Clausen and Jaquet. Rfc 3626: Optimized link state routing, 2003.

[7] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. Peernet: Pushing peer-2-peer down the stack. In *IPTPS*, 2003.

[8] Jakob Eriksson, Michalis Faloutsos, and Srikanth Krishnamurthy. Scalable ad hoc routing: The case for dynamic addressing. In *IEEE INFOCOM*, 2004.

[9] J. J. Garcia-Luna-Aceves, Marc Mosko, and Charles E. Perkins. A new approach to on-demand loop-free routing in ad hoc networks. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 53–62. ACM Press, 2003.

[10] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. In *INFOCOM*, pages 1360–1369, 2001.

[11] P. Gupta and P. Kumar. Capacity of wireless networks, 1999.

[12] Z. Haas. A new routing protocol for the reconfigurable wireless networks, 1997.

[13] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[14] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[15] Y.-B. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *ACM/IEEE MOBICOM*, 1998.

[16] Young-Bae Ko, Vinaychandra Shankarkumar, and Nitin H. Vaidya. Medium access control protocols using directional antennas in ad hoc networks. In *INFOCOM (1)*, pages 13–21, 2000.

[17] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MOBICOM '00)*, pages 120–130, August 2000.

[18] Ahamed K. Mohammed, Rudolf H.Johnson Reidi, David B., Peter Druschel, and Richard Baraniuk. Analysis of safari: An architecture for scalable ad hoc networking and services. Technical Report TREE 0304, Rice University, 2004.

[19] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.*, 1(2):183–197, 1996.

[20] Marc R. Pearlman and Zygmunt J. Haas. Determining the optimal configuration for the zone routing protocol. *IEEE Journal on Selected Areas in Communication*, 17(8), august 1999.

[21] G. Pei, M. Gerla, and X. Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *ACM MobiHOC'00*, 2000.

[22] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *ICC (1)*, pages 70–74, 2000.

[23] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang. A wireless hierarchical routing protocol with group mobility. In *WCNC*, 1999.

[24] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94*, 1994.

[25] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector rout-

ing. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90. IEEE Computer Society, 1999.

[26] Ram Ramanathan. On the performance of ad hoc networks with beamforming antennas. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 95–105. ACM Press, 2001.

[27] Ram Ramanathan and Martha Steenstrup. Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. *Mobile Networks and Applications*, 3(1):101–119, 1998.

[28] Venugopalan Ramasubramanian, Zygmunt J. Haas, and Emin Gün Sirer. Sharp: a hybrid adaptive routing protocol for mobile ad hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 303–314. ACM Press, 2003.

[29] John Sucec and Ivan Marsic. Clustering Overhead for Hierarchical Routing in Mobile Ad hoc Networks. In *IEEE INFOCOM 2002*, New York, NY, June 23–27 2002.

[30] John Sucec and Ivan Marsic. Hierarchical routing overhead in mobile ad hoc networks. *Mobile Computing, IEEE Transactions on*, 3, Jan 2004.

[31] C-K. Toh. Associativity-Based Routing For Ad Hoc Mobile Networks. *Wireless Personal Communications Journal*, 4(2):103–139, March 1997.

[32] Paul F. Tsuchiya. The landmark hierarchy : A new hierarchy for routing in very large networks. In *SIGCOMM*. ACM, 1988.