Optimal Tradeoffs for Location-Based Routing in Large-Scale Ad Hoc Networks

Taejoon Park, *Student Member, IEEE,* and Kang G. Shin, *Fellow, IEEE*

***Abstract*—Existing location-based routing protocols are not versatile enough for a large-scale ad hoc environment to simultaneously meet all of the requirements of scalability, bandwidth efficiency, energy efficiency, and quality-of-service routing. To remedy this deficiency, we propose an optimal tradeoff approach that: 1) constructs a hybrid routing protocol by combining well-known location-update schemes (i.e., proactive location updates within nodes' local regions and a distributed location service), and 2) derives its optimal configuration, in terms of location-update thresholds (both distance- and time-based), to minimize the overall routing overhead. We also build a route-discovery scheme based on an Internet-like architecture, i.e., first querying the location of a destination, then applying a series of local-region routing until finding a complete route by aggregating the thus-found partial routes. To find the optimal thresholds for the hybrid protocol, we derive the costs associated with both location updates and route discovery as a function of location-update thresholds, assuming a random mobility model and a general distribution for route request arrivals. The problem of minimizing the total cost is then cast into a distributed optimization problem. We first prove that the total cost is a convex function of the thresholds, and then derive the optimal thresholds. Finally, we show, via simulation, that our analysis results indeed capture the real behavior.**

***Index Terms*—Location-based routing, mobile ad hoc networks, optimal tradeoffs, random mobility model.**

I. Introduction

An AD HOC network is a self-configurable wireless network in which mobile, battery-powered devices dynamically create and change the network topology without relying on any infrastructure or administrative support. It offers unique benefits for certain environments and applications, but there are still several open issues to be resolved before realizing these benefits, including multihop routing, energy conservation, and mobility management. Specifically, we need an efficient routing protocol that discovers/maintains routes to any nodes in the network, minimizing computation/communication overhead and energy consumption. Key challenges in ad hoc routing are the large number of nodes, the wide range of node mobility and the limited battery energy.

Manuscript received August 27, 2002; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. Haas. This work was supported in part by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant F49620-00-1-0327 and by the Defense Advanced Research Projects Agency (DARPA) under AFRL Contract F30602-01-02-0527.

The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: taejoonp@eecs.umich.edu; kgshin@eecs.umich.edu).

Digital Object Identifier 10.1109/TNET.2005.845539

Traditionally, routing protocols have been classified as either *proactive* or *reactive*. Proactive protocols [1]–[4] are similar to Internet routing protocols in the sense that each node maintains routing information for all possible destinations. Whenever the topology changes, the proactive protocol updates routes even if no data traffic needs to be transported over them at the time of the topology change. Hence, as the number of nodes or average node mobility increases, the overhead of maintaining the routing information grows rapidly. Reactive protocols [5]–[8], on the other hand, discover routes only when they are needed to transport data. Changes in those parts of the network that are not in use do not necessarily generate route-update messages. However, they introduce a large initial search overhead which grows rapidly with the number of nodes or rate of route requests. Attempts have also been made to make a tradeoff between the proactive and reactive protocols, as reported in [9]–[12]. For instance, in ZRP [9], [10], each node uses proactive routing inside its local zone while initiating an on-demand search for those destinations located outside of the zone. The key advantage of this protocol is scalability, as it limits the route-update overhead by the zone size and reduces the route-search overhead through the use of local proactive knowledge.

The routing performance can be significantly improved by utilizing location information of nodes. That is, if each node is capable of accurately locating other nodes in the network (i.e., the destination and all its neighbors), it can *geographically* forward the packets toward its destination,[1] as shown in [13]–[15]. However, in practice, it is difficult to find/maintain node locations with accuracy in an ad hoc environment where nodes move around. More recent approaches [16]–[19] address this issue and present ways of distributing location updates and finding destination based on obsolete location information. DREAM [16] forces nodes to proactively flood their current location information over the entire network, enabling each node to build a complete location database. LAR [17] uses (possibly stale) geographic location to reactively determine the search space for a destination, hence reducing the number of route-discovery packets. However, these protocols do not scale well to large networks due mainly to their use of global flooding. By contrast, GLS [19] or landmark routing [20] achieves scalability by providing a location service that provides a mapping from node_id to the current location (similarly to the distributed lookup service for Internet peer-to-peer networks [21]–[26]). For instance, GLS lets each node maintain its current location at a small subset of network nodes, called the node's *location servers*. The route discovery for a destination

[1]In the geographic forwarding or greedy routing, a source or an intermediate node sends the packet to one of its neighbors closest to the packet's destination.

is then equivalent to recursively querying the location servers until the query packet arrives at the one having the destination's location. However, GLS has its own drawbacks: 1) it has poor route-discovery performance for a closer destination if it still relies on recursive location-queries; 2) it has limited support for discovery of energy-efficient or quality-of-service (QoS) routes due mainly to its use of single-path routing; and 3) it is difficult for each node to maintain or hand over location servers as they may move around.

Discovering and maintaining a route under the energy or QoS constraints (e.g., the total energy consumption, the minimum residual energy, end-to-end delay, the bandwidth reservation, and so on) is challenging, especially in a large-scale, mobile network. Single-path routing protocols like geographic forwarding or GLS are not suitable for this purpose. Multipath routing [27]–[30], on the other hand, is capable of finding energy-efficient, QoS routes as well as simplifying the route-recovery process, as it constructs multiple routes on-demand, one of which is used as the primary route while the rest as backups. It is, therefore, desirable to have a routing protocol that looks for enough candidate routes between any pair of nodes (*multipath*), without incurring too much routing overhead (*localized*).

A location-based routing protocol must have the following salient features: 1) scalability to a large number of nodes; 2) support for discovery/recovery of minimum-cost[2] routes for *any* destinations; and 3) bandwidth- and energy-efficiency in terms of minimizing the amount of routing overhead and prolonging the lifetime of each node. But, unfortunately, none of the existing routing protocols can simultaneously satisfy all of these requirements. Clearly, the problem of location-based routing is far from being solved (as stated in [33]), and there is a growing demand for efficient location management schemes. In this paper, we study how to achieve *optimal* location management and efficient route discovery on top of location-basd routing; we propose a tradeoff-based approach that: 1) combines several useful schemes to our advantage (an algorithmic tradeoff), and 2) optimally adjusts the relevant thresholds to minimize the routing overhead (a tradeoff within the protocol itself). In particular, the two main contributions of this paper are:

- development of a routing protocol as a hybrid of existing location-based schemes, that addresses the above requirements, and
- derivation of optimality conditions for the proposed location-update scheme, that enables each node to determine when to update its location.

To our best knowledge, this is the first analytical approach to formulate the minimum achievable routing overhead in a given location-update scheme and derive the optimal configuration associated with it.

The hybrid routing hinges on a location-update scheme that combines the proactive location updates within nodes' local regions and a distributed location service. That is, nodes update their current location information at a small number of other nodes either when their timers expire (*time-based*) or when their movement distances since their last update exceed a certain threshold (*distance-based*). The route discovery is then achieved using nodes' location caches as an Internet-like routing framework, i.e., it first queries the location of a destination, then applies a series of local-region routing to construct a complete route as a concatenation of partial (regional) routes. The proposed protocol scales well with the network size (thanks to its small storage overhead), and effectively provides energy-efficiency, QoS routing and route recovery (thanks to its localized, multipath routing).

Clearly, there exists a tension between keeping accurate location information and avoiding excessive overhead. The location-update overhead would increase with the frequency of location update, while the route-discovery overhead gets smaller as the frequency of location update is raised. On the other hand, the larger the update interval, the smaller the location-update overhead gets, while the more obsolete the cached location information becomes, thus increasing the route-discovery overhead. Hence, the thresholds should be so chosen as to keep the location-update overhead reasonably low while maintaining location information up-to-date enough to determine the asymptotic direction to each destination. To make an optimal tradeoff, we derive the condition for minimizing the routing overhead as a closed-form formula with all the relevant parameters. To this end, we first present models for route requests and node mobility,[3] then derive the total routing overhead of the network associated with both location update and route discovery, and finally establish a *distributed* optimization problem. For distributed optimization, we define the route-discovery overhead of node i as the expected amount of route-request packets that node i receives from all other nodes. We then show that the total overhead of i is a convex function of the location-update threshold of i only, hence guaranteeing existence of the optimal threshold. This implies that each node can determine its own optimal thresholds based on location-update and route-discovery overheads of itself; since the sum of overheads of all nodes yields the total overhead of the entire network, a set of thresholds determined as above is indeed globally optimal. We finally show, via simulation, that these analytical results closely match the real behavior, verifying their correctness.

The rest of this paper is organized as follows. Section II describes the proposed hybrid routing protocol. Section III presents models for node mobility and route requests, and then derives the routing cost and the optimal thresholds. Section IV presents the simulation results. Finally, the paper concludes with Section V.

II. Proposed Hybrid Routing

We propose a routing protocol, a hybrid of existing location-based protocols, that offers salient features such as scalability, bandwidth-efficiency, energy- and QoS-awareness, and support for efficient rerouting. The heart of the proposed protocol is the location-update scheme, in which each node: 1) updates its current location at nodes within its proximity of radius $\mathcal{R}$ as well as its location servers chosen distributively over the entire network,

[2]To achieve shortest-path, energy-aware and/or QoS routing, we may apply combinations of the following routing metrics: the number of hops, the total transmission power [31], the residual battery energy [32], the end-to-end delay [8] and so on.

[3]The incoming route requests are assumed to follow a general distribution to account for any real situation. The node mobility follows a Rayleigh distribution with a random mean squared speed, according to the random mobility model.

and 2) stores, as a recipient of location updates, locations of a small number of other nodes.[4] Based on nodes' location caches, we also develop a route-discovery scheme, featuring recursive location query, direction-based multipath route search, energy- and QoS-aware routing, and efficient route recovery. We start the protocol description by summarizing assumptions and formats of location databases and control packets, and then detail the location-update and route-discovery schemes.

1) Assumptions: Each and every node is equipped with a GPS device, and hence, can: 1) accurately determine its current geographic location and time, and 2) calculate its average speed for a certain period of time. Each node exchanges a BEACON packet (containing its current location) with all its neighbors[5] so it can keep its list of neighbors up-to-date. Each node may decide when to send the BEACON packet as described in [34]. If a node misses several BEACON packets from one of its neighbors, it removes the node from its neighbor list. Finally, links between two neighboring nodes i and j are assumed to be symmetric, i.e., if node i can send a packet to node j, so does node j to node i. This condition is easy to be met by adjusting the transmission power of the two nodes.

2) Location Database: Our location-update scheme (detailed in Section II-A) leads to the construction of two location databases at each node: 1) LocalLocDB storing the locations of all nodes inside the node's local region (of radius $\mathcal{R}$), and 2) DistributedLocDB for the locations of chosen nodes outside of the local region. An entry of these databases will be removed if it has not been updated for a certain timeout interval, and a new entry will be inserted if the node receives a location update from a new node. Both databases scale well with the network size, because LocalLocDB keeps only the proximity information regardless of the network size and DistributedLocDB grows in $\mathcal{O}(\log N_{\text{net}})$ where N_{net} is the total number of network nodes. Each node also maintains a list of locations of its neighbors based on BEACON messages exchanged.

3) Control Packets: We define and use the following control packets.

- **LINFO** (Location INFOrmation): is used by a node i to inform other nodes of its current location and speed, and contains: `packet_type`, either `server_id` (for `type_A`) or an update radius $\mathcal{R}$ (for `type_B`), `source_id` $(= i)$, the current location and speed of node i, and the timestamp when the packet was generated. It also delivers node i's residual battery energy (QoS parameters) for energy-aware (QoS) routing.
- **LQRY** (Location QueRY): is generated by a node s to search for the location of another node d (not cached in node s's DistributedLocDB). The LQRY packet is then geographically forwarded to node i chosen from node s's DistributedLocDB, whose ID is closest to that of destination d. The packet contains the ID and current location of node s, `destination_id` $(= d)$ and `query_id` $(= i)$.
- **RREQ** (Route REQuest): is sent by a node s (either a source or an intermediate node) to its selected neighbors when it wants to find routes to a target b whose location has been cached in node s's LocalLocDB. Neighbors that the RREQ packet should be sent to, are determined based on the past location and speed of node b (as described in Section III-E). The RREQ packet contains the ID and current location of node s, `target_id` $(= b)$, `sequence_number`, the number of hops taken so far, and the IDs of the intermediate nodes traversed thus far.
- **RREP** (Route REPly): is generated, and unicast back to the source s, by the recipient of the RREQ packet, node b. It contains `source_id` $(= b)$, `destination_id` $(= s)$, the number of hops (copied from the RREQ packet), the IDs of the nodes (copied from the RREQ packet) through which the RREP packet is to be forwarded.

A. Location Update

Our location-update scheme is a combination of the distributed lookup service [19], [21]–[26] and the proactive location management within the local region of each node [16]. Hence, each node i maintains its current location at a small subset of nodes, which consist of:

- location servers chosen according to the "closeness" in the node ID space throughout the entire network,[6] and
- all nodes within its local region of radius $\mathcal{R}$.

Each node triggers a location update either when its UpdateTimer expires (time-based) or when the distance displacement since the last update exceeds a certain threshold (distance-based). The location-update procedure then determines the node's current location and speed, builds and distributes the LINFO packets, and finally resets UpdateTimer. The LINFO distribution indeed performs two types of location updates as follows:

- **Updates to the Location Servers**: The sender generates as many LINFO packets as the number of its location servers, each marked as `type_A` and containing the ID of the location server the packet will be unicast to. The routing of LINFO packets follows, e.g., the procedure of GLS.
- **Updates to the Local Region**: The sender constructs, and floods within the region of radius $\mathcal{R}$, a LINFO packet of `type_B` and radius $\mathcal{R}$. Optionally, we can further reduce the overhead of LINFO flooding by utilizing the distance effect [16], i.e., as two nodes get farther away, they can do away with less accurate location information in each other's database. For instance, the update radius field of subsequent LINFO packets from the same node contains a sequence of values: $\mathcal{R}/4, \mathcal{R}/2, \mathcal{R}, \mathcal{R}/4, \ldots$. This sequence has the effect of updating nodes within radius $\mathcal{R}/4$ every time it broadcasts, while updating nodes within $\mathcal{R}/2$ and $\mathcal{R}$ at a lower rate.

Each node also executes a periodic cleanup task that checks and removes obsolete entries in DistributedLocDB/LocalLocDB.

[4]For the rest of the paper, $\mathcal{R}$ is regarded as a constant network parameter.

[5]Node i is said to be a *neighbor* of node j if node i is directly reachable by node j.

[6]For instance, each node elects and maintains, for each subarea of the network, a node with an ID close to node i's ID as a location server, as described in [19]. One may also apply the concept of distributed lookup service for Internet peer-to-peer networks such as [21]–[26].

Upon receiving the LINFO packet sent by node i, a node j processes and/or relays the packet as needed. If the timestamp in the LINFO packet indicates that the information in the packet is older than, or equal to, what is in DistributedLocDB or LocalLocDB (if any), then node j discards the packet else proceeds as:

U1. **if** `packet_type` equals `type_A`:
U1.1. **if** `server_id` equals node j's ID, node j updates its DistributedLocDB with the location, speed and timestamp of node i;
U1.2. **else** node j geographically forward the LINFO packet to `server_id` (e.g., as described in [19]);
U2. **if** `packet_type` equals `type_B`:
U2.1. node j calculates distance to node i, $\Delta_{i,j}$, using the locations of itself and node i;
U2.2. **if** $\Delta_{i,j} < \mathcal{R}$, node j updates (or inserts) its LocalLocDB with the location, speed and timestamp of node i, and then re-broadcasts the LINFO packet to its own neighbors.

B. Route Discovery

In this section, we develop an energy- and QoS-aware route-discovery protocol for a *large-scale* ad hoc network, incurring as small a routing overhead (including the amount of control traffic and the database size) as possible. The route discovery problem here is to find routes between a source s and a destination d (whose location is usually unknown to node s) satisfying the energy-efficiency or QoS constraints, when each node in the network maintains the location information for: 1) nodes within its proximity (in LocalLocDB) and 2) a small subset of the network nodes (in DistributedLocDB). To this end, we use the following routing components: the recursive location query of [19], the direction-based route search based on [16] and [17], and the geographic forwarding protocol [13]–[15]. Based on these components as well as the location databases, we propose a scheme capable of finding a sufficient number of candidate routes between nodes s and d and then selecting the best one among them.

To make it scalable, the proposed route discovery is built upon an Internet-like architecture in that: 1) it queries the location of a destination (similarly to the DNS lookup), and 2) it applies the route search within its local region (like the intra-AS routing) repeatedly until the route is discovered (like the inter-AS routing). To minimize the route-discovery latency, we apply steps 1) and 2) to the forward $(s \rightarrow d)$ and backward $(d \rightarrow s)$ paths, respectively. Following is a complete description of the route-discovery protocol:

D1. **if** node d's location is cached in node s's LocalLocDB:
D1.f. s initiates a (direction-based) procedure for searching for a route to d, as described in Section II-B2;
D1.r. d waits for RREQ packets from s for a certain period of time, decides which route to use, and informs s of the established route;
D2. **else**:
D2.f1. **if** d's location is found in s's DistributedLocDB, s geographically forward a LQRY packet to d;
D2.f2. **else**, s initiates a recursive location query for d's location (Section II-B1);

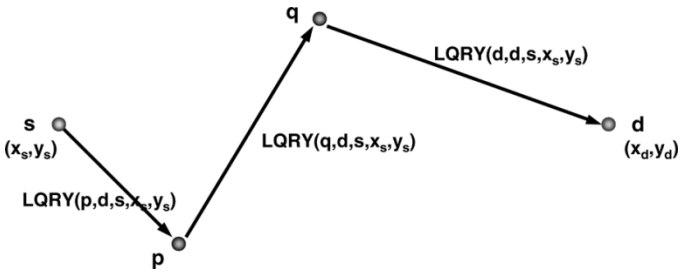


Fig. 1. Recursive location query from s to d.

D2.r1. d selects a border node b_1 in its LocalLocDB, then initiates a route-search procedure toward b_1 (Section II-B2);
D2.r2. b_1, after receiving RREQ packets from d, selects the best route between b_1 and d, and searches routes to its own border node b_2 (Section II-B2);
...
D2.rn. s receives RREQ packets from b_{n-1} and establishes a route between s and d.

In what follows, we describe the location-query algorithm, the direction-based route search, the concept of energy-aware routing for large-scale networks, and the route-recovery scheme.

1) Location Query: Since a source s maintains the location information of only a small subset of the entire node space, it usually has no knowledge of the destination d's location. Hence, we need a mechanism for s to query d's current location based on its DistributedLocDB only. The recursive query is such a mechanism: it attempts to reduce the *virtual* distance to d on the `node_id` space as the query progresses. That is, s sends a LQRY packet (using geographic forwarding) to the smallest ID node p, whose ID is greater than or equal to d's, for which s knows location information. Likewise, node p forward the LQRY packet, and so on. Hence, the LQRY packet will eventually reach a location server of d which will forward the packet to d. Fig. 1 illustrates this procedure: the LQRY packet initiated by s arrives at d via nodes p and q (see [19] for details of the location query process). The LQRY packet contains the current location of its initiator s. After receiving the LQRY packet, d may either send a reply packet back to s to inform its up-to-date location or initiates the route-search procedure. Note the latter effectively reduces the route-discovery latency.

2) Route Search: The direction-based route search can be viewed as a local-area routing protocol that discovers, based on initiator's LocalLocDB, routes between an initiator s and a target b located within s's local region. The route search from s to b consists of the following:

1) looking up s's LocalLocDB for b's past location, speed and timestamp;
2) estimating the range of b's current location using this information;
3) sending a RREQ packet to only those neighbors lying within the estimated range of the path toward b.

The process of determining the search scope is illustrated in Fig. 2. At time t_0, node b broadcasted its current location, (x_0, y_0) and speed, $\hat{\sigma}_b$, to all nodes. Thus, nodes i, j, k, and s updated their location databases with (x_0, y_0) and $\hat{\sigma}_b$. At a

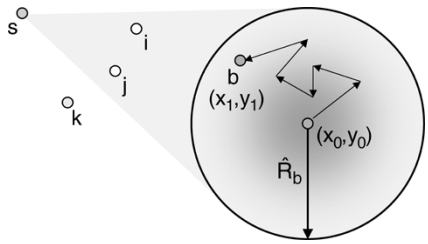


Fig. 2. Determination of the search scope at s.

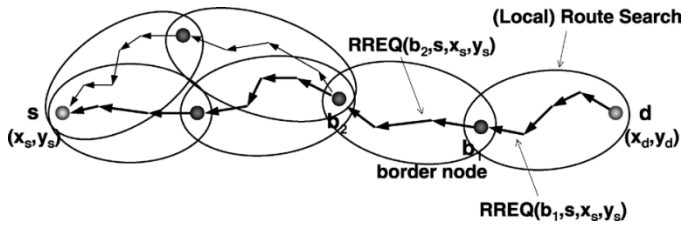


Fig. 3. Energy-aware route discovery for large-scale ad hoc networks.

later time t_1, node s initiates the search for a route to node b. Node s first decides the search scope for node b, which is fully captured in $\hat{R}_b$ and (x_0, y_0).[7] Then, node s determines which of its neighbors lie within the search scope, based on the location information of all its neighbors. As a result, node s sends a RREQ packet to nodes i and j, but not to node k.

If a node receives a RREQ packet seeking a route to b and if it knows how to get to b because it has recently communicated with b or is in the neighborhood of b, it relays the RREQ directly to b using the route it knows of. Otherwise, it estimates the range of b's location using the information in its own LocalLocDB, and relays the RREQ packet to those neighbors that lie within the estimated range of the path to b. This procedure repeats itself at each intermediate node until the RREQ packet reaches b. Each intermediate node that relays the RREQ packet will include its ID in the node list. This "cooperative" procedure makes the RREQ packet very likely to arrive at b, causing only local flooding. Upon receipt of the first RREQ packet originated from s, node b waits for several RREQ packets traversed different routes and determines the best one as the "winner". Then, b unicasts (via source routing) a RREP packet back to s, indicating that the route has been found.

When initiating a route search, s inserts a unique `sequence_number` into the RREQ packet to detect duplicate RREQ packets at intermediate nodes as follows. Each intermediate node maintains a list of [`source_id`, `sequence_number`] pairs that it has recently received (as shown in [5]). When a node receives a RREQ packet, it should look for the [`source_id`, `sequence_number`] pair of the received packet in its internal list of recently received RREQs, and, if found, should discard the packet.

The route search may fail for the following reasons. First, b has moved out of the search scope, and hence, no RREQ packet can reach b. To avoid this type of failure, s may retry the route-search with a wider search scope after the previous route-search timed out. Second, route-search failures may occur when $\hat{R}_b$ is much smaller than the distance to the destination, since no neighbors are likely to be within the search scope. This happens if the last-known speed of the destination is small or the destination's location has recently been updated. In this case, s either: 1) relies on the geographic forwarding or 2) forward the RREQ packet to two of its neighbors closest to the search scope, each chosen from the left-hand (right-hand) side.

3) Energy-Aware Routing for Large-Scale Networks: Based on the route-discovery protocol discussed so far, we propose an *energy-aware* routing protocol for large-scale networks that conserves energy while prolonging nodes' lifetime.[8] The energy-aware routing problem can be stated as that of finding multiple routes between two nodes knowing locations of each other, and then selecting the best energy-aware route among them. According to the proposed route discovery, s has made its location available to d by sending a LQRY packet (as shown in Fig. 1), and hence, d initiates energy-aware routing back to s, thus minimizing the route-discovery latency.

We want to disfavor the poor routes that may quickly drain the nodes' battery energy rather than finding the best energy-aware route. We achieve this by repeatedly applying the route-search algorithm (in Section II-B2) until the RREQ packet reaches s, as depicted in Fig. 3. That is, d chooses a *border* node, say b_1, from its LocalLocDB. To be eligible for the border node, b_1 must: 1) lie along the path toward s; 2) have sufficient residual energy; and 3) be the farthest from d among those in d's LocalLocDB. Node d then initiates the search for all partial routes to b_1 through nodes within the elliptical area. After receiving RREQ packets from d, b_1 declares one as the best energy-aware partial route, and then repeats the same procedure to find partial routes to its own border node b_2. This procedure repeats itself until a complete energy-aware route is constructed. Note that multiple border nodes may be chosen in each step of the route discovery (e.g., two border nodes in the third step of Fig. 3). This has the effect of finding more candidate routes at the expense of higher route-discovery overhead.

Since border nodes have sufficient battery energy and each partial route would be the most energy-efficient within the local region, the concatenated route from d to s will likely be the most energy efficient without any bottleneck node. Then, from the symmetry assumption on neighboring links, the route $s \rightarrow d$ is also energy efficient.

4) Route Recovery: In a network where nodes move rapidly, a node on an ongoing route moves away, thus breaking the route. This is more likely to happen to longer routes. Therefore, we need an efficient route-recovery scheme to patch up the broken route, still meeting the energy or QoS requirements. It is easy to realize the route recovery in our protocol, thanks to the localized, multipath routing. That is, if node i detects its next-hop node is no longer available: 1) it selects another node j participating in the route which is eligible to be a border node; 2) it initiates the route search to node j to find a new partial route; and 3) it fixes the route with a patch $i \rightarrow j$.

III. Optimal Tradeoffs

We derive the optimality condition for our proposed protocol as a closed-form formula that contains adjustable thresholds

[7]See Section III-E for determination of $\hat{R}_b$.

[8]One can apply the same methodology to QoS routing.

and all the relevant parameters. Our location-update scheme are both distance- and time-based. The distance-based location update is performed whenever the difference between current and last-sent locations is greater than a threshold γ. The time-based location update is triggered whenever a certain amount of time, T, has elapsed since the last location update and there were no distance-based updates during this period.

For optimization, we establish analytical models for route requests and node mobility as follows. The incoming route requests are modeled to have a *general* distribution with a constant mean arrival time. Hence, our model can determine the optimal cost and thresholds for *any* actual distribution. The model for node mobility is based on the *random mobility model*, and is characterized by the Rayleigh speed distribution with a mean squared speed parameter. Although the mean arrival time and the mean squared speed are, in reality, *random* and time-varying, we can always find good estimates for them, and hence, treat them as known constants during the time interval of interest.

We then derive the total cost associated with both location updates and route discovery as a function of the location-update threshold. The location-update cost is derived from the expected number of location updates per route request. It is a function of the location-update threshold γ. The route-discovery cost is derived using the expected search scope within a local region, which is a function of the interval between the last location update and a new route request. We then develop a *distributed* formula for determining a set of globally optimal thresholds, through each of which a node can find its own optimal location-update threshold using information available to itself only. For distributed optimization, we define the search cost of node i as the expected number of RREQ packets that node i receives from all other nodes. Next, we show that each individual search cost depends only on node's own threshold, then prove that the total cost of a node is a convex function of the threshold, and finally, derive the optimal location-update threshold of each node.

In this section, we describe our mobility and timing models, present key ideas for solving the optimization, derive location-update and route-discovery costs and the scope of route search, and finally, derive the optimal thresholds.

A. Mobility Model

1) Related Work: Efficiency in tracking nodes is of great importance to ad hoc routing. Random mobility models [35]–[39] are widely used for mobile ad hoc networks, and can be classified as either entity mobility models where each node moves independently of others, or group mobility models where a group of nodes take correlated movements. Here, we only consider the entity mobility model. The model of [35] assumed a Poisson random process for the change of each node's speed or direction, and approximated the distance traveled by the node during a time interval to follow a Rayleigh distribution. The authors of [38] presented a composite random mobility model that solves the speed-decay problem, i.e., average speed decreases until the speed converges to the long-term average. A realistic mobility model that incorporates the effects of obstacles was reported in [39].

2) Our Mobility Model: We establish an analytic model for node mobility based on the random mobility model to describe a node's movement during each location-update interval. This model is then used to derive statistical properties of the time interval for moving a given distance. We adopt the random mobility model since it is suitable for analysis, and, more importantly, it captures the crucial mobility patterns of nodes moving around in a local area; we may ignore the effects of nodes not following the random mobility model (e.g., moving straight without changing its direction, i.e., the direction does not follow a uniform distribution), since such effects are usually disfavored in our route discovery/recovery scheme. To account for the speed-decay problem, we model the mean squared speeds of nodes as random variables as described below.

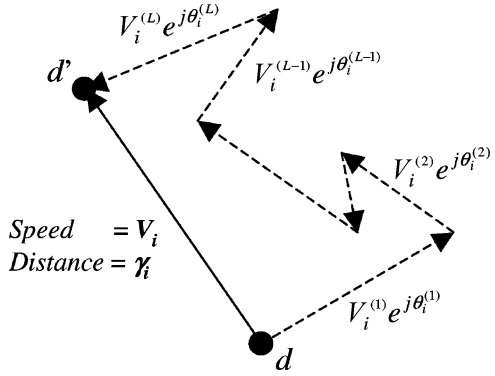


Fig. 4. Mobility model.

Fig. 4 depicts our mobility model. Each node's movements are comprised of a sequence of random intervals, called *mobility epochs*, during each of which a node moves along a constant direction at a constant speed. The speed and direction of each node varies randomly from epoch to epoch. The direction θ_i of node i's movement has a uniform distribution over $[0, 2\pi)$ and is independent of its speed. Node i's aggregate speed V_i between two consecutive location updates is modeled as a Rayleigh distribution with pdf $f_{V_i}(v)$ and mean squared speed σ_i^2, where σ_i is a random variable with pdf, $f(\sigma_i)$.[9] Moreover, given a good estimate $\hat{\sigma}_i$ of the speed, $f_{V_i}(v)$ is given by

$$f_{V_i}(v) = \frac{2}{\hat{\sigma}_i^2} v e^{\frac{-v^2}{\hat{\sigma}_i^2}}, \quad i = 1, \ldots, N. \tag{1}$$

To estimate σ_i, we use a *Maximum Likelihood* (ML) approach that finds the most likely σ_i, given past measurements of the node's speed $\underline{v}_i = [v_{i,1}, \ldots, v_{i,n}]^t$, satisfying $\hat{\sigma}_i = \arg\max_{\sigma_i \in [V_{\min}, V_{\max}]} f(\underline{v}_i|\sigma_i)$ where $f(\underline{v}_i|\sigma_i) = \prod_{k=1}^{n} (2v_{i,k}/\sigma_i^2) e^{-v_{i,k}^2/\sigma_i^2}$. Hence, the ML estimate $\hat{\sigma}_i$ is given by

$$\hat{\sigma}_i = \max\left[\min\left(\sqrt{\frac{1}{n}\sum_{k=1}^{n} v_{i,k}^2}, V_{\max}\right), V_{\min}\right]. \tag{2}$$

[9]By the Central Limit Theorem (CLT), the x and y components of V_i in the two dimensional Cartesian space can be approximated as uncorrelated, zero-mean Gaussian random variables. So, it is reasonable to model V_i follows the Rayleigh distribution.

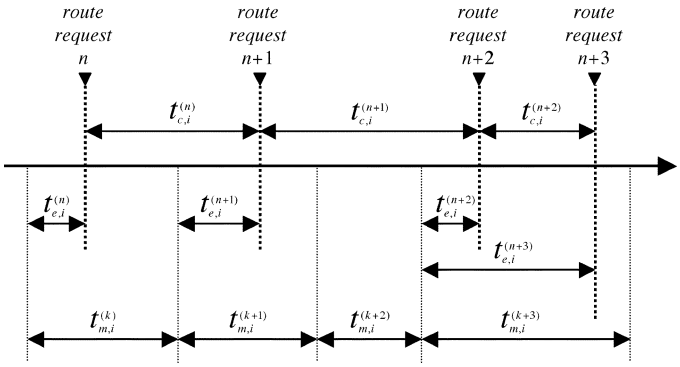


Fig. 5. Timing model.

B. Timing Model

Internet traffic measurements [40], [41] have shown that the conventional Poisson arrival assumption may sometimes fail to model the actual behavior, and that the intervals between route requests in the Internet have long-tail distributions, i.e., with tails that decay more slowly than exponentially. Hence, we adopt a very general model to account for actual characteristics including heavy-tailed traffic, in which the incoming route requests follow a *general* distribution, as shown in Fig. 5.

We use the following notation. Let $t_{c,i}$ denote the time interval at node i between two consecutive route requests with pdf $f_c(t)$ and mean $E[t_{c,i}] = \mu_c, \forall i$. Let $t_{m,i}$ denote the interval between two consecutive location updates by node i with pdf $f_{m,i}(t)$ and mean $E[t_{m,i}]$. Obviously, $t_{c,i}$ and $t_{m,i}$ are mutually independent. Finally, let $t_{e,i}$ denote the interval at node i between the last location update and the arrival of the next route-request with pdf $f_{e,i}(t)$. From Fig. 5, $t_{m,i}$ is the minimum of the time-based location-update interval, T_i, of node i, and also the time during which node i moves a distance γ_i with speed V_i. This accounts for both the distance- and the time-based location updates. Hence

$$t_{m,i} = \min\left(\frac{\gamma_i}{V_i}, T_i\right). \tag{3}$$

Using $f_{V_i}(v)$, we can derive $f_{m,i}(t)$ and $f_{e,i}(t)$. First, the pdf and cdf of γ_i/V_i, the interval for the distance-based updates, are expressed as

$$f_{\frac{\gamma_i}{V_i}}(t) = \frac{\gamma_i}{t^2} f_{V_i}\left(\frac{\gamma_i}{t}\right), \quad t \geq 0 \tag{4}$$

$$F_{\frac{\gamma_i}{V_i}}(t) = 1 - F_{V_i}\left(\frac{\gamma_i}{t}\right), \quad t \geq 0. \tag{5}$$

Hence, the pdf and cdf of $t_{m,i}$ are derived as

$$f_{m,i}(t) = \frac{\gamma_i}{t^2} f_{V_i}\left(\frac{\gamma_i}{t}\right) + F_{V_i}\left(\frac{\gamma_i}{T_i}\right)\delta(t - T_i), \quad 0 \leq t \leq T_i \tag{6}$$

$$F_{m,i}(t) = \begin{cases} 1 - F_{V_i}\left(\frac{\gamma_i}{t}\right), & 0 \leq t \leq T_i \\ 1, & t \geq T_i \end{cases}. \tag{7}$$

Then, we get

$$E[t_{m,i}] = \int_o^{T_i} \frac{\gamma_i}{t} f_{V_i}\left(\frac{\gamma_i}{t}\right) dt + T_i \cdot F_{V_i}\left(\frac{\gamma_i}{T_i}\right). \tag{8}$$

Finally, from the random observer property of the alternating renewal processes [42], [43], we can express $F_{e,i}(t)$ and $f_{e,i}(t)$ in terms of $f_{V_i}(v)$ as follows:

$$F_{e,i}(t) = \frac{E[\min(t_{m,i}, t)]}{E[t_{m,i}]} = \frac{\int_0^t [1 - F_{m,i}(u)]\, du}{E[t_{m,i}]} \tag{9}$$

$$f_{e,i}(t) = \begin{cases} \frac{1}{E[t_{m,i}]} F_{V_i}\left(\frac{\gamma_i}{t}\right), & 0 \leq t < T_i \\ 0, & t \geq T_i \end{cases}. \tag{10}$$

C. Key Idea in Solving the Optimization Problem

We propose that the time-based location-update threshold T_i for node i be dynamically adjusted according to the distance-based threshold and the node's speed. Specifically, T_i is updated to be proportional to γ_i and inversely proportional to $\hat{\sigma}_i$, whenever γ_i and $\hat{\sigma}_i$ are re-computed, as follows:

$$T_i = \kappa \cdot \frac{\gamma_i}{\hat{\sigma}_i} \quad i = 1, \ldots, N \tag{11}$$

where κ is a constant. This makes $E[t_{m,i}]$ a function of γ_i and $\hat{\sigma}_i$ only, and $E[t_{m,i}]$ is derived, from (8), as

$$E[t_{m,i}] = g(\kappa) \cdot \frac{\gamma_i}{\hat{\sigma}_i} \tag{12}$$

where

$$g(\kappa) = 2\sqrt{\pi} \cdot Q\left(\frac{\sqrt{2}}{\kappa}\right) + \kappa \cdot \left(1 - e^{\frac{-1}{\kappa^2}}\right) \tag{13}$$

and $Q(x)$ is defined by $Q(x) = (1/\sqrt{2\pi}) \int_x^\infty e^{-t^2/2} dt$, $x \geq 0$. By comparing the expressions for T_i and $E[t_{m,i}]$, we find that both T_i and $E[t_{m,i}]$ have the same structure, and $E[t_{m,i}]$ depends only on γ_i and $\hat{\sigma}_i$. Consequently, the location-update cost will have a simple structure appropriate for optimization.

The dynamic time-based location-update is essential for the following reasons. First, using time-based location-updates ensures convergence of the protocol, even when nodes do not move at all. Second, the probability that time-based updates take place is fixed and given by $1 - e^{-1/\kappa^2}$, where κ is a constant. Third, dynamic adjustment of the time-based location-update interval makes the route-discovery overhead independent of the node's speed. This, in turn, yields an important result: there exists a globally optimal configuration, which is a function of the location-update threshold only.

D. Location-Update Cost

We derive the location-update cost of node s when incoming route requests have an arbitrary distribution. One may think that the location-update cost can be expressed as the expected route-request interarrival time divided by the expected location-update interval. However, unfortunately, it is difficult to derive it using $f_c(t)$, $f_{m,s}(t)$ and $f_{e,s}(t)$. The main reason for the difficulty is that the independent increment property doesn't hold for any nonexponential distribution. Hence, deriving the expected location updates per route-request is very complicated, albeit not impossible.

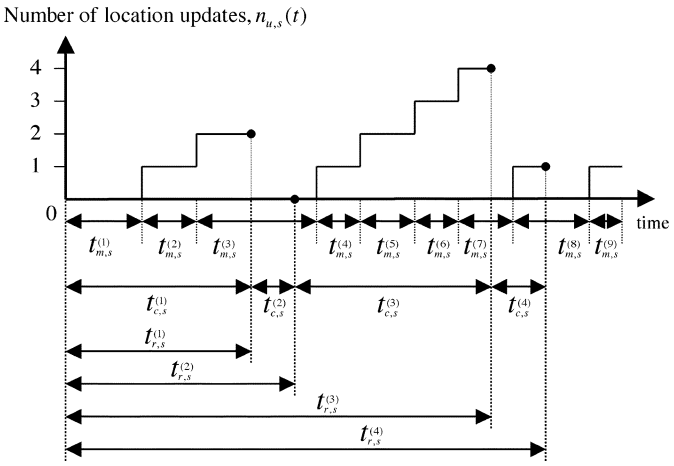


Fig. 6. Number of location updates at node s.

Our approach to this problem is to solve it in the time domain. We consider the average number of location updates instead of expected updates. Thanks to the ergodicity, this will give us the same result as the original problem. Let $N_{u,s}$ denote the average number of location updates by node s between two route requests. Let $\mathcal{U}_s$ be the the cost of s for each location update. Then, $\mathcal{U}_s$ is a (known) constant, since the expected number of nodes receiving the LINFO packets (i.e., those in the fixed local region of radius $\mathcal{R}$ and a constant number of distributed location servers) can be considered constant. Moreover, we apply, in deriving the location-update cost of node s, an average cost for each location-update, denoted as $\bar{\mathcal{U}}$, assuming $\bar{\mathcal{U}} \simeq \mathcal{U}_s, \forall s$. Let $C_u(s)$ denote the expected location-update cost per route-request of node s. $C_u(s)$ can be derived as the average number of location updates per route-request multiplied by $\bar{\mathcal{U}}$ as

$$C_u(s) = \bar{\mathcal{U}} \cdot N_{u,s}. \tag{14}$$

The key idea in deriving the update cost is to introduce the renewal processes associated with $t_{c,s}$ and $t_{m,s}$. Let $\{N_{c,s}(t), t \geq 0\}$ denote a renewal process of s for which the interarrival times, $\{t_{c,s}^{(n)}\}$ are independent and identically distributed with a distribution $f_c(t)$. Also, let $\{N_{m,s}(t), t \geq 0\}$ denote a renewal process of s for which the interarrival times are independent and identically distributed with a distribution $f_{m,s}(t)$. For $\{N_{c,s}(t), t \geq 0\}$, let

$$t_{r,s}^{(0)} = 0, \quad t_{r,s}^{(n)} = \sum_{k=1}^{n} t_{c,s}^{(k)}, \quad n \geq 1. \tag{15}$$

We can calculate $N_{u,s}$ by using the graphical description in Fig. 6 which plots the number of location updates, $n_{u,s}(t)$, at node s. Whenever a new route request arrives, $n_{u,s}(t)$ refreshes itself. $N_{u,s}$ is the time average of $n_{u,s}(t_{r,s}^{(n)})$ in the interval $[0, t]$, and is given by

$$N_{u,s} = \lim_{t \to \infty} \left[\frac{1}{N_{c,s}(t)} \cdot \sum_{n=1}^{N_{c,s}(t)} n_{u,s}\left(t_{r,s}^{(n)}\right) \right]. \tag{16}$$

$N_{u,s}$ can be expressed in terms of $N_{c,s}(t)$ and $N_{m,s}(t)$ as

$$N_{u,s} = \lim_{t \to \infty} \left[\left(\frac{t}{N_{c,s}(t)} \right) \cdot \left(\frac{N_{m,s}\left(t_{r,s}^{(N_{c,s}(t))}\right)}{t} \right) \right]. \tag{17}$$

According to the renewal theorem [43], we have

$$\lim_{t \to \infty} \frac{N_{c,s}(t)}{t} = \frac{1}{E[t_{c,s}]}, \quad \text{w.p.l.} \tag{18}$$

$$\lim_{t \to \infty} \frac{N_{m,s}\left(t_{r,s}^{(N_{c,s}(t))}\right)}{t} = \frac{1}{E[t_{m,s}]}, \quad \text{w.p.l.} \tag{19}$$

Therefore,

$$N_{u,s} = \frac{E[t_{c,s}]}{E[t_{m,s}]}. \tag{20}$$

Using $N_{u,s}$ and (12), we can derive the update cost $C_u(s)$

$$C_u(s) = \bar{\mathcal{U}} \cdot \frac{E[t_{c,s}]}{E[t_{m,s}]} = U \cdot \frac{\hat{\sigma}_s}{\gamma_s} \tag{21}$$

where

$$U = \frac{\bar{\mathcal{U}} \cdot \mu_c}{g(\kappa)}. \tag{22}$$

E. Route-Discovery Cost

We only consider the cost of RREQ packets, since the location-query overhead is, on average, constant and also smaller than route search. Let $\bar{\mathcal{N}}$ denote the average number of local-area route search per route-discovery. Then, the route-discovery cost is the expected cost of route search multiplied by $\bar{\mathcal{N}}$. We can consider the expected number of nodes per unit area receiving the RREQ packet to be constant (determined by the node density in the network). Let $\mathcal{S}$ be the (constant) cost for processing RREQ packets at nodes within a unit area. Then, we can relate the local-area search cost to the expected search area multiplied by $\mathcal{S}$, assuming that the distance between two randomly selected nodes is uniformly or exponentially distributed over the whole area.

As shown in Fig. 2, node s determines the search area to find the target b with the pre-specified success probability p_{succ}, and then sends a RREQ packet to nodes within that area only. Each intermediate node within the search area may receive multiple RREQ packets, as they are flooded within the search area. Moreover, the degree of replicating this RREQ packet gets higher as the search area gets widened. Hence, the route-search cost is proportional to the product of the power of estimated search scope $\hat{R}_b$ and the distance D ($\leq \mathcal{R}$) from node s to the last known location of b. Besides, the route-search algorithm forces each intermediate node to relay the RREQ packet to at least one or two of its neighbors, if any. This sets a certain lower bound, $C_{\min}$, of the search cost. (Note that the location-query cost may also be absorbed in $C_{\min}$.) Let $C_s(s, b)$ denote the expected search cost per route-request from s to b. Then, we have

$$C_s(s, b) = \mathcal{S} \cdot E\left[D \cdot \hat{R}_b^{\beta}\right] + C_{\min} \tag{23}$$

where β is a constant that accounts for the increase in the search cost due to duplicate RREQs. Let $C_s(s)$ denote the average search cost per route-request of node s. s initiates the route-search procedure to the border node b, instead of d outside s's local region, with an equally likely probability of choosing b. Therefore,

$$C_s(s) = \frac{1}{N_s - 1} \sum_{b \neq s} C_s(s,b) \tag{24}$$

where N_s is the average number of border nodes of s. Since $D \simeq \mathcal{R}$ if b is a border node, we have

$$C_s(s) = \frac{\mathcal{S}_\mathcal{R}}{N_s - 1} \sum_{b \neq s} E\left[\hat{R}_b^\beta\right] + C_{\min} \tag{25}$$

where $\mathcal{S}_\mathcal{R} = \mathcal{S} \cdot \mathcal{R}$. Note that the search cost is associated with three model parameters, $\mathcal{S}_\mathcal{R}$, β and $C_{\min}$, all of which are assumed to be known constants.

1) Determination of Search Scope: We want to determine the search scope such that the probability of b lying within that area is p_{succ}. The random variable, R_b $(= t_{e,b} \cdot V_b)$ denotes the distance moved by node b during time interval $t_{e,b}$ at speed V_b. Using the pdf of V_b, we can derive the conditional pdf, $f_{R_b}(r|t_{e,b})$, as follows:

$$f_{R_b}(r|t_{e,b}) = \frac{1}{t_{e,b}} f_{V_b}\left(\frac{r}{t_{e,b}}\right). \tag{26}$$

The conditional cdf of R_b is then

$$F_{R_b}(r|t_{e,b}) = F_{V_b}\left(\frac{r}{t_{e,b}}\right) = 1 - \exp\left[-\frac{r^2}{\hat{\sigma}_b^2 t_{e,b}^2}\right]. \tag{27}$$

The search scope $\hat{R}_b$ is chosen such that the probability of the destination node lying inside that scope is at least p_{succ} $(= 1 - p_{\text{fail}})$. So

$$F_{V_b}\left(\frac{\hat{R}_b}{t_{e,b}}\right) \geq p_{\text{succ}}. \tag{28}$$

By combining the above equations, $\hat{R}_b$ is determined as

$$\hat{R}_b = t_{e,b} \cdot \hat{\sigma}_b \cdot \left[\ln\left(p_{\text{fail}}^{-1}\right)\right]^{\frac{1}{2}} \quad p_{\text{fail}} = 1 - p_{\text{succ}} \neq 0. \tag{29}$$

The parameter $\hat{\sigma}_b$ corresponds to the square root of the mean squared speed of b. The search scope increases as b moves faster or the cached information gets obsolete. Also, a larger search scope is used for a higher "confidence" probability.[10]

2) Derivation of $E[\hat{R}_b^\beta]$: We then determine $E[\hat{R}_b^\beta]$ as follows. The β^{th} moment of $t_{e,b}$ can be derived, using (10), as

$$E\left[t_{e,b}^\beta\right] = \frac{1}{E[t_{m,b}]} \int_0^{T_b} t^\beta \cdot F_{V_b}\left(\frac{\gamma_b}{t}\right) dt = \frac{h(\kappa,\beta)}{\hat{\sigma}_b^\beta g(\kappa)} \cdot \gamma_b^\beta \tag{30}$$

[10]Since node movements are random, one can only guarantee that b still resides in the search scope with some confidence probability p_{succ}. By increasing p_{succ}, one can attain a higher route-search success probability, but it incurs higher search overhead. So, we would like to choose p_{succ} to guarantee a reasonably high probability of success in the search.

where

$$h(\kappa,\beta) = \frac{2}{\beta+1} \int_0^\kappa u^{\beta-2} \cdot e^{-\frac{1}{u^2}} du + \frac{\kappa^{\beta+1}}{\beta+1}\left(1 - e^{-\frac{1}{\kappa^2}}\right). \tag{31}$$

Note that we first used integration by parts, then plugged in (11) and (12). The expected search scope is then

$$\begin{aligned} E\left[\hat{R}_b^\beta\right] &= E\left[t_{e,b}^\beta\right] \cdot \hat{\sigma}_b^\beta \left[\ln\left(p_{\text{fail}}^{-1}\right)\right]^{\frac{\beta}{2}} \\ &= \gamma_b^\beta \cdot \frac{h(\kappa,\beta)}{g(\kappa)} \left[\ln\left(p_{\text{fail}}^{-1}\right)\right]^{\frac{\beta}{2}}. \end{aligned} \tag{32}$$

3) Derivation of $C_s(s)$: By combining the above equations, we obtain

$$C_s(s,b) = \gamma_b^\beta \cdot \mathcal{S}_\mathcal{R} \frac{h(\kappa,\beta)}{g(\kappa)} \left[\ln\left(p_{\text{fail}}^{-1}\right)\right]^{\frac{\beta}{2}} + C_{\min}. \tag{33}$$

Note that the route-search cost for arbitrary b is independent of b's mobility, as far as the time-based location-update thresholds of all nodes are inversely proportional to the mobility. The search cost (and the route-discovery cost, accordingly) depends only on the location-update threshold and the κ value. Therefore,

$$\bar{\mathcal{N}} \cdot C_s(s) = S \cdot \frac{1}{N_s - 1} \sum_{b \neq s} \gamma_b^\beta + C'_{\min} \tag{34}$$

where

$$S = \mathcal{S}_\mathcal{R} \bar{\mathcal{N}} \frac{h(\kappa,\beta)}{g(\kappa)} \left[\ln\left(p_{\text{fail}}^{-1}\right)\right]^{\beta/2} \quad C'_{\min} = \bar{\mathcal{N}} C_{\min}. \tag{35}$$

F. Optimal Thresholds

We want to derive a set of optimal thresholds $\{\gamma_i^* | i = 1, \ldots, N\}$, subject to $\gamma_i > 0$, $\forall i$, that minimizes C_{network}, the total cost of location update and route search per route-request of the entire network. Since all nodes are equally likely to be chosen as a border node, from (21) and (34), we have

$$\begin{aligned} C_{\text{network}} &= \sum_{i=1}^N \left[C_u(i) + \bar{\mathcal{N}} \cdot C_s(i)\right] \\ &= U \cdot \sum_{i=1}^N \frac{\hat{\sigma}_i}{\gamma_i} + S \cdot \sum_{i=1}^N \gamma_i^\beta + N \cdot C'_{\min}. \end{aligned} \tag{36}$$

Unfortunately, this optimization problem requires information on the entire network, which is not usually available. To solve this problem, we introduce a "target-oriented" search cost, which is defined as the expected number of RREQ packets that a node receives, as a border node, from all other nodes. Note that the target-oriented search cost has a useful property: the sum of all target-oriented search costs leads to the reconstruction of the same C_{network}. Let us derive a "distributed" formula for determining optimal thresholds that enables each node to calculate its own optimal location-update threshold

using information known to itself only. Let $C_s^{\dagger}(i)$ denote the target-oriented search cost of node i, defined as

$$C_s^{\dagger}(i) = S \cdot \gamma_i^{\beta} + C'_{\min}. \tag{37}$$

Then, we get

$$C_{\text{network}} = \sum_{i=1}^{N} \left[C_u(i) + C_s^{\dagger}(i) \right]. \tag{38}$$

Let $C(i)$ denote the total cost of node i, given by

$$C(i) = C_u(i) + C_s^{\dagger}(i) = U \cdot \frac{\hat{\sigma}_i}{\gamma_i} + S \cdot \gamma_i^{\beta} + C'_{\min}. \tag{39}$$

Then the optimization problem is equivalent to finding γ_i^* that minimizes $C(i)$ subject to $\gamma_i > 0$, for all i. We have

$$\frac{\partial^2}{\partial \gamma_i^2} C(i) = \beta(\beta-1) S \cdot \gamma_i^{\beta-2} + \frac{2U\hat{\sigma}_i}{\gamma_i^3}. \tag{40}$$

Since $U, S, \gamma_i > 0$, $(\partial^2/\partial\gamma_i^2)C(i) > 0$, $C(i)$ is a convex function. The optimum value, γ_i^*, satisfies the following relation:

$$\beta S \cdot (\gamma_i^*)^{\beta-1} - U\hat{\sigma}_i \cdot \frac{1}{(\gamma_i^*)^2} = 0. \tag{41}$$

Hence,

$$\gamma_i^* = \left[\frac{U\hat{\sigma}_i}{\beta S} \right]^{\frac{1}{\beta+1}} = \left[\frac{\bar{\mathcal{U}} \mu_c \hat{\sigma}_i}{\beta \mathcal{S}_{\mathcal{R}} \bar{\mathcal{N}} h(\kappa,\beta)} \left[\ln\left(p_{\text{fail}}^{-1}\right) \right]^{\frac{-\beta}{2}} \right]^{\frac{1}{\beta+1}}. \tag{42}$$

Finally, from (11), the optimal time-based location-update interval of node i is given by

$$T_i^* = \kappa \cdot \frac{\gamma_i^*}{\hat{\sigma}_i}. \tag{43}$$

Note that (42) and (43) include network parameters available to node i (i.e., $\hat{\sigma}_i$), hence ensuring optimal performance. Note also that a set of location-update thresholds given by (42) is indeed globally optimal, because the sum of all $C(i)$ equals C_{network}.

G. Discussion

The performance of our proposed protocol can be maximized when it is configured with the optimal thresholds, so chosen as to balance the update and search overheads. We have made several observations on the optimization results. First, the optimal distance-based update threshold, γ_i^*, is insensitive to the variations of the average route-request interarrival time, μ_c, and the average speed, $\hat{\sigma}_i$, since γ_i^* is proportional to $\sqrt{\mu_c}$ and $\sqrt{\hat{\sigma}_i}$, when $\beta = 1$. Second, (42) abstracts protocol-specific errors (e.g., imperfect location updates) or link-layer overheads (e.g., collisions/retransmissions) with parameters, $\bar{\mathcal{U}}$ and $\mathcal{S}_{\mathcal{R}}$. Thus, the location-update thresholds can be optimally adjusted if the network-wide estimates for these parameters are given.

Determining the exact values for γ_i^* and T_i^* requires prior knowledge of all parameters affecting the thresholds, as given by (42) and (43). However, they are actually unknown and vary with time and with network conditions. To keep track of the optimal thresholds in a time-varying network, we need to develop a parameter estimation algorithm or an adaptive scheme. This is our ongoing work and its results will be reported in a forthcoming paper. Assuming the existence of an algorithm for estimating the parameters at run-time, we can find the optimal thresholds corresponding to the current estimates of these parameters. Hence, each node periodically estimates the parameters and then calculates the optimal thresholds using the estimated parameters.

IV. Simulation Results

We would like to verify/confirm our protocol's ability to minimize the routing overhead while retaining a high route-discovery success probability. So, we simulated and measured the routing overhead, in bits per second per node, while varying the value of location-update threshold γ. We then compared the simulated routing overhead with the analytical results. Second, we show that dynamically adjusting γ to its optimum value is essential under the time-varying network condition. Thus, we evaluated the effect of the choice of γ on the routing performance. We measured the routing overhead and the probability of route-discovery success for several γ values by changing the route-request rate.

Note that we do not need detailed simulation of link-layer behavior, packet losses, and so on, because we are only interested in network-layer behaviors for various thresholds/parameters.[11] We, therefore, developed a customized simulator with a simple radio transmission model: at any time, each node can directly communicate with all others within its transmission range; the packet delivery to neighbors is instantaneous and error-free; and neither the fading or shadowing effects of the wireless links, nor the effects of obstacles are taken into account. We implemented and simulated the local-area routing protocol (in Section II-B2) along with the proactive location-update scheme (in Section II-A).

Our simulation environment is based on an ad hoc network of 100 nodes that move around in a square area of $1000 \times 1000\ [\text{m}^2]$. Each node has a radio transmission range of radius 250 [m]. Initially, each node is placed randomly in the area, except for the source node placed at the center of the network plane. Each nonsource node then chooses a new location to move to, according to the above mobility model. We generated the route-request interarrival times using the *Pareto* distribution (one of the heavy-tail distributions). Whenever a route-request event is triggered, the source randomly chooses another node, then initiates the route-search procedure as described in Section II-B2. We simulated and measured the control traffic only, without considering data traffic delivery, as we are interested in evaluating the control traffic overhead. We ran each simulation for the duration of 5 h (18 000 s) to gather sufficient samples. We fix the confidence probability p_{succ} to 0.97 and set the beaconing interval to 10 s. The sizes of LINFO, RREQ and RREP packets are set to 17, 82 and 82 bytes, respectively.[12]

[11] As described in Section III-G, link-layer overheads is absorbed in the estimation of $\bar{\mathcal{U}}$ and $\mathcal{S}_{\mathcal{R}}$.

[12] These packet sizes are computed by considering the MAC header, the payload fields defined earlier, and maximum hop count 16.

A. Incremental Node Mobility

A realistic model for node mobility generation is needed. One may consider trace-driven generation of movement patterns, which either directly replays the traces obtained from real wireless networks like Metricom [44] or uses a set of time-varying parameters derived from traces [45]. This approach intends to make evaluation results reproduce the real behavior. But, its main drawback is the requirement of a large amount of trace data for statistically meaningful results. So, a random mobility model is used widely for evaluating ad hoc routing protocols [9], [10], [16], [17], [35]. The speed and direction of node movements are generated randomly, but in a controlled manner to make them realistic. The advantage of this model-based simulation is its simplicity. Moreover, one can stress the routing protocol by letting all nodes move around constantly.

To be as objective as possible, we implemented the mobility model presented in [9] and [10] to generate nodes' movement patterns. It considers the relationship between a node's previous and current movement behaviors in speed and direction. According to the model, the speed and direction of a node's current movement evolve randomly every Δt from its previous speed and direction. The position of a node at (x, y) and its speed v and direction θ are updated as follows:

$$\begin{aligned} v(t+\Delta t) &= \min\left(\max\left(v(t)+\Delta v, 0\right), V_{\mathrm{MAX}}\right) \\ \theta(t+\Delta t) &= \theta(t)+\Delta\theta \\ x(t+\Delta t) &= x(t)+v(t)\cdot\cos\theta(t) \\ y(t+\Delta t) &= y(t)+v(t)\cdot\sin\theta(t) \end{aligned}$$

where V_{MAX} is the maxium velocity. Δv is generated from the uniform distribution $U(-A_{\mathrm{MAX}}\cdot\Delta t, A_{\mathrm{MAX}}\cdot\Delta t)$, and $\Delta\theta$ is from $U(-\alpha\cdot\Delta t, \alpha\cdot\Delta t)$. V_{MAX}, A_{MAX} and α are so chosen as to give the average speed of 10 [m/s].

B. Verification of the Optimization Results

To calculate theoretical curves for the location-update and route-search costs, we evaluated the model parameters such as U, S, β, and $C_{\min}$. They were evaluated based on our mobility model, where nodes have a Rayleigh-distributed speed with mean squared speed σ^2 and uniformly distributed direction changes. We set σ^2 equal to the mean squared speed of the incremental model, hence providing the same average speed. As a result, U, S, β, and $C_{\min}$ were determined to be 28092, 8.8353, 1, and 328, respectively. We used these values for graphing the theoretical costs. A node initiates a new route request to a randomly selected destination after a Pareto-distributed time-interval elapsed. The incoming route-request arrivals follow a Pareto distribution with rate 0.04 (i.e., $\mu_c = 25$ [s]).

Fig. 7 plots the analytical and simulated routing overheads of the ad hoc network of 100 nodes. The simulation result is close to the theoretical curve, although there exists a small discrepancy between simulated and theoretical search costs. Since we are concerned with the cost around the optimal location-update threshold, at which both update and search costs are relatively small, we can conclude that the formula for the optimal location-update threshold is consistent with that obtained from simulation. In Fig. 7, the optimal threshold occurred around 160–180 [m], which is close to the theoretic γ^* of 178.3 [m].

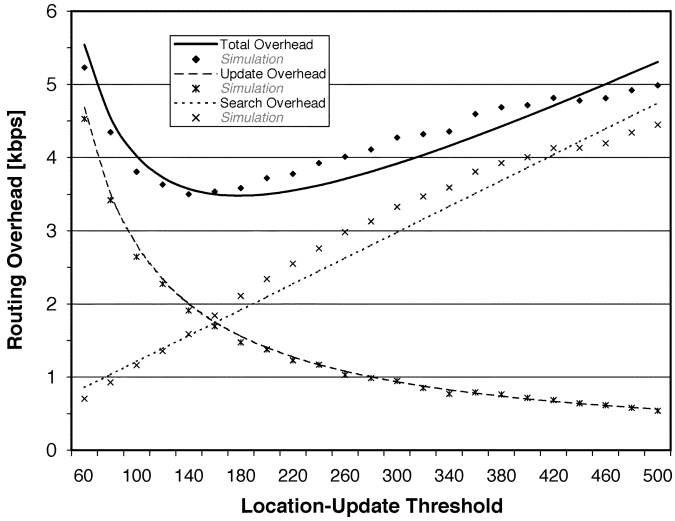


Fig. 7. Total cost for Pareto arrivals (in bytes).

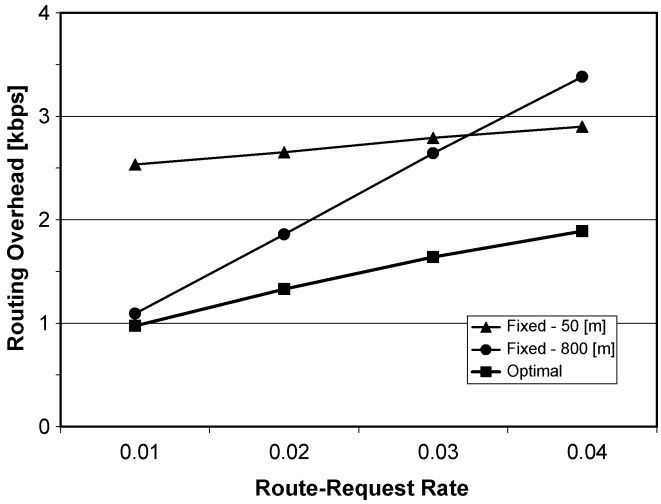


Fig. 8. Routing overhead of the proposed protocol.

Hence, the theoretic cost formula captures the actual behavior of the proposed protocol.

C. Optimal versus Fixed γ

The purpose of our simulation is to evaluate the effect of the choice of γ on the routing performance. We chose three modes of operation: optimally selected γ, fixed γ at 50 [m] and 800 [m]. The optimal γ value was selected using (42). The fixed γ value of 50 [m] means that the route search is operated in an overly proactive mode. The γ value of 800 [m] corresponds to the overly reactive mode of operation. We measured both the routing overhead and the probability of route-discovery success for several route-request rates, 0.01, 0.02, 0.03 and 0.04, which correspond to μ_c =100, 50, 33, and 25 [s], respectively. We used the LINFO broadcast scheme 1 with beaconing.

Figs. 8 and 9 show the routing overhead and the probability of route-discovery success. When γ is fixed at 50 [m], each node broadcasts its location frequently, introducing more update overhead than the search overhead. As a result, the total routing overhead is the highest at low route-request rates and the most insensitive to route-request rate variations. On the other hand,

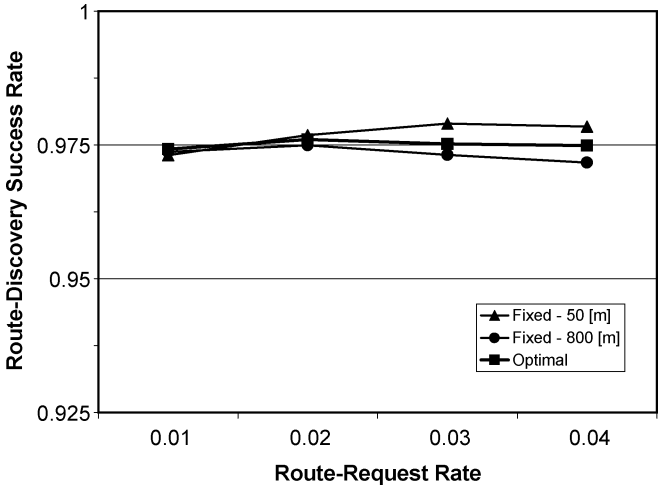


Fig. 9. Route-discovery success rate of the proposed protocol.

when γ is fixed at 800 [m], it tends to search a larger scope, incurring more search overhead than the update overhead. Hence, the routing overhead is very sensitive to the route-request rate. With the optimal γ, we achieve the maximum bandwidth-efficiency, while maintaining a high route-discovery success probability. For all three modes, the probability of route-discovery success is 0.975 and higher.

V. Conclusion

We proposed a tradeoff-based approach to devise a highly versatile and scalable routing protocol, consisting of: 1) a hybrid location-update scheme that combines well-known schemes such as proactive location updates within each node's local region of radius $\mathcal{R}$, and 2) a route-discovery scheme that realizes an Internet-like routing architecture by providing location-query, route-search, energy-aware (QoS) routing, and route-recovery mechanisms. The proposed protocol has the following advantages: 1) scalability, in terms of $\mathcal{O}(\log N)$ memory cost per node and localized route discovery; 2) versatility, supporting discovery and recovery of energy-aware and QoS routes; and 3) bandwidth and energy efficiency, minimizing the routing overhead and prolonging node's lifetime.

We also analyzed the optimality of our proposed protocol, and derived optimal location-update thresholds by which nodes can autonomously decide when to broadcast their own location updates to minimize routing overhead. Based on general models for node mobility and route-request arrivals, we first formulated costs for both location update and route discovery as functions of thresholds, and then solved a distributed optimization problem in which each node minimizes the overhead incurred by itself. The simulation results have shown that our analytical results closely capture the real behavior, thus verifying/confirming their correctness.

References

[1] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance vector routing (dsdv) for mobile computers," in *Proc. ACM SIGCOMM*, Sep. 1994, pp. 234–244.

[2] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM Mobile Networks Appl. J., Special Issue on Routing in Mobile Communication Networks*, pp. 183–197, Oct. 1996.

[3] V. Rodoplu and T. H. Meng, "Minimum energy mobile wireless networks," *IEEE J. Select. Areas Comm.*, vol. 17, no. 8, pp. 1333–1344, Aug. 1999.

[4] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T. W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE J. Select. Areas Comm.*, vol. 17, no. 8, pp. 1369–1379, Aug. 1999.

[5] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Norwell, MA: Kluwer, 1996, ch. 5, pp. 1369–1379.

[6] C. Perkins and E. M. Royer, "Ad hoc on demand distance vector (AODV) routing," Network Working Group, Internet Draft, Aug. 1998.

[7] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM*, Apr. 1997.

[8] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE J. Select. Areas Comm.*, vol. 17, no. 8, pp. 1488–1505, Aug. 1999.

[9] Z. J. Haas and M. R. Pearlman, "The perforamnce of query control schemes for the zone routing protocol," *IEEE/ACM Trans. Networking*, vol. 9, no. 4, pp. 427–438, Aug. 2001.

[10] P. Samar, M. R. Pearlman, and Z. J. Haas, "Hybrid routing: the pursuit of an adaptable and scalable routing framework for ad hoc networks," in *The Handbook of Ad Hoc Wireless Networks*. Boca Raton, FL: CRC Press, 2003.

[11] C.-C. Chiang, H. K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," in *Proc. IEEE SICON'97*, Apr. 1997, pp. 197–211.

[12] R. Sivakumar, P. Sinha, and V. Bharghavan, "Cedar: a core-extraction distributed ad hoc routing algorithm," *IEEE J. Select. Areas Comm.*, vol. 17, no. 8, pp. 1454–1465, Aug. 1999.

[13] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," ISI Report, ISI/RR-87-180, Mar. 1987.

[14] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Pers. Commun.*, vol. 8, no. 1, pp. 48–57, Feb. 2001.

[15] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," presented at the ACM/IEEE MobiCom'00, Boston, MA, Aug. 2000.

[16] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A distance routing effect algorithm for mobility (DREAM)," presented at the ACM/IEEE MobiCom'98, Oct. 1998.

[17] Y. B. Ko and N. H. Vaida, "Location-aided routing (LAR) in mobile ad hoc networks," presented at the ACM/IEEE MobiCom'98, Oct. 1998.

[18] I. Stojmenovic and X. Lin, "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1–10, Oct. 2001.

[19] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," presented at the ACM/IEEE MobiCom, Boston, MA, Aug. 2000.

[20] P. F. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," presented at the ACM SIGCOMM, Aug. 1988.

[21] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," presented at the ACM SIGCOMM, San Diego, CA, Aug. 2001.

[22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," presented at the ACM SIGCOMM, San Diego, CA, Aug. 2001.

[23] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems," presented at the IFIP/ACM Int. Conf. Distributed Systems Platforms, Nov. 2001.

[24] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: a scalable dynamic emulation of the butterfly," presented at the ACM Symp. Principles of Distributed Computing (PODC 2002), Monterey, CA, Jul. 2002.

[25] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information systems based on the XOR metric," presented at the Int. Workshop on Peer-to-Peer Systems (IPTPS 2002), Boston, Mar. 2002.

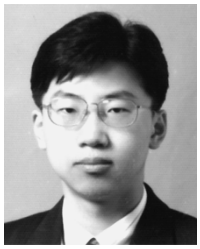
[26] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The impact of dht routing geometry on resilience and proximity," presented at the ACM SIGCOMM, Karlsruhe, Germany, Aug. 2003.

[27] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," presented at the IEEE INFOCOM, Kobe, Japan, Apr. 1997.

[28] J. Raju and J. J. Garcia-Luna-Aceves, "A new approach to on-demand loop-free multipath routing," presented at the IEEE ICCCN, Boston, MA, Oct. 1999.

[29] A. Nasipuri and S. R. Das, "On-demand multipath routing for mobile ad hoc networks," presented at the IEEE ICCCN, Boston, MA, Oct. 1999.

[30] S.-J. Lee and M. Gerla, "Aodv-br: backup routing in ad hoc networks," presented at the IEEE WCNC, Chicago, IL, Sep. 2000.
[31] K. Scott and N. Bambos, "Routing and channel assignment for low power transmission in pcs," presented at the IEEE ICUPC, Cambridge, MA, 1996.
[32] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," presented at the ACM/IEEE MobiCom, Oct. 1998.
[33] I. Stojmenovic, "Location updates for efficient routing in ad hoc networks," in *Handbook of Wireless Networks and Mobile Computing*. New York: Wiley, 2002.
[34] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information dissemination in partitionable mobile ad hoc networks," presented at the IEEE Symp. Reliable Distributed Systems, Oct. 1999.
[35] A. B. McDonald and T. F. Znati, "A mobility-based framework for adaptive clustering in wireless ad hoc networks," *IEEE J. Select. Areas Comm.*, vol. 17, no. 8, pp. 1466–1487, Aug. 1999.
[36] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless networks," presented at the IEEE/ACM MSWiM'99, Seattle, WA, Aug. 1999.
[37] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communication and Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking*, vol. 2, no. 5, 2002.
[38] J. Yoon, M. Liu, and B. Noble, "Sound mobility models," presented at the ACM/IEEE MobiCom, Sep. 2003.
[39] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri, "Toward realistic mobility models for mobile ad hoc networks," presented at the ACM/IEEE MobiCom, Sep. 2003.
[40] A. Feldmann, "On-Line Call Admission for High-Speed Networks," Ph.D. thesis, Sch. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, 1995.
[41] A. Feldmann, "Modeling characteristics of TCP connections," AT&T, AT&T Laboratories Report, 1996.
[42] W. Feller, *An Introduction to Probability Theory and Its Application*. New York: Wiley, 1966, vol. 1.
[43] S. M. Ross, *Stochastic Processes*. New York: Wiley, 1983.
[44] D. Tang and M. Baker, "Analysis of a metropolitan-area wireless network," presented at the ACM/IEEE MobiCom'99, Seattle, WA, Aug. 1999.
[45] B. D. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz, "Trace-based mobile network emulation," *ACM Comput. Commun. Rev.*, vol. 27, pp. 51–61, Oct. 1997.



Taejoon Park (S'04) received the B.S. degree (*summa cum laude*) from HongIk University, Seoul, Korea, in 1992, and the M.S. degree from the Korea Advanced Institute of Science and Technology, Taejon, Korea, in 1994. Since Fall 2000, he has been with the University of Michigan, Ann Arbor, where he is currently a Research Assistant working toward the Ph.D. degree in electrical engineering and computer science.

He worked at LG Electronics, Seoul, as a Research Engineer from 1994 to 1999, then as a Senior Research Engineer in 2000. His research interests include security and routing in infrastructureless networks such as sensor, mobile ad hoc, and peer-to-peer networks.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and the M.S. and Ph.D degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is the Kevin and Nancy O'Connor Professor of Computer Science and Founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA, IBM T. J. Watson Research Center, Software Engineering Institute at Carnegie Mellon University, and HP Research Laboratories. He also chaired the Computer Science and Engineering Division, EECS Department, University of Michigan, for three years beginning January 1991. His current research focuses on QoS-sensitive networking and computing as well as on embedded real-time OS, middleware and applications, all with emphasis on timeliness and dependability. He has supervised the completion of 51 Ph.D. theses, and authored or co-authored around 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has co-authored (with C. M. Krishna) the textbook *Real-Time Systems* (New York: McGraw Hill, 1997).

Dr. Shin has received a number of best paper awards, including the IEEE Communications Society William R. Bennett Prize Paper Award in 2003, the Best Paper Award from the IWQoS'03 in 2003, and an Outstanding IEEE TRANSACTIONS ON AUTOMATIC CONTROL Paper Award in 1987. He has also co-authored papers with his students which received the Best Student Paper Awards from the 1996 IEEE Real-Time Technology and Application Symposium, and the 2000 UNSENIX Technical Conference. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Service Excellence Award in 2000, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award in 2004 from the University of Michigan, a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002, and 2003 IEEE RTC Technical Achievement Award. He is a Fellow of the ACM, and a member of the Korean Academy of Engineering. He is serving as the General Chair for the Third ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys'05), was the General Chair of the 2000 IEEE Real-Time Technology and Applications Symposium, the Program Chair of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chair of the 1987 RTSS. He was the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems, a Program Co-Chair for the 1992 *International Conference on Parallel Processing*, and served numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991–1993, was a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, and an Area Editor of the *International Journal of Time-Critical Computing Systems, Computer Networks*, and *ACM Transactions on Embedded Systems*.