Efficient Content Location in Wireless Ad Hoc Networks

Jivodar B. Tchakarov
Department of Computer Science
University of Illinois at Urbana-Champaign

Nitin H. Vaidya
Coordinated Science Laboratory, and
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Abstract

*The advances in wireless networking have enabled new paradigms in computing. An abundance of information and services provided by remote servers is expected to become available to wireless users. A fundamental issue in this environment is efficiently locating needed content. Such content may be in the form of files, services, or any other kind of data. In this paper, we describe an algorithm for efficient content location in location-aware ad hoc networks. The Geography-based Content Location Protocol (*GCLP*) makes use of physical location information to lower proactive traffic while reducing query cost. The results of our analysis show that* GCLP *performs favorably in terms of overhead and scalability.*

1. Introduction

Explosive growth of the Internet has resulted in increasing availability of information and services to networked users. It has made sharing of data easier than ever with a few simple clicks of the mouse button. File sharing service such as *Napster* have demonstrated the need for creative protocols for location and sharing of data. The need for avoiding centralized responsibility and increasing stability has been the driving force behind a variety of new approaches that have been proposed for peer-to-peer file sharing.

The rise of mobile computing has further increased the pervasiveness of devices capable of storing data and requiring the ability to efficiently locate content and services. Printers, for example, are a common need for a variety of applications. Given a pervasive wireless network, the user may not always be expected to know the location and characteristics of the closest device/server.

A variety of algorithms have been proposed for resource location (not necessarily in ad hoc networks). Some of the first approaches to appear followed the centralized client-server architecture. Some examples of such approaches are presented in [1, 2, 3, 4, 5]. These models rely upon a centralized storage that would handle queries by users. Decentralized approaches [6, 7, 8, 9, 10] remove the reliance upon a central directory server but do not take link cost into account when computing routes. This can make them impractical for use in wireless ad hoc networks; ad hoc networks are formed by wireless hosts without necessarily making use of an infrastructure.

Some protocols have been proposed specifically for ad hoc networks as well [11, 12, 13, 14, 15]. A novel approach to disseminating service information is described in [16, 17]. [16] proposes the use of location information for routing. The protocol provides for all nodes to periodically send advertisements along geometric trajectories. Any node that wishes to communicate with another node need simply send a query along a path that intersects with the advertisement path. The host that receives a query may then send a reply to the requesting node. A similar idea is also presented in [17]. [17] proposes propagating the advertisements and queries in cross-shaped trajectories, thus guaranteeing two intersections. Queries are answered by nodes at the intersection of the advertising and query trajectories. This is a simple and elegant approach that may be modified to work for a variety of resources available in a network. However, as the number of advertising servers grows, the amount of proactive traffic becomes prohibitive. Both of these algorithms assume that each node will advertise a unique resource. This is not true, however, in many cases since duplicate content may well exist in a network. An example of such duplicate content may be several replicas of a file hosted by different servers or an identical service provided by several nodes in the network. Under these conditions, the above algorithms will not scale well since they do not take measures to limit the overhead for duplicate resources.

In this paper, we present a content location service, the *Geography-based Content Location Protocol* (GCLP), that takes physical location information into account to provide an efficient content location service to nodes in an ad hoc network. The content location service allows a user host to *locate* a server that can provide desired content – "lo-

cating" a server can mean knowing its IP address, and/or its physical location, whichever information is necessary to subsequently route requests from the client to the content server. Note that our content location protocol does use physical location information. However, the routing protocol used to deliver requests from clients to servers may or may not use physical locations (the two protocols can be independently designed). The proposed content location protocol works best in dense networks. GCLP assumes that all devices in the network know their own physical location. Since GCLP is meant to operate in an ad hoc network, it is important to summarize some of the properties of the environment as they relate to the content location service. First, we cannot assume a static topology. Nodes may join and leave the network at any time and node mobility is an accepted occurrence. Second, the cost of making sure that everyone knows about everything is prohibitive. Thus, to locate a specific content, a device need not be aware of all content available on the network. In such an environment, GCLP nodes make use of geographic information to periodically advertise content they are hosting to nodes along several geographical directions. Nodes that want to locate content should be able to locate any one server for that content, preferably the closest one. The protocol proposed in this paper tries to allow each client to find a nearby server, where the geographical distance is used as the distance metric (not number of hops). In a uniformly distributed dense network, shorter geographical distance would typically translate into smaller number of hops as well. While the protocol is not designed for a specific network topology, we show that it performs better in dense networks.

2. Geography-based Content Location Protocol (GCLP)

2.1. Protocol Overview

Nodes in the network may assume any of the following roles, more than one if required. A Content Server (CS) is a node that hosts one or more resources that may be used by other nodes on the network. Such nodes are responsible for advertising their hosted resources to the rest of the network. Content Location Servers (CLS) are nodes that host *location* information about one or more available resources – *location* of a resource may mean IP address of the corresponding server, and/or physical location of the server, or any other information useful for routing to the server. The content location servers are responsible for providing responses to queries about specific content. *Clients* are nodes that request resources on behalf of an application or any other higher layer.

The basic protocol follows the scheme described in [17]. Periodically, a content server will transmit *update* messages to specific nodes in the network. These updates advertise available resources and the content server that hosts them. Update messages follow a suitable trajectory through the network similar to the trajectory-based schemes described in Section 1 [16, 17]. This significantly decreases the amount of proactive traffic as it is limited to nodes along the trajectories. Nodes along these trajectories cache the information received from the updates. A node that stores such information becomes a content location server (CLS). If it receives a query about content it knows the server of, it will reply with the server *location*.

A client may locate any content on the network by sending out a *query* message. The query is similarly propagated along suitable trajectories. In a dense network, these query trajectories are likely to intersect at at least one update trajectory. The content location server (CLS) at the intersection point that receives the query responds with a reply message that is sent back to the client. Upon receipt of a reply, the client may establish a direct connection with the content server using the underlying routing protocol to make use of the available resource. The queries follow a forwarding scheme that keeps their cost low while finding the closest available server in the vast majority of situations (*closest* is defined here in terms of physical distance). Finding the closest content server available is an important benefit as it generally (though not always) results in fewer hops between the client and the server, which, in turn, translates into lower network overhead.

To make sure that all nodes in the network know the physical location of all their neighbors in order to be able to select next hop for the updates and queries, a third type of message is used. A *hello* message is periodically broadcast by each node in the network to its one-hop neighbors to advertise the node's physical position.

2.2. Protocol Details

This section discusses the content advertisement and content discovery mechanisms in *GCLP*.

2.2.1. Content Advertisement: Content advertisement is performed by periodically sending update messages through the network, similar to [17]. Each content server periodically initiates advertisements by sending update messages in four geographical directions, or trajectories – North, South, East, and West. Each advertisement specifies the location of a resource at the content server (CS). To accomplish this, the content server uses the geographic location of its immediate neighbors to select the next content location server (CLS) in the given direction. The algorithm for selecting next-hop nodes is described in detail below. The update message is then sent to that node. Upon receiving the update, the chosen content location server adds the information to its *Content Lo-*

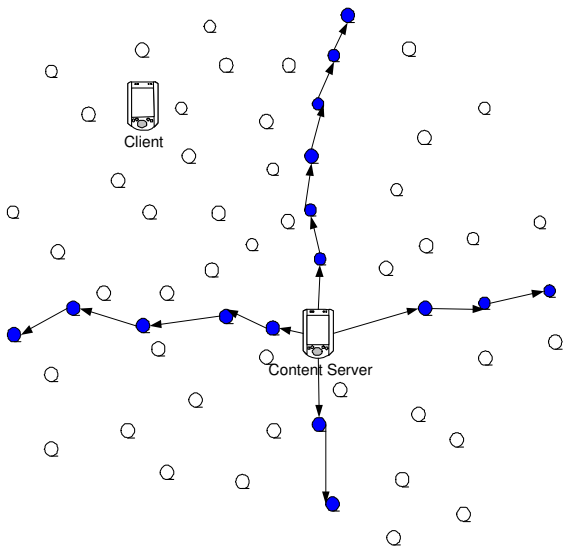

Figure 1. A Content Server propagating updates through the network. The server's selected content location servers are shaded.

cation Table and uses the same algorithm to decide which node in the direction of the update will be the next content location server. This continues until a node on the fringe of the network discovers that there are no neighbors in the sector along the update's direction.

This basic advertising algorithm is exemplified in Figure 1. Here, update messages are being propagated for a particular content server through the network in the four directions. The nodes that are chosen to become content location servers are shaded. Note that, on each hop, a certain node is the intended receiver of the updates from a content server, all nodes that overhear the updates also cache the content server's location information. Only the intended receiver is responsible for forwarding the update, however.

An important part of the protocol is the selection of the content location server (CLS) nodes along the geographic direction of an update message. In designing an algorithm several heuristics may be used:

- Picking nodes closest to the line in the desired direction (e.g., South). This keeps the line of content location servers straighter but may not be as efficient since some content location servers may be needlessly close to each other even if nodes are available farther away generally in the desired direction.
- Picking nodes with greatest distance from the current node. This may be more efficient as it provides for greater spaces between neighboring content location servers. However, this approach may lead to update trajectories that are not straight.
- A combination of the two. This is the approach that we take and we describe the exact algorithm below.

Note that, in our scheme, each node attempts to forward the update in the desired direction (e.g., South) with respect to its own location, not with respect to location of the originator of the update. (However, the latter alternative may also be worth evaluating.)

In [17], the authors propose using the first two approaches above for selecting next hop nodes in the update path, i.e., select nodes farthest away from the current node or select nodes closest to the trajectory line. We modify this basic selection algorithm as follows. When selecting next hop content location servers, we would like to achieve two things–cover a larger distance between content location servers and keep the update trajectory as straight as possible. To accomplish this, each node in the 90° sector along the trajectory is assigned a rating based on Equation 1, where *R* is the rating for the given node in the sector, *d* is the distance from the node making the decision, and *r* is the offset from the geographical direction line. This algorithm allows for nodes farther away and closer to the perfect trajectory to have highest ratings. It also means that the trajectory is defined in terms of the current node making the decision and not only the original server.

$$R = d/r \tag{1}$$

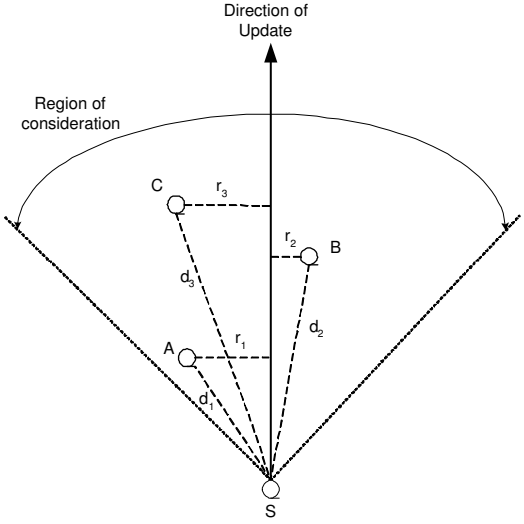

Figure 2. A node picks the next hop for an update message among nodes in the appropriate sector.

This is further clarified by Figure 2. A node, *S*, is considering three possible candidates in the desired sector, *A*, *B*,

and *C*. Based on the formula provided, node *B* will be selected as the next hop in the trajectory as it will have the highest rating.

The selection of 90° as the sector size is dictated by the need to increase node availability in each sector, particularly in sparse networks. Small sector sizes would not be suitable since the above rating will tend to prefer nodes closer to the trajectory (when the sector size is too small). Thus, we can only benefit from allowing more nodes to be considered for selecting next hop neighbors in the update and query trajectories. If a node cannot find a neighbor in a given sector that it is trying to transmit to, the trajectory in the given direction is interrupted. (In mobile environments, when deciding whether to interrupt a trajectory or not, the staleness of the cached information about other known servers can be taken into account. This is, however, not implemented in the simulated scheme.)

As described until now, the protocol is simply a variation on [17] and, like [17], does not scale well. It does not deal with the possibility of more than one content server (CS) hosting the same resource in the network. Next, we describe a major component of our protocol that provides for protocol scalability. If update messages for duplicate resources are allowed to propagate throughout the network, the resulting traffic will be overwhelming since it would increase linearly with the number of servers hosting the same content. Instead, each content location server (CLS) chooses whether to forward an update message or not. If a content location server receives multiple advertisements for a particular resource, it will only forward updates from the content server closest to it. In case there is a tie, the content location server will continue to propagate updates from the first server it received advertisements from. (A content location server may initially be forwarding advertisements from a distant content server before it learns of the closest server; however, once it learns of a closer server, advertisements from distant servers are not forwarded anymore.) The resulting advertisement grid allows for scalability of the protocol as each additional replica of a resource will introduce less and less proactive traffic into the network. An example is shown in Figure 3 – note that the updates sent by all the servers do not always travel to the edges of the network. This method also has the important property that content location servers will typically know the location of servers close to them.

2.2.2. Content Discovery: To locate content on the network, a client sends out a query message through the network in a manner identical to the update messages. A query is sent in the four geographical directions. The next hop in the query trajectory is selected using the same algorithm described in Section 2.2.1. A *content location server* that receives a query message will send a query reply message to the requester. To reduce the possibility of a query slipping

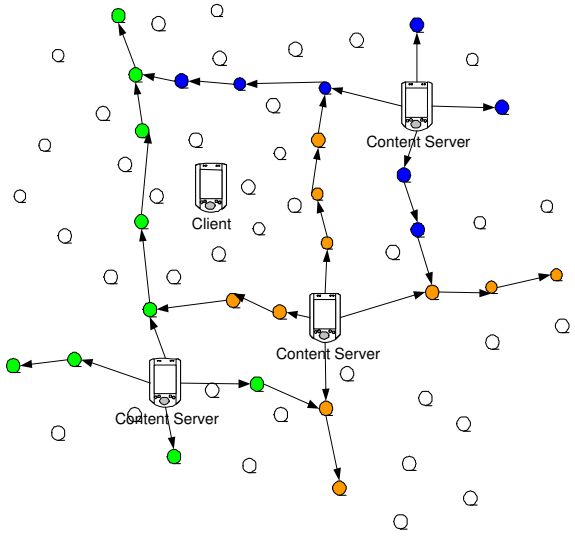

Figure 3. Several servers advertise availability of the same resource.

through the advertising trajectory without finding a content location server, all nodes in the range of a node sending or forwarding a query may answer – thus, the requester may receive multiple replies. While this modification does not completely eliminate the above possibility, the modification significantly reduces the possibility as shown by our simulations. The reply messages follow greedy geographic routing with each node forwarding the packet to its neighbor closest to the destination; other more sophisticated geographic routing schemes, or other routing protocols may also be used. Potentially, the cost of forwarding the multiple replies can be reduced by requiring each intermediate node to forward at most one reply; such an optimization is not evaluated in this paper.

The content discovery process is exemplified in Figure 4. Here a client sends out queries through the network in four directions. Once the queries reach a content location server along the update trajectories, it answers with a response message. The client may receive more than one response messages to the same query. In such a case, it picks the response identifying a content server closest to its geographical location.

3. Analysis of GCLP

In this section we provide a simplified analysis of the Geography-based Content Location Protocol. It is important to note that the network under consideration in this section is a hypothetical dense network with nodes at each point in space. Thus, physical distance is proportional to

hop count (in the context of the hypothetical dense network, a hop is assumed equal to one transmission range). In practice, the results of this section will apply best to very dense networks. However, they also provide hints on how the protocol will perform in networks that may not be very dense.

3.1. Correctness Argument

We argue that, in a dense network, at least one intersection of update trajectories and query trajectories exists. Even with some update trajectories being interrupted by updates from closer content servers, we need to show that at least one trajectory will propagate in each one of the four geographical directions. The correctness argument is simple and is illustrated in Figure 5. We have two content servers, *S1* and *S2*. Two content location servers, *A* and *B*, interrupt some update trajectories and forward others. To see that at least one trajectory is propagated in each direction, let us observe what happens at a point where a trajectory is interrupted, such as point *B* in the figure. It is sufficient to observe that in order for one trajectory to get interrupted, such as the trajectory from *S1*, there must exist a server, such as *S2*, whose perpendicular trajectory causes the interruption at that point. Such a server would be propagating its own updates in the direction of the interrupted trajectory. Thus, there is at least one update trajectory in each direction.

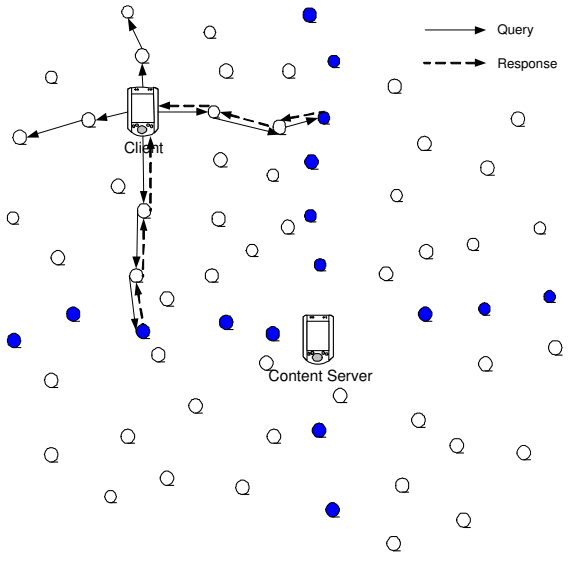

Figure 4. A client attempts to locate content by sending queries through the network. The queries are answered once they reach a content location server.

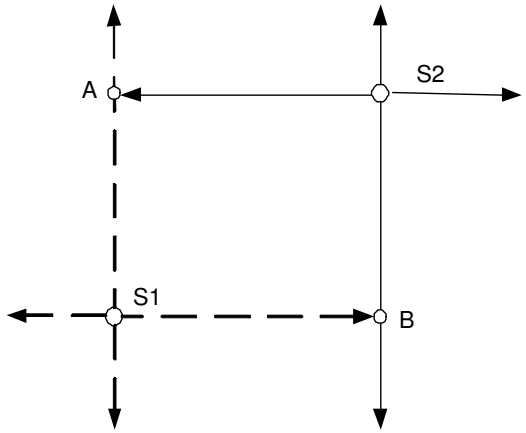

Figure 5. At least one trajectory will propagate in the direction of an interrupted update trajectory.

3.2. Scalability Analysis

To provide analytical model of our protocol, we examine the performance of the protocol in the two extreme cases: when there is a single server in the network and when all nodes in the network are servers. We then consider how the proactive traffic introduced by update messages changes with growing the number of servers in between the two extremes.

In the first case, we have a single content server in the network for a given resource. Consider a network of dimensions *w* by *l* hops. Any update sent by the content server in the four directions will travel along the length and breadth of the network area. Thus, in the case of a single content server, the cost is *O(w+l)*.

In the other extreme, consider the case where all nodes in the network host the same content. In this case, the updates from each host will be ignored by their next hop neighbors because they are themselves content servers serving the same content. Thus, the cost of each individual update is *O(1)*, giving a total cost in the network of *O(n)*, where *n* is the number of content servers in the network. In between the two extremes, we argue that the protocol overhead generated per server drops rapidly as new servers are added to the network [18].

Our simulations show that the overhead traffic of the protocol indeed scales well and adding servers leads to a rapid drop in update traffic per server. The scalability of the protocol is also seen in the simulation results presented in Section 4.

3.3. Response Analysis

A desired property for the proposed protocol is the ability of queries to locate the closest content server available. However, under certain conditions, this may not be possible. This anomaly is shown in Figure 6. There are two servers in this setup, *S1* and *S2*. They propagate their updates according to the protocol described in Section 2.2.1. A content location server, *CLS1*, at the intersection of the two trajectories from *S1* and *S2* will only propagate updates from the server closer to it. In the figure, this means that only updates from *S1* will be propagated by *CLS1*.

Let us consider the different possible positions of a client in this environment defined by the update trajectories from *S1* and *S2* and the line, l, which defines the set of points of equal distance to both servers. If a client is on the side of l where *S1* is located, then *S1* is the closest server to that client. Queries sent in that sector will intersect at least one update trajectory from S1 thus resulting in discovering the closest server. If a client is located on the side of l towards *S2* and not between l and the downwards update trajectory from *S1* (i.e., not in the region R in the figure), then it is closest to *S2* and queries from such a client will intersect at least one update trajectory from *S2*, thus again finding the closest server.

The problem occurs when a client, Q, is located in the region *R* bounded by the line l and the downwards update trajectory from *S1*. In this region, *S2* is the closest server. However, a query will be answered by a location server, *CLS2*, positioned on the downwards trajectory from *S1* resulting in locating a server that is not the closest one, *S1* instead of *S2*. Notice, that a similar region exists, R' where $S1$ is the closest server but only $S2$ can be found. In this section we prove that even in this situation, in the worst case, the error is relatively small when compared with the actual distance to the closest content server. This error is measured as the ratio *QS1/QS2*, where *QS1* is the distance between *Q* and *S1* and *QS2* is the distance between *Q* and *S2*.

First, let us examine how the motion of *Q* along a line m perpendicular to the line l affects the ratio. It is easy to see that any move of *Q* farther from l is moving it farther away from *S1* and closer to *S2*, increasing the ratio *QS1/QS2*. Thus, as *Q* is closer to the update trajectory, the ratio will be larger, giving us a greater error. For the purposes of the following analysis, we assume the worst case where *Q* is actually on the trajectory line itself.

Let us express the ratio of *QS1/QS2* in terms of the distances *a*, *b*, and *c*, as shown in Figure 6. Equation 2 shows this formula.

$$QS1/QS2 = (a+c)/(\sqrt{b^2+c^2}) \quad (2)$$

Starting from the above expression, it can be shown that,

$$QS1/QS2 \leq \sqrt{2} \quad (3)$$

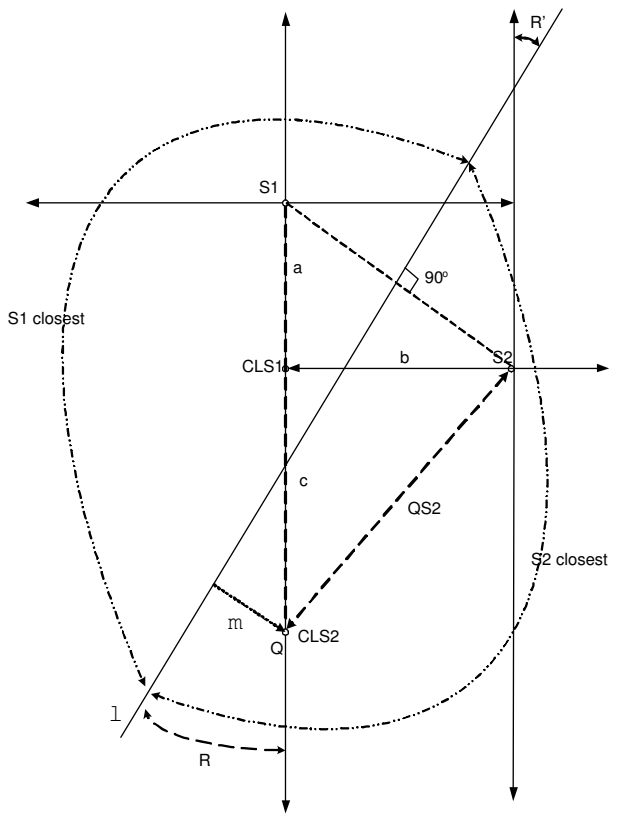

Figure 6. A client, Q, may locate a resource on a more distant server, QS1, instead of the closer QS2.

According to Equation 3, our protocol allows (in the hypothetical dense network) a host to learn about a content server that is at a distance no larger than $\sqrt{2}$ times the distance from the closest content server. In practical networks, which are not very dense, this result is not directly applicable, but suggests that the proposed protocol has the potential to locate nearby servers.

4. Simulation Results

In this section we evaluate the performance of our protocol in a simulated environment. The *ns-2* simulator was used to simulate a variety of network conditions. The area over which the simulated network was situated was 2000 by 2000 meters. Several network densities were simulated. For each case, 20 simulations were run and the results were averaged to produce the presented data. A sparse network of 50 nodes, moderate density networks of 100 and 200 nodes, and a dense network of 500 nodes were simulated. Static topologies are first evaluated. Nodes are placed at randomly chosen locations within the given area. Different

node placements are chosen for each of the 20 runs. In ns-2, we created the nodes during the first 5 seconds, and they were designated as content servers at between 10 and 15 seconds from start of the simulation (thus, initially, the created nodes are not servers, but become servers over time). Queries are performed after all desired servers have been instantiated. Each content server sends updates every 30 seconds. To study the effects of mobility, a moderate density network of 100 nodes was evaluated with varying nodes speeds. The Random Waypoint mobility model from ns-2 simulator was used to simulate node mobility (all nodes including the content server move when mobility is considered). Transmission range is assumed to be 250 meters. We use the following metrics to evaluate the proposed protocol:

- **Update Cost**: This measure captures the normalized cost incurred by content servers performing location updates. For each update initiated by each server, the update message travels several hops (in four different directions). Each such hop is counted as one update message. The *update cost* is calculated as the total number of update messages required, divided by the total number of updates initiated by the servers, and by the total number of content servers.
- **Query Cost**: This measure is used to quantify the cost of searching the network for required content. When a node initiates a query, it may be forwarded on several hops (in different directions). *Query cost* is calculated as the total number of hops taken by all the queries, divided by the number of queries initiated. In our simulations, each node in the network initiates a query once. Generating query once per node is enough in static networks, since the average cost of the query will not vary over time. For mobile scenarios, this is not always adequate, and additional evaluations need to be performed.
- **Success Rate**: The accuracy of our protocol is measured by this metric. Success rate is calculated as the number of queries that receive responses, divided by the total number of queries initiated. If a query receives more than one response, it is only counted as successful once.

4.1. Effects of Network Density

This section studies the effects of node density on the performance of *GCLP*. While the protocol is not designed for a specific network topology, we show that it performs better in denser networks.

4.1.1. GCLP in Moderate-Density Networks As seen in Figures 7 and 8, the update cost per server decreases rapidly with the addition of new servers, and then becomes essentially constant. This pattern is also implied by the analysis in Section 3.2. (Due to the manner in which the servers are instantiated during first 15 seconds of the simulation, as elaborated earlier, we believe that that update overhead is slightly higher, as compared to the case when all servers are instantiated at the start of the simulations.)

The amount of query traffic also decreases rapidly with the addition of servers. This is to be expected as the grid grows denser and more nodes have the ability to answer queries. This observation is supported by the data in Figures 9 and 10. Query cost becomes zero when number of servers is large, because with large number of servers, each host is either a content server or already knows location of a nearby content server (due to updates sent by the content servers).

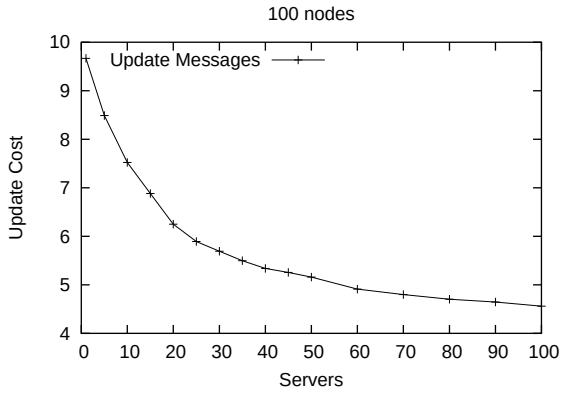

Figure 7. Update cost in a network of 100 nodes as a function of number of servers.

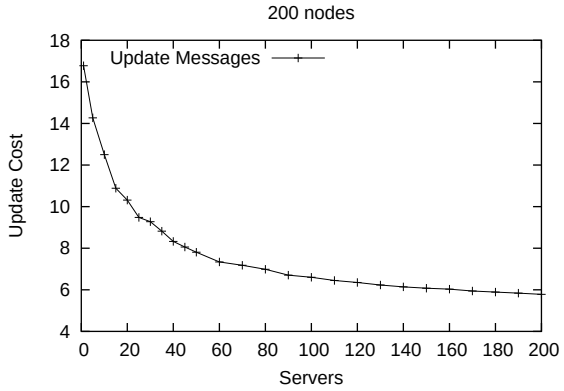

Figure 8. Update cost in a network of 200 nodes as a function of number of servers.

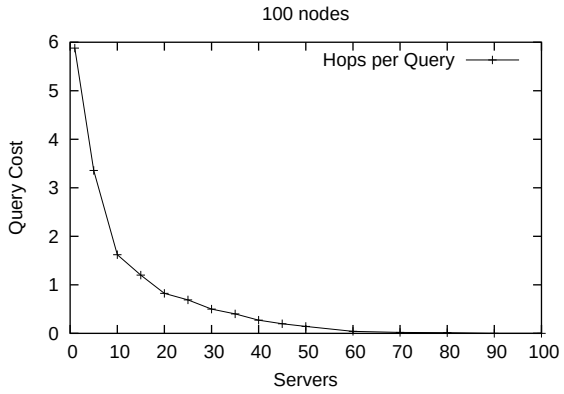

Figure 9. Query cost in a network of 100 nodes.

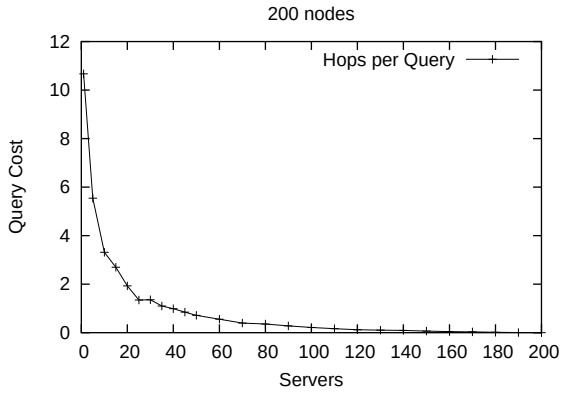

Figure 10. Query cost in a network of 200 nodes.

The success rate for queries is virtually 100 percent in many cases. This is not true, however, for the cases of small numbers of servers and particularly in the less dense 100-node network. When the network is not very dense, some query paths may not intersect with any update path. However, as redundancy is introduced with the addition of new servers, the success rate quickly grows. This trend is seen in Figures 11 and 12.

4.1.2. GCLP in High-Density Networks To simulate a high-density network, 500 nodes were placed in an area of 2000 meter by 2000 meter. Figure 13 shows the scalability property of the protocol as a growing number of servers rapidly leads to lower update traffic per server and, eventually, a constant update traffic per server for large numbers of servers (as expected from Section 3.2).

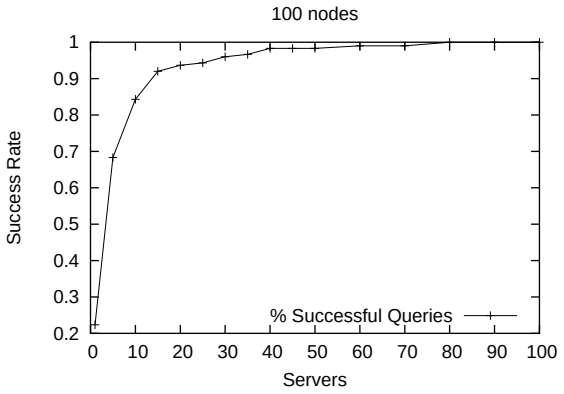

Figure 11. Success rate in a network of 100 nodes as a function of the number of servers available on the network.

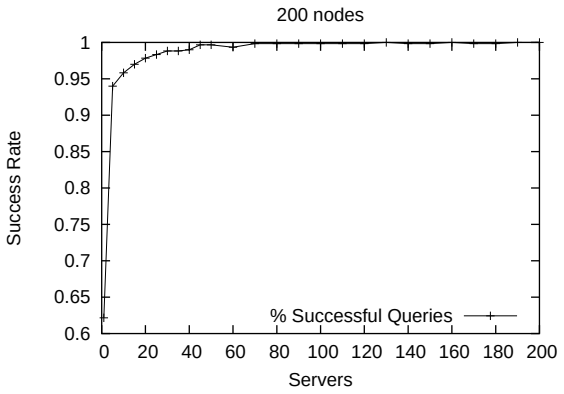

Figure 12. Success rate in a network of 200 nodes as a function of the number of servers available on the network.

The cost of queries also decreases quickly with growing numbers of servers. This is shown in Figure 14. At the same time, the success rate is virtually 100 percent for any number of servers, as shown in Figure 15. This is due to the good connectivity of the network with high density.

4.1.3. GCLP in Sparse Networks The performance of *GCLP* in sparse networks is evaluated next. Figure 16 shows that proactive traffic per server does not seem to vary significantly with the number of servers. This is due to the low connectivity of the network. In such an environment the performance of the protocol is lowered by the inability of nodes to establish connections with each other due to limitations of their transmission range.

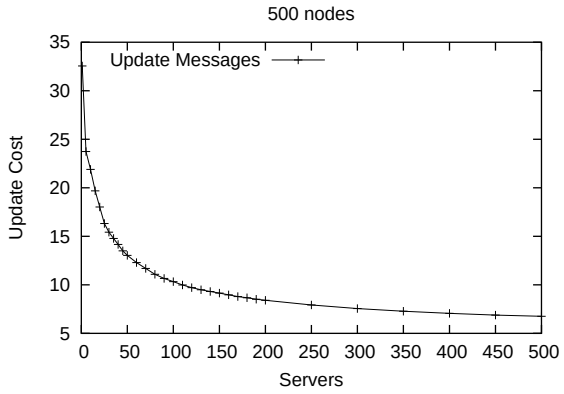

Figure 13. Update cost in a network of 500 nodes as a function of number of servers.

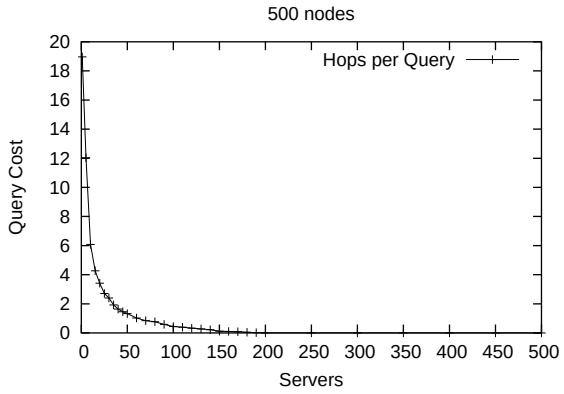

Figure 14. Query cost in a network of 500 nodes.

In the sparse network, the cost of queries does not decrease as fast with the number of servers as in denser networks. This is shown in Figure 17. The reason behind this is again the lower connectivity. As seen in Figure 18, the success rate grows slowly with the number of servers because of the increased difficulty of propagating the updates and queries in a low-density environment.

4.2. Effects of Mobility

We have done some preliminary evaluation in mobile environments. The results here should not be considered conclusive, since additional experimentation is necessary to gain adequate confidence in the data. The network under examination is a moderately dense network of 100 nodes with 250 meters transmission range, deployed in an area of 2000

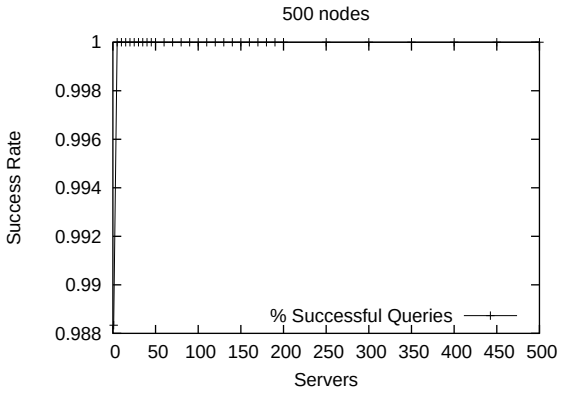

Figure 15. Success rate in a network of 500 nodes as a function of the number of servers available on the network.

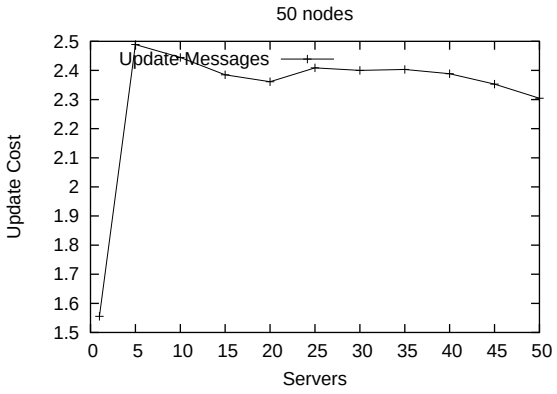

Figure 16. Update cost in a network of 50 nodes as a function of number of servers.

by 2000 meters. To measure the effects of node mobility, scenarios with node speeds of 10 m/s, 20 m/s, and 30 m/s are compared against the data from a static network. The same measures of overhead traffic and query efficiency are used to analyze performance as described in Section 4.

Figure 19 show that overhead traffic remains virtually unchanged when mobility is introduced into the network. Thus, our protocol appears to perform well under conditions where node mobility exists. However, we believe that more extensive simulations are necessary to confirm this observation. With mobile content servers, accuracy of the physical location information changes with time. This accuracy is not measured in our evaluation with mobility, and can become important if geographical routing is to be used by clients to communicate with the content servers (on the

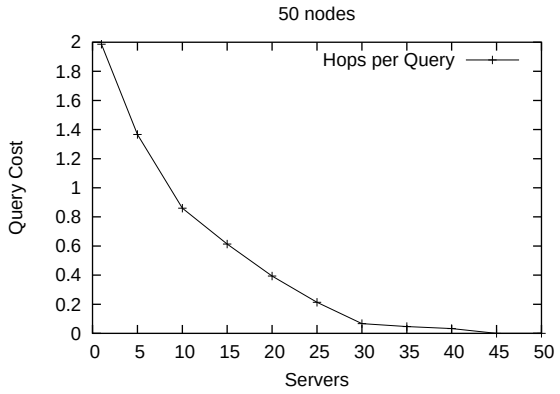

Figure 17. Query cost in a network of 50 nodes.

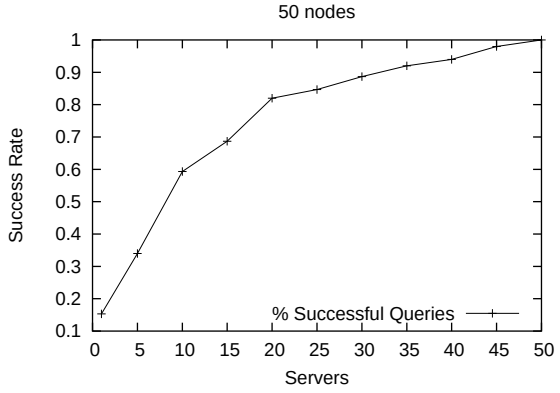

Figure 18. Success rate in a network of 50 nodes as a function of the number of servers available on the network.

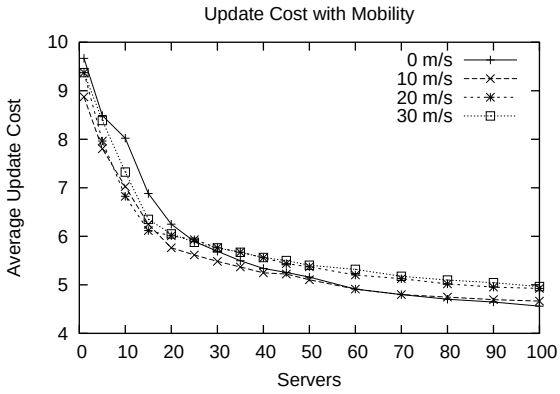

Figure 19. Update cost in a network of 100 nodes as a function of number of servers for different node speeds.

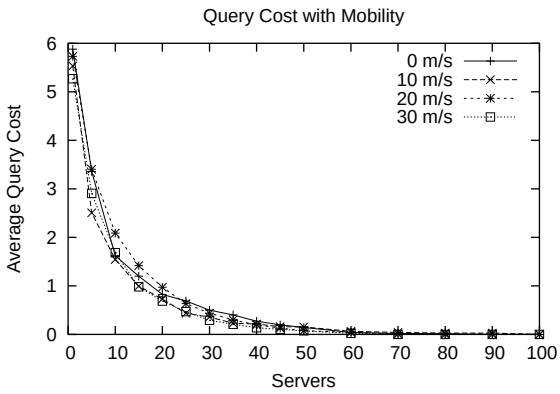

Figure 20. Query cost in a network of 100 nodes for different node speeds.

other hand, if our content location service is merely being used to determine the IP address of a content server, then the accuracy of the physical location of the content server is not as important).

The cost of locating content does not seem to vary significantly with the changes in node speed. As Figure 20 shows, the hops travelled by query messages per location attempt is similar for the static and dynamic topologies.

An interesting phenomenon occurs when considering query success rates. As Figure 21 shows, increased node mobility actually seems to lead to small increases in query success rates for small numbers of servers. This is due to the fact that mobile nodes will lead to location information being present in more parts of the network as nodes formerly located along the update paths travel through the network and answer queries.

5. Conclusions and Future Work

In this paper we have presented a protocol for content discovery in location-aware mobile ad hoc networks. The protocol, Geography-based Content Location Protocol, *GCLP*, uses location information to achieve scalability and cost effectiveness as measured by distance between clients and discovered servers. Simulation results demonstrate that *GCLP* is indeed an efficient content discovery scheme.

Several optimizations and variations on *GCLP* are plausible.

- One optimization is to allow nodes to suppress their

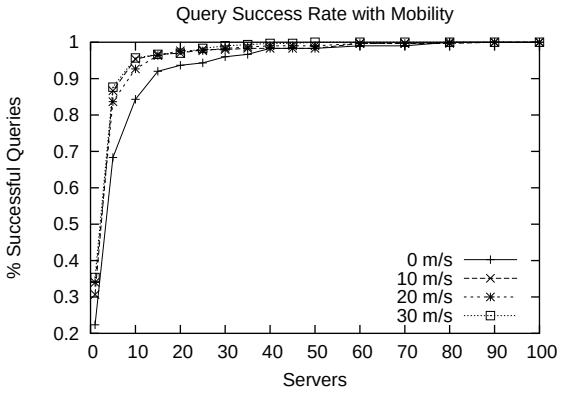

Figure 21. Success rate in a network of 100 nodes as a function of the number of servers available on the network for different node speeds.

updates in a specific direction if they already have a neighbor in the given direction that is serving the same content. This can reduce update cost. Alternatively, the nodes may selectively send updates in a greater or smaller (than four) number of directions, with the actual directions and their number being chosen to meet a desired update cost versus query cost trade-off. Similarly, the directions of updates may also be changed over time.

- Attempting to route queries and updates in a straight line can become difficult when their are obstacles that create "holes" in the network topology. Thus, possible optimization that aims at rerouting trajectories around such regions in the network (similar to geographical routing schemes). Care must be taken to achieve favorable trade-off between the update overhead and the ability to disseminate updates along desired directions, when such rerouting schemes are used.
- Another variation is to use a different heuristic for selecting next hops on update trajectories. For instance, the heuristic may give more weight to the distance between the nodes or to the deviation from the perfect trajectory (e.g., using rating $R = d^2/r$). Also possible is the consideration of other factors. For example, the age of entries in the neighbors table may be used to give more weight to more recent entries.
- In our algorithm, we assumed that each node knows the physical location of the nearby nodes. This information is used in choosing the next node on update trajectories. An alternative is to allow each node to simply broadcast an update including its own location information, and the desired direction for the next hop, without explicitly specifying the next node that should forward the update. Each node hearing this update can independently decide whether it should forward the update or not. One approach for implementing this as follows: When a node B hears an update message transmitted by node A (including A's physical location), node B first determines whether it is "approximately" in the direction in which the message is to be propagated. If the answer is affirmative, then node B chooses a "backoff interval" that varies inversely with its "rating R" (R may be evaluated in a variety of ways – note that B knows its own location, so B can take into account locations of A, B and the desired direction in evaluating rating R). Essentially, nodes that have a higher rating will tend to choose a lower backoff interval. Subsequently, node B waits for duration of the backoff interval (analogous to the IEEE 802.11) and forwards the update if B does not hear another node forwarding the same message in the desired direction. Clearly, many variations on this basic mechanism are also possible.

Further work is also needed to evaluate content location in mobile environments. This paper presented a preliminary evaluation. However, the trade-off between accuracy of location information obtained by clients, and the overhead in propagating the location information has not been evaluated in this paper. New techniques may have to be developed to improve this trade-off.

Acknowledgements

This research is supported in part by the National Science Foundation. We thank Bill List for his help in developing the single-server version of the proposed scheme.

References

[1] E. Guttmann, C. Perkins, J. Veizades, and M. Day, "Service location protocol," *IETF Internet Draft, RFC 2608*, 1999.

[2] H. Chen, D. Chakraborty, L. Xu, and T. Joshi, "Service discovery in the future electronic market," in *Proc. Workshop on Knowledge Based Electronic Markets, AAAI*, 2000.

[3] H. Chen, A. Joshi, and T. W. Finin, "Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether," *Cluster Computing*, vol. 4, no. 4, pp. 343–354, 2001.

[4] S. D. Gribble, M. Welsh, J. R. von Behren, E. A. Brewer, D. E. Culler, N. Borisov, S. E. Czerwinski, R. Gummadi, J. R. Hill, A. D. Joseph, R. H. Katz, Z. M. Mao, S. Ross, and B. Y. Zhao, "The ninja architecture for robust internet-scale systems and services," *Computer Networks*, vol. 35, no. 4, pp. 473–497, 2001.

[5] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An architecture for a secure service discovery

service," in *Mobile Computing and Networking*, pp. 24–35, 1999.

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, Apr. 2001.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, vol. 2009, pp. 46–67, 2001.

[8] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329–??, 2001.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pp. 161–172, ACM Press, 2001.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proceedings of the 2001 ACM SIGCOMM Conference*, pp. 149–160, 2001.

[11] D. Doval and D. O'Mahony, "Nom: Resource location and discovery for ad hoc mobile networks," in *Med-hoc-Net 2002*, (Sardegna, Italy), September 2002.

[12] V. Verma, S. Helal, N. Desai and C. Lee, "Konark a service discovery and delivery protocol for ad-hoc networks," in *Third IEEE Conference on Wireless Communication Networks (WCNC)*, (New Orleans, LA), March 2003.

[13] Q. Zhang, B. Li, J. Liu, K. Sohraby and W. Zhu, "Resource discovery in mobile ad hoc networks," in *Handbook on Ad Hoc Wireless Networks*, CRC Press, 2002.

[14] O. Ratsimor and D. Chakraborty, "Allia: Alliance-based service discovery for ad-hoc environments," in *ACM Mobile Commerce Workshop*, September 2002.

[15] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Gsd: A novel groupbased service discovery protocol for manets," in *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002)*, 2002.

[16] B. Nath and D. Niculescu, "Routing on a curve," in *HOTNETS-I*, (Princeton, NJ), October 2002.

[17] I. Aydin and C.-C. Shen, "Facilitating match-making service in ad hoc and sensor networks using pseudo quorum," in *11th IEEE International Conference on Computer Communications and Netwroks (ICCCN)*, (Miami, FL), October 2002.

[18] Jivodar Tchakarov, "Efficient content location in mobile ad hoc networks," *Research Thesis, University of Illinois at Urbana-Champaign*, 2003.