Efficient Content Location in Mobile Ad hoc Networks

Jivodar B. Tchakarov
Nitin H. Vaidya
University of Illinois at Urbana–Champaign
{tchakaro,nhv}@uiuc.edu

June 9, 2003

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 location information to lower proactive traffic while minimizing query cost. The results of our analysis show that* GCLP *performs favorably in terms of overhead, latency, 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. The rise and fall of a popular file sharing service such as *Napster* has led to the need for new and more creative protocols for location and sharing of data. While previous approaches were following the client-server model, the need for avoiding centralized responsibility and increasing stability has been the driving force behind a variety of new approaches that have been set forth in the context of 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 available on the Web. Cell phones and wirelessly connected PDA's have become a new kind of storefront for e-tailers. Such technical novelties have further increased the need to efficiently locate not only general content but specific services available on the network. 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 device closest to him.

To deal with this problem, a variety of algorithms have been presented to solve the problem of resource location 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 [2, 3, 4, 4, 5, 6, 3]. What all these models have in common is the reliance upon a centralized storage that would handle queries by users. This assumption violates the requirements of ad hoc networks where all nodes should be considered equal and no one node should be given extra responsibility when compared to its peers.

Decentralized approaches [12, 9, 10, 7, 8] remove the reliance upon a central directory server but do not take link cost into account when computing routes. This makes them impractical for use in ad hoc networks. Some protocols have been proposed specifically for such resource poor environments [19, 16, 22, 18, 20] but these still rely heavily on the use of broadcast making them too expensive to operate.

A novel approach to disseminating service information is described in [24]. The authors propose the use of location information for routing. The protocol provides for all nodes to periodically send advertisements along geometric trajectories. At each node in the trajectory, a backwards pointer is set up establishing paths leading to the source host. Any node that wishes to communicate with another node need simply send a query along a path that intersects with the advertisement path. The query is then forwarded by the node on the path to the desired host. The host that receives a query may then send a reply to the requesting node. This idea is further developed in [25]. The authors propose 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. Solving this scalability problem is a contribution of this work.

In this paper, we present a content location service, the *Geography-based Content Location Protocol* (GCLP) that takes location information into account to provide an efficient content location service to nodes in an ad hoc network. GCLP assumes that all devices in the network know their own 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 attempt to locate content need only contact one of these nodes to become aware of the presence of the desired content on the network and the closest available server. The proposed protocol must be scalable, fault-tolerant, adaptable, and accurate. Since we attempt to work in an environment where communication cost is high, we also want queries to be relatively cheap while still allowing for lower overhead cost and scalability of the protocol. Finally, a response should specify the server that would result in the cheapest connection as measured by distance between the server and client. In a uniformly distributed dense network, a shorter distance would translate into smaller number of hops and thus smaller cost.

2 Geography-based Content Location Protocol

In this section we provide the design description of *GCLP*. We start by giving a brief overview of the protocol before describing the details of the design.

2.1 Protocol Overview

The proposed protocol treats all nodes in the network as equal. Thus, no single node takes more responsibility than others. Nodes 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*'s) are nodes that host location information about one or more resources available. These nodes are responsible for providing timely and efficient 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 [25]. Periodically, a *CS* will transmit update messages (*UM*) to specific nodes in the network. These updates advertise available resources and the *CS* that hosts them. *UM*'s follow a predefined trajectory through the network similar to the trajectory-based schemes described in Section 1 [24, 25]. 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 *CLS*. If it receives a query about content it knows the server of, it will reply with the server address.

A client may locate any content on the network by sending out a query message (*QM*). The query is similarly propagated along predefined trajectories. In a dense network, these trajectories are guaranteed to intersect at least one update trajectory. The *CLS* at the intersection point that receives the query responds with a reply message (*RM*) 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. Finding the closest *CS* available is an important benefit as it generally means fewer hops between the client and the server, which, in turn, translates into lower connection cost.

To make sure that all nodes on the network know the location of all their neighbors when selecting next hop hosts 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 position.

The following sections explain the functionality of all three types of nodes in detail. Proofs are provided to demonstrate the correctness of the protocol and its ability to keep cost low.

2.2 Protocol Details

This section provides details about the design of *GCLP*. It describes the three phases of the protocol – neighbor discovery phase, content advertisement, and content discovery. It also presents proof of protocol correctness and efficiency.

2.2.1 Content Advertisement

Content advertisement is performed by periodically sending Update Messages (*UM*) through the network, similar to [25]. Each *CS* periodically initiates advertisements by sending *UM*'s in four opposite geographical directions, or trajectories–North, South, East, and West. Each advertisement specifies the location of a single resource at the *CS*. To accomplish this, the *CS* uses the geographic location of its immediate neighbors to select the next *CLS* in the given direction. The algorithm for selecting next-hop nodes is described in detail below. The *UM* is then sent to that node. Upon receiving the update, the chosen *CLS* adds the information to its *Content Location Table* and uses the same algorithm to decide which node in the direction of the update will be the next *CLS*. 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 *CLS*'s are shaded.

An important part of the protocol is the selection of the *CLS* nodes along the geographic direction of a *UM*. In designing an algorithm several heuristics may be used:

- Picking nodes closest to the line of the direction. This keeps the line of *CLS*'s straighter but may not be as efficient since some *CLS*'s may be needlessly close to each other even if nodes are available farther away in the sector.

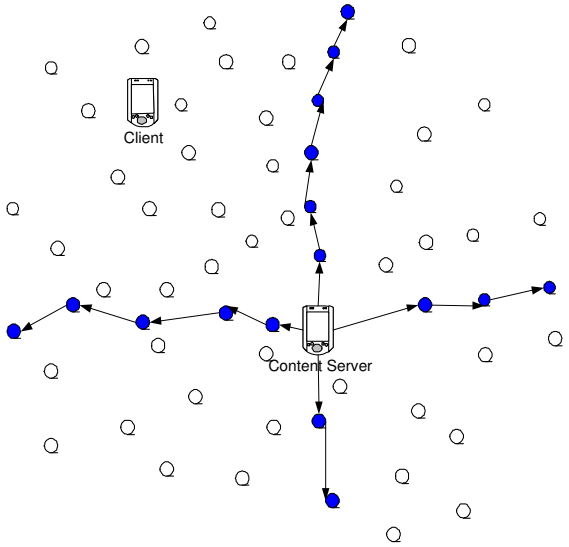

Figure 1: A Content Server propagating updates through the network. The server's selected content location servers are shaded.

- Picking nodes with greatest distance from the current node. This may be more efficient as it provides for greater spaces between neighboring *CLS*'s. 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.

In [25], 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 *CLS*'s, we would like to achieve two things–cover a larger distance between *CLS*'s 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.

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

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,

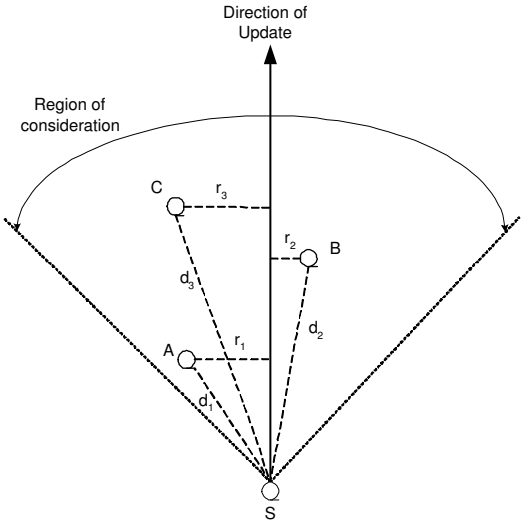

Figure 2: A node picks the next hop for an update message among nodes in the appropriate sector.

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 given 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.

As described until now, the protocol is simply a variation on [25] and, like [25], does not scale well. It does not deal with the possibility of more than one content server hosting the same resource in the network. Next, we describe a major component of our protocol that provides for protocol scalability and cost efficiency. If *UM*'s 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 *CLS* chooses whether to forward a *UM* or not. If a *CLS* receives multiple advertisements for a particular resource, it will only forward updates from the *CS* closest to it. In case there is a tie, the *CLS* will continue to propagate updates from the first server it received advertisements from. 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. This method also has the important property that each *CLS* will know the location of servers closest to them and thus answer queries resulting in data connections of smallest cost possible as measured by distance (which is proportional to the number of hops in dense networks).

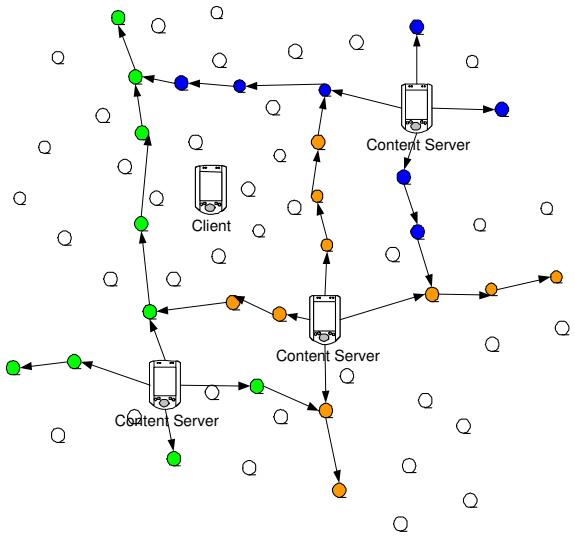

Figure 3: Several servers advertise availability of the same resource. The content location servers for each server are given in different shades.

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 *UM*'s. 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 *QM* will send a query response message (*RM*) to the requestor. The *RM* follows regular greedy geographic routing with each node forwarding the packet to its neighbor closest to the destination.

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 *CLS* 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 *CS* closest to its geographical location.

3 Analysis of GCLP

In this section we provide an analysis of the Geography-based Content Location Protocol, including proof of correctness. It is important to point out that the network under consideration in this section is a dense network with nodes at each point in the logical space. Also, in the context of this discussion and of a dense network, a hop is assumed to mean one transmission range.

3.1 Proof of Correctness

To prove that at least one intersection of update trajectories and query trajectories still exists, even with some update trajectories being interrupted by updates from closer *CS*'s, we need to prove that for at least one trajectory will propagate in each one of the four geographical directions. The proof is simple and is illustrated in Figure 5. We have two *CS*'s, *S1* and *S2*. Two *CLS*'s, *A* and *B*, interrupt some update trajectories and forward others. To prove 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 from one trajectory to get interrupted, such as the trajectory form *S1*, there must exist a server whose perpendicular trajectory causes the interruption at that point, *S2*. 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.

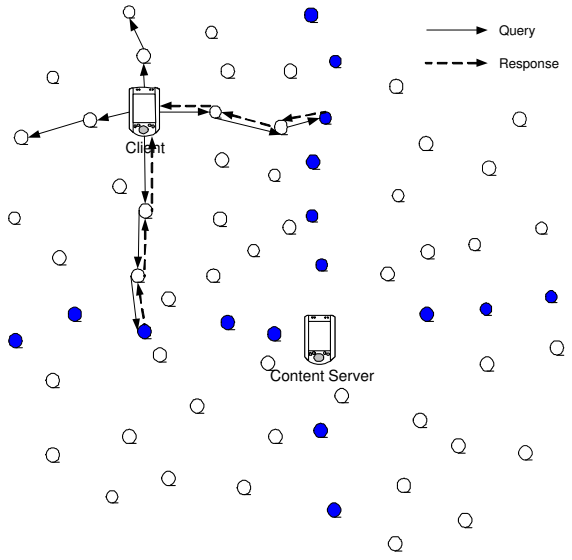

Figure 4: A client attempts to locate content by sending queries through the network. The queries are answered once they reach a content location server.

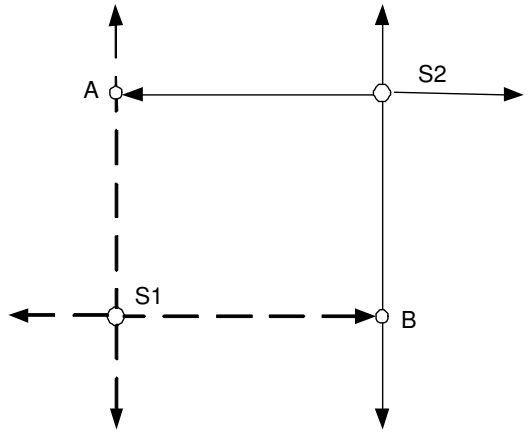

Figure 5: At least one trajectory will propagate in the direction of an interrupted update trajectory.

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 *CS* in the four directions will have a total cost of $w+l$. Thus, in the case of a single *CS*, 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 *CLS*'s because they already host the resource. 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 *CS*'s in the network.

$$Rays\ per\ Server = \frac{6}{2^{k+1}+1} \tag{2}$$

In between the two extremes, we can prove that the protocol overhead generated per server drops exponentially as new servers are added to the network. It can be shown that the amount of proactive traffic per server as new and evenly spaced servers are introduced into the network will follow a pattern described in Equation 2. Notice that, as mentioned above, for large numbers of servers the traffic introduced by new servers will be constant and the complexity of the algorithm approaches $O(n)$.

Our simulations show that the overhead traffic of the protocol does indeed scale well and additional servers lead to exponential drops in overhead traffic per server. The scalability of the protocol is clearly seen in the simulations presented in Section 4.

3.3 Response Analysis

An 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 CLS, *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. In this case, the closest server is discovered.

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.

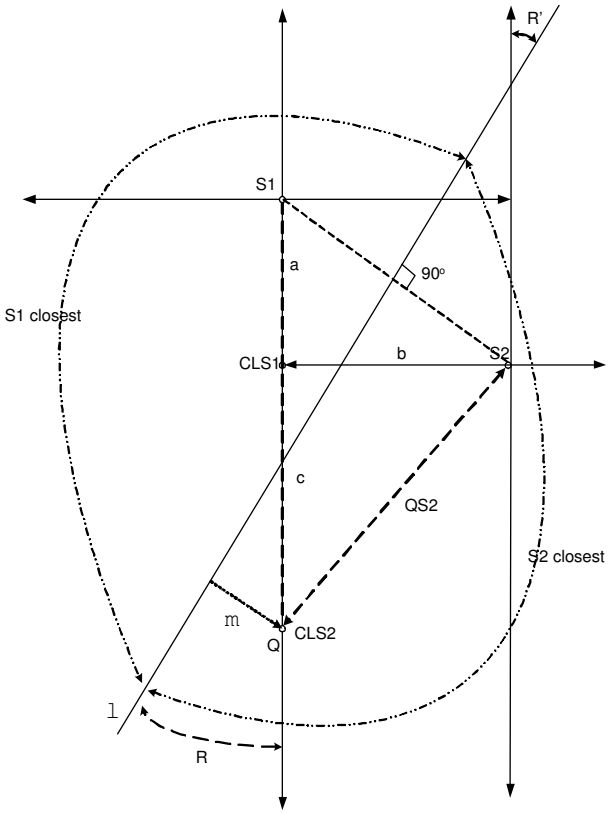

Figure 6: A client, Q, may locate a resource on a more distant server, QS1, instead of the closer QS2.

Let us express the ratio of *QS1/QS2* in terms of the distances a, b, and c, as shown in Figure 6. Equation 3 shows this formula.

$$QS1/QS2 = (a + c)/(\sqrt{b^2 + c^2}) \qquad (3)$$

It can be algebraically shown that this gives us a worst case result as follows:

$$QS1/QS2 = \sqrt{a^2 + b^2}/b \leq \sqrt{2b^2}/b = \sqrt{2} \qquad (4)$$

According to Equation 4 we can conclude that, in the worst case, *QS1/QS2* $\leq \sqrt{2}$. Thus, the maximum error produced by our protocol as measured by the ratio between the distance found and the actual closest distance is relatively small, $\sqrt{2}$ times the optimal in a dense network.

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. A variety of 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 presented for networks of all densities. To study the effects of mobility, a moderate density network of 100 nodes was evaluated with varying nodes speeds. The Random Waypoint mobility model was used to simulate node mobility. Transmission range of the wireless nodes used for the simulations was 250 meters.

Several measures were considered when evaluating the protocol. The proactive traffic due to Update Messages was a major factor to the scalability and adaptability of our *GCLP*. The cost and success rate of queries is another crucial piece of information. To evaluate these properties, the following measures are considered in evaluating protocol performance:

- **Updates per Initialization per Server (UpIpS)**: This measure is computed by dividing the *Updates per Initialization* by the number of servers present in the network. This metric is directly related to the *Rays per Server* metric discussed in Section 3.2 since the cost of the ray is directly proportional to the number of hops in the network and thus to the number of update messages necessary.

$$UpIpS = \frac{UpI}{Number\ of\ Servers}$$

- **Hops per Query (HpQ)**: A goal of the protocol is to keep query cost low. This measure is used to quantify the cost of searching the network for content. In our simulations, each node in the network initiates a query once. The Hops per Query is calculated by taking the number of all transmissions of *QM*'s and dividing by the number of queries initiated.

$$HpQ = \frac{Total\ Query\ Messages}{Number\ of\ Queries\ Initiated}$$

- **Success Rate (SR)**: The accuracy of our protocol is measured by this metric. It is computed by taking into account the number of queries that receive responses and dividing by the total number of queries. If a query receives more than one response, it is only counted as successful once.

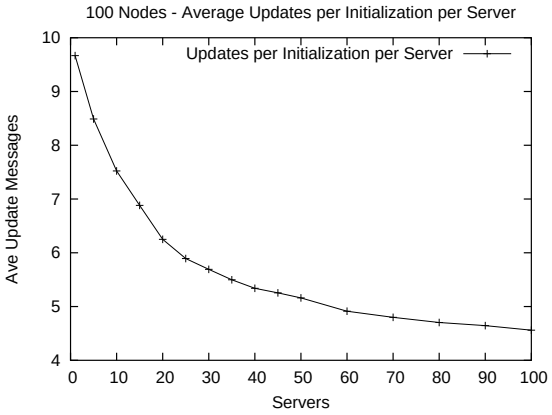

Figure 7: Proactive Update traffic per server in a network of 100 nodes as a function of number of servers.

$$SR = \frac{Successful\ Queries}{Total\ Number\ of\ Queries\ Initiated}$$

These results of the simulations, as measured by the above metrics, are described in the following sections.

4.1 Effects of Network Density

This section studies the effects of node density in the network on the performance of *GCLP* as measured by the factors specified in Section 4.

4.1.1 GCLP in Moderate-Density Networks

As seen in Figures 7 and 8, the amount of update messages per server shrinks rapidly with the addition of new servers. This pattern relates directly to the analysis in Section 3.2. The quick drop can be seen in update traffic per server with the addition of servers, as well as the constant complexity for large numbers of servers. This is due to the construction of the update grid. The above mentioned figures clearly demonstrate the scalability of the protocol.

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 property is shown in Figures 9 and 10.

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. We stipulate that the reason is the non-perfect connectivity in the not too dense network. However, as redundancy is introduced with the addition of new servers, the success rate quickly grows to reach a virtual 100 percent.

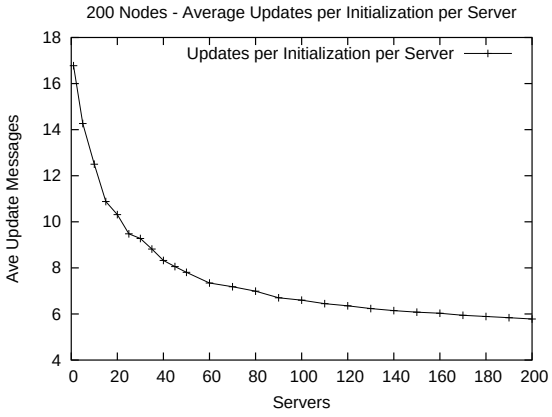

Figure 8: Proactive Update traffic per server in a network of 200 nodes as a function of number of servers.

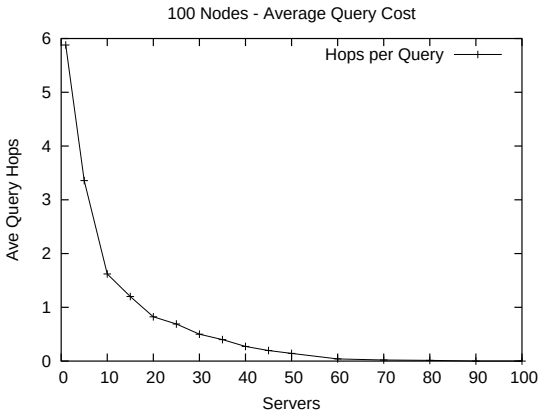

Figure 9: Query messages generated per application layer query in a network of 100 nodes.

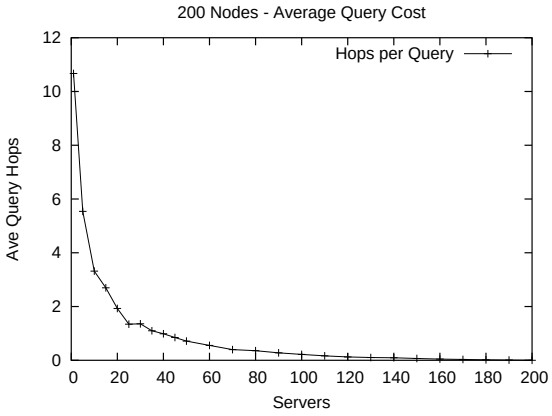

Figure 10: Query messages generated per application layer query in a network of 200 nodes.

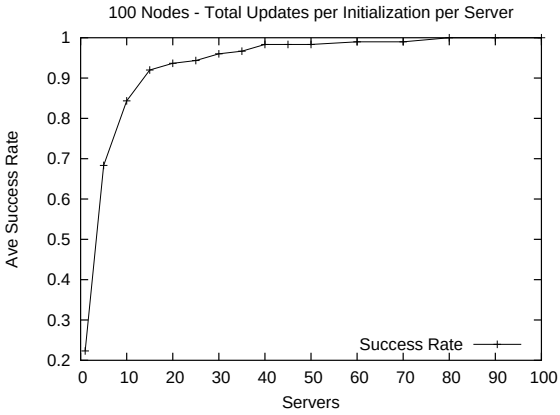

Figure 11: Query success rate in a network of 100 nodes as a function of the number of servers available on the network.

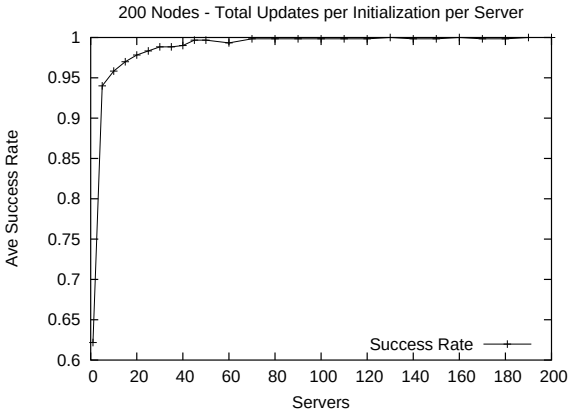

Figure 12: Query success rate in a network of 200 nodes as a function of the number of servers available on the network.

These tendencies are shown in Figures 11 and 12.

4.1.2 GCLP in High-Density Networks

To simulate a high-density network, 500 nodes were placed on the same area of 2000 by 2000 meters.

Figure 13 shows the good scalability property of the protocol as a growing number of servers rapidly leads to less proactive traffic per server and, eventually, a constant overhead traffic per server for large numbers of servers as expected from Section 3.2. This is due to the construction of the update grid.

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

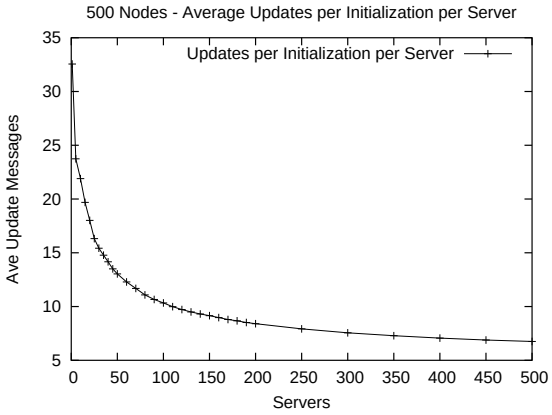

Figure 13: Proactive Update traffic per server in a network of 500 nodes as a function of number of servers.

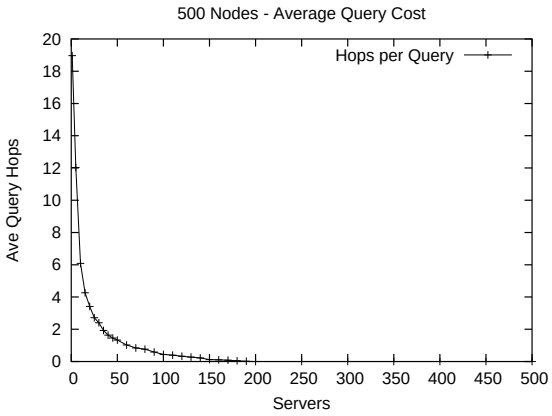

Figure 14: Query messages generated per application layer query in a network of 500 nodes.

Figure 15. This is due to the good connectivity of the network and its high density.

4.1.3 GCLP in Sparse Networks

The performance of *GCLP* in sparse networks is evaluated next. Figure 16 show 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.

The cost of queries does not decrease as fast with the number of servers. This is shown in Figure 17. The reason behind this is again the lower connectivity and the inability of the protocol to reliably build the

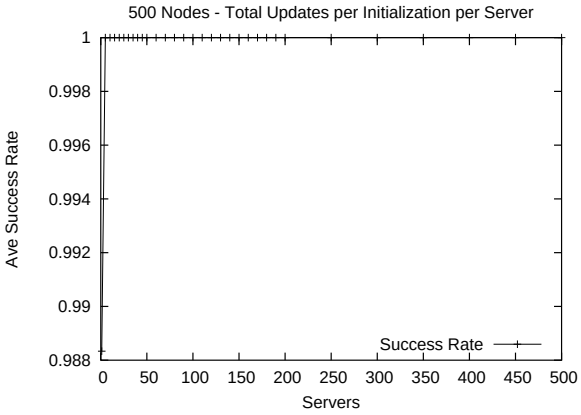

Figure 15: Query success rate in a network of 500 nodes as a function of the number of servers available on the network.

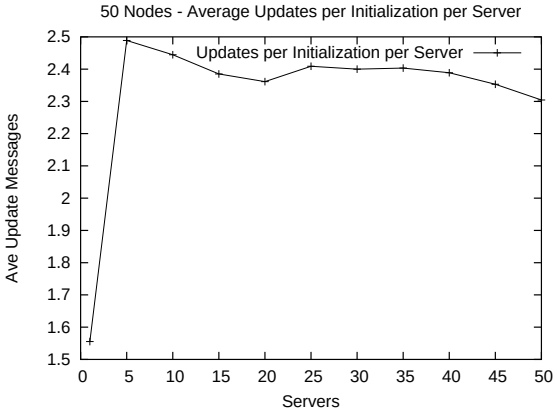

Figure 16: Proactive Update traffic per server in a network of 50 nodes as a function of number of servers.

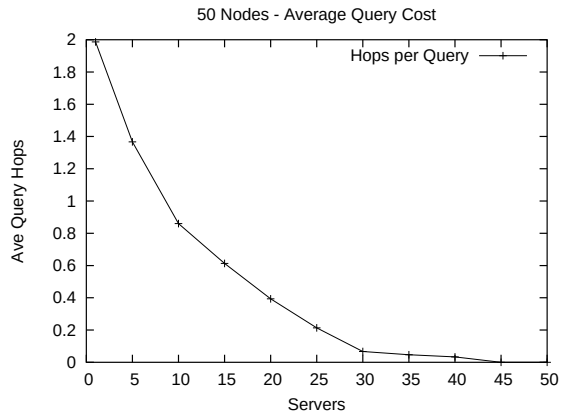

Figure 17: Query messages generated per application layer query in a network of 50 nodes.

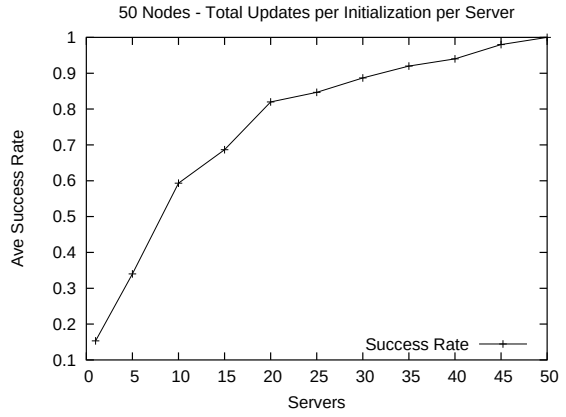

Figure 18: Query success rate in a network of 50 nodes as a function of the number of servers available on the network.

update grid. This is further shown in Figure 18 which shows that the success rate grows slowly with the number of servers because of the increased difficulty of constructing the update grid in the low-density environment and the lower reliability of the queries.

4.2 Effects of Mobility

Mobility is a property of ad hoc networks that leads to problems for many systems deployed in such networks. It is important that mobility does not have a significant effect on the performance of our protocol or its adaptability and efficiency in such an environment will be questionable.

The network under examination is a moderately dense network of 100 nodes with 250 meters transmission range, deployed in an area of 2000 by 2000

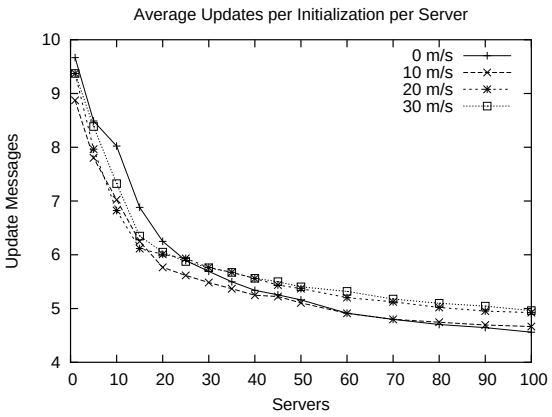

Figure 19: Proactive Update traffic per server in a network of 100 nodes as a function of number of servers for different node speeds.

meters. Several factors played a role in selecting such a network topology. First, a network of such density is of higher possibility of deployment than the very dense 500 node networks. Also, the *ns-2* network simulator used to run the simulations imposes the use of a less dense network topology since it does not scale well with higher numbers of mobile nodes. To measure the effects of node mobility, scenarios with node speeds of 10 m/s, 20 m/s, and 30 m/s were 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. We stipulate that the small fluctuations in the data are due to the different network topologies and not the presence of node mobility. Thus, our protocol performs well under conditions where node mobility exists. Adaptability was one of the requirements of our protocol and it has been met according to our measurements.

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 topology and the three dynamic topologies.

An interesting phenomenon occurs when considering query success rates. As Figure 21 shows, increased node mobility actually seems to lead to minimal 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 grid travel through the net-

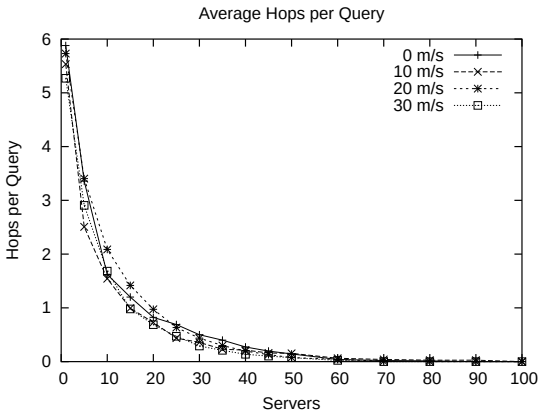

Figure 20: Query messages generated per application layer query in a network of 100 nodes for different node speeds.

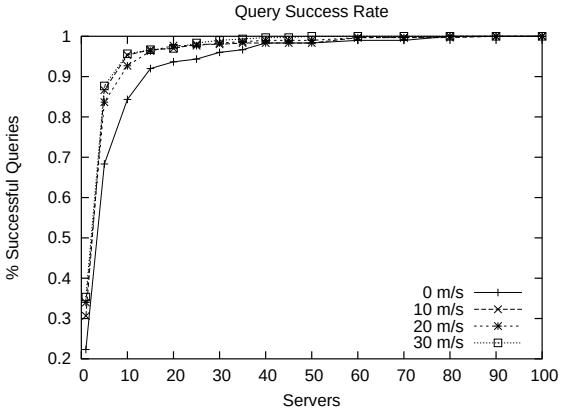

Figure 21: Query success rate in a network of 100 nodes as a function of the number of servers available on the network for different node speeds.

work and answer queries.

5 Conclusion and Future Work

There are a number of possible optimizations to *GCLP*. These optimization are the subject of future work and have not been studied extensively from the point of view of performance in real-world conditions. Such optimizations may be used to improve protocol reliability and efficiency. One optimization is allowing nodes to suppress their updates in a specific direction if they already have a neighbor in the given direction that is serving the same content. A possible optimization that aims at rerouting trajectories around empty regions in the network may be employed to increase fault-tolerance. When a node chooses a next hop in a trajectory and notices that it has no neighbors in the desired sector, instead of interrupting the trajectory, it may lookup neighbors in the adjacent sectors and pick one of them to continue the trajectory. A third variation of the protocol may use a different heuristic for selecting next hops in trajectories. A number of other algorithms are possible that may give more weight to the distance between the nodes or to the deviation from the perfect trajectory (e.g., rating $R = d^2/r$). Also possible is the inclusion of other factors in the algorithm. For example, the age of entries in the neighbors table may be used to give more weight to more recent entries.

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. We have also presented a mathematical analysis of our protocol as well as results from our simulations. Our analysis and simulation results demonstrate that *GCLP* is indeed a viable and efficient content discovery scheme.

References

[1] "www.napster.com."

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] A. I. T. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *Symposium on Operating Systems Principles*, pp. 188–201, 2001.

[12] 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.

[13] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distributed object location in a dynamic network," Tech. Rep. UCB/CSD-02-1178, Apr. 2002. Updated version to appear in SPAA 2002.

[14] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *Proceedings of ACM ASPLOS*, ACM, November 2000.

[15] Y. Chen, R. H. Katz, and J. D. Kubiatowicz, "Scan: A dynamic, scalable, and efficient content distribution network," in *Proceedings of the International Conference on Pervasive Computing (Pervasive 2002)*, (Zurich, Switzerland), August 2002.

[16] V. V. Sumi Helal, Nitin 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.

[17] L. Cheng and I. Marsic, "Service discovery and invocation for mobile ad hoc networked appliances," in *Proceedings of the 2nd International Workshop on Networked Appliances (IWNA2000)*, (New Brunswick, NJ), November 2000.

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

[19] 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.

[20] 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.

[21] C. Perkins, "Ad-hoc on-demand distance vector routing," in *MILCOM '97 panel on Ad Hoc Networks*, 1997.

[22] Q. Z. B. L. Jiangchuan Liu, Kazem Sohraby and W. Zhu, "Resource discovery in mobile ad hoc networks," in *Handbook on Ad Hoc Wireless Networks*, CRC Press, 2002.

[23] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *MOBICOM'02*, (Atlanta, GA), September 2000.

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

[25] 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.