

Adaptive Strategies for Efficiently Locating Internet-based Servers in MANETs

Hongbo Jiang
Division of Computer Science, EECS
Case Western Reserve University
Cleveland, OH 44106
hongbo.jiang@case.edu

Shudong Jin
Division of Computer Science, EECS
Case Western Reserve University
Cleveland, OH 44106
shudong.jin@case.edu

ABSTRACT

Providing Internet access to Mobile Ad hoc Networks (MANETs) can greatly extend their applications, increase their scalability, and improve the quality of service. However, a critical problem is how the mobile hosts can locate Internet-based servers efficiently in a dynamic, unstructured network. Neither reactive strategies, where the hosts initiate on-demand server discovery, nor proactive strategies, where the servers periodically advertise their availability information, are optimal. To that end, this paper studies adaptive strategies that (1) combine both proactive advertising by the servers and on-demand discovery by the mobile hosts, and (2) determine the relative rate of proactive advertising and on-demand discovery adaptively according to the network characteristics including the host mobility level and the offered load. We propose and evaluate two novel, integrated algorithms. First, to determine the rate of proactive advertising, we propose an exponential backoff algorithm to probe the optimal operating point. Second, to reduce the network traffic due to reactive (on-demand) discovery, we propose a novel controlled flooding algorithm. Our simulation study reveals that, compared to the previous proactive strategies and reactive strategies, our adaptive strategies reduce network traffic for locating the servers by several times when the network has a moderate offered load and a low or high level of host mobility.

Categories and Subject Descriptors

C.2.1 [Computer Communication Networks]: Network Architecture and Design—*Wireless communication*; C.2.2 [Computer Communication Networks]: Network Protocols—*Routing protocols*

General Terms

Algorithms, Performance

Keywords

Mobile ad hoc networks, search algorithms, controlled flooding

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSWiM'05, October 10–13, 2005, Montreal, Quebec, Canada.
Copyright 2005 ACM 1-59593-188-0/05/0010 ...\$5.00.

1. INTRODUCTION

In a Mobile Ad hoc Network (MANET), a collection of wireless mobile hosts form a temporary network without the aid of any established infrastructure. A pair of hosts may communicate with each other with the help from other intermediate hosts. Many applications in both the commercial domain and the military domain can be supported by these networks, such as content distribution, information dissemination, ad hoc multimedia streaming, and distributed games, just to name a few.

The approach of combining MANETs with the Internet infrastructure has recently attracted attention [7]. This approach can greatly enrich the applications, increase the scalability, and improve the quality of service. First, the Internet supports a much larger set of popular applications. These applications, if made available via the interfaces between the Internet and MANETs, will further facilitate universal accessibility. Second, MANETs are infrastructure-less and they do not scale up to a large number of hosts. For example in large networks, routing and target discovery may incur extremely high traffic. On the other hand, the Internet has a well-established infrastructure. This infrastructure has excellent scalability and abundant network resources (so at least compared to MANETs). An Internet-based MANET has a more hierarchical architecture: at the high level the Internet connects a set of interfaces (between the Internet and MANETs) while at the low level a native MANET connects the mobile hosts. Such a hierarchical approach leads to a more scalable system. Third, in MANETs the network topology changes dynamically and the multi-hop communication route between a pair of hosts can be broken due to host mobility and possibly host joining/leaving. Therefore, it is difficult to provide high-quality service to the applications. On the other hand, by connecting MANETs to the Internet, they will likely provide better service to the applications. If the MANETs are well connected to the Internet via many interfaces, a pair of hosts can communicate with each other via an Internet shortcut.

However, an important question is how to locate the interfaces between the Internet and MANETs. Hereafter, we call the interfaces *Internet-based servers* in MANETs. In other words, how can the mobile hosts be informed of the existence of the servers and how can they reach the servers? In a stationary network this problem is trivial. However, in a mobile network, every time when a host requests the Internet service, it may have to locate a new server and find the path to this server. Due to the infrastructure-less nature of MANETs, it is expensive to do so. The usual search methods (of using query flooding) generate potentially very high network traffic. Making things worse, when the request rate increases, naive search methods require that search traffic be increased in proportional to the request rate, and hence overload the network.

This paper focuses on strategies for locating the servers at minimum network cost. Broadly there are two categories of strategies: reactive strategies and proactive strategies. With reactive or on-demand strategies, a mobile host initiates server discovery only when there is a request. The host floods a query into the network, hoping to hit at least one of the servers and to discover the path to the server. With proactive strategies, the servers advertise their availability by flooding packets into the entire network. In this way the mobile hosts obtain the routes to the servers. There are tradeoffs between these two categories of strategies. A number of factors play important roles in these tradeoffs. At a high level of host mobility, reactive strategies are favored since routes to servers (as the result of proactive advertising) will frequently become stale. At a high offered load (or request rate), proactive strategies will likely cause lower network traffic than reactive strategies do.

We study adaptive strategies for locating the servers in MANETs. By adaptive strategies, we mean (1) they are hybrid strategies that combine both proactive advertising by the servers and on-demand discovery by the mobile hosts, and (2) the relative rate of proactive advertising and on-demand discovery is determined by the current network characteristics (including the request rate and the mobility level). Our main contributions are two novel, integrated algorithms. First, to determine the rate of proactive advertising, we propose an *exponential backoff algorithm* to probe the optimal operating point. Second, to reduce the network traffic due to reactive (on-demand) discovery, we propose a *novel controlled flooding algorithm*, which generates low network traffic. Our simulation study reveals that, the combined use of these two algorithms is close to the optimal in minimizing network traffic. Compared to the previous proactive strategies and reactive strategies, our adaptive strategies reduce network traffic by several times when the network has a low or high mobility level and a moderate request rate.

The remaining of this paper is organized as follows. In the next section, we first describe our network model, contrast proactive strategies and reactive strategies, and illustrate the intuition behind our adaptive strategies. Section 3 describes our exponential backoff algorithm for proactive advertising, and Section 4 describes our novel flooding algorithm. Section 5 describes our simulation and performance evaluation. Section 6 briefly describes related work and finally Section 7 summarizes the paper.

2. PRELIMINARIES

In this section we describe the network model, contrast reactive strategies and proactive strategies, and then illustrate the intuition behind our adaptive strategies.

2.1 Network Model

We consider an ad hoc network with a large number of mobile hosts. Each host has a wireless communication interface such as IEEE 802.11b. We assume the interface has a fixed transmission range and the host is able to communicate with its physical neighbors who are within the transmission range. Any pair of hosts may communicate with each other directly, or via one or more intermediate hosts. The hosts can move in any direction. Therefore, the network topology changes dynamically. We ignore those hosts disconnected from the network.

A small subset of these mobile hosts have also interfaces connected to the Internet. We call these mobile hosts Internet-based servers. Therefore, the entire network consists of two levels: the high level Internet-based servers and the low level mobile hosts, as illustrated in Figure 1. Compared to the native ad hoc network, the Internet has much better performance (higher bandwidth, shorter latency, and lower packet loss rate, etc) so that we can almost as-

sume the Internet resources are infinite. Therefore, we ignore any performance limitation of the Internet, and focus on the mobile ad hoc network itself.

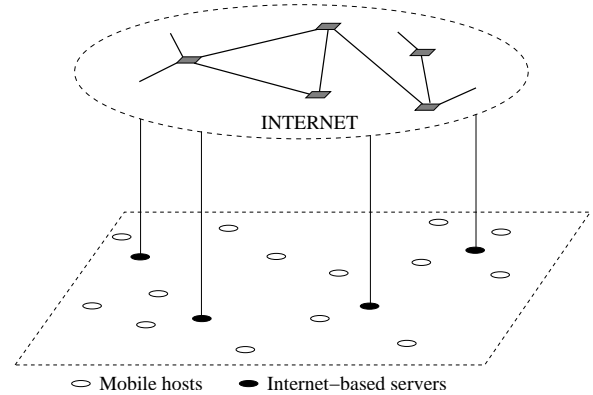


Figure 1: Two levels of a MANET with some hosts as interfaces to the Internet.

Any mobile host can request the Internet service at any time. If the mobile host has a direct connection to the Internet, the problem becomes trivial and we ignore this case. In general, the mobile host is one or multiple hops away from any servers. This mobile host will need to locate one of the servers and find a route. This is a network layer problem to be solved in this paper. Assume this host (source) has obtained a route to a server, or it has an up-to-date route in its cache¹. It includes the route in the packets to be sent and forward the packets to the next hop. Intermediate hosts just faithfully forward the packets according to the route specified by the source host. Concisely, we assume a source routing approach at the network layer, just like in the dynamic source routing (DSR) protocol [12]. However, the route discovery mechanism of DSR is considered inefficient to locate servers in our application. In DSR, a mobile host floods a query packet into the MANET in an on-demand fashion. As we will discuss later in this section, this strategy can be expensive.

2.2 Reactive Strategies

With reactive strategies, the mobile hosts locate the servers in an on-demand fashion. On each new request, a mobile host first checks if its cache contains the current route to a server. If so the host will use the cached route. If not, the source routing protocol will detect route failure, and the host will initiate a new route discovery. For example, DSR implements such a mechanism.

To perform server discovery in MANETs, the mobile hosts can use flooding techniques. A source host propagates a query packet to the network until it hits a server. Then the server replies the query via the reverse path. There are unrestricted (uncontrolled) flooding and controlled flooding techniques. With unrestricted flooding (such as the one used in DSR), query packets are propagated to the entire network. On the other hand, controlled flooding techniques avoid propagating query packets to more than enough hosts. Flooding stops as soon as any server is located. An example of controlled flooding techniques is the TTL-based flooding. A time-to-live (TTL) value in the packet indicates how many hops the packet can travel. Initially a small TTL value is used. Every time when an attempt to locate any server fails, the source host increases the TTL value until it eventually succeeds. In our application the goal

¹We assume the mobile host maintains a route cache which include routes to the discovered servers. The routes can be stale though.

is to locate just one of many servers. Therefore controlled flooding techniques are suitable.

2.3 Proactive Strategies

With proactive strategies, the servers broadcast their availability information (called an advertisement) periodically, regardless of the actual requests from the hosts. An advertisement is a packet with a unique message identifier. To broadcast one advertisement, the set of servers first coordinate with each other via the Internet and agree on the message identifier. Then they flood the advertisement into the network. When a mobile host receives an advertisement, it checks if it has received this advertisement earlier from any of the servers. If so the advertisement is discarded. Otherwise the route (which is included in the advertisement) is extracted and stored in the host's cache. Finally, the host attaches itself to the route in the advertisement, and broadcasts the advertisement in case any of its neighbors has yet to receive the advertisement.

On each request, the mobile host obtains the latest route to any server and uses this route to access the Internet. With those strictly proactive strategies, the mobile host itself will not initiate server discovery even when the cached route is stale. Thus, there will be frequent failures in accessing the Internet, especially if the proactive advertisements are not broadcast frequently. For this reason we assume that when the mobile host has a stale cached route, it will initiate server discovery.

2.4 Reactive versus Proactive Strategies

There are tradeoffs between reactive and proactive strategies. The tradeoffs are determined by a number of factors, including the level of host mobility and the offered load (*i.e.* the request rate). First, the level of host mobility is important. When the mobility is low, less likely cached routes will become stale. Thus, the servers need only to advertise their availability information at a relatively low frequency, which is high enough to avoid frequent reactive server discovery by the mobile hosts. Therefore, proactive strategies are effective. On the contrary, if reactive strategies are used, each mobile host still has to initiate discovery frequently. Second, the offered load is also important. When the request rate is low, proactive strategies become less efficient. The servers still need to advertise their availability information at a certain level (otherwise the cached routes would be stale), but few hosts will benefit from such proactive advertising due to the low request rate.

Figure 2 intuitively shows the regions where reactive strategies perform better and where proactive strategies perform better. There are two regions. (1) When the level of mobility is very high compared to the request rate, proactive strategies are ineffective. Therefore, there is a region for reactive strategies only². (2) When the request rate is moderate or very high compared to the level of mobility, proactive strategies become effective. However, even when proactive advertising is frequent, there is still a need for reactive discovery (*i.e.* there are always stale routes). Hence, above the region for reactive strategies, there is a single region for *hybrid* strategies that combine both proactive advertising and reactive discovery. Within this region, if the request rate increases or the mobility level decreases, the optimal operating point is "more proactive"; otherwise the optimal point is "more reactive".

2.5 Intuition behind Our Adaptive Strategies

Our general discussion in the last subsection indicates that, the key problem is how to find the optimal operating point. This opti-

²When the mobility level is very high, the cost due to increased proactive advertising always outweighs the reduced reactive discovery. Hence purely reactive strategies are optimal.

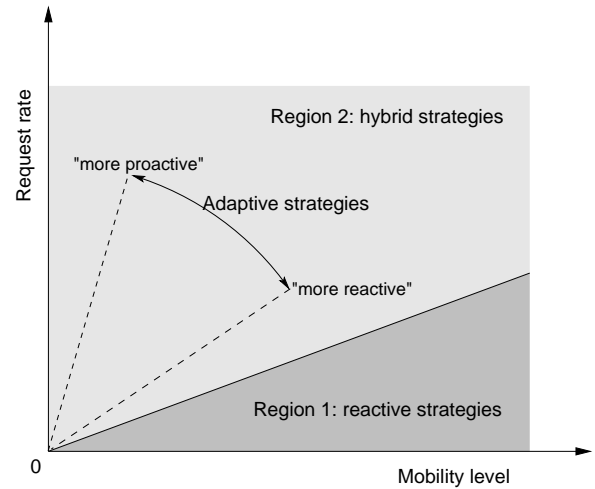


Figure 2: Regions for reactive strategies and proactive strategies. High mobility level favors reactive strategies and high request rate favors proactive strategies.

mal point could be purely reactive, or a balancing between proactive advertising and reactive discovery. To that end we need first define the *cost* function of the search strategies.

Consider a general strategy. The servers advertise (using flooding techniques) their availability information at a frequency $f \geq 0$ advertisement per request. The mobile hosts cache the discovered routes to the servers. On each request, if a mobile host finds the cached route is stale, it will initiate server discovery using controlled flooding techniques. We define the cost as the total network traffic. This cost includes two parts, the part due to proactive advertising and the part due to reactive discovery:

- **Cost due to proactive advertising** Let the cost of one advertisement (unrestricted flooding) be the unit cost. This unit cost corresponds to one broadcast per host in the network. Hence, for each request the average cost due to proactive advertising is f .
- **Cost due to reactive discovery** Let $p(f)$ denote the probability of having a stale cached route, and let $c_{controlled}$ denote the average cost of one controlled flooding. Hence, for each request the average cost due to reactive discovery is $p(f)c_{controlled}$. Here the constant $c_{controlled}$ depends on the controlled flooding algorithms. It is defined as the average number of broadcast per host in order to discover at least one server. Since there are many servers, a reasonable controlled flooding algorithm has $c_{controlled} < 1$.

To summarize, for each request the average total cost is

$$Cost = f + p(f)c_{controlled} \quad (1)$$

When we increase f , there are two effects: (1) the network traffic due to proactive advertising increases, and (2) the probability of reactive discovery decreases due to the increased freshness of cached routes. The probability of route staleness is intuitively shown in Figure 3. Later we use simulation to support this figure. Overall the total network traffic for locating servers is intuitively shown in Figure 4. There are two possibilities. If the request rate is very low, then an increased f causes more total traffic as the increased proactive traffic outweighs reduced reactive traffic. Therefore the optimal solution is purely reactive strategies. If the request rate is

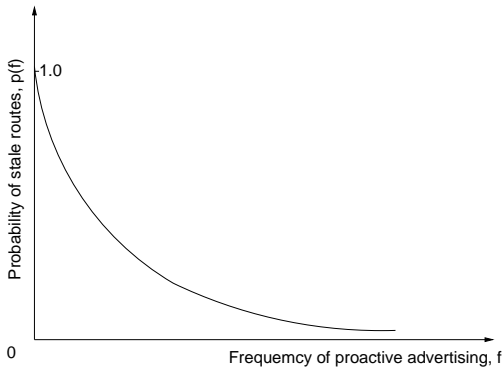


Figure 3: An intuitive example showing that the probability of route staleness is a monotone, decreasing function of the frequency of proactive advertising.

high, then proactive advertisements is likely to be effective. Therefore, the optimal solution is hybrid strategies.

Equation (1) suggests that we may reduce the cost in two ways. First we can find the balanced, optimal operating point between reactive and proactive strategies. An adaptive strategy will achieve this. Second we can develop more efficient controlled flooding algorithms to decrease the constant factor $c_{controlled}$. We will do both in the next two sections.

3. EXPONENTIAL BACKOFF FOR PROACTIVE ADVERTISING

The exponential backoff algorithm probes the optimal operating point between proactive and reactive strategies. The algorithm is depicted in Figure 5. The general process is described as follows.

The system divides time into slots such that there are N requests in each slot. Initially in the first slot from line (1) to (5), the frequency of proactive advertising is set to $c_{controlled}$ advertisement per request. In this slot, the servers count how many times the mobile hosts initiate reactive discovery. Let $N_{discovery}$ denote this number. Thus the total cost per request is $f + \frac{N_{discovery}}{N} c_{controlled}$, calculated according to Equation (1). After the first N requests, the system enters two end-less loops, labeled as loop-down and loop-up, respectively.

In loop-down from line (6) to (16), the system attempts to decrease the frequency f . Each time the frequency is decreased by multiplying it by a positive constant $\beta < 1$. The servers count how many times the hosts complete reactive discovery. The servers calculate the total cost again. If the cost is reduced, the system will stay in loop-down and attempts to further decrease the frequency. Otherwise, the system will enter loop-up. In this second loop from line (17) to (27), the system just does the opposite: it attempts to increase the frequency f until the cost is no longer reduced.

3.1 Properties of Exponential Backoff

CLAIM 1. *In the worst case, the average total cost is twice that of the purely reactive strategies.*

We set the initialization point $f = c_{controlled}$. In the worst case, proactive advertising leads to all stale routes. Thus on each request a mobile host must perform reactive discovery. The wasted proactive advertising requires cost equal to $c_{controlled}$ per request. Therefore, the average total cost is twice that of purely reactive strategies in the worst case.

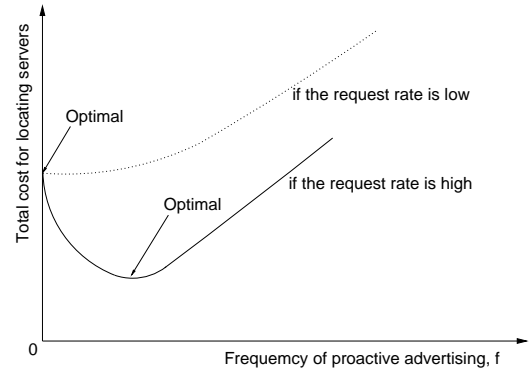


Figure 4: An intuitive example showing that to minimize total network traffic for locating servers, reactive strategies or hybrid strategies work at low/high request rate.

Exponential Backoff Algorithm:

- 1) $f \leftarrow c_{controlled}$
 - 2) for the next N requests
 - 3) broadcast an advertisement every $1/f$ requests
 - 4) $N_{discovery} \leftarrow$ the number of reactive discoveries
 - 5) $Cost_{old} \leftarrow f + N_{discovery}/N * c_{controlled}$
 - loop-down:
 - 6) $f \leftarrow f * \beta$
 - 7) for the next N requests
 - 8) broadcast an advertisement every $1/f$ requests
 - 9) $N_{discovery} \leftarrow$ the number of reactive discoveries
 - 10) $Cost_{new} \leftarrow f + N_{discovery}/N * c_{controlled}$
 - 11) IF $Cost_{new} < Cost_{old}$ THEN
 - 12) $Cost_{old} \leftarrow Cost_{new}$
 - 13) GOTO loop-down
 - 14) ELSE
 - 15) $Cost_{old} \leftarrow Cost_{new}$
 - 16) GOTO loop-up
 - loop-up:
 - 17) $f \leftarrow f/\beta$
 - 18) for the next N requests
 - 19) broadcast an advertisement every $1/f$ requests
 - 20) $N_{discovery} \leftarrow$ the number of reactive discoveries
 - 21) $Cost_{new} \leftarrow f + N_{discovery}/N * c_{controlled}$
 - 22) IF $Cost_{new} < Cost_{old}$ THEN
 - 23) $Cost_{old} \leftarrow Cost_{new}$
 - 24) GOTO loop-up
 - 25) ELSE
 - 26) $Cost_{old} \leftarrow Cost_{new}$
 - 27) GOTO loop-down
-

Figure 5: Pseudo code of the exponential backoff algorithm

Note we set the initialization point so because it is certain that the optimal frequency must be smaller than this initial point (at least at the point $f = 0$, the cost is only $c_{controlled}$ per request), and the algorithm will enter loop-down. Nevertheless, it is fine if initially we set f to a relatively small value. Assume this initial value is smaller than the optimal point, then the algorithm will find in loop-down that the cost is increased. Thus the algorithm will enter loop-up to increase f .

CLAIM 2. *In steady-state the system operates between f^*/β^2 and $f^*\beta^2$, where f^* is the optimal point.*

As shown in Figure 6, in steady-state the system operates at any of the three points, f_0 , f_1 , and f_2 . Each of them is between f^*/β^2

and $f^* \beta^2$, since the optimal point is either between f_0 and f_1 or between f_0 and f_2 . In addition, this result has two implications. First, theoretically the maximum range of oscillation is $f^*(1/\beta^2 - \beta^2)$. Second, this range of oscillation can be as small as possible because we can always adjust β such that the length approaches zero.

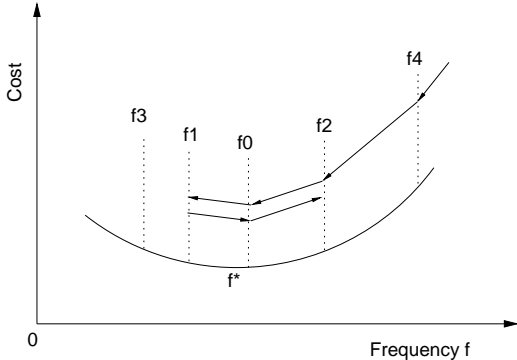


Figure 6: In steady-state the system operates at a point which is between f^*/β^2 and $f^* \beta^2$.

CLAIM 3. *When the mobility level changes or the request rate changes, the system can react to the changes and eventually move its operating point towards the optimal point.*

Assume the request rate increases drastically but the level of host mobility does not change. Theoretically, the new optimal point should be a smaller frequency. There are two possibilities. (1) The algorithm is in **loop-down** at the time. Notice that the length of each slot becomes shorter. If we keep the same frequency of proactive advertising, then the probability of having stale routes will decrease. Thus the number of reactive discoveries and the total cost decrease. Therefore the algorithm will stay in **loop-down** and move the operating point towards the new optimal point. (2) The algorithm is in **loop-up** at the time. Since the total cost decreases, the algorithm will stay in **loop-up** and thus move its operating point away from the new optimal point. After the next slot, however, the algorithm will move back towards the new optimal point.

Similar effect is achieved if the request rate does not change but the mobility level decreases, or if either the request rate decreases or the mobility level increases. We omit detailed discussions here.

CLAIM 4. *The parameter β determines a tradeoff between the stability of the system in steady-state and the convergence speed during transient periods.*

The primary benefit of choosing a large value for β is, the system can stay very close to the optimal value. On the other hand, a small value for β may cause oscillation. The primary benefit of choosing a small value is, the system reaches the near-optimal point as soon as possible. This is important in the beginning of the algorithm, or when the network conditions change.

Further optimizations are possible though. For example, the algorithm can set β to be a variable. When the system is in steady-state, the algorithm can increase the value of β such that it is closer to 1.0. Therefore, the system will operate in a range very close to the optimal point. On the other hand, when the network conditions change, the algorithm can decrease the value of β , *e.g.* such that it is close to 0.5, *i.e.* binary backoff. Therefore, the system will move towards the new optimal point faster.

4. CONTROLLED FLOODING FOR REACTIVE DISCOVERY

In this section we describe a novel flooding algorithm to minimize the cost of reactive discovery. Recall that in Equation (1) the constant $c_{controlled}$ represents the per-discovery cost using controlled flooding, *i.e.* the average number of broadcast per host in order to locate a server.

One popular algorithm is called *expanding ring*, as shown in Figure 7. Briefly, this algorithm performs multiple rounds of flooding with increased TTL values. However, this algorithm has potentially very high search cost. First there are overhead due to repeated broadcasts caused by inappropriate choices of TTL. For example in Figure 7, the hosts inside the inner rings may receive and broadcast query packets multiple times. Second the expanding ring algorithm may also cause severe overshooting. If the TTL is increased very fast, potentially packets are propagated to many more hosts than necessary. To reduce repeated broadcast, TTL should be increased fast, while to minimize overshooting, TTL should be increased slowly. Therefore, the high overhead of the expanding ring algorithm is doomed. Recent theoretical work [4] showed that any expanding ring algorithm requires average search cost multiple times that of the lower bound.

We use the following novel *spiral* flooding algorithm. The source host divides the entire 2-D space into four quadrants, with itself as the origin, as shown in Figure 8. In the first round of flooding, the source host propagates the packet toward the hosts in the network. The packet contains (1) a field to indicate this is the first round of flooding, (2) the geographical location of the the source, and (3) a TTL value to indicate how far it can travel. If a host in the first quadrant receives this packet, this host will check if the TTL value in the packet is larger than the length of the shortest route from the source to this host, and if yes, this host will re-broadcast the packet. If a host (which is on the boundary of two quadrants) in another quadrant receives the packet, it may keep the shortest route (if this is the shortest route) in its route cache, but the host will not re-broadcast the packet. If after the first round of flooding, the source host fails to locate a server, it starts the second round of flooding towards the hosts in the second quadrant, with a larger TTL value. Similarly only those hosts in the second quadrant re-broadcast the packet. If after four rounds of flooding, the source host has yet to locate a server, it will again flood a packet to the first quadrant.

The algorithm implicitly assumes that each host can know its own approximate geographical location via Global Position System (GPS) or by other means. We do not assume the hosts can know the positions of other hosts, since in a dynamic network it is expensive to provide location service such as the one in [14].

In the algorithm the hosts maintain the discovered shortest routes in their caches. This is meaningful if the shortest route from the source to a server will go through hosts in different quadrants. We should note that these shortest routes are maintained temporarily for flooding only, and they can be discarded soon. The cost should not be high.

4.1 Properties of Spiral Controlled Flooding

This simple algorithm is effective in reducing the overhead of server discovery. The algorithm has the following properties:

CLAIM 5. *The proposed algorithm reduces the number of repeated broadcasts by the hosts.*

This is because, the algorithm attempts to discover a server in a new space after every failed attempt. If the algorithm does not locate a server after four rounds of flooding, it will propagate the packet to the first quadrant again. However, by that time, the TTL has been

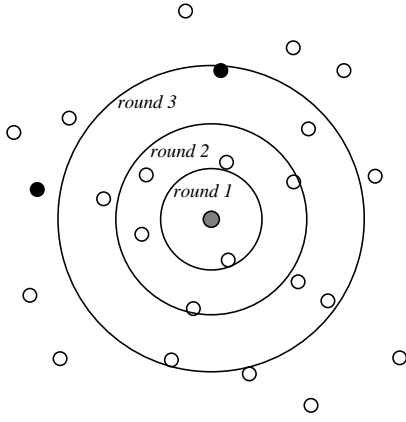


Figure 7: Expanding ring algorithm requires high search overhead due to repeated broadcasts and big overshooting.

increased four times already. Therefore, the probability of repeated broadcasts is not high.

CLAIM 6. The proposed algorithm minimizes the overshooting of the search space.

This is because, the algorithm can increase the TTL more slowly than the expanding ring algorithm does. Thus less likely the packet will be broadcast by too many hosts. In addition, although each time the TTL is increased slowly, when the algorithm returns to the first quadrant, the TTL value is much larger than it was the last time. Hence we can have both smaller overshooting and fewer repeated broadcasts.

5. SIMULATION STUDY

This section evaluates the performance of our new adaptive strategies for locating the servers in MANETs. First, we investigate the probability of route staleness as a function of the rate of proactive advertising via simulation. Second, we compare the performance of various strategies using the number of transmitted packets as the performance metric. These compared strategies may or may not use proactive advertising. They may use the expanding ring flooding or our spiral flooding for reactive server discovery.

5.1 Simulation Setup

For our experiments, an event-driven simulator is developed to measure the performance of the strategies under a variety of conditions. The algorithm parameters and network configuration parameters can be adjusted. These parameters include the number of mobile hosts and the number of Internet-based servers, the pattern and speed of the movement, and the radio range of the hosts and servers, etc.

The simulated MANET consists of 800 mobile hosts and 20 servers in a 1000m x 1000m square area. Each host or server has a radio range of 50 meters. We assume this range does not change over time, and it is not affected by the interference during communication. We have found on average each node (host or server) can communicate directly with about 6 neighbors at any point of time. The hosts and servers move according to the random waypoint model [3]. In our simulation we consider both a low mobility level and a high mobility level. At the low mobility level the mobile hosts and servers move with a maximum velocity of 2 m/s, while at the high mobility level they move with a maximum velocity of

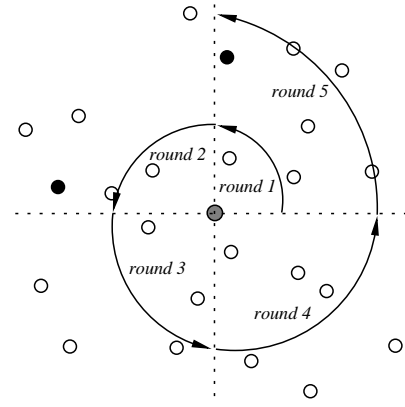


Figure 8: The proposed spiral flooding algorithm reduces repeated broadcasts and minimize overshooting.

10 m/s. At both levels, a mobile host or server moves for 10 seconds and then pauses for 10 seconds. This move/pause process is repeated by turns.

Initially the frequency of server proactive advertising is set to 0.05, *i.e.* after every 20 requests, all the Internet-based servers will broadcast their availability information once. This frequency value is updated after every 200 requests using the exponential backoff algorithm with the parameter $\beta = 0.8$. That is, first, each time slot of our exponential backoff algorithm contains 200 requests. Second, theoretically the maximum range of oscillation as we mentioned in Section 3 is $0.9225f^*$.

The request rate is 10 per second. In every 0.1 second, one randomly chosen host requests to locate a server. This host first checks if its own cache contains a fresh route to a server. If the path is stale or no path to a server can be found, it will begin a reactive discovery using the controlled flooding algorithms in Section 4. Each broadcast by a host takes 0.1 second plus a random delay below 0.001 second. During the controlled flooding, the timeout parameter is set to 0.2 second (which is slightly larger than the sending time) times the TTL value. For instance, if the TTL is 4 then the timeout we set is 0.8 second. In both the expanding ring and in the spiral flooding, the TTL value is increased by one after each round of flooding. The process will stop after the TTL value reaches the threshold 8. We assume the source host is disconnected from the network if it cannot reach any server in this many hops.

In order to obtain accurate results, the simulator ran for more than 1000 seconds. We emphasize that throughout the simulations reported in this paper, we use the same parameter values. This, from another point of view, suggests that our adaptive strategies are robust in a variety of environments.

5.2 Simulation Results

First we evaluate the relationship between the probability of route staleness and the frequency of proactive advertising when the mobility level is high or low. Figure 9 plots the probability of route staleness $p(f)$ as a function of the frequency of proactive advertising f . In this experiment we run the simulator for 1000 seconds and compute the average probability of route staleness. Intuitively, the probability of staleness should decrease when the frequency increases, as illustrated in Figure 3. Our experiment shows a similar result in Figure 9. From the figure we observe that with the low mobility, a frequency of 0.03 advertisement per request is enough to provide 80% freshness of the routes in the caches, while with the

high mobility, the frequency needs to be larger than 0.18. Besides, we can see that the probability of staleness decreases fast at first but slowly when the frequency of advertising becomes large.

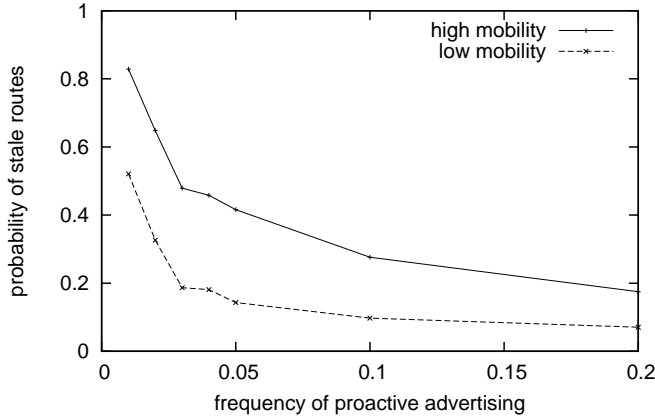


Figure 9: Probability of route staleness, $p(f)$, as a function of the frequency of proactive advertising when the mobility level is high or low.

In our next experiment we compare the performance of different algorithms when the mobility level is low. The average communication cost per request is used as the metric for performance evaluation, since the algorithms aim at reducing the network traffic. Figure 10 compares the performance of our adaptive strategies and purely reactive strategies (without proactive advertising) in terms of the number of broadcast packets. This quantity is proportional to the cost definition in Equation (1). For each strategy, either the expanding ring algorithm or our spiral flooding is used. In this figure the x-axis is the sequence number of the time slots. The simulation lasts for 1000 seconds and there are 10000 requests. Therefore, we have 50 time slots, each with 200 requests. The y-axis is the average cost per request for the 200 requests in each time slot.

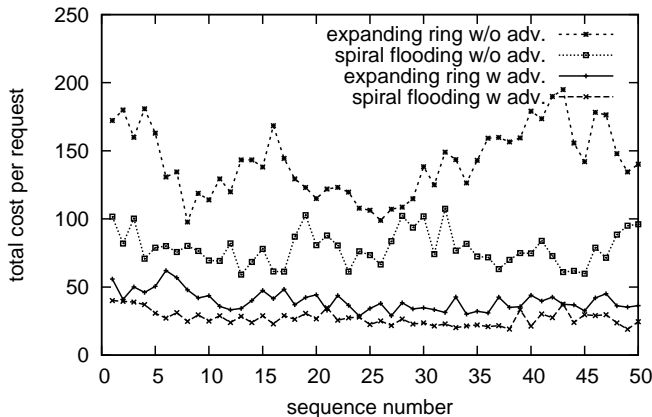


Figure 10: The total cost (number of broadcasts by all hosts) per request when the frequency of proactive advertising changes, and the level of mobility is low. Both expanding ring and spiral flooding are compared.

This figure leads to two important conclusions. First, proactive advertising significantly reduces the number of broadcast packets per request. Without proactive advertising, we estimate the average number of broadcast packets per request is over 140 if the expand-

ing ring algorithm is used in flooding, and close to 80 if our spiral flooding algorithm is used. With proactive advertising, these two quantities become about 40 and 27, respectively. Notice that with low mobility, proactive advertising is more efficient in providing fresh server information at low cost. Our exponential backoff algorithm is able to find the optimal operating point which has very low network traffic. Second, our spiral flooding algorithm gains noticeable performance gain over the previous expanding ring algorithm. In both cases of with and without proactive advertising, the use of expanding ring causes about 50% more broadcast packets.

Figure 11 shows the performance when the mobility level is high. We do not plot the performance when proactive advertising is disabled, as it is the same as in Figure 10. We observe that at high mobility level, the algorithms' performance degrades as we expected. On average the number of broadcast packets is about 70% larger than that in the low mobility case. Once again, we observe that the spiral flooding is better than the expanding ring. However, the variation of the results is a little more than that in the low mobility configuration. We guess this is because high mobility causes more oscillations. It is more likely that the optimal point f^* will oscillate over the time.

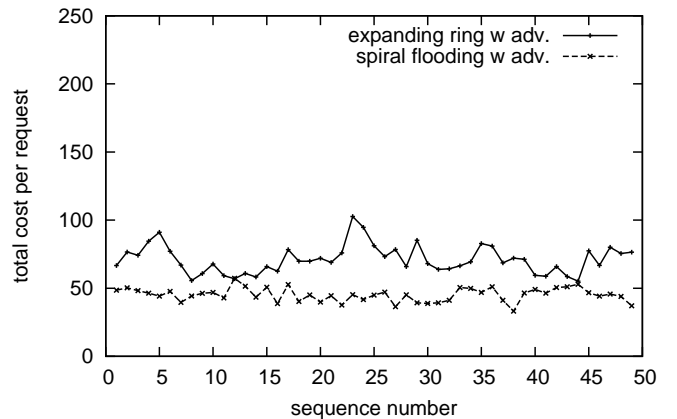


Figure 11: The total cost (number of broadcasts by all hosts) per request when the frequency of proactive advertising changes, and the level of mobility is high. Both expanding ring and spiral flooding are compared.

6. RELATED WORK

The tradeoffs between reactive and proactive strategies have been investigated in the context of (1) ad hoc routing, (2) caching in ad hoc networks, and (3) search in sensor networks.

For ad hoc routing there are three categories of protocols. Reactive protocols, such as AODV [19] and DSR [12], delay route discovery until a route is required. Proactive protocols, represented by DSDV [18], exchange routing information periodically between hosts. Hybrid protocols, such as ZRP [9] and SHARP [20], combine proactive and reactive routing strategies. The SHARP protocol is the closest to ours in that it finds the balanced point between proactive and reactive routing. There are three main differences between our work and SHARP. First, SHARP focuses on exploiting spatial locality to reduce routing protocol overhead. It is not designed to solve the problem of locating the servers in MANETs. Second, SHARP optimizes user-defined performance metrics, such as loss rate, routing overhead, or delay jitter, whereas our goal is to minimize network cost. Third, we consider how to improve flooding algorithms to reduce network traffic.

Caching in ad hoc networks is considered an approach to improving user-perceived quality and reducing network cost [10, 17, 22, 13, 8]. There is a tradeoff between push-based (proactive) and pull-based (reactive) strategies. For instance, Nuggehalli *et al.* [17] studied how far to push content towards the areas of high demands. The related cache invalidation problem has been studied in [11, 15]. In particular, Hara [11] suggested to periodically update data items. On the other hand, Lim *et al.* [15] examined several push-based and pull-based cache invalidation schemes, and found that a pull-based strategy provides low latency and low communication overhead.

Search in sensor networks must be efficient since energy consumption is a bigger concern it is in ad hoc networks. Neither query flooding, where a host (*i.e.* sensor node) propagates queries to locate an event in an on-demand fashion, nor event flooding, where the source of an event propagates notifications towards all hosts, is energy-efficient. Recent studies [2, 21] show that random walk based search methods are good choices. Both the source of an event and an interested host can inject packets into the network, hoping that they detect each other eventually. While random walk based methods are appropriate for sensor networks, three disadvantages prevent them from being directly adopted in our application. First, these methods do not guarantee that the hosts can locate the servers. Second, they do not provide an efficient route (not to mention the shortest route) to a server. Third, these methods often require very long search latency.

Finally, there are many studies on flooding algorithms, which are popularly used in ad hoc networks and unstructured peer-to-peer networks. TTL-based controlled flooding techniques have gained much attention [16, 6, 4, 5]. In particular, recent theoretical work [1, 4, 5] has provided much important results on the performance of such flooding algorithms. Various flooding strategies (with different approaches to setting TTL) have been studied, including those with worst case cost ratio of 4 and with average case cost ratio below 3. Subsequent studies [4, 5] derived more sophisticated randomized algorithms to achieve worse case and average case cost ratio below 3, to the limit. In this paper we propose a new flooding algorithm whose cost ratio is much lower. Thus our algorithm can be used to minimize the cost due to reactive server discovery.

7. CONCLUSIONS

In this paper we have studied adaptive, hybrid strategies for locating the servers in MANETs. Our exponential backoff algorithm can rapidly probe the near-optimal operating point which balances proactive advertising by the servers and reactive discovery by the mobile hosts. The algorithm can adapt to the dynamic network conditions such as the host mobility level and the offered load. Furthermore, we have proposed a novel controlled flooding algorithm to effectively bound the total network traffic due to reactive discovery. The algorithm outperforms the previous expanding ring algorithms. Our event-driven simulation using random waypoint model reveals that, the combination of these two algorithms is close to the optimal in minimizing network traffic. At a low or high mobility level and a moderate request rate, our proposed algorithms reduce network traffic by several times.

We look forward to working in two directions. First, we will theoretically study the optimality of adaptive strategies for locating servers, and will consider the mobility level and the request rate in our analysis. Second, we are considering to conduct comprehensive simulation study. We hope to consider other mobility models, *e.g.* group mobility models, and to consider the related issues at different layers, for example congestion and transport protocols, and contention at the link layer.

Acknowledgments

The authors would like to thank Ray-Yaung Chang who is involved in developing the simulator, and Limin Wang for discussions on flooding search techniques.

8. REFERENCES

- [1] Y. Baryshnikov, E. Coffman, P. Jelenkovic, P. Momcilovic, and D. Rubenstein. Flood search under the california split rule. *Operations Research Letters*, 32(3), 2004.
- [2] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of IEEE ICDCS*, July 2002.
- [3] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Comm. and Mobile Computing (WCMC) Special Issue on Mobile Ad Hoc Networking*, 2(5):483–502, 2002.
- [4] N. Chang and M. Liu. Revisiting the TTL-based controlled flooding search: Optimality and randomization. In *Proceedings of ACM MobiCom*, September 2004.
- [5] N. Chang and M. Liu. Controlled flooding search in a large network. In *Proceedings of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, March 2005.
- [6] Z. Cheng and W. B. Heinzelman. Flooding strategy for target discovery in wireless networks. In *Proceedings of ACM International Workshop on Modeling, analysis, and simulation of wireless and mobile systems (MSWiM)*, September 2003.
- [7] M. S. Corson, J. P. Macker, and G. H. Cirincione. Internet-based mobile ad hoc networking. *IEEE Internet Computing*, pages 63–70, August 1999.
- [8] R. Friedman, M. Gradinariu, and G. Simon. Locating cache proxies in manets. In *Proceedings of ACM MobiHoc*, May 2004.
- [9] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, 1997.
- [10] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proceedings of IEEE INFOCOM*, April 2001.
- [11] T. Hara. Replica allocation methods in ad hoc networks with data update. *ACM-Kluwer Journal on Mobile Networks and Applications (MONET)*, 8(4):343–354, 2003.
- [12] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.
- [13] B.-J. Ko and D. Rubenstein. A distributed, self-stabilizing protocol for placement of replicated resources in emerging networks. In *Proceedings of IEEE ICNP*, November 2003.
- [14] J. Li, J. Jonnotti, D. D. Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of ACM MobiCom*, August 2000.
- [15] S. Lim, W.-C. Lee, G. Cao, and C. R. Das. Performance comparison of cache invalidation strategies for Internet-based mobile ad hoc networks. In *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, October 2004.
- [16] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of International Conference on Supercomputing*, November 2002.
- [17] P. Nuggehalli, V. Srinivasan, and C.-F. Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *Proceedings of ACM MobiHoc*, June 2003.
- [18] C. Perkins. Highly dynamic destination-sequenced distance-vector routing (dsv) for mobile computers. In *Proceedings of ACM SIGCOMM*, August 1994.
- [19] C. Perkins. Ad hoc on-demand distance vector (aodv) routing. IETF MANET, Internet Draft, 1997.
- [20] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer. SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks. In *Proceedings of ACM MobiHoc*, June 2003.
- [21] S. Shakkottai. Asymptotics of query strategies over a sensor network. In *Proceedings of IEEE INFOCOM*, March 2004.
- [22] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. In *Proceedings of IEEE INFOCOM*, March 2004.