Strategies for Service Discovery over Ad Hoc Networks

Michel Barbeau, Evangelos Kranakis and Honghui Luo *

Abstract

Service discovery is an important and necessary component of ad hoc networks. To fit within the context of such networks, a post-query model with several service discovery strategies (named post-query strategies) are proposed, with a focus on locating services. Strategies consists of a sequence of post-query protocols executed in rounds and based on the location method employed they include the conservative, greedy, incremental, (l, l'), uniform memoryless, with demand distribution, and with memory post-query. We analyze some of these strategies and present a performance evaluation in combination with the DSR and DSDV protocols.

Keywords: ad hoc network, service discovery, resource discovery, service location protocol, service discovery protocol, post-query strategy

1 Introduction

An ad hoc network consists of set of nodes with wireless network interfaces, which is self organizing (i.e., its topology is established and maintained automatically), short-lived and does not involve routing across the Internet (i.e., the availability of a network infrastructure is not required) [16]. Service discovery is defined as the problem of locating servers that fulfill requirements of clients [9]. A service discovery protocol is a set of message formats and rules that can be used by clients to find servers over a network. There is a number of service discovery protocols, for general networks, such as Service Location Protocol (SLP) [5] and Jini [13]. Besides, there are ad hoc network specific discovery protocols such as Service Discovery Protocol (SDP) [1], DEAPSpace [7], Konark [6], and the work of Koodli and Perkins [11].

*The authors graciously acknowledge the financial support received from the following organizations: Natural Sciences and Engineering Research Council of Canada (NSERC) and Mathematics of Information Technology and Complex Systems (MITACS). Address of authors is: School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Canada K1S 5B6 Email: {barbeau,kranakis}@scs.carleton.ca

SLP is designed for TCP/IP networks. Key abstractions are the user agent, which locates services on behalf of a client, service agent, which provides information about a service, and directory agent, which caches service information received from service agents and replies to service requests from user agents. SLP can operate in a distributed manner (without directory agents) or in a centralized manner (with one or several directory agents). The Jini architecture consists of the service provider, which is the front-end of a service available on a node, client, which is a user of services, and lookup service, which is a central node where service providers register their services (similar to the directory agent of SLP). Bluetooth, a short range ad hoc network technology, employs SDP for locating services provided by or available on other devices. The SDP architecture consists of the server and client. To find a service, an client unicasts a request to servers one by one. Servers either send back responses or error responses. These service discovery protocols use a request-reply communication model. They can, however, be used in several different ways that we term *strategies*. Strategies aim at maximizing the probability of success in locating a service and minimizing the waiting time and total number of postings and queries.

In the context of service discovery in ad hoc networks, this paper addresses the following important questions. How can these service discovery protocols be best used on ad hoc networks? How can different service discovery strategies be characterized? In particular, how is it possible to obtain a characterization with respect to tradeoffs between expected cost and probability that a given client succeeds in finding a server?

In this paper, to fit within the context of ad hoc networks, a post-query model with several service discovery strategies named post-query strategies are proposed. This model focuses on locating services. It defines ways of using service discovery protocols on ad hoc networks. The post-query model is based on the distributed match-making paradigm described by Mullender and Vitányi [14] and Kranakis and

Vitányi [12]. Their work refers only to traditional networks. Our work goes beyond this and differs in two important respects, all related to modeling the message transmissions of service location protocols in ad-hoc networks. Firstly, the post-query strategy is taking place in a sequence of rounds. In each round posting is followed by querying.

This is intended to account for the fact that post-query protocols are executed at different frequencies, as well as the fact that repeated rounds may be necessary to account for collisions, faults, and service availability. Secondly, services are requested by clients following a probability distribution (called the client demand distribution) which indicates the probability that client c requests a service of a certain type (and posted by servers).

In the post-query model, each server posts its service(s) to other nodes according to the Posting protocol, and each client queries its desired services according to the Querying protocol. A pair of Posting and Querying protocols forms a post-query protocol. In order to adapt to topological changes over time in an ad hoc network, several post-query strategies are proposed. A post-query strategy is a sequence of post-query protocols executed in rounds. In the context of ad hoc networks, the Posting and Querying protocols are independent. We propose seven post-query strategies, namely, the conservative, greedy, incremental, (l, l'), uniform memoryless, with demand distribution, and with memory post-query strategies. we introduce a novel framework which core is a probabilistic model in which service discovery strategies can be modeled and evaluated in a uniform manner. We show how strategies for discovering services in ad hoc network can be represented, analyzed and evaluated through simulations, combined with the DSR protocol [8] and DSDV protocol [15]. Our methodology for evaluating the performance and efficiency of such service discovery protocols can be useful in developing efficient search strategies for ad-hoc networks.

Service discovery in ad hoc networks is reviewed in Section 2. Our strategy representation model is introduced in Section 3. Strategies represented in this mode are discussed in Section 4. Section 5 presents analytical results of post-query strategies. Section 6 presents simulation results of post-query strategies. We conclude with Section 7.

2 Service Discovery in Ad Hoc Networks

Existing standards for dynamic configuration of nodes (see for example Guttman *et al* [5]) are inadequate for ad hoc networks (see Perkins [17]). A natural question posed by Perkins [16] is: *where do services reside?* The difficulty of answering this problem is compounded by the fact that in addition to the topological changes affecting ad hoc networks over time, it may not be possible either for service discovery to be inserted into a network layer (due to vertical design obstacles) or for a node to specify the exact service it wants in a network layer (see Perkins [17]).

Bluetooth is an ad hoc network technology for nodes that are very close to each other [1]. They are constructed from piconets in order to form larger networks called scatternets. A piconet is a set of two to seven interconnected Bluetooth devices while a scatternet is a set of interconnected piconets. Bluetooth defines the Service Discovery Application Profile [1] for finding services over such networks. To find a service, a client sends a *ServiceSearchRequest* message to every device on the network, one after the other using unicast. They each respond with the *ServiceSearchResponse* message, which can be either positive of negative.

IBM has developed DEAPspace that addresses the service discovery problem in single-hop ad hoc networks [7]. A push model is used (servers send unsolicited service advertisements). Each node maintains a cache to store all the services offered in the network. Nodes participate in the advertising of services through a broadcast mechanism. Periodic broadcasts are scheduled in a proactive way using an adaptive backoff mechanism. A performance evaluation of DEAPspace shows that this approach uses about the same bandwidth as other push model solutions while the time required for the discovery of available services is better, since more service information is being propagated when one node broadcasts its entire cache than when each node of the network only broadcasts the service information which it offers. However, maintaining such a tight convergence in a highly dynamic ad hoc network may not be worth the effort. Unnecessarily repeated broadcasts occur in DEAPspace, consuming network bandwidth.

Konark [6] is designed specifically for the discovery and delivery of services in multihop ad hoc networks. As for the service discovery aspect, Konark supports

the push model and pull model. It assumes multicast support from the underlying ad hoc routing protocol. Konark attempts to balance the convergence time and network bandwidth use. In contrast to DEAPspace, when a node receives a service message, it only multicasts the new information, if there are any. Hence, network traffic is reduced.

Perkins pointed out (see [17]) that existing standards for dynamic configuration of nodes (see for example Guttman *et al* [5]) are inadequate for ad hoc networks. A natural question posed by Perkins [16] is: *where do services reside?* The difficulty of answering this question is compounded by the fact that in addition to the topological changes affecting ad hoc networks over time, it may not be possible either for service discovery to be inserted into a network layer (due to vertical design obstacles) or for a node to specify the exact service it wants in a network layer. For the purposes of service discovery in ad hoc networks, Koodli and Perkins define extensions to ad hoc network routing protocols [11]. Their extensions together with a routing protocol can find services and routes to services at the same time. Service information and route information become simultaneously available. The extensions can be combined with both proactive and reactive routing protocols. With proactive protocols, service information is included in packets about the network topology that are exchanged between routers. With reactive protocols, service discovery uses the same messages as the one defined for route discovery, with extensions.

In light of the above service discovery protocols for ad hoc networks, we make the following assumption in the sequel of this paper: routes to services are discovered before or at the same time of services.

3 Strategy Model

A network is modeled as a graph $G = (N, L)$ where $N = \{1, 2, \ldots, n\}$ is a set of n nodes and $L \subseteq N \times N$ is a set of directed links. Whenever $(i, j) \in L$, then there is a communication path between the nodes i and j and i can talk to j. Given a set of nodes N, an ad hoc network is modeled as a discrete sequence of graphs $G_0, G_1, G_2, \ldots$. At time t, $G_t = (N_t, L_t)$, with $N_t \subseteq N$, modeling the set of active nodes, and $L_t \subseteq N_t \times N_t$, modeling the set of active links. Whenever $(i, j) \in L_t$, then there is a communication path between the nodes i and j and i can talk to j at time t. This model is similar to the one defined by Farago and Syrotiuk [4]. To model the uncertainty in ad hoc networks, we define the probability p, with $0 \leq p \leq 1$, that there is a communication path between two given nodes. Let $1, 2, \ldots, k$ be the k different types of services offered in the network and let $K = \{1, 2, \ldots, k\}$.

A strategy is abstracted as post-query protocols. They are time dependent protocols executed in rounds modeling request-reply interactions between client and servers on a network. The post-query model described by Mullender and Vitányi [14] and Kranakis and Vitányi [12] refers only to traditional networks. In this paper, we adapt and extend their post-query model to ad hoc networks.

At any given time a client wants to locate a service that has been posted by a server.[1] Servers post services and clients query nodes in order to locate the desired services. *Post-query protocols* are algorithms for posting and querying services. A client may request services of different types.

Definition 1 *A* post-query protocol *is a pair of functions* (P, Q). $P : N \to 2^{N \times K} : s \to P(s)$ *is the* posting protocol*, and* $Q : N \to 2^{N \times K} : c \to Q(c)$ *is the* querying protocol.[2]

Each server s posts (respectively, client c queries for) services in nodes of the set $P(s)$ (respectively, $Q(c)$) according to the following rules. Server s posts service i to node u if and only if $(u, i) \in P(s)$. Similarly, client c queries node u for service i if and only if $(u, i) \in Q(s)$. In the sequel, it will be useful to adopt the following notations. $N_P(s) = \{u \in N : (\exists i \in K)((u, i) \in P(s))\}$ be the set of nodes to which services are posted by server s. Let $K_P(u, s) = \{i \in K : (u, i) \in P(s)\}$ be the set of services posted to u by server s and

$$K_P(s) = \bigcup_{u \in N} K_P(u, s)$$

the set of all services posted by server s. We define in a similar manner the set $N_Q(c)$ of nodes queried by client c and the sets $K_Q(v, c)$ (respectively, $K_Q(c)$) of services requested by client c when querying node v (respectively, nodes of the network).

[1] Although in the application model we have in mind all nodes are treated as computationally identical, we will still find it convenient to use the client/server terminology in order to distinguish between nodes querying/posting for services, respectively.

[2] In this paper, we use 2^S to denote the set of all subsets of a set S.

Several post-query protocols fall under the category of our model. Here we illustrate with two examples of simple protocols. A server s (respectively, client c) may not post (respectively, query) at all, i.e., $P(s) = \emptyset$ (respectively, $Q(c) = \emptyset$). A server s (respectively, client c) may post (respectively, query) all services everywhere, i.e., $P(s) = N \times K$ (respectively, $Q(c) = N \times K$).

The cost of the posting and querying protocols is the number of postings and queries made by all nodes, i.e.,

$$C(P) := \sum_{s=1}^{n} |N_P(s)| \text{ and } C(Q) := \sum_{c=1}^{n} |N_Q(c)| \quad (1)$$

The cost of the post-query protocol (P, Q) is an averaged sum of the costs of the posting and querying protocols, i.e.,

$$C(P, Q) = \frac{1}{n} \sum_{s,c=1}^{n} (|N_P(s)| + |N_Q(c)|) \quad (2)$$

Definition 2 *A client succeeds in finding all services it wants if every service it wants has been posted by some server to a node that it queries.*

In view of the aforementioned notation, $K_Q(c)$ is the set of services requested by client c. Therefore, success is assured if for all services $i \in K_Q(c)$ there exists a server s and a node u such that $(u, i) \in P(s)$ (i.e., service i has been posted at node u) and $(u, i) \in P(s)$ (i.e., client c requests service i from node u). We extend this model to capture the unknown environment of an ad hoc network. We modify Definition 1 so that P, Q are random variables.

Definition 3 *A* post-query protocol *is a pair (P, Q) of functions. $P : N \to 2^{N \times K} : s \to P(s)$ is the* posting protocol *and $Q : N \to 2^{N \times K} : c \to Q(c)$ is the* querying protocol *such that $P(s)$ and $Q(c)$ are random variables.*

The *expected cost* of the posting and querying protocols is defined by

$$E[P] := \sum_{s=1}^{n} E[P(s)] \text{ and } E[Q] := \sum_{c=1}^{n} E[Q(c)] \quad (3)$$

respectively, where

$$E[P(s)] := \sum_{i=1}^{n} i \cdot \Pr[|N_P(s)| = i] \text{ and}$$

$$E[Q(c)] := \sum_{c=1}^{n} i \cdot \Pr[|N_Q(c)| = i]$$

The *expected cost* of the post-query protocol (P, Q) is an averaged sum of the expected costs of the posting and querying protocols, i.e.,

$$\begin{aligned} E(P, Q) &:= E(P) + E(Q) \\ &= \frac{1}{n} \sum_{s,c=1}^{n} (E[P(s)] + E[Q(c)]) \end{aligned} \quad (4)$$

As given in Definition 3, we are interested in maximizing the probability that a given client $c \in N$ *succeeds* in finding a service. This can be expressed compactly as the probability

$$\Pr[\forall s(N_Q(c) \cap N_P(s) \neq \emptyset)] \quad (5)$$

A lower bound on the expected cost $E(P, Q)$ can be derived easily as in the main result of Kranakis and Vitányi [12]. Namely, if we define the random variables U_i as *the number of occurrences of the node i in an intersection $N_P(s) \cap N_Q(c)$* then

$$E(P, Q) \geq \frac{2}{n} \sum_{i=1}^{n} E[\sqrt{U_i}] \quad (6)$$

Protocols matching this lower bound exist, e.g. based on a projective model (for more details see References [12], [14]). This is, however, not very useful in the unknown environment of ad hoc networks.

Remark 1 *In the context of ad hoc networks it is reasonable to assume that the post and query protocols $P(s), Q(c),\ s, c = 1, 2, \ldots, n$, are independent and identically distributed random variables. A similar remark applies to the case of post-query strategies which are post-query protocols executed in rounds, although in the latter case a post-query protocol at a given round may depend on the protocol of the previous round.*

In general, in the model with k types of services we are interested in maximizing the probability that a given client succeeds in obtaining all services, namely

$$\begin{aligned} p_{(P,Q)}(c) := \Pr[&\forall i \in K(\exists s \in N \\ &(i \text{ available in } N_Q(c) \cap N_P(s)))] \end{aligned} \quad (7)$$

The dynamic changes affecting an ad hoc network indicate that a post-query protocol is not by itself sufficient to locate a service efficiently. For example, during the execution of a posting (respectively, querying)

protocol nodes may not be available either because they are operating in a non-active mode or in an active mode but at a different frequency. To overcome this problem we consider post-query strategies. *Post-query strategies* are post-query protocols that are executed in a sequence of rounds, each round consisting of a post-query protocol.

Definition 4 *A* post-query strategy *is a sequence* $(P_1, Q_1), \ldots, (P_r, Q_r), \ldots, (P_R, Q_R)$ *of post-query protocols executed in* R *rounds.*

At each round r, the nodes of the network first post services they have available to other nodes of the network according to the *posting protocol* P_r and then query other nodes of the system according to a *querying protocol* Q_r. Thus, post-query strategies comprise a sequence of post-query protocols that adapt to changes in the network. Execution of the strategy is in R rounds, so that during each round $r = 1 \ldots R$ a post-query protocol is executed. Rounds are necessary in order to adapt to topological changes over time in an ad hoc network.

For the sake of simplicity, we use the notation (P, Q) to denote a post-query strategy consisting of a sequence $(P_1, Q_1), (P_2, Q_2), \ldots, (P_R, Q_R)$ of post-query protocols. The quantity R is a parameter indicating an upper bound on the number of rounds within which clients and servers need to terminate execution of their respective strategy. We can extend the definition in Equation 4 to define the expected cost of a post-query strategy when executed in R rounds as the sum over all the rounds of the expected costs of its constituent post-query protocols, namely

$$E(P, Q) := \sum_{r=1}^{R} .E(P_r, Q_r) \tag{8}$$

In general, a given post-query strategy (P, Q) is executed by node s (s can be either a client or a server) in at most R rounds or until a desired service is located. The specific algorithm in detail is as follows.

Algorithm 1 (Post-Query Strategy (P,Q) (s))

1. **for** $r := 1$ **to** R
   or until service i is located
   (which ever comes first) **do**
2. *for all nodes u,*
3. *s posts service i at node u*
   if and only if $(u, i) \in P_r(s)$
4. *for all nodes u,*
   s queries for service j at node u
   if and only if $(u, j) \in Q_r(s)$

In round r, node s first posts services according to the posting set $P_r(s)$ and subsequently queries nodes for services according to the query set $Q_r(s)$. Each posting is followed by querying. The algorithm terminates either when it succeeds in finding the desired service or when $r := R$, which ever comes first.

In the model of Definition 3, we are interested in maximizing the probability that in the post-query protocol (P, Q) a given client $c \in N$ *succeeds* in finding a service (see Formula 5). In the case of a post-query strategy (P, Q), we are interested in minimizing the waiting time, i.e., the time until client c succeeds in finding a service according to the post-query startegy (P, Q). This is given by the formula

$$W_{(P,Q)}(c) := \sum_{r=1}^{\infty} r \cdot p'_{(P_r, Q_r)}(c) \tag{9}$$

where $p'_{(P_r,Q_r)}(c)$, is the probability that client c succeeds in finding a service at exactly the rth round. The *maximum waiting time* is the maximum taken over all clients in order to acquire a service and is given by the formula

$$W_{(P,Q)} := \max_{c \in N} W_{(P,Q)}(c) \tag{10}$$

4 Strategies

Post-query strategies aim to find a balance between usage of scarce ad hoc network resources, success, and waiting time. In the sequel, we propose several post-query strategies.

4.1 Conservative

The conservative post-query strategy requires that in each round all servers (clients) post to (query from) all their one-hop neighbors using a one-hop broadcast communication mechanism. It is a non-adaptive, round invariant strategy.

4.1.1 Greedy, Post-Greedy, and Query-Greedy

These post-query strategies follow the post-to-all, query-all rule. In a greedy post-query strategy all nodes post to all nodes, and all nodes query all nodes of the network using the network broadcast mechanism, formally, $N_P(s) = N_Q(c) = N$, for all $s, c \in N$. This post-query strategy is non-adaptive and does not change with each round. It is possible to consider two alternatives to the greedy post-query strategy: *post-greedy* and *query-greedy*. In the former case the servers post to all nodes and the client queries only one node, while in the latter the servers post to only one node while the clients query all the nodes.

4.2 Incremental

The incremental post-query strategy starts by posting and querying a small number of nodes in the first round and gradually increases the number of nodes posted to and queried from. The strategy aims at conserving valuable resources in an ad hoc network, e.g. power consumption. The incremental post-query strategy (P, Q) is a sequence $(P_r, Q_r), r = 1, 2, \ldots, R$ of post-query protocols such that (P_r, Q_r) is defined to be the (r, r)-post-query strategy, for all $r \leq R$. Two variants of this strategy are the *post-incremental* and *query-incremental* strategies. In the former, only the posting set is incremented, i.e., (P_r, Q_r) is defined to be the $(r, 1)$-post-query strategy, for all $r \leq R$, while in the latter case, only the querying set is incremented, i.e., (P_r, Q_r) is defined to be the $(1, r)$-post-query strategy, for all $r \leq R$.

In all the incremental strategies above, the probability of success in the r-th round is calculated as in the memoryless post-to-all, query-to-all strategy, see Subsection 4.4. However, the waiting times $W_{(P,Q)}(c)$ and $W_{(P,Q)}$ are calculated by using the formulas in Identities 9 and 10.

4.3 (l, l')-strategy

The (l, l') post-query protocol works by following the post-to-l and query-l' rule, where $l, l' \leq n$ are positive integers. In detail, this means that each server posts to a random set of l nodes all the services it has to offer while each client queries a random set of l' nodes. The (l, l') post-query strategy consists of rounds of uniform and memoryless repetition of the (l, l') post-query protocol. Analysis of this strategy is given in the Subsection 5. The expected cost of the post-query protocol is $(l + l')n$ since each server posts to exactly l nodes and each client queries exactly l' nodes. In such a protocol, it is easy to calculate the waiting time from the probability of success because it obeys the geometric distribution. Hence, the waiting time until client c succeeds in finding all services in this post-query protocol is equal to $1/p_{(P,Q)}(c)$.

4.4 Uniform memoryless

In the uniform memoryless post-query strategy, the same post-query protocol is executed in each round and the posting and querying protocols are identical for all the nodes. In such protocols, it is easy to calculate the waiting time from the probability of success because it obeys the geometric distribution. Hence, the waiting time until client c succeeds in finding all services in this post-query protocol is equal to $1/p_{(P,Q)}(c)$.

The first class of strategies we consider post to a random set of nodes and query a random set of nodes. The size of the set of nodes used affects the waiting time but also the network overload and number of collisions.

4.5 With demand distribution

We now extend our model in order to capture supply/demand conditions in an ad-hoc network. We augment the model to account for access costs of the services. Each service $i \in K$ is associated with a weight w_i indicating *some kind* of access cost, like, monetary cost in relation to the size of the program in bits. To each client, we associate a random variable Y_c such that $Y_c = i$ is the event that client c requests service of type $i \in K$. Let $q_c(i) := \Pr[Y_c = i]$ be the probability that client c requests service i. Probability distributions q_c may or may not depend on the client executing the protocol.

We can extend and interpret all previous definitions in this more general setting in order to quantify the access cost, demand probability Distribution, and waiting time of acquiring item of type i by the client. Thus, it is easy to define weighted versions of the formulas given by Identities 2, 3, and 10. For example, as in Identitiy 9 we can define the waiting time $W_{(P,Q)}(c, i)$ for client c to locate item i. Given the demand probability distribution $q_c(i)$, we can define the *demand cost* of client c in order to locate all services

when following the strategy (P, Q) by the formula

$$D_{(P,Q)}(c) = \sum_{i=1}^{k} W_{(P,Q)}(c, i) \cdot w_i \cdot q_c(i) \qquad (11)$$

where w_i is the access cost of service i. The demand cost of the strategy is the maximum of the demand cost of the strategy (P, Q) over all clients c, namely

$$D_{(P,Q)} := \max_{c \in N} D_{(P,Q)}(c) \qquad (12)$$

4.5.1 q-Demand balancing strategy

It is clear from Formula 11 that in order to minimize the demand cost, the posting strategy of the servers must allocate the services in such a way that the most popular services are located faster. Assume that all clients c use the same demand distribution $q(i) := q_c(i)$, and this is known to the servers. The (l, l') *q-demand balancing strategy* is to post services in proportion to their popularity to l nodes and query for services according to their popularity from l' nodes.

1. **Posting:** each server selects l nodes at random and for each of these nodes the server posts service i at this node with probability $q(i)$.
2. **Querying:** each client selects l' nodes at random and queries each of these nodes for service i with probability $q(i)$.

In a sense, the (l, l') *q-demand balancing strategy* is a refinement of the (l, l') post-query strategies because it takes into account the demand distribution of the services to post the most popular services, while in the latter case all. services are posted and queried at once. We note that several variants of this strategy are possible. E.g., by combining the number of nodes being posted/queried with the number of services. We present an outline of the analysis of this protocol in the appendix.

An important issue that must be raised is whether or not the servers have a priori knowledge of the distributions used by the clients. If not, there are several types of distributions that could be considered for the q-demand balancing post-query strategies. A Zipfian distribution is the most realistic. It is related to the "80-20" rule of thumb stating that 80 percent of transactions are dealing with the most active 20 percent of a list of items [10]. The Zipf distribution [18] satisfies $q(i) = \frac{1}{iH_k}$, where $H_k = \sum_{i=1}^{k} 1/i$ is the harmonic number. In a uniform distribution all services are requested with equal probability, i.e., $q(1) = q(2) = \cdots = q(k) = 1/k$. In the geometric ditribution the probability the i-th service will be requested satisfies $q(i) = a^{-i}$, for $i = 1, \ldots, k-1$, and $q(k) = 2 - (1 - a^{-k})/(1 - a^{-1})$, where $a > 1$. If the ditribution is arbitrary then it will satisfy $q(1) + q(2) + \cdots + q(k) = 1$.

4.6 With memory

In a with memory post-query strategy, the clients build a cache of nodes they visited and services they found in the previous round. Each new round involves searching only new (previously unused) nodes for undiscovered services.

5 Analytical results

In this section, we present a detailed analysis of the (l, l') post-query strategy.

Consider the events $A_{i,s,c}$: i is not available in $N_Q(c) \cap N_P(s)$, $B_{s,c}$: $N_Q(c) \cap N_P(s) \neq \emptyset$, and $B^m_{s,c}$: $|N_Q(c) \cap N_P(s)| = m$. Using the notation above, observe that for a given c,

$$\begin{aligned}
&\Pr[i \text{ is not available in } N_Q(c) \cap N_P(s)] \\
&= \Pr[N_Q(c) \cap N_P(s) = \emptyset] + \\
&\quad \Pr[(i \text{ is not available in } N_Q(c) \cap N_P(s)) \textit{ and } \\
&\quad (N_Q(c) \cap N_P(s) \neq \emptyset)] \\
&= \Pr[N_Q(c) \cap N_P(s) = \emptyset] + \\
&\quad \sum_{m=1}^{\min\{l,l'\}} \Pr[(i \text{ is not available in } \\
&\quad N_Q(c) \cap N_P(s)) \ \&\ |N_Q(c) \cap N_P(s)| = m] \\
&= \Pr[N_Q(c) \cap N_P(s) = \emptyset] + \\
&\quad \sum_{m=1}^{\min\{l,l'\}} \Pr[A_{i,s,c} | B^m_{s,c}] \Pr[B^m_{s,c}] \\
&= \left\{ \begin{array}{cc} 0 & \text{if } l + l' > n \\ \frac{\binom{n-l'}{l}}{\binom{n}{l}} & \text{if } l + l' \leq n \end{array} \right\} + \\
&\quad \sum_{m=1}^{\min\{l,l'\}} (\Pr[A_{i,s,c}])^m \, \frac{\binom{l'}{m}\binom{n-l'}{\min\{l,l'\}-m}}{\binom{n}{l}}
\end{aligned}$$

Recall we are assuming that the post-query functions are independent random variables, and the events "i is available in $N_Q(c) \cap N_P(s)$" are independent for $i \in K, s, c \in N$. If we substitute the last formula in the equation below, it follows that the probability

that a client c succeeds in the post-query protocol can be computed as follows

$$\begin{aligned} p_{(l,l')}(c) &= \Pr[\forall i \in K \exists s \in N \\ &\quad (i \text{ is available in } N_Q(c) \cap N_P(s))] \\ &= (\Pr[\exists s \in N \\ &\quad (i \text{ is available in } N_Q(c) \cap N_P(s)])^k \\ &= (1 - \Pr[\forall s \in N \\ &\quad (i \text{ is not available in } N_Q(c) \cap N_P(s)])^k \end{aligned}$$

with

$$\begin{aligned} &\Pr[\forall s \in N(i \text{ is not available in } N_Q(c) \cap N_P(s)] \\ &= \prod_{s=1}^{n} \Pr[i \text{ is not available in } N_Q(c) \cap N_P(s)] \\ &= \prod_{s=1}^{n} (A + B) \end{aligned}$$

$$A = \left\{ \begin{array}{cc} 0 & \text{if } l + l' > n \\ \frac{\binom{n-l'}{l}}{\binom{n}{l}} & \text{if } l + l' \leq n \end{array} \right\}$$

and

$$B = \sum_{m=1}^{\min\{l,l'\}} (\Pr[A_{i,s,c}])^m \frac{\binom{l'}{m}\binom{n-l'}{\min\{l,l'\}-m}}{\binom{n}{l}}$$

It follows that the expected waiting time of the (l, l') post-query strategy is equal to

$$\left(1 - \prod_{s=1}^{n} (A + B)\right)^{-k} \tag{13}$$

The expected cost of the post-query protocol is $(l + l')n$ since each server posts to exactly l nodes and each client queries exactly l' nodes.

Assume that the probability that a given type i of service is offered by a server is at least q and at most p. It is clear that $1-p \leq \Pr[A_{i,s,c}] \leq 1-q$. Using this and Formula 13 we can obtain upper and lower bounds on the expected waiting time $W_{(l,l')}(c)$ that a client c following the (l, l') post-query strategy succeeds in locating all services. Using the approximation, $1-x \approx e^{-x}$, for x a small positive real number, we can derive easily the following estimates for special cases of the (l, l') post-query protocol.

$(1, 1)$ **post-query strategy:** $n \geq 2$

$$\begin{aligned} q^k &\approx \left(1 - \left(\frac{1-q}{n}\right)^n\right)^k \\ &\leq p_{(1,1)}(c) \\ &\leq \left(1 - \left(\frac{1-p}{n}\right)^n\right)^k \\ &\approx p^k \end{aligned}$$

$(1, l')$ **post-query strategy:** $n \geq l' + 1$

$$\begin{aligned} (l'q)^k &\approx \left(1 - \left(\frac{1-l'q}{n}\right)^n\right)^k \\ &\leq p_{(l,1)}(c) \\ &\leq \left(1 - \left(\frac{1-l'p}{n}\right)^n\right)^k \\ &\approx (l'p)^k \end{aligned}$$

$(l, 1)$ **post-query strategy:** $n \geq l + 1$

$$\begin{aligned} (lq)^k &\approx \left(1 - \left(\frac{1-lq}{n}\right)^n\right)^k \\ &\leq p_{(1,l')}(c) \\ &\leq \left(1 - \left(\frac{1-lp}{n}\right)^n\right)^k \\ &\approx (lp)^k \end{aligned}$$

(l, l) **post-query strategy:** $n \geq 2l$

We have that for a given c

$$\Pr[i \text{ is not available in } Q(c) \cap P(s)] = \frac{\binom{n-l}{l}}{\binom{n}{l}} + \sum_{m=1}^{l} (\Pr[A_{i,s,c}])^m \frac{\binom{l}{m}\binom{n-l}{l-m}}{\binom{n}{l}}$$

and the probability of success of the post-query protocol and the expected waiting time is:

$$\begin{aligned} p_{(l,l)}(c) &= \left(1 - \prod_{s=1}^{n} (A + B)\right)^k \\ W_{(l,l)}(c) &= \left(1 - \prod_{s=1}^{n} (A + B)\right)^{-k} \end{aligned}$$

with

$$A = \frac{\binom{n-l}{l}}{\binom{n}{l}}$$

and

$$B = \sum_{m=1}^{l} ((\Pr[A_{i,s,c}])^m \frac{\binom{l}{m}\binom{n-l}{l-m}}{\binom{n}{l}}$$

5.1 Analysis of the (l, l') q-Demand post-query protocol

This is similar to the analysis of the (l, l') post-query protocol given above. We can compute the probability $p_{(l,l')}(c, i)$ that client c finds a given service i at a server in the network. Indeed, we have that

$$\begin{aligned} & p_{(l,l')}(c,i) \\ &= \Pr[\exists s \in N(i \text{ is available in } N_Q(c) \cap N_P(s))] \\ &= 1 - \Pr[\forall s \in N \\ &\quad (i \text{ is not available in } N_Q(c) \cap N_P(s)] \\ &= 1 - \prod_{s=1}^{n} \Pr[i \text{ is not available in } N_Q(c) \cap N_P(s)] \end{aligned}$$

Now note that the event that service i is not available in $N_Q(c) \cap N_P(s)$ is equivalent to the event that service i was not posted in $N_Q(c) \cap N_P(s)$. Since the size of the set $N_Q(c) \cap N_P(s)$ ranges from $m = 0$ to $\min\{l, l'\}$, we obtain that the probability of this last event is equal to

$$\Pr[N_Q(c) \cap N_P(s) = \emptyset] + \sum_{m=1}^{\min\{l,l'\}} (1 - q(i))^m$$

From this we can compute the waiting time $W_{(P,Q)}(c, i)$ as well as the demand cost

$$D_{(P,Q)}(c) = \sum_{i=1}^{k} W_{(P,Q)}(c, i) \cdot w_i \cdot q_c(i)$$

where w_i is the access cost of service i, and maximal demand cost $D_{(P,Q)}$ from Identities 11 and 12.

6 Simulation results

6.1 Simulation environment

The Network Simulator (NS) is the simulation tool used for this work [3]. The Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs is used as the MAC layer protocol. The transmission range is about 250 meters. The signal propagation model combines both a free space propagation model and a two-ray ground reflection model.

A network of 50 nodes is simulated. Ten of them are servers and 40 of them are clients. There are ten kinds of services offered in the network. We assume that all the services follow a uniform distribution. The Random Waypoint Mobility Model [3] is used to generate three different kinds of network scenarios:

- scen300-01: a 300 second pause time and one m/s maximum node speed;
- scen100-10: a 100 second pause time and 10 m/s maximum node speed;
- scen30-30: a 30 second pause time and 30 m/s maximum node speed.

To characterize the mobility of the nodes of an ad hoc network, a parameter called *dynamic ratio* (DR) is introduced. To calculate it, we denote the total number of nodes in the network as N and the total number of link changes of all the nodes during the entire simulation time as LC. The dynamic ratio is then defined as:

$$DR = LC/N$$

A high DR reflects a high mobility. We define the maximum round number R as equal to ten. It is an upper bound on the number of rounds after which servers and clients terminate their execution. If all the clients successfully locate their wanted services, the simulation is terminated. Round is a time interval of 15 seconds.

6.2 Performance metrics

We denote as N, the total number of nodes; N_c, the total number of clients; and N_{succ} the total number of successful clients. For each round r, we denote as $M_{post,r}$, the number of post messages; as $M_{query,r}$, the number of query messages, and as $M_{reply,r}$, the number of reply messages. Each successful client c receives one to m reply messages associated with waiting time $t_{c,1}, t_{c,2}, \ldots, t_{c,m}$. We use three performance metrics: success rate (SR), number of transmitted messages (NTM), and average waiting time (AWT).

SR is the percentage of clients which successfully locate the services for which thay are loooking for. It is calculated using the following formula:

$$SR = N_{succ}/N_c \times 100\ (\%)$$

NTM is the number of messages transmitted in each round by all the nodes in the network. It includes the

post, query and reply messages. It is calculated using the following formula:

$$NTM = \sum_{n=1}^{N} M_{post,r} + M_{query,r} + M_{reply,r}$$

AWT is the minimum time period, in seconds, averaged over all the clients, starting from the sending of a query message and ending with the receiving of a reply message. A client which fails to locate a service after r rounds has the waiting time r times the round interval seconds. It is calculated using the following formula:

$$AWT = \frac{\sum_{n=1}^{N_c} min(t_{1,c}, t_{2,c}, \ldots, t_{m,c})}{N_c}(s)$$

Given a post-query strategy, good performance means a high SR, a short AWT, and a low NTM.

6.3 Detailed results greedy strategy

We review in detail the simulation results of the greedy post-query strategy. The simulation results associated with the conservative, incremental, uniform memory less, and with memory post-query strategies are summarized in Table 1. The greedy post-query strategy requires that all servers post their services to all the nodes, and that all clients query their services from all the nodes in the network in each round. To achieve this goal, a flooding algorithm is used. Each node tries to forward every message to every neighbor. This results in every message eventually being delivered to all reachable parts of the network. A message ID based mechanism is used to avoid duplicate deliveries and infinite loops.

Figures 1 show that this strategy can achieve a SR of 100% when combined with the DSR and DSDV protocols, which means that all the clients successfully locate the desired services. The flooding algorithm used in this strategy results in a large number of nodes being posted to or queried from within a low number of rounds. The higher the DR of the network is, the lower the SR this strategy has. As the DR increases, the DSR protocol has more routing overhead, and the DSDV protocol has a lower packet delivery ratio [2] which result in a lower SR. When this strategy is combined with the DSR protocol, it has a lower SR than the one when combined with the DSDV protocol. The flooding algorithm requires the underlying routing protocols to accommodate the heavy routing demands. In such circumstance, the DSR protocol is aggressively adopted and it has more routing overhead than the DSDV protocol.

Figures 2 show the NTM of the greedy strategy as a function of the number of rounds, over the DSR protocol or DSDV protocol. This strategy is costly in terms of NTM. As the DR becomes higher, the NTM becomes lower. When this strategy is combined with the DSR protocol, it has a lower NTM than when combined with the DSDV protocol.

Figure 3 presents the AWT for different DRs of the greedy combined with the DSR protocol or DSDV protocol. As the DR increases, the AWT becomes longer. With the DSR protocol, this strategy has longer AWT than with the DSDV protocol. As an on-demand routing protocol, DSR has to accommodate heavy routing demands of establishing routes from all nodes to all nodes of the network, which is very time-consuming. As a table-driven routing protocol, in the DSDV protocol, routes are discovered before messages are sent, which will save time.

Table 1 lists the simulation results for the conservative, greedy, incremental, uniform memory less, and with memory post-query strategies when combined with the DSR protocol or DSDV protocol. Note that what is important here is not so much the absolute numbers, but their relative values. In other words, better or worst absolute numbers can be obtained by changing the simulation conditions, but their relations will remain similar.

Each strategy, given a routing protocol, was tested with three different DRs. There is a total of 30 combinations. The following conclusions could be drawn. The greedy strategy consumes a significant amount of network resources to achieve a high SR. Among all the strategies, it has the highest NTM and shortest AWT. When requiring a service discovery strategy with a high SR and a short AWT, regardless of NTM, the greedy strategy combined with the DSDV protocol is a good choice. The conservative strategy can also achieve a high SR with a low NTM and a short AWT in a high DR ad hoc network. In a low DR ad hoc network, the SR is low because numerous nodes may not be posted to or queried from at all. For a high DR ad hoc network, this strategy combined with the DSR or DSDV protocol is the best choice. The incremental strategy can achieve a high SR with a low NTM, while it has the longest AWT of all the strategies. When requiring a high SR with a low NTM, regardless of AWT, this strategy combined with the

Strat.	**RP**	**DR**	**Max. SR**	**Total NTM**	**AWT**
Conservative	DSR	8.06	82.25%	2909	36.55
		61.48	99.75%	5455	20.12
		190.72	100%	3620	8.42
	DSDV	8.06	81.5%	2122	37.62
		61.48	99.5%	4519	27.81
		190.72	100%	3408	10.09
Greedy	DSR	8.06	100%	16215	7.50
		61.48	100%	15601	7.85
		190.72	98.75%	13100	10.42
	DSDV	8.06	100%	16557	4.93
		61.48	100%	15777	5.03
		190.72	100%	14887	5.46
Incremental	DSR	8.06	96%	2900	79.72
		61.48	94.75%	2894	81.32
		190.72	96%	2899	83.76
	DSDV	8.06	97%	2898	83.23
		61.48	87.75%	2856	90.80
		190.72	81.75%	2828	107.15
Uniform memoryless	DSR	8.06	89%	2602	54.55
		61.48	92.5%	2610	54.30
		190.72	87.5%	2602	57.72
	DSDV	8.06	88.25%	2596	59.27
		61.48	83.25%	2577	64.57
		190.72	75.25%	2555	87.64
With memory	DSR	8.06	100%	2845	46.96
		61.48	100%	2833	47.29
		190.72	100%	2832	53.05
	DSDV	8.06	100%	2816	51.89
		61.48	98.5%	2741	56.12
		190.72	97.75%	2686	77.27

Table 1: The performance of the post-query strategies over the DSR protocol or DSDV protocol

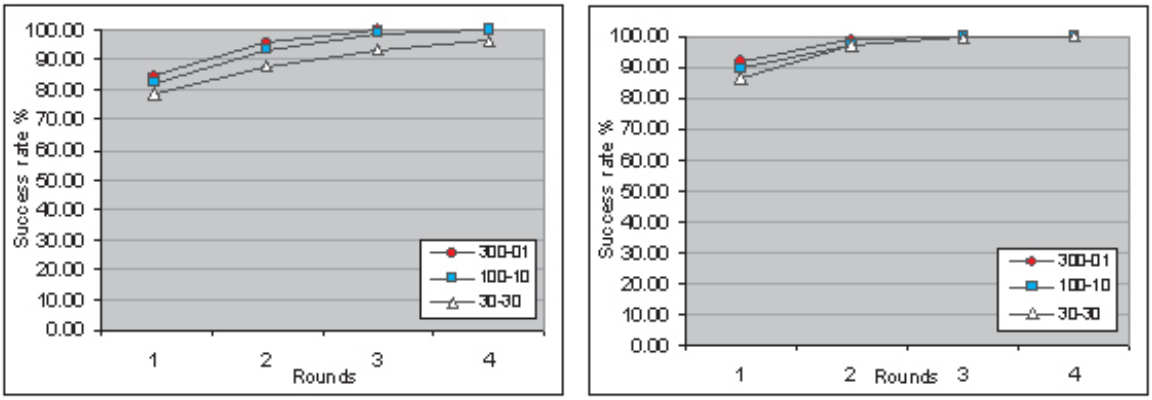

Figure 1: The SR of the greedy strategy over DSR(left) and DSDV (right).

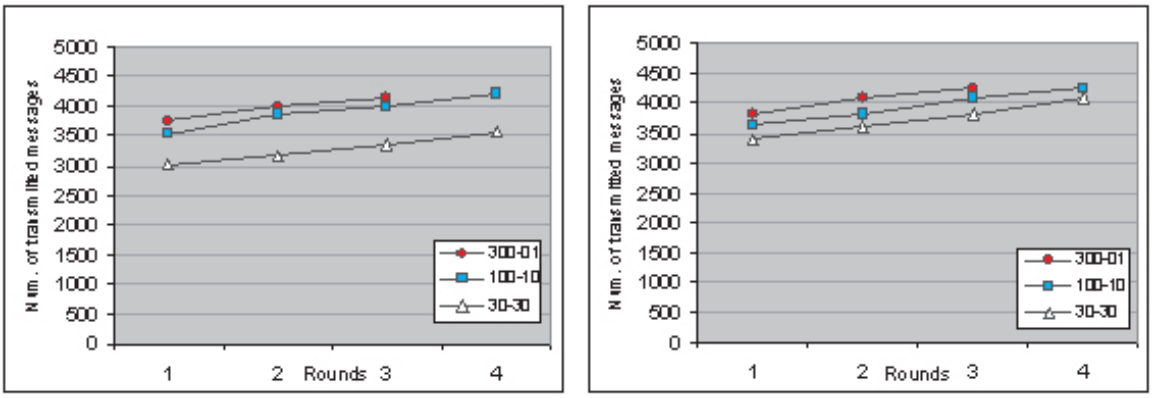

Figure 2: The NTM of the greedy strategy over DSR (left) and DSDV (right).

DSR protocol is the best. With the uniform memoryless strategy, if the sizes of posting and querying sets are relatively low, the NTM is low and, in each round, almost remains constant. It uses a relatively low amount of network bandwidth. For an ad hoc network which can only spare a dedicated bandwidth for service discovery, this strategy combined with the DSR protocol or DSDV protocol is the most appropriate. The with memory strategy aims to improve the SR. It can achieve a high SR with a low NTM and a long AWT. In contrast with the uniform memoryless strategy, the SR of the with memory strategy is higher. But the tradeoff is that each node is required to build a cache of about 0.2 KB to store the IDs of previously visited nodes. When requiring a high SR and a low NTM and can bear with more memory usage, regardless of the AWT, the with memory strategy combined with the DSR protocol is the most suitable.

7 Conclusion

In this paper, we have defined post-query strategies as time dependent post-query protocols that are executed in rounds. We have created a general methodology for evaluating the performance and efficiency of service discovery strategies in ad hoc networks. Several interesting problems remain for further investigation. We believe it is an interesting mathematical challenge to refine our methodologies and propose optimal strategies. Our heuristics are under the assumption that all clients are employing the same probability distribution. What are appropriate heuristics when the number of network nodes is unknown and clients do not necessarily employ identical demand probability distributions? A more difficult problem is what strategies to consider when the demand distribution used is unknown.

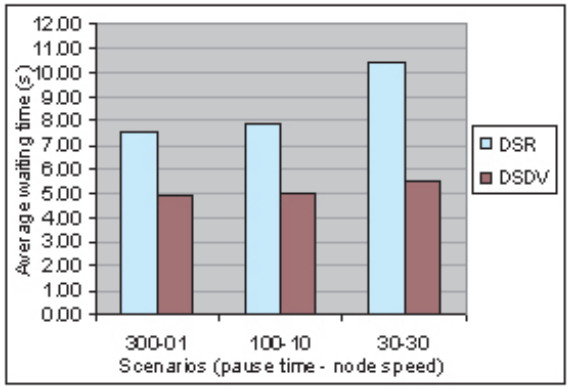

Figure 3: The AWT of the greedy strategy with DSR or DSDV.

References

[1] Bluetooth. Specification of the Bluetooth system, November 2003.

[2] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, Dallas, TX, October 1998.

[3] K. Fall and K. Varadhan. NS notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, 1997.

[4] A. Farago and V.R. Syrotiuk. Merit: A unified framework for routing protocol assessment in mobile ad hoc networks. In *SIGMOBILE*, pages 53–60. ACM, 2001.

[5] E. Guttman, C. Perkins, J. Veizades, and M. Dago. *Service Location Protocol, Version 2*, volume RFC 2608. IETF, 2000.

[6] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - a service discovery and delivery protocol for ad-hoc networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, volume 3, pages 2107–2113, New Orleans, March 2003.

[7] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade. Deapspace - transient ad-hoc networking of pervasive devices. *Computer Networks*, 35:411–428, 2001.

[8] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad-hoc wireless networks. *Mobile Computing*, 353, 1996.

[9] J. Kempf and P. St. Pierre. *Service Location Protocol for Enterprise Networks*. Wiley, 1999.

[10] D. Knuth. *The Art of Computer Programming: Volume 3, Sorting and Searching*. Addison Wesley, 3 edition, 1998.

[11] R. Koodli and C.E. Perkins. Service discovery in on-demand ad hoc networks. Internet-Draft, 2002.

[12] E. Kranakis and P. Vitányi. A note on weighted distributed match-making. *Mathematical Systems Theory*, 25:123–140, 1992.

[13] Sun Microsystems. Jini architectural overview. Technical White Paper, December 1999.

[14] S. Mullender and P.Vitányi. Distributed match-making. *Algorithmica*, 3:367–391, 1988.

[15] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, October 1994.

[16] C.E. Perkins, editor. *Ad Hoc Networking*. Addison Wesley, 2001.

[17] C.E. Perkins, editor. *Is the Client Server Model Viable?*, pages 353–354. In Perkins [16], 2001.

[18] C. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison Wesley, Reading Mass., 1949.