A Reinforcement-Learning Approach to Service Directory Placement in Wireless Ad-hoc Networks

Sergio González-Valenzuela, Son T. Vuong, *Senior Member*, *IEEE*, and Victor C. M. Leung, *Fellow, IEEE*

***Abstract*— In this paper we address the problem of service directory placement in wireless ad-hoc networks. In particular, we introduce the Service Directory Placement Protocol, or SDPP, which enables directory-based service discovery in wireless ad-hoc networks. In our approach, the service directory is moved from one host to another in response to the dynamics of a service discovery overlay system and the network topology, as opposed to having a service directory confined to a single host. The objective is to incur in less bandwidth overhead by conveniently placing the directory in a network neighbourhood wherein a significant amount of service discovery queries are likely to be generated at that point in time. To accomplish this objective, we formulate the directory placement problem as a Semi-Markov Decision Process. The challenge here is to find an efficient policy for directory placement by means of the Reinforcement Learning technique within the context of a low-mobility wireless ad-hoc network. Performance measurements depicting the degree of effectiveness introduced by our approach, as well as its limitations are presented as part of an ongoing investigation.**

***Index Terms*—Directory Placement, Service Discovery, Ad-hoc Wireless Networks, Reinforcement Learning.**

I. Introduction

Earlier research on routing protocols for wireless ad-hoc networks, along with recent advances in wireless communications has revamped an interest in the potential applicability of this kind of networks for commercial applications. The incorporation of both novel technologies supporting higher data-rates and miniature devices with increased data storage capacity, will leverage a more effective market penetration of small portable devices with multimedia capabilities in the highly-profitable entertainment sector [1]. However, although many pieces of the complex wireless technology puzzle are finally coming into place, a few important issues remain to be solved. To this effect, we regard service discovery as a key piece that brings awareness to the user applications about network facilities, and enables users to collaborate through the lower-layer infrastructure that the wireless communications network provides.

The role and scope of a service discovery protocol (SDP) in a wireless ad-hoc network differs from that in wire-line networks. Devices in a wire-line network employ a well-defined procedure to connect to service-providing devices that are normally located at fixed places for extended periods of time. In contrast, locating either fixed or mobile service-providing devices through other mobile hosts in wireless ad-hoc network requires an altogether different approach. Therefore, enhanced mechanisms are required to enable user applications find services given unpredictable topology changes in an ad-hoc wireless network.

Unfortunately, existing SDPs, such as Universal Plug-and-Play (uPnP) [2], Jini [3], Salutation [4], and the Service Location Protocol (SLP) [5] did not take into consideration their eventual deployment over wireless ad-hoc networks. In fact, little or no performance evaluations of these SDPs over mobile ad-hoc networks (MANET) have been reported in the literature. Instead, most of the available papers on the subject limit themselves to guessing about the adaptability of existing SDPs to the wireless ad-hoc network realm, and produce conclusions based mainly on qualitative analysis [6-9].

Protocols like SLP and Jini share fundamental similarities, being the most evident their relying on a framework of federated devices that form a network. To this regard, user devices are hierarchically organized at the application level by assuming the role of either provider (server) or consumer (client) within a service-oriented overlay network. A third entity known as a directory is also defined, although its availability is considered optional. In fact, service-providing devices are often expected to serve as service information repositories too. In any case, no service directory placement strategies have been convened or agreed upon for any of the most popular SDPs described above.

Presumably, employing a directory on a wireless ad-hoc network would make little or no sense. The reason for this is that clients would have to locate the directory each time that changes in the topology render its latest known location obsolete. This perhaps is the main reason behind the directory-less approach being widely deemed as default given the circumstances inherent to wireless ad-hoc networks. In the next section we provide supporting arguments for directory-based service discovery in wireless ad-hoc networks, and

Manuscript received April 11, 2005; revised May 26, 2005. This work was supported by the National Sciences and Engineering Research Council of Canada through grants STPGP 257684 and RPGIN 2515.

Sergio González-Valenzuela and Victor C. M. Leung are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T1Z4 Canada (email: sergiog;vleung@ece.ubc.ca, phone: 604-822-2872; fax: 604-822-5949).

S. T. Vuong is with the Department of Computer Science, University of British Columbia, Vancouver, BC V6T1Z4 Canada (e-mail: vuong@cs.ubc.ca, phone: 604-822-6366, fax: 604-822-5485).

discuss previous work on the subject. In Section III we introduce the basic concepts behind Markov Decision Processes (MDPs) as background for our problem-solving methodology. The actual formulation of the service directory placement problem as an MDP is then presented in Section IV. In Section V, we describe our implementation of SDPP, and comment on the results obtained so far. Finally, we proceed with a discussion in Section VI for improvements and future work.

II. Previous work

In its basic form, the directory placement problem can be directly related to the facility location problem [10]. The objective here is to find an optimal location for a service facility, so that the incurred cost when honouring requests generated by spatially distributed consumers is minimal. The facility location problem can also be generalized for the case of more than one facility, and is well known in the field of Graph Theory to be NP-hard [11]. A variant of the facility location problem is known as the minimum k-median problem, which differs from the former by not considering the costs associated with the creation of a service facility. This problem is known to be NP-hard too.

Algorithms that find sub-optimal solutions to both of these problems have already been found, and employed in determining the placement of web-server replicas on the Internet [12]. However, the applicability of algorithms that solve the previously mentioned problems for service directory placement in ad-hoc networks is dubious. The reason to this is that neither of these approaches takes into consideration the fact that the network topology changes in time. This has led to the proposal of alternative algorithms that do take into account the issue of network mobility. A few other solutions in the area of distributed content placement over ad-hoc networks have been proposed also [13, 14].

We contend that the directory-based approach for this type of networks is purportedly flawed because such directory is assumed to remain anchored to a single host during the network's lifetime. However, the possibility of having a mobile directory whose current location would depend not necessarily on the network topology, but on the dynamics of the service querying process introduces new perspectives from which the service discovery problem can be approached. An efficient decision mechanism for directory placement that enables the possibility of bandwidth savings, while incurring in minimal overhead hence becomes of foremost importance.

Therefore, we turn our attention to the applicability of MDPs as an efficient tool employed to solve a wide variety of engineering problems. MDPs have already been successfully used to solve mobility-related problems in the area of cellular networks. In fact, Dynamic Programming has been considered as a solving method for MDPs in determining the placement of web proxies on the Internet [16]. In this paper, we study the applicability of MDPs for directory placement in wireless ad-hoc networks.

III. Solving Markov Decision processes Using reinforcement Learning

An MDP [15] is comprised of an *agent* that makes decisions based on the current *state* of the system to which the procedure is being applied. It then chooses one of the predefined *actions* in an attempt to maximize the amount of *reward* received in the long run. The set of rules that maximizes this reward is called a *policy*. System states are mapped to values that serve as a measure of their worthiness. This is known as a Value Function (VF), and is used by the agent as a reference when choosing actions. In general, MDPs can be solved by means of a technique known as Reinforcement Learning (RL), and in particular, by employing one of the following three methods: Dynamic Programming (DP), Monte Carlo (MC), and Temporal Difference learning (TD). The objective of these methods is to learn the true mapping of states to state values, i.e. learn the VF. These initially arbitrary values are gradually adjusted to their true values by the RL process over the period of time in which the algorithm executes.

Formally, the initial VF can be depicted as the sum of a true values plus some error for each of the system's states at time *t*:

$$V(S_t) = V^*(S_t) + e(S_t), \tag{1}$$

wherein V^* denotes the optimal VF. Thus, the objective of the RL algorithm is to minimize the error $e(S_t)$:

$$e(S_t) = \lim_{t \to \infty} [V(S_t) - V^*(S_t)]. \tag{2}$$

In order to do this, the agent must learn which actions yield the largest rewards over an extended period of time. MDPs can additionally be classified as either finite or infinite-horizon. For this investigation we are interested in infinite-horizon processes, wherein there is no absorbing final state that the process might reach. A mechanism known as *discounting* is thus introduced in order to safeguard the computation of an unbounded sum of rewards when dealing with infinite-horizon processes. In addition, a discounting factor γ with an initial value (0,1] is employed to reduce the weight of future rewards. That is, the reinforcement value is gradually attenuated as time elapses during the execution of the RL algorithm:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{3}$$

wherein R_t is the accrued sum of rewards at time t for an infinite-horizon RL process.

Once $V^*(S)$ has been computed the agent can choose actions that maximize R_t, leading to the optimal policy π^*, under which the final value of any state s_i equals the expected sum of rewards when the process starts at an initial state s_0 and chooses actions a from the set $A(S)$ thereafter:

$$V^{\pi}(s) = E\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, t = 0\right\}. \quad (4)$$

Here, E denotes the expectation of the sum of reinforcements as per the probability distribution that maps all possible successor states to actions. Since it might be possible to take more than one action at any given state, state-action pairs can then be individually assigned their own reward Q. The Q-Learning algorithm [17] maps state-action pairs to their corresponding Q value. Therefore, the Bellman optimality equation ensures that an optimal mapping can be achieved by assigning the maximum reward to state-action pairs:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a' \in A(s')} Q^*(s',a'), \quad (5)$$

wherein $P^a_{ss'}$ denotes the probability of going from state s to state s' when an action $a \in A(S)$ is taken and policy π is followed thereafter. Equation (5) can be rewritten in terms of the mean (expected) value as:

$$Q^*(s,a) = E\left\{r_{t+1} + \gamma \max_{a' \in A(s')} Q^*(s',a') \mid s_t = s, a_t = a\right\}. \quad (6)$$

Working with the state-action pairs as opposed to just the state values provides us two desirable features [18]:

1) The availability of optimal expected rewards of individual state-action pairs (Q^*) eliminates the need to conduct one-step look-ahead searches for all possible subsequent states that are otherwise necessary if only state values were available. Instead, the availability of a Q^* value allows the selection of actions without the need to know all of the possible successor states, which also eliminates the need to know about the underlying environment's dynamics. In other words, *a model of the system's environment being studied is no longer necessary*, as seen in MC RL methods, but not in DP. This is a powerful feature of Q-Learning as per the potentially large state space often encountered when solving MDPs, which otherwise renders the calculation of $P(s'|s,a)$ and $R(s'|s,a)$ intractable.
2) There is no need to calculate the Q value of all possible actions $a \in A(S)$ in state s. Instead, the Q-Learning algorithm updates $Q(s,a)$ for individual actions asynchronously during disjoint time steps until they converge to their optimal values, as long as the algorithm executes an infinite number of iterations. Hence, learning takes place gradually as preliminary Q values slowly converge to their optimal value. This is known as bootstrapping, a feature not observed in MC methods, but available when employing DP, which results particularly appealing for infinite-horizon RL processes.

Q-Learning can thus be considered as a subcategory of Temporal Difference learning since updates to Q values are based on the difference between two estimates at disjoint points in time. The Q-learning model is shown in Figure 1.

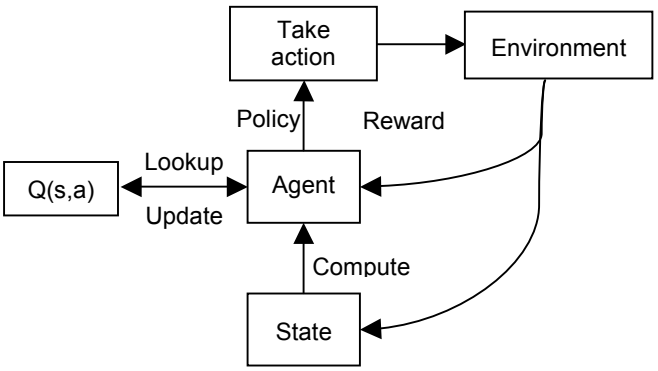

Fig. 1. The Q-learning model

IV. Formulating the Service Directory problem as a Markov Decision Process

We devise SDPP as a discrete event system since service-related queries occur randomly at discrete times. However, the amount of time between these events is actually real-valued, which is why the directory-positioning problem is formulated as a *Semi*-Markov Decision Process (SMDP). If the amount of time elapsed between two events were discrete and constant, then we would have an MDP. The sequence of events observed in this system can be depicted in the time-line shown in Figure 2, where ε_i represents the decision epoch at which the learning agent decides to take an action according to a service-related query, and τ_i is the elapsed time between two decision epochs.

A. States

The state of the network at ε_i is an abstract representation that takes into account the following factors as measured for the duration of each time epoch:

1) The difference between the number of neighbouring hosts at the location of the querying node and that at the current location of the directory.
2) The hop distance between the directory and the querying host for the current query.
3) The average hop distance that service-related query packets travel as perceived by the directory at its current location.
4) The difference between queries processed by the directory at its current location and queries generated in the neighbourhood of the querying host.

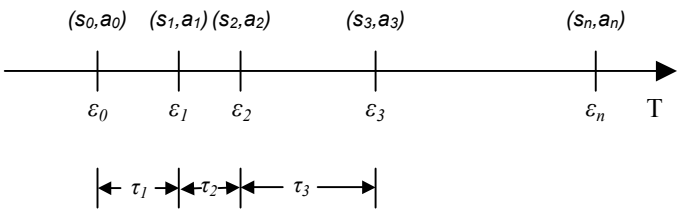

Fig. 2. Time-line of events for the formulation of SDPP as an SMDP

Information about the estimated number of neighbouring nodes can be readily obtained either from the MAC layer or from a topology formation protocol, if available, whereas the hop distance values are obtained directly from the query packets. The use of knowledge about the network topology as a whole is avoided altogether since forwarding topology information across the network would consume a prohibitive amount of bandwidth. Moreover, the mobility issue, along with network size would only aggravate matters in trying to maintain current topology information. Instead, we would like to capture the system dynamics of the service-oriented overlay network. In fact, knowledge of the whole network topology might be of little use, since a directory is expected to contribute to bandwidth savings in the network region in which is deployed. This is the motivation for the first factor being considered in formulating the system's state. The hop distance factor is introduced to account for the amount of bandwidth consumed if the directory is moved to another host. This could also be an indicator of the extent of the service discovery broadcast process. However, the hop distance is considered part of the reward signal that is feedback to the agent for the following epoch, which helps in evaluating the goodness of the action previously taken. The average hop distance factor is introduced in order to take into account the closeness of the directory to neighbouring hosts querying the directory. The fourth and final factor is intended to provide the agent with a better estimate of the current system dynamics, so that it can ponder the potential reward obtained if the directory is moved.

B. Actions

The agent can take on one of the following actions: let the directory stay at the current host, or, hop to the host that generated the last service discovery query. The lack of initial knowledge about the goodness associated with an action for every system state suggests choosing one of the two actions in a random fashion. This is in fact the starting point for the learning agent, although the action-choosing strategy changes as the learning process evolves.

C. Rewards

The reward signal that the agent receives must reflect the amount of packets generated due to service-oriented queries. The reward is thus computed by dividing the number of service-oriented packets by the number of queries processed for the current time epoch:

$$r(s,s',a) = -\frac{\int_{t=\varepsilon}^{t_{\varepsilon+1}} c(W_t, s_t, a_t)\, dt}{\int_{t=\varepsilon}^{t_{\varepsilon+1}} \upsilon_t \, dt}, \tag{8}$$

wherein W_t represents the state of the natural process, and υ_t is the number of queries at time t during the n^{th} epoch. This equation yields a negative average reward signal that is proportional to the number of queries generated. Thus, less negative values depict a larger reward, whereas more negative values represent less reward. The number of service-oriented queries is therefore employed here as a normalizing factor.

D. Definition of SDPP as an SMDP

The Q-Learning function features a discount factor γ that is re-defined given that the states' sojourn time is continuous random variable, not discrete and constant [19]. A reward rate $\rho(s,a)$ and a probability distribution $F_{ss'}(\cdot|a)$ for the states' sojourn time are accordingly employed to account for this effect, leading to the redefinition of the Bellman optimality equation described above (5) as:

$$\begin{aligned} Q^*(s,a) = \sum_{s' \in S} p(s'|s,a) \cdot \int_0^\infty \int_0^t e^{-\beta s} \rho(s,a) ds\, dF_{ss'}(t|a) \\ + \sum_{s' \in S} p(s'|s,a) \cdot \int_0^\infty e^{-\beta t} \max_{a' \in A(S)} Q^*(s'a')\, dF_{ss'}(t|a), \end{aligned} \tag{9}$$

yielding the following rule for Q-learning in SMDPs:

$$\begin{aligned} Q_{t+1}(s,a) = Q_t(s,a) + \\ \alpha_t \left[\frac{1-e^{-\beta t}}{\beta} r(s,s',a) + e^{-\beta t} \max_{a' \in A(s)} Q_t(s',a') - Q_t(s,a) \right]. \end{aligned} \tag{10}$$

Here, a learning rate α is introduced in order to allow for the algorithm to converge to one of several possibly existing policies, and β is a factor that controls the rate of exponential decay. Both values are initially set in the (0,1] range and are gradually decremented as the algorithm executes until it reaches 0. An effective policy learned by the agent will help SDPP determine whether is best for the directory to be kept in its current location, or be moved to another host according to the system state, which is computed when a service discovery query is received as depicted in Figure 3. The learning process can therefore take place off-line, and once an efficient directory placement policy is available, it can be used by the protocol made available to all hosts running their SDP of choice (e.g. Jini, SLP, etc.) The efficiency of the SDPP will be measured in part by its ability to meet the following constrain:

SD signalling cost + SDPP overhead < cost of not using SDPP.

Actions are selected by employing the well known *ϵ-greedy* scheme, which indicates either the choosing of action *a* in *A(s)* that has the largest Q-value for the current state with probability *1-ϵ*, or taking one of the other actions in *A(s)* as an exploratory move. ϵ is slowly decremented to 0 during the course of the learning process, which ensures that the algorithm converges to the optimal policy π^*.

V. System Implementation and Simulation Results

Our protocol is implemented using the Mobility Framework of the OMNeT++ discrete event simulator [20]. The mobility model employed is equivalent to the random-waypoint model [21], except that no pause times are employed. We simulated a generic wireless ad-hoc network in which low-mobility client hosts are randomly dispersed in an area of 500 m. x 600 m.

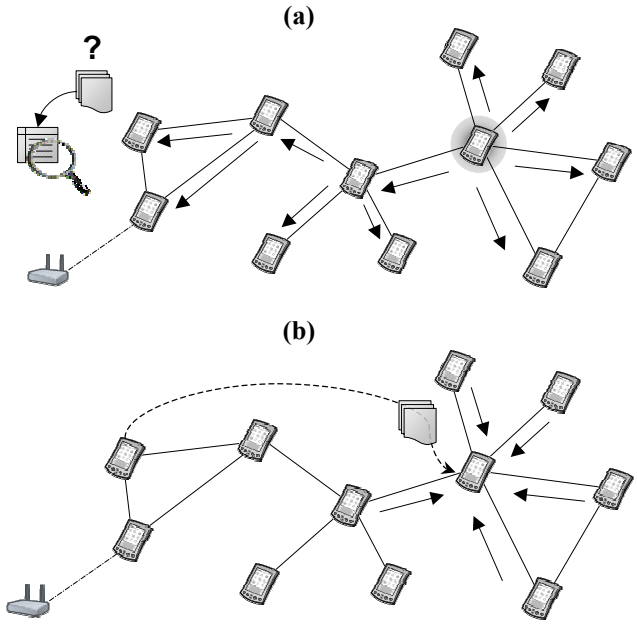

Fig. 3. The mobile directory concept in a wireless ad-hoc network

creating a mesh network. For simplicity, a generic AODV/DSR-like routing protocol is employed too. It is assumed that all hosts are running the same SDP protocol (e.g. SLP), which in turn makes use of SDPP.

Service discovery queries are broadcasted at disjoint times by each client through the topology-changing network following an incremental breadth-first approach. TTL values are initially set to 2, and are doubled for subsequent queries should the previous one fail to reach the service provider, as opposed to employing brute-force flooding at once. When the service provider is reached, a reply signal is sent back to the querying node, which in turn employs the route found for subsequent unicast queries to the service provider. Clients query for services employing exponentially distributed inter-arrival times with a mean value of 2 minutes. Failure to receive a reply signal of a subsequent service query triggers the initial broadcast process to re-discover a new path for the service provider or the mobile directory as depicted in the Figure 4.

At the beginning of the simulation the only directory available is located within the service-providing device, which is itself positioned halfway along one of the 600 m. edges of the rectangular test area. The time at which the directory receives a service discovery query is considered the ending period of the current time epoch and the beginning of a new one, wherein the learning agent chooses among one of two available actions according to the ϵ-*greedy* scheme, with ϵ initially set at 0.1.

We say that the service directory is detached from the service provider when one or more service advertisement entries are registered into the data-empty directory of a remote host, which will now contain a copy of the directory. The details of this process depend on the actual SDP being used. SDPP's task is therefore limited to recommending the underlying SDP when and where to create a copy of the service directory. This directory-moving scheme also implies that, except for the initial directory detachment from the

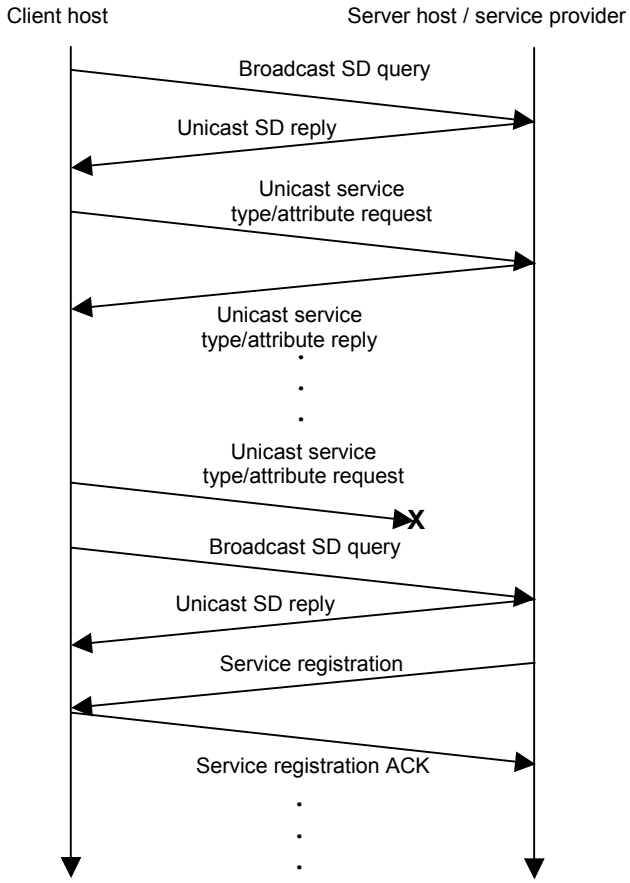

Fig. 4. Signalling sequence for service discovery/query in SDPP

service provider, subsequent service advertisement registrations are accompanied by service de-registrations at the host where the service information used to reside. This procedure is repeated as necessary according to the decision taken by the learning agent during the simulation's lifetime. For simplicity, a single service registration/de-registration event at the remote/local host is respectively performed in our simulations. In reality, the service registration process depends on the actual SDP being employed. While a particular SDP might register/de-register single directory entries at a time, another SDP might transfer the whole directory as an encapsulated software object at once, perhaps as compressed data. Alternatively, an SDP employing a mobile agent-based scheme would transfer not only the whole directory data at once, but the very agent that processes service queries too.

Once the service directory is detached from the service provider, both the host where the new directory copy now resides and the service provider may answer to a service discovery query when they hear one. However, the querying host will only keep the address of the first one to answer, so that a second reply heard by a client is subsequently discarded.

Q(s,a) values are computed and stored in a table at the end of a time epoch by employing equation (10), which takes into account the duration of the decision epoch, the amount of reward received, as well as parameters α and β, whose initial values are both set to 0.1. All of α, β, and ϵ are decremented linearly until they reach a value of 0 at the very end of the simulation according to the following equation:

$$x'(t) = x(t) - \left\lfloor \frac{x(t) \cdot t}{T} \right\rfloor, \quad (11)$$

wherein x is the value being computed (α, β, or ϵ), t is the current simulation time, and T represents the total simulation time. Results are collected in runs equivalent to 10000 simulation hours per run. Packets are counted in a moving-window fashion by resetting the count every ten simulated hours during the course of the simulation in order to measure the performance of the learning system as time elapses. As mentioned before, our simulations take into consideration mobile ad-hoc networks whose hosts move slowly in order to simulate people walking at a low pace of 0.5 m/s, a normal pace of 1 m/s, and a fast pace of 2 m/s. For now, two network sizes of 15 and 25 hosts were employed in order to assess the efficiency of employing only one service directory.

Figure 5(a) shows the number of packets averaged over 500-hour periods for a smoother plot for hosts moving at 0.5 m/s in a 15-host network. It can be observed that, although there appears to be a point in time in which small bandwidth savings can be obtained, in the end the result is not encouraging for this scenario. In fact, the average amount of bandwidth preserved due to service discovery/query packets per hour when employing SDPP is just slightly lower than that of the directory-less approach.

We attribute the previous result to the low mobility of the network hosts, meaning that the known path of each host toward the directory remains somewhat stable. Since these paths are longer lived, the number of service attribute/type queries is evidently higher than the number of service discovery queries. As a consequence, sporadic service discovery queries received by the directory cause SDPP to recommend its relocation, and hosts that had been relying in it re-launch the cost-expensive service discovery broadcast process once they realize it's no longer there. The end result is that, SDPP's response to service discovery queries in low-mobility scenarios yields no bandwidth gains.

Conversely, Figure 5(b) does show a noticeable improvement in bandwidth gain when hosts move twice as fast at 1 m/s. Here, the average amount of packets generated is just over 200000 packets for the last 500-hour period, against 257700 averaged for the directory-less approach – a mere 22% gain in bandwidth savings. We attribute this gain to SDPP's ability to foresee potential packet savings if the directory is relocated to the host where the query was issued, as opposed to staying at the current host. This decision is driven by the perceived network state inferred from the information contained in the service discovery query generated at that particular location. However, there is a cost associated with this action, which is incurred by service discovery queries generated by hosts that were relying on the directory staying at its previous location. Therefore, the reward obtained by the learning process when deciding to relocate the directory must be greater than the one obtained if the directory is allowed to stay at the current host. The amount of overhead attributed to the directory's relocation process is minimal when compared to the one incurred by the service discovery queries. It can also be observed that the optimal policy is almost reached by the time the agent has collected information for approximately 1500 simulated hours. In this case, minimal bandwidth gains are attained afterwards.

Similar bandwidth gains can be obtained for the same network sizes, but with hosts moving now at 2 m/s. Here a directory-less approach yields an average of almost 294000 packets per each 10-hour period, whereas SDPP reduces this quantity to around 222500, allowing bandwidth savings of almost 25%. In fact, the mere existence of a service directory being moved throughout the network without any initial strategy proved more bandwidth efficient than its counterpart.

Additional experiments were conducted to quantify the effectiveness of our approach, in which the size of the network was increased to 25 hosts. Figures 6(a-c) depict results for SDPP with the same parameters for the learning system as above. Both charts make evident the performance degradation of the mobile directory scheme as the network size grows.

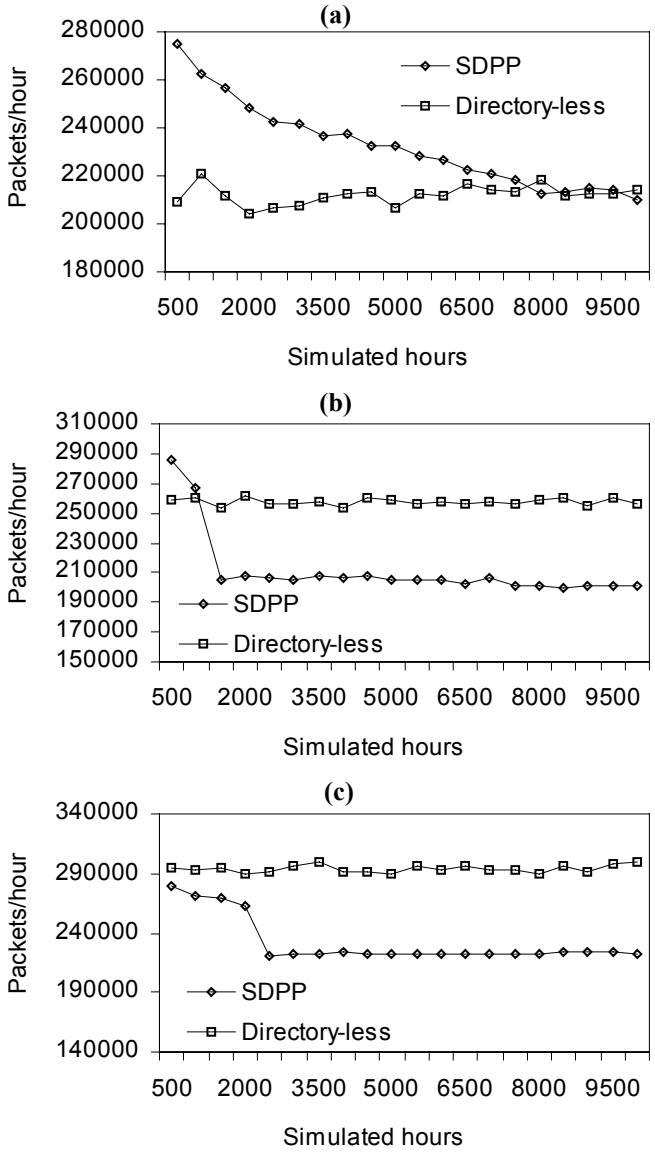

Fig. 5. SD packets per hour, averaged over 50-hour periods for a 15-host network with a host velocity of: (a) 0.5 m/s, (b) 1.0 m/s, and (c) 2.0 m/s.

Though by smaller margins, SDPP was still able to outperform the directory-less approach for the same mobility speeds as in the 15-host case. Figure 6(a) merely confirms previous observations for the deployment of SDPP in networks of very-low mobility. Figure 6(b) however does show an improvement introduced by the learning approach that SDPP relies on. Here, an average 12% of bandwidth gain can still be obtained when comparing the 1 million-plus packets observed every 10-hour period by using SDPP, against the 1140000-plus packets incurred by the directory-less approach. Figure 6(c) shows that SDPP approach is also able to outperform its directory-less counterpart also by an average of 12%. However, here the learning approach is unable to produce any striking improvement when 2 m/s host velocities are considered.

Table I shows the number of visited states in the learning system employed in our simulations. Notwithstanding the fact that only four factors were employed to represent the system's state, the number of states visited fluctuates in the range of thousands. Clearly, employing the Dynamic Programming approach to solve the SMDP would have resulted in a daunting task, given the large number of state transition probability equations that would have needed to be solved. Similarly, the state space ended up being relatively conservative, given that the formulation of our problem as an SMDP points to a continuous space as per the interarrival time of the service discovery queries at the directory. However, the interarrival time of these queries was not taken into account as part of the system's state representation. Instead, this parameter is employed to compute the *Q(s,a)* value by means of equation (10) through the terms that adjust both the reward, and the *max Q(s',a')* values. Thus, the epoch duration is directly employed in the calculation of the reinforcement signal's magnitude as a delayed reward, whose value is measured in seconds by the learning system and incorporated accordingly in equation (10).

VI. Discussion and Further Work

We have presented the Service Directory Placement Protocol for service discovery in wireless ad-hoc networks. SDPP is designed to work in conjunction with any SDP that enables directory-based service discovery. We observed that the formulation of SDPP's decision scheme as a Semi-Markov Decision Process proved useful in defining a policy that helps to accomplish our bandwidth-savings objective. The use of Reinforcement Learning, and in particular the Q-Learning technique, also proved an invaluable tool to solve the SMDP. Results obtained through computer simulations helped to measure both the effectiveness and the limitations of the directory-based approach against the directory-less approach for different network sizes and host velocities.

The results obtained for a low-mobility 15-host network proved the usefulness of the directory-based approach, with average bandwidth savings of up to 22%. Results observed for the 25-host network revealed the performance degradation experienced when SDPP is employed here. Still our mobile directory approach was able to obtain bandwidth gains of up to 12% on average. Beyond this point, a directory-less approach might suffice since no substantial gains would be expected by employing the directory-based approach.

Existing SDPs, such as Jini and SLP do provide the necessary structure to deploy a mobile directory-based service advertisement/discovery scheme. SDPP may then become a useful asset to these and other SDPs.

It's interesting to note that SDPP can further benefit from the use of Q-Learning by keeping ϵ at a low value, say, 0.01, once off-line learning is finished. This would allow for the algorithm to make exploratory decisions every now and then, so that a shift in policies can be accomplished should the dynamics of the underlying service discovery system change. In other words, learning would always be taking place, allowing SDPP to become environment-adaptable.

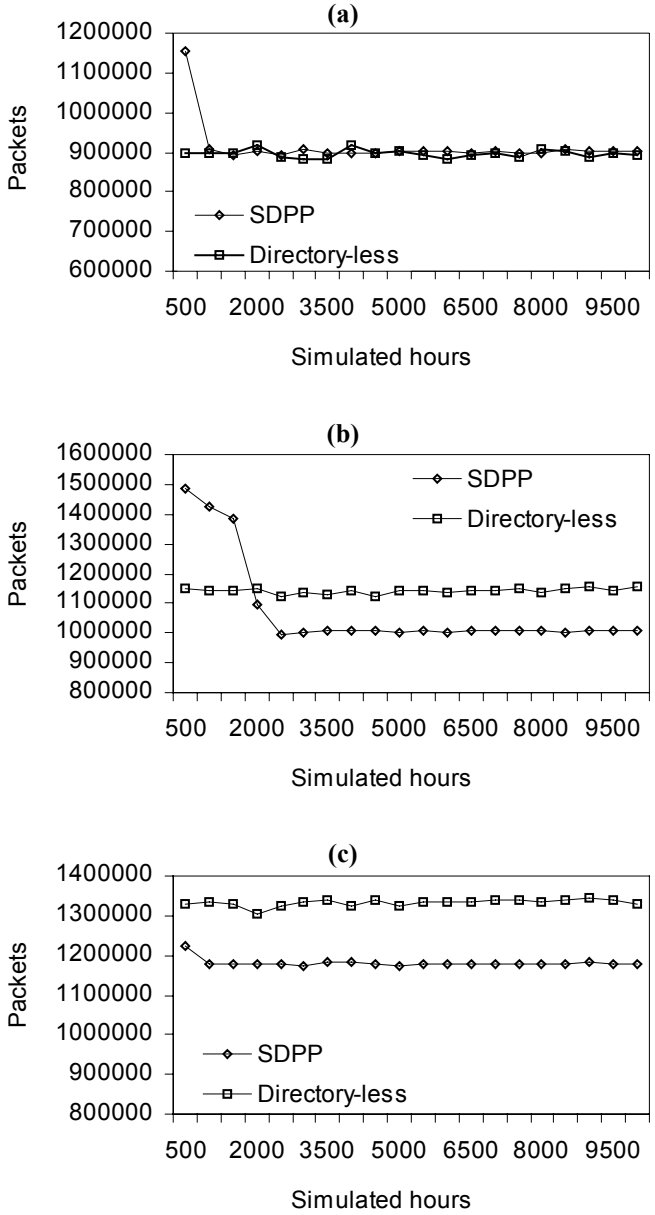

Fig. 6. SD packets per hour, averaged at 50-hour periods for a 25-host network with a host velocity of: (a) 0.5 m/s, (b) 1.0 m/s, and (c) 2.0 m/s.

TABLE I
NUMBER OF VISITED STATES FOR THE 15-HOST NETWORK

Velocity (m/s)	States Visited (15-host network)	States Visited (25-host network)
0.5	11808	5697
1.0	6911	11847
2.0	8060	5064

Our mobile-directory framework breeds a new set of research and performance evaluation challenges. We plan to introduce several improvements to this initial effort, such as the introduction of AODV instead of a generic data routing algorithm, and the use of a neural network-based method for function approximation instead of the table based-approach. An enhanced version of SDPP is currently being formulated, in which more than one mobile directory can be employed. As far as performance evaluations are concerned, we plan to further measure the usefulness of some of the provisions made by existing SDPs, but in the wireless ad-hoc networks realm.

Finally, the directory-forwarding mechanism that an existing SDP could employ if SDPP were being used could also have a measurable impact on its overall performance. For example, while Bluetooth's SDP would clearly rely on message passing to move a directory from host to host, Jini technology would employ Remote Method Invocation (RMI) and mobile objects for the same purpose. The mobile agent paradigm would also appear to be a suitable approach for this kind of task, since the directory as a query-processing device, along with its associated data, could move in an independent fashion. However, the effectiveness of each of these individual approaches remains yet to be explored.

REFERENCES

[1] El-Sayed, M. and Jafee, J. "A View of Telecommunications Network Evolution." *IEEE Communications Magazine,* Volume 40, Issue 12, December 2002.

[2] The Universal Plug-and-Play Forum, *http://www.upnp.org*.

[3] The Jini Community, *http://www.jini.org*.

[4] The Salutation Consortium, *http://www.salutation.org*.

[5] Service Location Protocol v2, IETF RFC2608, *http://www.ietf.org/html.charters/svrloc-charter.html*.

[6] Bettstetter, C and Renner, C. "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol." In *Proc. of the 6th EUNICE Open European Summer School: Innovative Internet Applications (EUNICE'00)*, Twente, Netherlands, September 2000.

[7] Rakotonirainy A. and Grooves. G. "Resource Discovery for Pervasive Environments." In *Proceedings of the 4th International Symposium on Distributed Objects and Applications*, October 2002.

[8] McGrath, R. "Discovery and its Discontents: Discovery Protocols for Ubiquitous Computing." *Presented at Center for Excellence in Space Data and Information Science*, *NASA Goddard Space Flight Center*, April 2001.

[9] Golden, R. "Service Advertisement and Discovery: Enabling Universal Device Cooperation." *IEEE Internet Computing,* September 2000.

[10] Cornuejols, G., Nemhauser, G. L., and Wolsey, L. A. "The Uncapacitated Facility Location Problem." In *Discrete Location Theory, Mirchandani, P., and Francis, R., editors*, P.119-171. John Wiley and Sons, New York, 1990.

[11] A compendium of NP optimization problems *http://www.nada.kth.se/~viggo/problemlist*.

[12] Qiu, L., Padmanabhan, V. N., and Voelker, G. M. "On the placement of Web server replicas", in *Proceedings of IEEE INFOCOM*, April 2001.

[13] Hara, T. "Replica Allocation Methods in Ad Hoc Networks with Data Update." *Mobile Networks and Applications (MONET)*, 8, 343–354, Kluwer Academic Publishers, 2003.

[14] Aiofi, M. et al. "Mobile Dynamic Content Distribution Networks." In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems* MSWiM'04, Venice, Italy, October 4-6, 2004.

[15] Puterman, M. L, "Markov Decision Processes." *Wiley Interscience*, New York, USA, 1994.

[16] Li, B. et. al. "On the Optimal Placement of Web Proxies in the Internet." In *Proceedings of IEEE INFOCOM*, P.1282-1290, March 1999.

[17] Watkins, C. J. C. H., "Learning from Delayed Rewards." PhD thesis, Cambridge University, Cambridge, England, 1989.

[18] Sutton, R. S. and Barto, A. G., "Reinforcement Learning – An Introduction." *MIT Press*, Cambridge, USA.

[19] Bradtke, S. J. and Duff, M. O. "Reinforcement Learning Methods for Continuous-Time Markov Decision Problems." In *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky and T. K. Leen, Editors, MIT Press, Cambridge, USA 1995.

[20] The OMNeT++ discrete event simulator environment: http://www.omnetpp.org.

[21] Bettstetter, C., Resta, G. and Santi, P. "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad-hoc Networks." *IEEE Transactions on Mobile Computing,* 2(3): 257-269, July-September 2003.