Service discovery in MANET via biased random walks *

Roberto Beraldi
Dipartimento di Informatica e Sistemistica (DIS) - Via Ariosto, 25, Roma (Italy)
University of Rome, "La Sapienza"
beraldi@dis.uniroma1.it

ABSTRACT

In this paper we tackle the problem of service discovery in mobile wireless networks by exploiting an informed random walk based search, which is implemented directly atop the data link layer. The algorithm can be used both to discover a service and route data packets back to the requesting node, without any routing layer support. The information driving a search consists of the estimated distance of nodes from a service and the previously visited nodes. The suitability of the approach is shown through simulations and supported by an analytical model.

1. INTRODUCTION

Service discovery is a key functionality to enable Service Oriented Architectures (SOAs). In a SOA environment, nodes make resources available to other participants in the network as independent services that the participants access in a standardized way [7]. In a mobile wireless network the proximity to a given service, as well as the kind and the number of services may vary unpredictably and abruptly with time, [10]. As a consequence service discovery in these networks is a real challenge.

Two main architectures have been proposed in the literature to address the problem of service discovery. The first one is to store all information about the services in one or more directory servers, e.g. Jini [9], UDDI [2] and Salutation [3]. The directory server announces its presence by periodically flooding the network (optionally within a given scope) with announcement messages. A service provider can then register its service via a unicast service registration message sent to a directory server, while a service is discovered by the client using a similar unicast request message sent to a directory. The deployment of several directory services, as for example proposed by Kozat and Tassiulas [13] or Sailhan and Issarny[6], is also an option. In this case however, some form of virtual topology is required, which can be difficult to maintain efficiently on top of a time varying one.

The second architectural solution to service discovery rests on a directory-less architecture model. In this case, services and clients directly discover each other by pulling or pushing services. Examples of architectures of this kind include SLP [8] and UPnP [4]. Hybrid architectures are also an option [5]. The main drawback of directory-less architectures is the potentially high overhead they can produce due to the independent discoveries. Recently, a field theoretical approach has also been proposed [14]. Service providing and service requesting nodes both are modelled as electrical charges with different signs. The algorithm is proactive, i.e., the field the charges generate is propagated periodically through network-wide broadcasts.

In this paper we face the service discovery problem in MANET (Mobile Ad Hoc Network) by exploiting a probabilistic approach. At the heart of the proposal there is the exploitation of an informed random walk. Unlike a pure random walk, in which all nodes forward a received message (the walker) to a randomly selected neighbor, an informed walk uses some form of information to bias the direction of the walker towards the target. The information we exploit is very simple and intuitive and it consists of the (estimated) distance of nodes from a service providing node.

The contribution of the paper is twofold. First, we present an analytical study of biased random walks running on a regular grid topology. This study is useful to understand the influence of biasing on the random walk performance. It was found that a bias as small as 0.65 is able to improve the performance considerably.

Second, we present a searching protocol based on a biased random walk algorithm tailored for mobile wireless networks. The aim of the protocol is to determine if a service is provided in the network. It uses a simple yet efficient selection mechanism running directly on the MAC layer. The selection of the next node leverages the broadcast nature of wireless transmissions; it assures a high reliability, efficiency and resilience to topological changes. Data packets, e.g., the reply from the provider to the searching node, could also be routed using the same protocol.

The rest of this paper is organized as follows. In the following section we present a simple model of an informed walk search, based on external oracle; in Section 3 we present the proposed protocol; the evaluation study is reported in Section 4 and conclusions are given in Section 5.

*This work is supported by the Italian Ministry of Education, University, and Research (MIUR) under the "IS-MANET" project and by the European Union under the "ReSIST" Network of Excellence (Grant number 026764).

AUTONOMICS 2007, 28-30 October 2007, Rome, Italy

DOI 10.4108/ICST.AUTONOMICS2007.2208

2. SEARCHING VIA A BIASED RANDOM WALK IN A REGULAR GRID

To gain some preliminary insight on the search performance of a biased random walk, we consider a lattice grid with degree four. This regular topology is simple enough to be treated mathematically while it can be considered a particular case of wireless network, e.g., it can be though as a snapshot of a mobile network which fits the Manhattan-like topology. The random walk is implemented by a packet generated by a source node and aiming at reaching a target node, say t, whose ID is carried in the packet. A step of the walk consists of a node sending the packet to one of its four neighbors. It is assumed that each node i knows the set $NL(i)$ of its current neighbors (lookahead one). This means that the packet hits the target the next step it reaches one of the target's neighbors. When the packet reaches a node i which is not a target's neighbor, i queries an oracle. The oracle tags the nodes in $NL(i)$ either as downstream or not (a neighbor j of i is said to be a downstream w.r.t. t when the distance of j from t is less than the i's one). The node then selects a downstream node at random. The oracle correctly tags a node in $NL(i)$ with probability ρ. We call such a random walk, random walk with bias ρ and one hop lookahead.

The aim of the analysis is to determine the expected number of steps that the packet generated at node s makes before it hits the target node t for the first time, hereafter called the hitting time h_{st}.

Hitting time.

Let a be the number of downstream nodes and b the number of not downstream ones, hereafter called upstream nodes. The probability the oracle correctly tags k_1 nodes as downstream and erroneously k_2 nodes is:

$$p_{SEL}(k_1,k_2,a,b)=\binom{a}{k_1}\rho^{k_1}(1-\rho)^{a-k_1}\binom{b}{k_2}(1-\rho)^{k_2}\rho^{b-k_2}$$

When $k_1+k_2>0$ nodes are tagged as downstream, the probability a node sends the message to some of them is $\frac{k_1}{k_1+k_2}$, while when $k_1+k_2=0$ this probability is $\frac{a}{a+b}$. The probability a node sends the message to some downstream is then:

$$p_{DWN}(a,b)=\sum_{k_2=0}^{b}\sum_{k_1=1}^{a}\frac{k_1}{k_1+k_2}p_{SEL}(k_1,k_2,a,b)+$$

$$\frac{a}{a+b}p_{SEL}(0,0,a,b)$$

The probability the message is sent exactly to a given downstream node is:

$$p_{dwn}(a,b)=\frac{p_{DWN}(a,b)}{a}$$

while the probability the message is sent to a specific upstream node is:

$$p_{up}(a,b)=\frac{1-p_{DWN}(a,b)}{b}$$

The hitting time can be derived by exploiting a Discrete Time Markov Chain (DTMC) with an absorbing state. Let the state of the chain represent the node currently processing the message (this mapping is straightforward if nodes are numbered from 0 to N^2-1), the absorbing state corresponds to the target node, the initial state to the requesting node. The transition matrix $\mathbf{P}$ is readily obtained from the probabilities determined above; the element P_{ij}, i.e., the probability that the message is sent by node i to node j is given by:

$$P_{ij}=\begin{cases}p_{dwn}(a,b) & j\neq t, j\in D(i), a=|D(i)|, b=|U(i)|\\ p_{up}(a,b) & j\neq t, j\in U(i), a=|D(i)|, b=|U(i)|\\ 1 & j=t, j\in U(i)\cup D(i)\cup\{i\}\\ 0 & \text{otherwise.}\end{cases}$$

where $D(i)$ ($U(i)$) is the set of downstream (upstream) nodes. Let $\pi_{i,j}^{(n)}$ be the probability that at time n the state is i, given that the initial state of the chain was j, and $\pi_j^{(n)}=[\pi_{0,j}^{(n)},\pi_{1,j}^{(n)},\ldots,\pi_{N^2-1,j}^{(n)}]$. We have $\pi_j^{(n)}=\pi_j^{(0)}\mathbf{P}^n$, where $\pi_j^{(0)}$ is a vector of all zero but one in position j. Hence the hitting time of node t starting from node s is:

$$ht_{st}=\sum_{k=1}^{\infty}k[\pi_{t,s}^{(k)}-\pi_{t,s}^{(k-1)}]$$

This is because $\pi_{t,s}^{(k)}$ is the probability that the state of the chain at time k is t given that at time $k=0$ it was s, i.e., the target was reached at a time $w\leq k$ starting from s, so that the term inside the square brackets is the probability that is was reached exactly at time k. The hitting time for a 20×20 lattice is reported in Figure 1. The searching node is placed at $(2,10)$, the target at $(17,10)$; similar results were obtained for other deployments of these two nodes. It is interesting to note how the hitting time sharply decreases with ρ. For a correctness of $\rho=0.65$, the hitting time passes from 1000 to 100, i.e., it is one order of magnitude less than the one of a natural random walk. Note that by using flooding 400 transmissions are required.

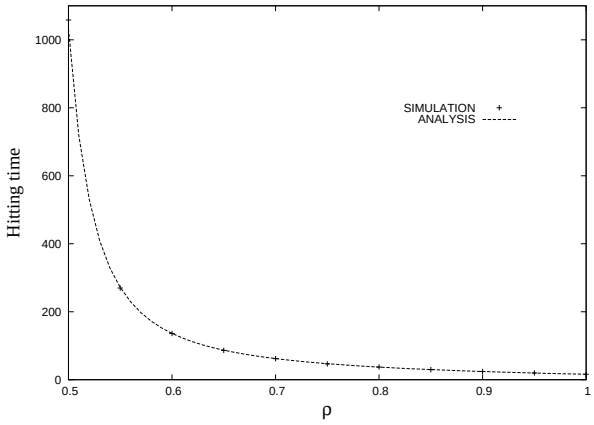


Figure 1: The effect of biasing on a 20×20 grid.

Directionality.

Another interesting performance metric can be obtained from the equation $\mathbf{N}=(\mathbf{I}-\mathbf{Q})^{-1}$, where $\mathbf{Q}$ is the fundamental matrix for the chain $\mathbf{P}$. The entry N_{ij} represents the expected number of times the chain will be in the state

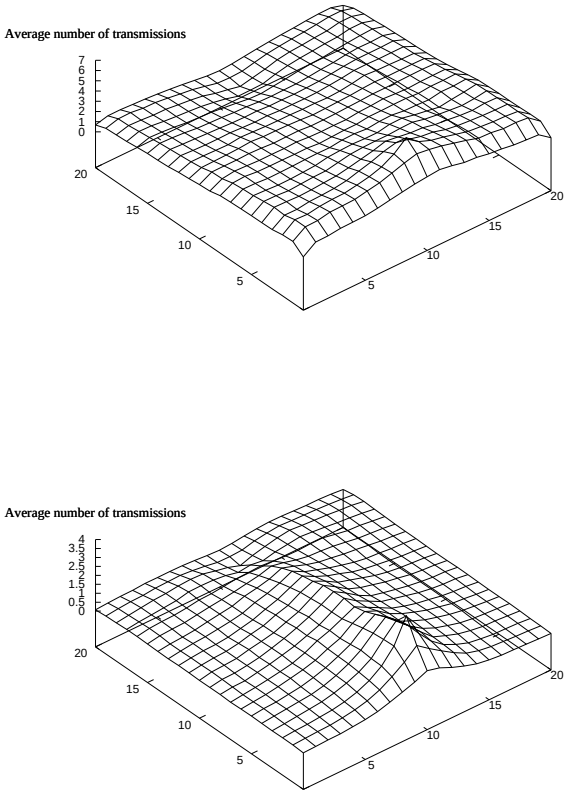


Figure 2: Average number of transmissions per node in a 20×20 grid. Top: no bias, bottom: bias $\rho = 0.55$.

j before absorbtion when it is started at i. For i equals to the id on the searching node, this value can be interpreted as the average number of transmissions of a node j before the target is reached.

Figure 2 reports such a quantity for $\rho = 0.5$ and $\rho = 0.55$ assuming the the target and source placed at the above positions. While in a natural random walk the packet visits almost isotropically the network, when a light polarization appears we can see how the nodes away from the shortest path send the message a fairly less number of times.

3. PROPOSED PROTOCOL

To implement a random walk with bias we need a method to implement the oracle. In our proposal, the functionality of the oracle is implemented by a mechanism for estimating the proximity of the neighbors of a node to the target, i.e., a service providing node. It works as follows.

Each service providing node, say node t, sends an advertising service packet containing the description of the service, say SRV, via a local broadcast every ΔT time units. A receiving node, say i, caches this information into a local service table. At time k, the proximity of i to t is

$$d_{it}(k) = \frac{k - last_{it}}{beacon_{it}}$$

where $beacon_{it}$ is the number of beacons received by i from t and $last_{it}$ the time of the last received beacon. If i has no entries for t into its cache then the proximity is ∞, while when $last_{it} > k - \Delta T$, i.e., the last beacon was received no earlier than ΔT time units ago, it is $d_{it}(k) = 0$. The proximity of i to a service SRV, $d_{iSRV}(k)$, is the lowest proximity of i to any node that announced SRV.

The beacon counter is reset if i starts hearing a beacon again after three missed beacons. The number of beacons is used as an estimation of the relative speed between i and t. Thus, the rationale of the above proximity metric is to assume the expected value of $d_{it}(k)$ be proportional to the expected distance of i from t. This relationship has been proved experimentally for the random way point mobility model and analytically for a simpler mobility model, see [11].

When a node j has to forward a packet searching for a service, j uses the proximity of its neighbors to the service for determining which of them has the highest chances of being a downstream node, and then it forwards the packet to it. More precisely, let m be the packet to be forwarded. Node j sends m to a node i via a three phases handshaking protocol, which also embeds the neighbor selection logic.

The selection/forward protocol is implemented atop the link layer (all packets are sent using a single local broadcast transmissions); it works as follows. When a node j has to forward m, it sends m via a local broadcast and then it waits for the first RTF (Request To Forward) packet received from a neighbor, say i. Upon receiving such a control packet, j broadcasts a short ack packet whose goal is to inform i that it is in charge of keep forwarding m and the other neighbors that the packet has been successfully forwarded.

A neighbor of j, say w, behaves as follows. On receiving m, w stores the packet locally and waits δt time units. If during this period of time w hears no other control packets from j, i sends an RTF control packet to it and waits for hearing the corresponding ack packet. When w hears the ack from j destined to it, it knows that it is the winner of the selection and thus it has to forward m using a new run of the handshake protocol. If the ack is not destined to w, w discards m. The packet is also discarded and no RTF packets is sent if w hears an ack during δt. Reliability in this selection/forwarding procedure is archived via time-outs.

The selecting node j selects the node from which the first RTF is received. The delay $\delta t_i(SRV)$ of node i must then be proportional to d_{iSRV} in expectation. This assures that the node with the lowest proximity is preferred to the others with a probability determined by its proximity. In order to also regulate the degree of such a proportionality, we propose to set $\delta t_i(SRV) = \tau \xi_{iSRV}$, where ξ_{iSRV} is a random variable with values in (0..1) and τ a constant representing the maximum delay. We propose to set the ξ_{iSRV}'s probability distribution function to a power law function, given by:

$$f_{\xi_{iSRV}}(x) = (\beta + 1)(1 - x)^{\beta / d_{iSRV}}$$

The parameter β of the function captures the believe in estimations and determines the sharpness of the distribution. If $\beta = 0$, the random variable is uniformly distributed, i.e., the

value of the proximity ignored. As β increases the node with the lowest proximity generates the lowest delay with a probability proportional to β. In this way, the node increases the chances of being selected.

Finally, in order to avoid loops nodes already selected to forwarded m cannot reply with the RTF packet again, while if $d_{iSRV}(t) = 0$ then $\delta t_i(SRV) = 0$.

Exploitation as a routing protocol.

By substituting the description of the service with the specific address of a node t, the same forwarding protocol can be used to route a packet to t. This is useful when the service providing node has to send a short reply back to the requesting node. Exploiting a routing protocol can in fact be not convenient in this case: if a reactive protocol is in place, sending the reply means to discover a path and then trigger a network flooding; a cost not amortized by the short amount of data.

4. EVALUATION

In this section we report some preliminary simulation results. To assess the suitability of our proposal we have conducted a simulation study by exploiting a custom discrete event simulator, already used in [11]. We have simulated 300 nodes moving according to the mobility models proposed in [12]. The first one is a round trip mobility model with waypoints (RWP) defined in a square shaped region with edge 2.5 km, speed selected in the range $[1..v_{max}]$ m/s and no pause time. The other mobility model is a particular instance of the random waypoint on a general connected area. The area represents a city (City model); the space graph is defined in an ASCII format that contains: road id, average road speed, coordinates of the road endpoints. We have used the code available at [1] with the data representing a real-world road map of a section of Houston close to Rice University. The area of the section is roughly a square with a side of length 1200 meters. The scenario consists of 383 intersections and 594 road segments, with mobiles moving with uniform distribution on the interval from 0.01 to 9.99 meters per second. Statistical data were collected for 3000 s and no warm-up period is adopted (mobility models are already at the steady state).

The simulator has the following main characteristics. Packet transmissions are governed by an ideal scheduler, a broadcast packet is served (i.e., initiated) after the channel is free for a Random Assessment Delay (RAD) randomly chosen in the range [0..500] ms, the packet reception is notified to a sender's neighbor provided that it remained for the whole duration of the transmission within the transmission range and such that no collisions with other transmissions occurred, a FIFO buffer of 20 packets in size is used at each node, packets are 512 bytes in length, the transmission bandwidth is 11 Mbps, the transmission radius 250 m and the beacon interval $\Delta T = 2.5$ s.

As far as the traffic is concerned, a service requesting message is generated by a randomly selected node at a rate of one request per second. A request is satisfied when any node providing the service is reached by the requesting message.

Numerical results and discussions.

The plot at the top of Figure 3 shows the hitting time (number of packet retransmissions) as a function of the polarization β. When $\beta = 0$ the algorithm behaves like a random walk with local memory, i.e., no nodes are used twice to forward the same message. This memory property alone is capable of highly reducing the hitting time (the hitting time in natural random walk was estimated to be roughly 90 steps). By coupling this restriction with biasing the hitting time is reduced to approximatively 25 steps for $\beta \approx 20$. As β is further increased the information provided by the proximity is not exploited, since the delay values after which nodes send their RTF packets is always very small (for $\beta \to \infty$ the delay is always zero and thus the selection becomes again at random).

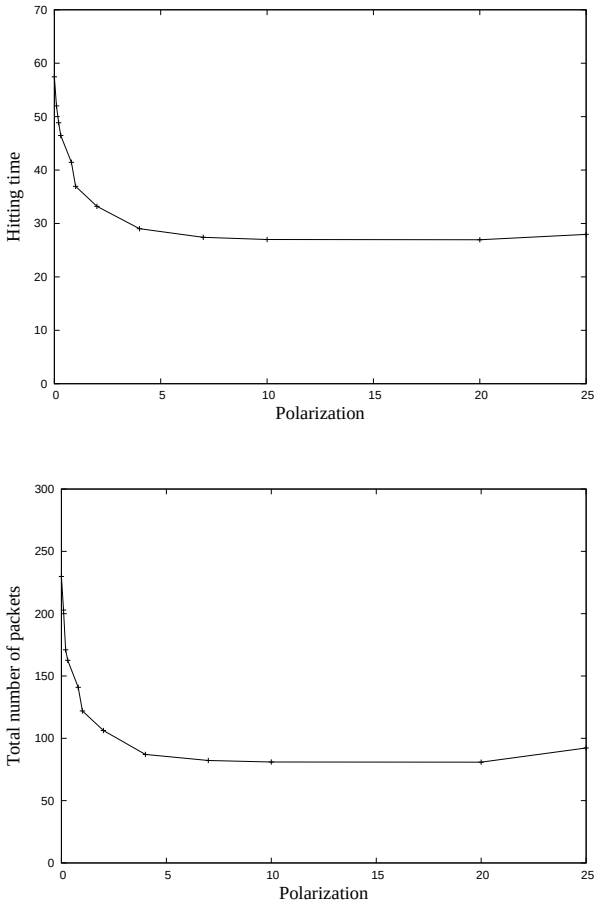
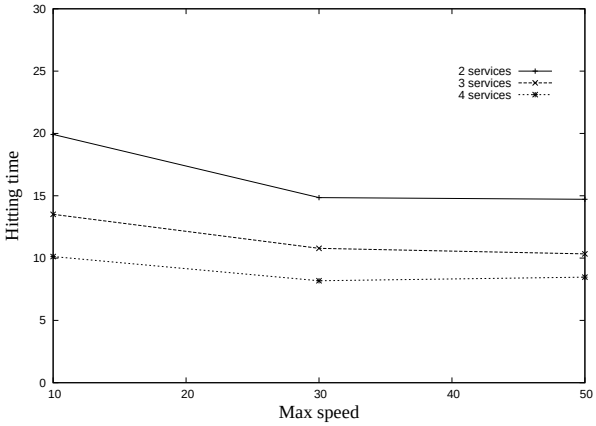


Figure 3: Effect of the polarization on the hitting time.

The other plot in 3 shows the total number of link layer packets sent (beacons are also included). For $\beta = 20$ approximatively 80 packets are required. Note that a flood search will require 299 link layer broadcast transmissions (the cost of flooding can be decreased using a gossip protocol; however, also in this case the cost of our search is lower[1]).

The impact of mobility on the search performance has been studied by varying the maximum speed, v_{max}, see Figures 4. In general, the effect of increasing the speed is

[1]In fact, the forwarding probability at each node must be higher than a critical threshold p_c, where $p_c \approx 0.6$.



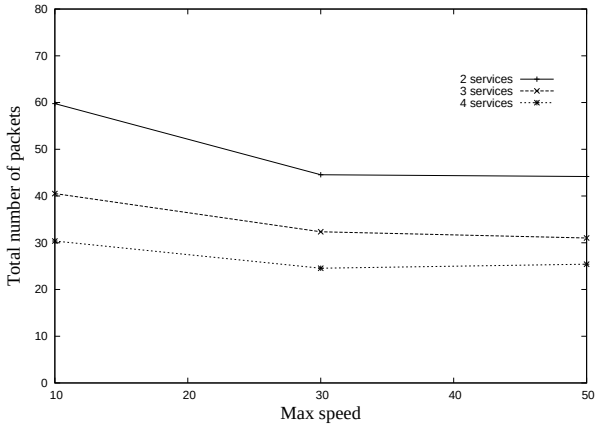
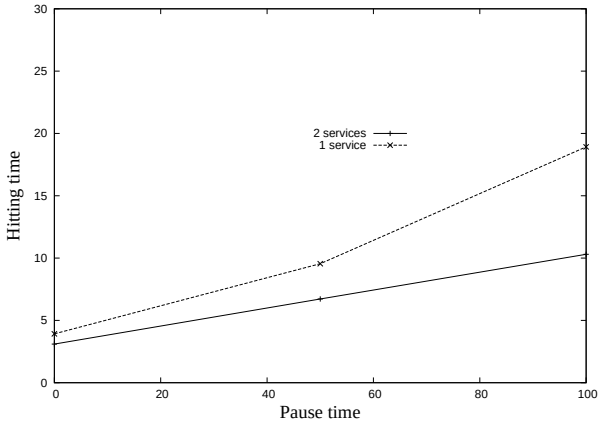


Figure 4: Hitting time vs maximum speeds (top) and related cost (bottom).



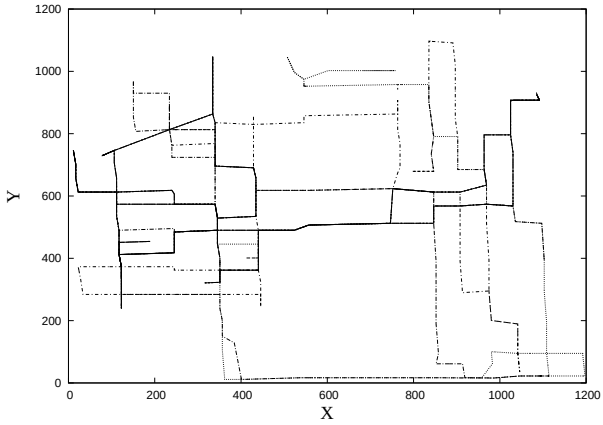


Figure 5: Hitting time as a function of the pause time for the city model (top); city map (bottom)

twofold. On one hand, a high speed helps nodes to come in contact with each other and thus potentially having fresh estimations of their proximity. On the other hand, as the speed is increased the validity of an estimation is shortened. Thus, when the speed is increased estimations are more precise but also for a shorter amount of time. The existence of both the effects discussed above explains why the overhead first increases with the speed and then decreases. This behavior is observed for any number of service providers. We can see how the overhead varies as the hitting time does and that it is roughly three times the hitting time. This simply because the implementation of a step of the walk requires almost always the transmission of three packets. The last set of experiments aimed at assessing the impact of the mobility model. Figure 5 reports the effects of mobility for the city model; mobility is regulated by varying the pause time. The hitting time is reported in the top of this figure. The bottom plot reports a map of the city (lines are traces of the mobile devices).

Under a low mobility level (high pause time) the estimation mechanism fails to provide useful information; this mainly happens since the service providing node meets only a few other nodes; thus, for the lack of information the random walk is forwarded at random. As the pause time is decreased estimations are available and useful to reduce the hitting time.

5. CONCLUSION

In this paper we have addressed the problem of service discovery in mobile wireless networks. Such a problem is very important to enable dynamic cooperation among software components. We proposed a probabilistic approach with its root in the exploitation of a random walk based search. To increase the search performance of a natural random walk, we have exploited the notion of proximity of nodes to a service. When a node meets another one providing a service, the proximity among the two after they are no longer neighbors is the ratio of the time elapsed since their link was broken with the duration of the link. This information can be effectively exploited to bias the selection of the next node of the walk and thus pushing a search toward a service providing node. The same protocol could also be used to send short amount of data packets, thus completely avoiding the need of any routing protocol. The presented simulation results show that the solution represents a very appealing alternative to the existing architectures. We are

currently considering possible optimizations of the protocol and further simulation assessments.

6. REFERENCES

[1] http://ica1www.epfl.ch/randomtrip/.

[2] Oasis Consortim. The universal description, discovery and integration (uddi). http://www.uddi.org.

[3] Salutation Consortium. Salutation architecture specification version 2.1. http://www.salutation.org, 1999.

[4] Microsoft. Corporation. Universal plug-and-play (upnp) forum. http://www.upnp.org., 1999.

[5] P. E. Engelstad and Y. Zheng. Evaluation of service discovery architectures for mobile ad hoc networks. In *Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services (WONSŠ05)*, St Moritz, Switzerland, 2005. IEEE.

[6] V. Issarny. F. Sailhan. Scalable service discovery for manet. In *Proceedings of the 3rd IEEE Int'l Conf.on Pervasive Computing and Communications ((PerCom 2005)*, Kauai Island, HI, USA, 8-12 March 2005. IEEE.

[7] G. Coulouris, J. Dallimore and T. Kindberg. *Distributed Systems, Concepts and Design (fourth edition)*. Addison Wesley, 2005.

[8] E. Guttman, C. Perkins, J. Veizades, M. Day. Service location protocol, version 2. RFC 2608, Internet Engineering Task Force (IETF), June 1999.

[9] Sun Microsystems. Jini network technology. http://www.jini.org, 1999.

[10] S. Nesargi and R. Prakash. Issues pertaining to service discovery in mobile ad hoc networks. *ACM Workshop on Principles of Mobile Computing*, 2001.

[11] R. Beraldi, L. Querzoni, R. Baldoni. A hint-based probabilistic protocol for unicast communications in manets. *Elsevier Ad Hoc Networs*, 2005.

[12] S. PalChaudhuri, J.-Y. Le Boudec, M. Vojnovic. Perfect simulations for random trip mobility models. In *38-th Annual Simulation Symposium*, San Diego, California, USA, April 2005.

[13] U.C. Kozat and L. Tallius. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, (2)2004.

[14] V. Lenders, M. May, B. Plattner. Service discovery in mobile ad hoc networks: A field theoretic approach. *Elsevier Pervasive and Mobile Computing*, 2005.