Accelerating Service Discovery in Ad-hoc Zero Configuration Networking

Se Gi Hong, Suman Srinivasan and Henning Schulzrinne
Columbia University, New York, NY
{segihong, sumans, hgs}@cs.columbia.edu

***Abstract*—Zero Configuration Networking (Zeroconf) assigns IP addresses and host names, and discovers service without a central server. Zeroconf can be used in wireless mobile ad-hoc networks which are based on IEEE 802.11 and IP. However, Zeroconf has problems in mobile ad-hoc networks as it cannot detect changes in the network topology. In highly mobile networks, Zeroconf causes network overhead while discovering new services. In this paper, we propose an algorithm to accelerate service discovery for mobile ad-hoc networks. Our algorithm involves the monitoring of network interface changes that occur when a device with IEEE 802.11 enabled joins a new network area. This algorithm allows users to discover network topology changes and new services in real-time while minimizing network overhead.**

I. Introduction

When mobile devices are intermittently connected to the Internet, information can be shared with peers using ad-hoc networking. These wireless mobile ad-hoc networks do not have any central servers, such as a DNS server and a DHCP server, though they primarily run on IP. For these reasons, mobile ad-hoc networks require Zero Configuration Networking (Zeroconf) [1]. Zeroconf provides for the assignment of IP addresses, host naming, and service discovery without any central servers or human adminstration. There are several applications that are designed for wireless mobile ad-hoc network and intermittently connected networks. These applications provide a variety of services. Service discovery is a vital part of those applications, as it allows those applications to automatically discover services as well as announce their services. Therefore, Zeroconf plays an important role in order for such applications to work properly.

However, Zeroconf suffers from problems in highly mobile ad-hoc networks. Thus, the frequency of devices joining and leaving a local ad-hoc network is very high. Network topology changes are not announced to other devices, and there is no algorithm in Zeroconf to detect these frequent network topology changes. When a device joins a new network area, the network resources of the node, such as IP address and host name in the area, may possibly conflict with the attributes of the existing members of that network.

Another potential problem is that services should be announced or browsed for frequently to be discovered in a highly mobile network, but this causes high network overhead. Bonjour [2], the most popular implementation of Zeroconf, uses an exponential back-off scheme to reduce overhead, but it causes slow detection as well as loss of discovery for new services.

We analyze the relationship between the interval of service browsing, average residence time of devices in a local ad-hoc network, and the probability that new services announced by new joining peers are not discovered. We then propose a new algorithm that accelerates the service discovery protocol and allows devices to discover network topology changes and new services in real time while minimizing network overhead. In our algorithm, each device can detect whether it has joined a new network area. Monitoring changes to the network interface allows a device to detect the network topology changes. If a node detects that it has joined a new network, it resolves possibly conflicting of IP address and host names, and announces or browses for services.

The remainder of this paper is structured as follows. In section II, we introduce Zeroconf. Section III discusses potential problems of Zeroconf in mobile ad-hoc networks. Section IV describes our new service discovery algorithm. Section V describes implementation and performance of our new algorithm. Finally, we describe related work in Section VI.

II. Overview of Zeroconf

The IETF Zero Configuration Networking (Zeroconf) Working Group [1] has proposed four main requirements [3] for Zeroconf: IP interface configuration, translation between host name and IP address, IP multicast address allocation, service discovery. Zeroconf specification is complete about IP interface configuration, but is not complete about the other requirements. In January 2007, Link-local Multicast Name Resolution (LLMNR) [5] was approved as an informational RFC by the IETF to address the name resolution requirement. Currently, LLMNR is only implemented on Windows CE. Bonjour [2] is one of most popular implementations of Zeroconf, and it satisfies the requirements of Zeroconf.

A. IP interface configuration

The configuration of IP address and netmask without a central server is key to ad-hoc networking operations. Zeroconf specifies the use of the IPv4 link-local address [4], which includes address selection and address conflict resolution. After a host randomly selects an IP address within the 169.254/16 subnet, it announces this address by broadcasting ARP announcements in the local area to detect possible IP address conflicts. If the host receives an ARP response from

another host which already uses this IP address, it will select another random IP address. Windows and Mac OS implement the auto-configuration of IPv4 link-local addressing. Linux and a few other operating systems will require installation of a daemon that implements link-local addresses.

B. Translation between host name and IP address

Since a host name is much more user-friendly than an IP address, Zeroconf also requires auto-configuration of a host name without a central server. The Zeroconf host name protocol allows host names to be mapped into IP addresses and vice versa. In addition, it resolves naming conflicts. Bonjour has proposed Multicast DNS (mDNS) [6] for host naming in a local area network. Multicast DNS allows translation between names and IP addresses and provides DNS-like operation without a central DNS server.

In mDNS, each device first selects their own local host name. The form of the name is "single-dns-label.local.", and the ".local." means that it is link-local and meaningful only on local network. The naming conflict resolution is similar to IPv4 link-local address conflict resolution. A host announces the name in local area by multicasting to detect possible name conflicts.

C. IP multicast address allocation

An IP multicast address allows the efficient use of bandwidth since all multicast receivers can receive multicast packets. The range of IP multicast address is from 224.0.0.0 to 239.255.255.255. Bonjour uses the mDNS multicast address, 224.0.0.251, as its IP multicast address [6].

D. Service discovery

Zeroconf allows users to discover services and choose the services without knowing the location of the service provider in advance to communicate with the providers. Bonjour has proposed multicast DNS-Based service discovery (mDNS-SD) [7] for service discovery for Zeroconf. This service discovery protocol allows users to find all service instances of a particular service type and to resolve IP address, port number and additional information for the service using PTR, SRV and TXT records. The PTR lookup of the format ⟨Service⟩.⟨Domain⟩ is used to obtain all service instances which have the same ⟨Service⟩.⟨Domain⟩. The name of a service instance is of the format ⟨Instance⟩.⟨Service⟩.⟨Domain⟩.

The SRV record provides the port number and IP address of the service provider. TXT records are used for storing additional information about services as attribute-value pairs.

III. Zeroconf in Mobile Ad-hoc Networks

A. Network topology changes

Since the topology of mobile ad-hoc networks changes frequently, network resources such as IP addresses and local host names should be announced to detect possible conflicts in the new network area [3]. However, Zeroconf does not include an algorithm to detect network topology changes. Furthermore, Zeroconf does not specify an algorithm to resolve possible conflicts during the topology changes.

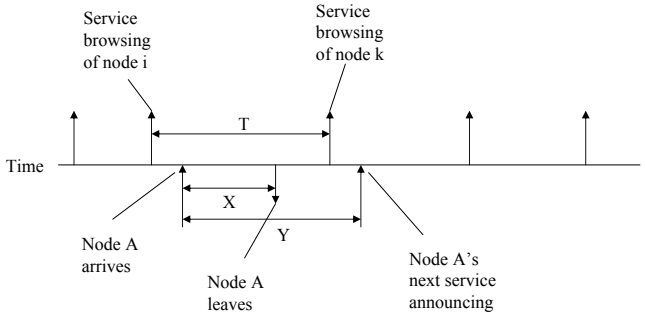


Fig. 1. Mobile node joins and leaves a local network

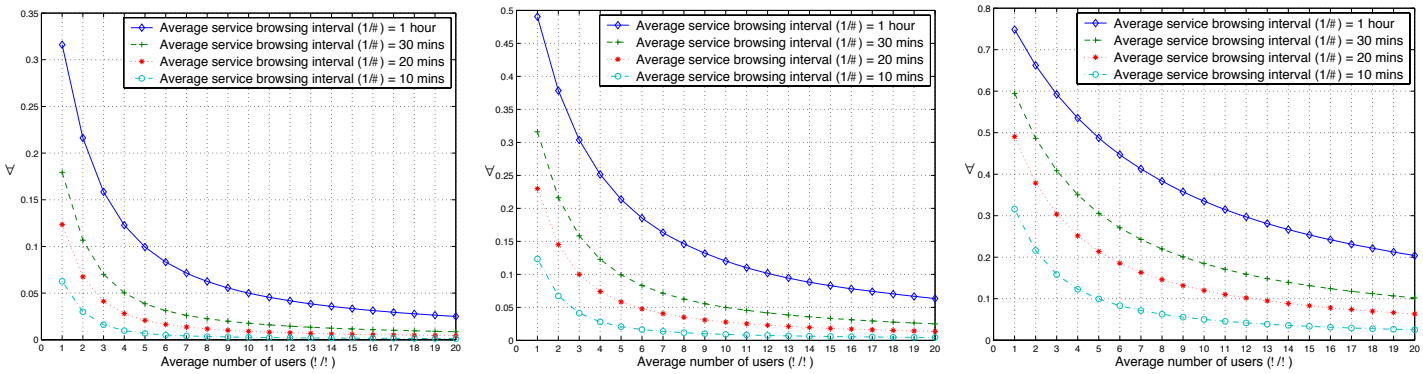
B. Network traffic overhead

In highly mobile networks, devices frequently join and leave a local network. In this scenario, if the services of mobile nodes are not frequently announced or looked up, the services might not be discovered by other mobile nodes. Thus, services should be announced and looked up frequently. However, this causes high network overhead. We analyze the relationship between the probability that new services announcements are not discovered, average residence duration of devices in a local area and the average interval of service announcing or browsing in this section. We assume that the service announcing and browsing interval follows an exponential distribution with rate α; the node arrival rate to a local ad-hoc area follows an exponential distribution with rate λ; the node residence time follows an exponential distribution with rate μ; the number of nodes in a current local ad-hoc area is n. For simplicity, a new node A announces a service, and other nodes browse the service.

Fig. 1 shows a mobile node joins and leaves a local ad-hoc network with Zeroconf. In Fig. 1, T is the interval between the last service browse request before a new node A arrives and the next service browsing request. T is min$\{T_1, T_2, ...,T_n\}$, where T_k is the interval between the next service browsing request time of node k and the last service browsing time of node i, where i is the last service browse request before node A arrives and k is the first service browse request after node A arrives. X is the residence time of node A. Y is the interval between node A's next service announcement time and its arrival time.

The service loss probability, $Q(n)$, is the probability that node A cannot hear any browsing requests and does not announce services during residence in a local area.

$$\begin{aligned} Q(n) &= \Pr(\mathrm{T} > \mathrm{X})\Pr(\mathrm{Y} > \mathrm{X}) \\ &= \int_0^\infty \int_0^\infty \Pr(\mathrm{T} > \mathrm{X})\Pr(\mathrm{Y} > \mathrm{X})\mathrm{dxdy} \\ &= \left(\frac{\mu}{n\alpha + \mu}\right)\left(\frac{\mu}{\alpha + \mu}\right). \end{aligned}$$



(a) Average residence time $(1/\mu)$ is 60 minutes (b) Average residence time $(1/\mu)$ is 30 minutes (c) Average residence time $(1/\mu)$ is 10 minutes

Fig. 2. The mean service loss probability of new services

The mean service loss probability, θ, is the mean of $Q(n)$.

$$\theta = \sum_{n=0}^{\infty} Q(n)\Pr(\mathrm{N}=\mathrm{n}) = \sum_{n=0}^{\infty} \left(\frac{\mu}{n\alpha+\mu}\right)\left(\frac{\mu}{\alpha+\mu}\right)\left(\frac{\left(\frac{\lambda}{\mu}\right)^n}{n!}\right)\exp\left(\frac{-\lambda}{\mu}\right).$$

Fig. 2 shows the result of the analysis. The figure plots the mean service loss probability, θ, against the average number of users. The average number of users affects the mean service loss probability. If there are many users in the local network area, the mean service loss probability decreases. It is reasonable since the browsing requests are not synchronized between devices, so a new peer has more chances of receiving browsing requests when there are more users. We can observe from Fig. 2 that as the average residence time of a mobile node decreases and the average service browsing interval increases, the service loss probability, $Q(n)$, increases. We are interested in the proper average service browsing interval in the case of a highly mobile environment, such as devices in cars on a highway. As we can see in Fig. 2(c), if average service browsing interval is one hour and the average number of users is four, the mean service loss probability is above 0.5. Therefore, to reduce the probability, service browsing should be made more frequent. If the average service browsing interval is 10 minutes, the probability decreases to around 0.1. This, however, increases network traffic.

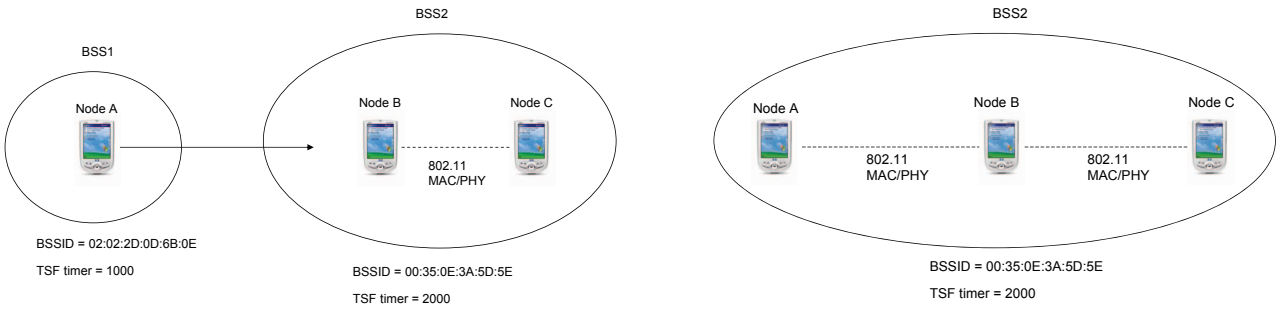
There are two phases in discovering services. One of them is the announcing of new services, and the other is browsing for and resolving the services. If there are a lot of service announcements, or if service browsing is done frequently, then services can be discovered with higher probability. As we can see in the result of the analysis, the services should be announced or browsed for frequently in order to be discovered in highly mobile networks. However, this causes high overhead. Bonjour uses an exponential back-off scheme [6] to reduce overhead. The intervals between the subsequent service queries are doubled; the minimum interval at startup is 1 second, and it increases exponentially till it reaches the maximum interval of 1 hour, and it remains constant after that. This exponential back-off scheme can reduce network overhead, but it causes slow detection of services or loss of chances to detect services that arriving peers may have.

IV. Accelerating Service Discovery in Zeroconf

A. Mobile device joining in IEEE 802.11 ad-hoc networks

The IEEE 802.11 standard [8] specifies when to generate beacon frames in ad-hoc mode. In ad-hoc mode, all nodes participate in generating beacon frames. Each node maintains `aBeaconPeriod` which specifies the length of a beacon interval, and its value is typically 100 ms. `aBeaconPeriod` is same for all nodes in the same Basic Service Set (BSS). The beacon period is included in beacon frames and probe response frames. When nodes join the local ad-hoc network, they adopt the beacon period. All nodes maintain their own time synchronization function (TSF) timer for `aBeaconPeriod`. The value of the TSF timer is an integer with modulus 2^{64}. When a network interface starts, the value of the TSF timer is set to zero and incremented every microsecond. The TSF timer value of all nodes in the same BSS is the same. In ad-hoc mode, each node has a random backoff timer. It chooses a random delay uniformly distributed in the range between zero and twice `aCWmin` $\times$ `aSlotTime`. The default values of `aCWmin` and `aSlotTime` are 31 and 20 μs, respectively. If a node receives another beacon frame before the timer expires, it cancels the remaining random delay as well as the pending beacon transmission. If no beacon arrives during the random delay, a node sends a beacon frame. Thus, there is only one beacon frame in each `aBeaconPeriod` in the same BSS.

To join a network, each node scans all the channels. There are two ways of scanning channels: passive scanning and active scanning. In passive scanning, a node listens to all channels by hopping each channel. A node listens to beacon frames which contain a desired Service Set Identification (SSID). In active scanning, a node generates probe request frames which contain a desired SSID. The node that generated the last beacon frame generates the probe response frame for the response of the probe request frame if the value of SSID in



(a) Before joining

(b) After joining

Fig. 3. Mobile node joins the network

the probe request is a broadcast address or matches the address of its own interface. Beacon frames and probe response frames contain the TSF timer value in the timestamp field and BSSID. When a node receives a beacon frame or a probe response frame, it adopts the information, such as the BSSID and the TSF timer, only if the SSID in the frame matches and the value of the timestamp is later than the node's TSF timer. Therefore, the BSSID and the TSF timer (of the node which receives the beacon frame) become the same value as that of the node which sends the beacon frame.

Fig. 3 shows a new device joining a local ad-hoc network. In Fig. 3(a), a mobile node A in area BSS1 moves to area BSS2, and it receives a beacon frame from a node B. The beacon frame contains the BSSID and the TSF timer value of BSS2. Before node A joins BSS2, node A's BSSID is 02:02:2D:0D:6B:0E and the TSF timer value is 1000. The BSSID value of all nodes in BSS2 is 00:35:0E:3A:5D:5E and the TSF timer value is 2000. Fig. 3(b) shows node A joining BSS2. After joining BSS2 networks, node A's BSSID becomes 00:35:0E:3A:5D:5E. The TSF timer value of node A becomes 2000.

B. Accelerating service discovery in mobile ad-hoc networks

In order to accelerate service discovery, we detect changes in network topology. When a node detects that it has joined a new network area, it re-triggers the service discovery protocol. This enables mobile devices to discover services in real-time. In addition, the algorithm can reduce network overhead because devices do not need to announce or browse for services periodically and frequently. As we can see in Fig. 3, when two ad-hoc networks are merged, one of the networks changes its BSSID value. We can check this BSSID value to detect if the node has joined a new network.

After a node detects that it has joined a new network, it announces or browses services to support real-time service discovery. If a node has services, it announces the services in the area. If a node is interested in some services, it browses for the services. Furthermore, network resources, such as IP addresses and local names, are announced in order to avoid possible conflicts in the new network area. In our algorithm, a mobile node announces or browses services in two cases:

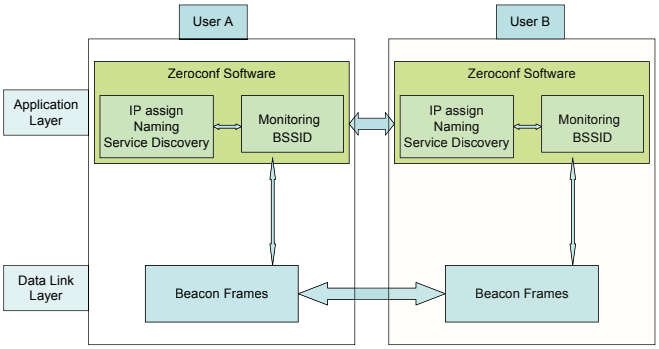


Fig. 4. Accelerating service discovery architecture

when it starts Zeroconf and when it detects that it has joined a new network area. Therefore, our new algorithm minimizes overhead and new services can be detected by other devices in real-time.

Fig. 4 presents the architecture of our service discovery system. Beacon frames in the data link layer can be used to detect new devices, and monitoring the BSSID value at the application layer allows mobile devices to detect network topology changes and notify the changes to the service discovery system for announcing or browsing services and resolving any possible conflict of IP addresses and host names in the new network area.

V. Implementation and Performance

In Linux, we can read the BSSID value of a wireless network interface using the standard wireless extensions for Linux and the Wireless LAN API for Windows. In Linux, we using the `ioctl()` function with the SIOCGIWAP option which enables us to get the interface's BSSID value. On Windows, we can use the `WlanGetNetworkBssList()` function.

We used three Linux laptop computers to test our service discovery system. Our test modeled a situation where a laptop computer A joins a local ad-hoc network which already contains two laptop computers B and C. Computer A announces a service and the other two computers browse for the service.

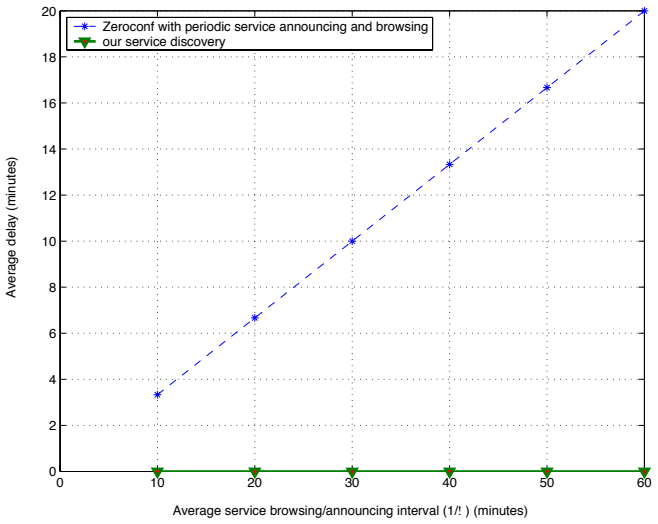


Fig. 5. Delay measurement of the service discovery protocol

The result of our testing shows that whenever computer A joins the local ad-hoc network, the other two computers can properly discover the service that computer A has.

We also measured the service discovery delay. The delay represents the period between the moment when a node joins a network and the moment when that new node's service is discovered. We compared the result with an analysis of the delay of Zeroconf with periodic service announcing and browsing. In Zeroconf we assume that all service announcing and browsing interval follows an exponential distribution with rate α. D is the delay. t_A, t_B, and t_C is the next announcing or browsing time of computer A, B and C.

$$\begin{aligned} \Pr(\mathrm{D} > \mathrm{t}) &= \Pr(\min(\mathrm{t_A}, \mathrm{t_B}, \mathrm{t_C}) > \mathrm{t}) \\ &= \Pr(\mathrm{t_A} > \mathrm{t})\Pr(\mathrm{t_B} > \mathrm{t})\Pr(\mathrm{t_C} > \mathrm{t}) \\ &= e^{-3\alpha t}. \end{aligned}$$

Fig. 5 shows that the delay versus the average service announcing and browsing interval. As we can see in Fig. 5, the delay of Zeroconf is increased as the mean service announcing and browsing interval is increased. However, the delay of our algorithm is not affected by the average service interval. It means that services do not need to be periodically and frequently announced or browsed in our system. Furthermore, the delay in our system is negligible. It is because we use an algorithm that can detect network topology changes.

VI. Related work

There are many papers about service discovery protocols in mobile ad-hoc networks, but almost all of them are concerned about scalability, building service discovery support in the network layer, and routing problems [10][11][12]. In contrast, our objective is to determine how to discover new services in real-time while minimizing overhead.

BEAD [13] also uses beacon frames to get information about its neighbor nodes. However, it uses the information for routing tables. Thus, BEAD is not related to service discovery supported by Zeroconf. BEAD does not properly implement reading beacon frames at the application layer. In the model, each node can only communicate with other nodes through a forwarding node. This differs significantly from our system.

VII. Conclusion

Zeroconf allows a local ad-hoc network to work on IP networking without central servers. However, it has some potential problems in mobile ad-hoc networks. Zeroconf does not support detecting network topology changes. Because of the frequent changes of topology, services need to be announced or browsed frequently to be properly discovered. However, it causes network traffic overhead. We have proposed a new algorithm to accelerate service discovery, especially in a highly mobile and low density network environment. It allows mobile users to detect network topology changes and new services in real-time while minimizing network overhead. Since in the service discovery systems, services are announced and browsed only if Zeroconf is initially started or there are network topology changes, it can reduce network overhead. Since it supports detecting network topology changes, new services can be discovered in real-time.

References

[1] Zeroconf working group. [Online]. Available: http://www.zeroconf.org/
[2] Bonjour home page. [Online]. Available: http://developer.apple.com/networking/bonjour/
[3] A. Williams. Zeroconf IP Host Requirements. *draft-ietf-zeroconf-reqts-10.txt.* February 2002. Work in Progress.
[4] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927. May 2005.
[5] B. Aboba, D. Thaler and L. Esibov. Link-Local Multicast Name Resolution (LLMNR). RFC 4795. January 2007.
[6] Stuart Cheshire and Marc Krochmal. Multicast DNS. *draft-cheshire-dnsext-multicastdns-06.txt.* August 2006. Work in Progress.
[7] Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery. *draft-cheshire-dnsext-dns-sd-04.txt.* August 2006. Work in Progress.
[8] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std. 802.11–1997. pp. 123-151 New York, New York, 1997.
[9] IEEE Computer Society LAN MAN Standards Committee. IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture IEEE Std. 802–1990. pp. 23-27 New York, New York, 1990.
[10] S. Motegi, K. Yoshihara and H. Horiuchi. Service Discovery for Wireless Ad Hoc Networks. In *Proceedings of the 5th International symposium on Wireless Personal Multimedia Communications (WPMC),* Honolulu, Hawaii, October 2002.
[11] U. C. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Proceedings of the IEEE INFOCOM,* San Francisco, April 2003.
[12] J. Tyan and Q. H. Mahmoud. A Comprehensive Service Discovery Solution for Mobile Ad Hoc Networks. In *Mobile Networks and Applications (MONET), Volume 10, Pages 423 - 434,* Aug. 2005.
[13] L. Raju, S. Ganu, B. Anepu, I. Seskar and D. Raychaudhuri. Beacon Assisted Discovery Protocol (BEAD) for Self-Organizing Hierarchical Ad-Hoc Networks. In *Proceedings of the IEEE GLOBECOM,* Dallas, Texas, Dec. 2004.