Distributed Naming System for Mobile Ad-Hoc Networks

Xiaoyan Hong, Jun Liu and Randy Smith
Computer Science Department,
University of Alabama, Tuscaloosa, AL 35487
Yeng-Zhong Lee
Computer Science Department,
University of California, Los Angeles, CA 90095

Abstract*— Mobile ad hoc networks (MANETs) are self-organized networks envisioned to deploy with zero (or less)-configuration effort in places where infrastructure networks are not available. In such a network, connectivity is enabled by automatic IP address allocation and dynamic routing. MANETs are highly dynamic with members joining and rejoining, leaving, moving around freely, network partitioning and merging, dynamic address allocation according to location closeness or according to host's logical affiliation. These dynamics lead to potential frequent changes of a node's IP address. Therefore, application programs are more likely to use well known names (Domain Names) that are unique, easy to remember and/or that bear logical semantics of the parties. On-the-scene translation from a node's well known name to its newly assigned network address faces new challenges in MANETs, which obsoletes legacy Internet Domain Name Systems. In this paper, we present a distributed naming system that provides service robustly in the presence of mobility and node failures. We evaluate the proposed scheme through simulation and compare it with a centralized ideal naming system.

Keywords – Mobile Ad Hoc Networks, Domain Name System, Clustering, Fault Tolerance.

I. Introduction

Mobile ad hoc networks (MANETs) are self-organized networks envisioned to deploy with zero-configuration effort in places where infrastructure networks are not available. In such a network, connectivity is enabled by automatic IP address assignment [1], [2], [4], [5] and dynamic routing [13], [14], [15]. MANETs are highly dynamic with members joining and rejoining, leaving, moving around freely, network partitioning and merging [3], dynamic address allocation according to location closeness [6] or according to host's logical affiliation [16]. These dynamics lead to potential frequent changes of a node's IP address. Therefore, communications among network users are more likely using well known names (Domain Names) that are unique, easy to remember and/or that bear logical semantics of the parties. Translation from a node's well known name to its newly assigned network address, which may indicate its current network location or reflect a routing hierarchy, has to be made before a real connection can be established.

When an ad hoc network is almost static and is small in scale, a straight forward adaptation from Internet Domain Name System (DNS) and manual configuration would be sufficient. However, this research recognizes that many MANET applications are large in terms of mobile hosts involved and complex in terms of the tasks performed. In this context, traditional static hierarchical domain name server structures (corresponding to Internet domain name hierarchy) faces challenges in maintaining the hierarchy in a dynamic environment. Manual configuration of query paths among distributed databases does not work either.

We identify the following requirements for name services concerning the unique features of MANETs. First, MANET name system should be able to support name conventions that either reflect logical organizational affiliation or a totally unstructured name. Second, MANET name system must be able to deal with frequent topology changes and node failures. Redundancy is considered an advantageous feature that provides fault tolerance to the naming system. Third, low overhead and load balance are highly desired to sustain the resource limited (bandwidth, battery power) mobile devices. In this paper, we present a *MANET Distributed Naming System (ADNS)*. ADNS supports unstructured name convention. It includes a distributed server system that allows redundancy for fault tolerance, a lookup scheme that reduces query overhead and balances the query/response load, and a server maintenance scheme that enables the server overlay network to evolve with mobility.

Without a naming service, resolution could use brute force flooding issued by either the node that experiences an address change to announce the newly obtained address; or,

Yeng-Zhong Lee was supported by ONR "MINUTEMAN" project under contract N00014-01-C-0016.

by the source that requires a resolution of the destination's name. Address changes can also be handled using Mobile IP. However, how to locate a node's home agent becomes another non-trivial problem. Zhou *et al* [10] tried to avoid Mobile IP but used an address handoff scheme for MANET. In the scheme, two IP addresses are bound to a network interface so that the new address can be used for route reestablishment while the old address can maintain on-going connections. In other research, design of naming services for MANET is tied to routing protocols. Existing work [11], [12] use hierarchical routing protocols associated with hierarchical address schemes indicating nodes' network positions in the routing hierarchy. In such a network, when mobility causes network topology change, it directly results in the change of a node's position in the routing hierarchy and hence its address. Thus address lookup for a specific identifier is frequently made and typically occurs in several recursive steps. Each of the steps moves the query to a closer intermediate node for final address resolution.

Two recently proposed name systems can be used for MANETs. Linklocal Multicast Name Resolution (LLMNR) [7] is designed for local links and supports small networks including ad hoc networks. However, it serves only as a secondary name resolution mechanism to DNS for scenarios where a conventional DNS name resolution is not possible. Thus it is not feasible for MANET as a primary use. Multicast based MANET Name Directory Service (NDR) [8], [9] restricts name resolution in a link-local domain or a site-local domain for IPv6. The scheme requires nodes to join a special multicast group. A name can be resolved by sending a query along the tree. When the site-local domain is the whole network (most likely the case for MANET), a query will be flooded over the entire network.

Different from the above work, we are concerned about the influence of a name system on network performance. Facing the aforementioned challenges, our design features dynamically elected distributed server system, a load-balanced registration and lookup protocol, and mechanisms handling server mobility. The rest of this paper is organized as follows: Section II gives the design of the distributed name system. It includes server election, registration and lookup protocols and system maintenance in dealing with network dynamics. The name system is then evaluated in Section III. Section IV concludes the paper.

II. Distributed Naming System for MANETs

A. Network Assumptions

We assume each network member has a well known name as its unique identifier. This well known name may or may not contain hierarchical organizational structure. In our design, the names are considered to be flat. The argument is that a MANET is a highly mobile and dynamic network, hierarchical names will not necessarily help name resolving like Internet DNS. Before a node joins the network, it is preloaded with all the well known names in the network. Or at least, it is preloaded with a set of well know names that it wishes to communicate with during its network operation lifetime. It is also possible that a search engine in a MANET provides URL information leading to an unique host name. With this name, any node will be able to find a destination's IP address by querying a name server. Our naming service will allow registrations and updates of $< name, address >$ bindings as well as lookup requests.

For the mobile network, we assume that the frequency of address change is far less than the frequency of the occurrence of communication sessions. That is to say, the network traffic is more intense compared to the frequency of topology changes. Also we assume that the time for all the servers to converge to a new $< name, address >$ binding is far less than the time a binding stays stable. These two assumptions are considered reasonable in the sense that nothing can be really accomplished in a network that changes too quickly.

B. Overview

The *MANET Distributed Naming System (ADNS)* provides registration and lookup service for each node about its current IP address. It includes a distributed server system that allows redundancy in its repository of binding records, a lookup scheme that tries to reduce the query overhead, and a server maintenance scheme where the server structure evolves with mobility. The design is distributed in order to avoid single-point-of-failure in a mobile environment and also to balance the query/response load. Multiple name servers are chosen to store an individual name-to-IP address mapping. In other words, node A's $< name, address >$ binding is maintained at a small subset of the current stable name servers, which are referred to as A's name servers. Different names have different subset of servers from all the name servers. So a query can be resolved using a nearby name server. The redundant repository also provides fault tolerance to query failures, namely, when a query fails at one server, a node can be looked up at another server. The servers that an address registration or lookup is made to are determined completely based on A's well known name. The components of ADNS are described below, they include server selection, indexing structure, name hashing and server selection, address registration and query, and data repository maintenance.

C. Name Servers, Registration and Query

1) Elected Name Servers: Members in MANET are dynamic both temporally and spatially. There either lacks a way of predesignating a set of nodes as servers or it is not feasible to permanently assign a node as a name servers. Our name servers are selected among the network members through clustering algorithms such as the ones presented in [19], [20], [21]. The cluster heads server as name servers and announce themselves to the entire network. Particularly, we use k-hop Random Competition based Clustering [22] for a more stable set of servers. [22] has shown that the algorithm is more stable than ID-base (Lowest ID) and connectivity-based (Highest Degree) algorithms. The elected

servers are distributed among all the nodes and dynamically adapt to network topology changes. Running on top of a proactive MANET routing protocol, the clustering messages are propagated by piggybacking in other routing messages, so overhead is minimized.

2) Indexed and Segmented NS Table: Each node maintains a name server table (NS table) for all the name servers upon receiving server advertisements. The table is sorted in descending order based on the names of the servers and indexed. Thus, all the nodes see the same NS table. Like a routing table, an entry in the NS table stores a server's name, IP address and next hop towards it through a shortest path.

The NS table is then segmented for redundancy. Let l be the degree of redundancy, i.e., a name record will be stored at l servers. Usually, l can be small. So there will be l segments in the table. If the NS table has a size of n, each segment has $m = \lceil n/l \rceil$ name servers ($m < n$). Please note that the l^{th} segment may not have a full size of m name servers. However, this fact will affect only the redundancy of a part of the name records stored at several servers, but not the operation of the name service. m effectively gives the actual number of name servers used for distributing naming service load.

3) Name Hashing and Server Selection: To deal with the changes of the servers, we use the index of a NS table for server selection. That is, a host name is always corresponding to a sever in a particular location in the NS table through hashing. In order to provide redundancy, several indexes will be used for a single name. The servers at the indexed positions are the name servers of the node.

A node N's well known name is hashed into an integer in the range of [0 , m-1], say, $i = H(N)$, where H is the hash function, and $0 <= i <= m-1$. Function H can be as simple as $i = Val(N) \ mod \ m$. The name servers for node N, then, will be the servers at the i^{th} position of each segment. The indexes for the servers are $\{j : j = k * m + i; j < n, k = 0, 1, ..l-1\}$. When j falls into the last segment and is greater than or equal to n, no correspondent server exits. Thus for the nodes whose last server index falls out of the range, the server redundancy is one less than the other nodes. However, since $m < n$, each node will have at least one name server.

4) Address Registration and Lookup: Each node N registers its $< name, address >$ binding to each of its name servers through unicast. If nodes' unique well known names are associated with uniformly distributed value and the servers are elected according to a not-name-related weight (here, random competition based on time), the hashing function results in a uniform distribution in server selection. Thus the distances from one arbitrary node to its servers are uniformly distributed over the network. The load of registration is uniformly distributed among all the servers as is the storage overhead.

When source A wants to communicate with node N, it uses the same hash function H to calculate the indexes of N's name servers. Source A then uses the indexes to gather all of node N's name servers into a set NS_N. From set NS_N, A chooses the closest server and sends the query message. In case of a tie, i.e., more than one server has the same closest distance, A randomly picks one. Upon receiving a failure message or experiencing a query timeout, A then turns to a less closer server. In case the failure is caused by a change of the name server at the index position, A will delay querying a server that is recently elected, instead, it turns to the next less closer one so to avoid inconsistency. More details on name server mobility are discussed next.

5) An Example: Figure 1 shows an example of the operation of the ADNS. The network has four name servers notated by $A, B, C,$ and D respectively. They are ordered in the same way. The redundancy degree l is set to 2. Thus, a NS table has two segments. At one moment, node U hashes its name to the first entry of each segment. The entries indicate that U's name servers are A and C. Node U then sends registration messages to them. Later, when node V needs to communicate with U, it picks node C from U's name server set $\{A, C\}$ as C is closer.

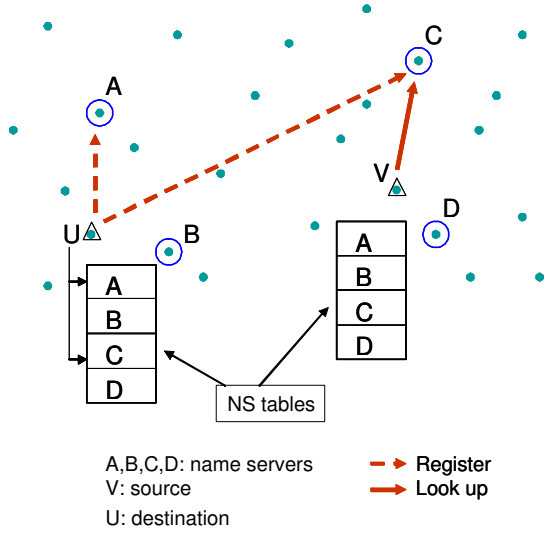

Fig. 1. Example: ADNS

D. Facing Mobility: Database Maintenance

In mobile networks, mobility or mission reorganization will cause a mobile node to change its address; or when topology changes too dramatically, the clustering algorithm will re-elect new cluster head(s). In the former case, a node will acquire a new address, thus need to update its servers with the new address. When the node is a name server itself, the new address will be advertised to the network with the routing update messages. When the latter situation happens, i.e, election causes name server to change, issues to be addressed involve server handovers. Both are discussed below.

1) Address Update and Lookup: Whenever node N changes its address, it calculates its server indexes and collects its server set. The node sends its new $< name, address >$ binding to its current address servers. The update messages are sent to all the servers regardless of their distances. Since the

change of server structure is much slower than the occurrence of communication traffic, sending updated name-address bindings to a few registration servers will not generate significant overhead to disturb routing and data communications.

2) Server Handover and Database Maintenance: As a result of cluster re-election, new server(s) will replace old server(s), which, will cause replacing one NS table entry or reordering of the indexing table (the worst case is that all the entries are reordered even though most old servers remain as servers). While a well known name still hashes to the same set of indexes, the servers may have changed to different nodes. There are two ways to deal with databases stored at old/new servers during the change of name servers, namely, node proactive registration and server database handover.

Detecting Server Change. In order to capture changes of the servers at the indexed NS table, a Boolean field $recentlyChanged$ is associated with each entry. For example, if a server L occupies the 3^{rd} place in the NS table, the field is marked *FALSE* when the network is in a stable state. When re-election of cluster head (server) occurs and a new server M occupies this 3^{rd} entry, the field is set to *TRUE*. The value of *TRUE* will keep for a short $holdingTime$ and reset to *FALSE* after it is believed that M has been updated with the most recent database. The short $holdingTime$ depends on many factors, such as: the server handover scheme, whether nodes proactively update their name-address bindings, how frequent nodes update their bindings, and the network diameter (measured in hop distance). Related operations on the field will be described in later paragraphs.

No matter the type of the underlying routing protocol, the server advertisement assembles a Distance Vector routing algorithm. Thus, when a node periodically updates its NS table based on a newly received NS updating message, the node will use a secondary NS table to store and sort the new NS updates first. It then compares the secondary table with the existing NS table. The node can easily tell whether at each index, the entry indicates different servers. If there is a difference, the server from the secondary NS table will replace the existing entry. This entry will be marked *TRUE* at its $recentlyChanged$ field. With the propagation of NS advertisement, the new server information will be notified to all the nodes in the network. As a consequence, all the nodes will update and mark their NS tables accordingly.

Proactive Registration. When a node detects a change of its name server sets, e.g., node A hashes its well known name into the same indexes, but finds some content of the indexes have changed to new servers where fields $recentlyChanged$ are marked $TRUE$, node A immediately updates those servers of its current name-address binding. Corresponding $recentlyChanged$ fields are then set to $FALSE$ after A updates the servers. A single server replacement may result in several nodes registering to the new server. The old servers simply discard their databases. Note that a query to a previous server will return a failure, so the source will look up the other servers. This situation only happens during the time when new server information is still propagating through the network. A short period of inconsistency of NS tables at different nodes causes the failure.

Database Handover. This approach requires servers explicitly handover their databases. When a node is defeated in an election, it checks in the new NS table for the entry that it previously occupied to see which is the current server. It then sends its database to the server and discards its own database. The new server marks its $recentlyChanged$ field $FALSE$ after receiving the database. If a new server causes a reordering of several servers in the NS table, all these servers will handover their old databases to the ones newly occupying their old positions. They will receive databases from the nodes previous referenced at their newly occupied indexed entries. When the network is large in scale and each segment contains only a few servers, the database can be very large. Local traffic could increase temporarily during the handover period.

However, the database handover among servers is considered redundant given that proactive registration is always needed in order to keep name-address databases fresh. A node can always detect whether its servers have changed or not, so as to update the new servers immediately. In some extreme cases, e.g., network partitioning or merging, generation/replacement of servers may associate with address changes of some nodes as a result of the dynamic nature of MANETs. Then database handover will be accompanied by address registration. Thus in our design, we adopt the proactive registration approach for the database maintenance.

3) Avoiding Invalid Queries: Having the $recentlyChanged$ field indicating the validation of a server, a source S can avoid sending a query that is doomed to failure. S will not use a server that has been marked $TRUE$ in its $recentlyChanged$ field. Instead, S queries the next closer server for the destination. If S finds that all the hashed entries are marked $TRUE$, it delays for a while (the $holdingTime$ can be pre-configured) before it references again to the servers in the hashed entries and chooses the closest one. The delay gives the new servers time to populate their databases. Typically, after the short delay period, each node remarks the indexing table and resumes its full functionality.

Whether an ongoing application will be affected by a node address change depends on how quickly the new address is resolved. It is possible that a few packets will be lost before the application discovers packet loss (due to the address change). This loss is typically the same as packets loss incurred by a broken route. If the application uses reliable data transmission protocols, data packet loss will be recovered. As for how long packets have to be buffered in looking up for an address, several factors are concerned, including: the round trip time for the unicast lookup query and its response; in case of handover, the number of iterative queries to several servers; or the holding time for the database to converge. For large-scale networks, the holding time could be large in order to gather fresh bindings from distant destinations.

III. Simulation Evaluation

We evaluate ADNS using a very dynamic application scenario where a large number of mobile hosts are involved. The nodes are organized in logical groups, e.g., tank battalions, infantry companies, UAV swarms, etc. Complex missions are launched which typically comprise several such teams. As the missions evolve, groups of nodes are reconfigured and new groups are often created by rearranging and regrouping the assets in response to new emergencies. These activities cause the network members to change their IP address frequently. The operation of the network is supported by automatic group detection at the network layer through recognition of motion affinity among members belonging to the same logical groups and motion differences between different groups [18]. Landmark Ad Hoc Routing Protocol (LANMAR) [16] is used to provide scalable routing by exploring the group behavior exhibited. The challenge of the network is that nodes' addresses are designed to reflect the address aggregation over motion groups to enable CIDR-like [17] scalable routing. The automatic group formation causes dynamic address change when they perform their tasks. In simulating the network, group motion evolution is the norm of the network. Building on top of LANMAR, ADNS coordinates server election with the group formation algorithm and also updates server information with the landmark routing update messages.

We presents two set of evaluations for the network scenario. The first set evaluates the ADNS scheme with different redundant factors. Metrics evaluated are (i) *average name registration and lookup distance* – the average hop distance that registration and query messages travel; and (ii) *query overhead* – the total number of query messages transmitted in the network. The other set investigates data communications when ADNS is used for such a network with comparisons to an ideal name service. With the ideal name service, a source can lookup for a destination's most current address immediately. Results from this scheme provide an upper bound for all the performance metrics. The metrics include: (iii) *packet delivery fraction* – the ratio between the number of data packets received and those originated by the sources; and (iv) *average end-to-end packet delay* – the time from when the source generates the data packet to when the destination receives it.

A. Simulation Scenario

The simulations use the GloMoSim simulation platform [23], a discrete-event, detailed simulator for wireless network systems. The message exchange uses a MAC layer that realizes the default characteristics of the distributed coordination function (DCF) of IEEE 802.11, where RTS/CTS/DATA/ACK mechanism is used to provide virtual carrier sensing for *unicast* data packets, and CSMA/CA is used for *Broadcast* packets. The radio model uses characteristics similar to a commercial radio interface (e.g., Lucent's WaveLAN). The channel capacity and transmission range are 2 Mbits/sec and 200m respectively.

The simulations run in a network occupying a square field of 1000m X 1000m, with 100 initial uniformly distributed mobile nodes. The mobile nodes are in 4 groups, each having 25 nodes. The nodes initially occupy a quarter of the area without overlapping. The motion of the mobile groups is modelled using the *Reference Point Group Mobility (RPGM)* model [24], i.e., sets of nodes move in different common trajectories with a little randomness. Nodes have no group identities at the beginning. They dynamically form logical groups after simulation starts [18]. If ADNS is used, nodes will register/update to their name servers. For the referencing ideal naming service, nodes will register/update to a single centralized data structure available for immediate access by all the nodes all the time. The name server update messages are sent with the LANMAR routing updates.

Constant Bit Rate (CBR) data sessions with randomly chosen source-destination pairs are used. The CBR sessions are short lived. When one pair stops communication, another pair will be generated. So the network keeps constant offered load over the simulation time. Each CBR session sends a packet of 512 bytes every second for two minutes. Each simulation runs for 10 minutes with a warm up period (which is required by the dynamic group formation algorithm) of 4 minutes (to suit all the mobility cases). The communications start after the warm up period.

B. Results

Our first result (Figure 2) gives the average hop distance travelled by registration and query messages as a function of offered load. Mobility is moderate at 6 m/s. The result compares server redundancy at 1 and 2. The figure shows that query distances are shorter than registration distances as expected since the queries are always sent to the closest server. The figure also reports that for registration, redundancy does not make a big difference since servers are uniformly distributed, while for query, redundancy helps reducing the distance since a node is offered more severs in choosing a closest one.

Figure 3 gives the total number of query messages transmitted in the network. We compare ADNS with the brute force query flooding. The figure shows a much slower increasing trend of overhead of ADNS than for query flooding, showing the benefit of using a name system. The figure also shows that the query messages are not affected by redundancy. While expectation might be that redundancy should bring lower overhead, the network scale simulated here is not sufficiently large to emphasize the distance advantage (Figure 2) in reducing overall number of query messages.

The second set of simulation varies mobility so to observe how the proposed scheme reacts to the dynamic group membership change. 10 pairs of CBR traffic is offered and redundancy is 2.

Figure 4 gives packet delivery fraction as a function of mobility. Generally both performances degrade when mobility increases. The figure shows that the ADNS scheme degrades faster than the ideal case when mobility increases. Since high

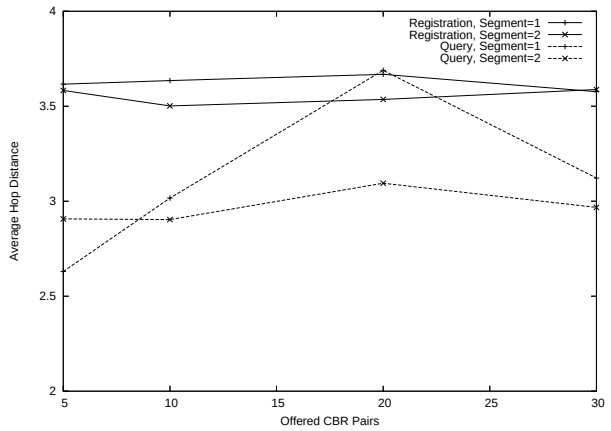

Fig. 2. Lookup Distance

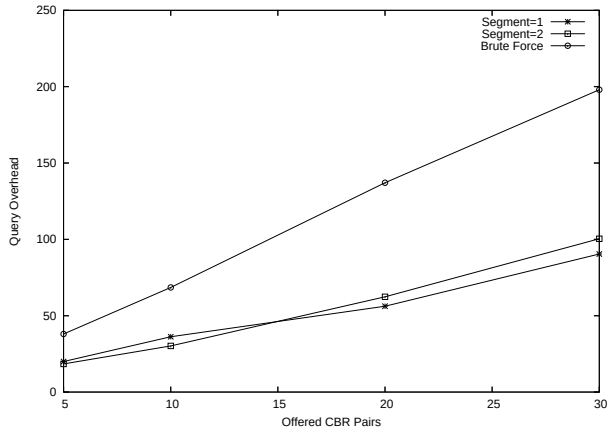

Fig. 3. Total Transmitted Query Messages

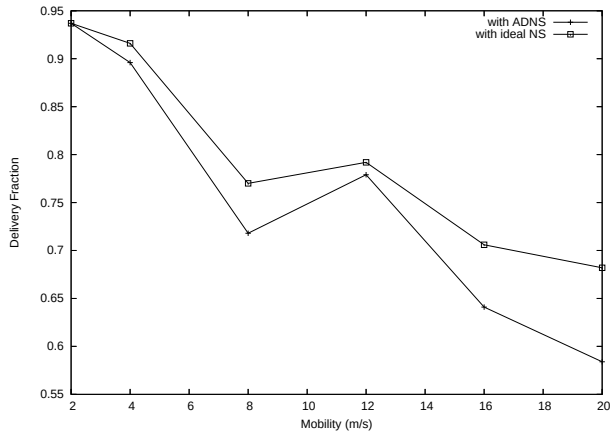

Fig. 4. Data Packet Delivery Fraction

mobility causes more frequent address changes of the mobile nodes and more server reelection, the further degradation of ADNS counts for server address changes and server handover when communications are in session (note that we evaluate ADNS in a very stressful scenario where dynamic group formation affects routing and addressing). In addition, registration and query massages could be lost when they can not find the intended servers. Currently we are working on improving both the formation algorithm and ADNS scheme.

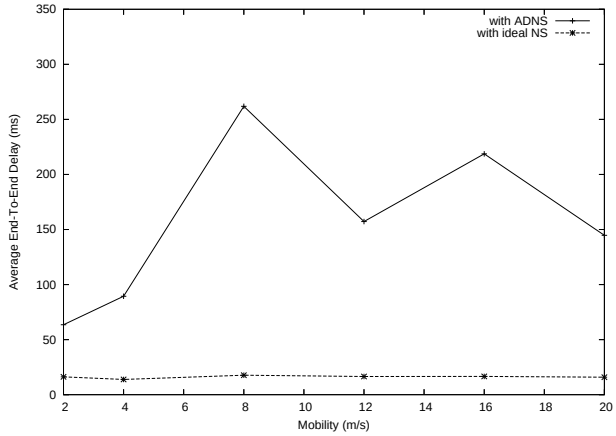

Fig. 5. Average End-to-End Delay

Figure 5 reports changes in average packet end-to-end delay when mobility increases. Using ADNS, packets experience longer end-to-end delay compared to the ideal case due to the buffering for name resolutions. With the ideal lookup scheme, mobility has no influence at all. However, higher mobility increases ADNS's delay because more queries go to the second server, which increases the buffering time, hence the end-to-end delay. From our specific network configuration, the delay falls within a reasonable range in the sense that applications (using TCP) other than CBR (using UDP) can still recover from timeouts caused by a few individual packets.

IV. Conclusions

The dynamic nature of mobile ad hoc networks requires name service to support on-the-scene translation of a mobile node's permanent well known name to its current network address which possibly changes frequently. This paper proposes a Distributed Naming Service for MANETs (ADNS) to meet the need. ADNS provides a distributed and redundant server structure, balances the query load and provides fault tolerance. The simulation results show that the distribution of name servers helps to decrease the lookup traffic, and the service reacts to dynamic organization of the mobile nodes promptly and accurately, providing valid name resolution with reasonable delay. Future work will include investigating naming system for networks using on-demand routing protocols.

References

[1] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network", *Proceedings of INFOCOM 2002*, New York, June 23-27, 2002.

[2] C. Perkins, J. Malinen, R. Wakikawa, E. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks", Internet draft, draft-ietf-manet-autoconf-01.txt, Nov. 2001.

[3] R. Wakikawa, J. Malinen, C. Perkins, A. Nilsson, and A. Tuominen, "Global connectivity for IPv6 Mobile Ad Hoc Networks", Internet draft, draft-wakikawa-manet-globalv6-01.txt.

[4] G. Chelius and E. Fleury, "IPv6 Addressing Architecture Support for mobile ad hoc networks", Internet Draft, draft-chelius-adhoc-ipv6-00.txt, September 2002.

[5] H. Zhou, L. Ni, and M. Mutka, "Prophet Address Allocation for Large Scale MANETs," *Proceedings of IEEE INFOCOM 2003*, San Francisco, March, 2003.

[6] J. Eriksson, M. Faloutsos, S. V. Krishnamurthy, ”Scalable Ad Hoc Routing: The Case for Dynamic Addressing”, *IEEE INFOCOM 2004*, Hong Kong, 2004.

[7] Levon Esibov, Bernard Aboba, Dave Thalerx, ”Linklocal Multicast Name Resolution (LLMNR),” *INTERNET-DRAFT draft-ietf-dnsext-mdns-38.txt*, DNSEXT Working Group, 19 February 2005.

[8] Jaehoon Jeong, Jungsoo Park and Hyoungjun Kim, ”NDR: Name Directory Service in Mobile Ad-Hoc Network”, *ICACT 2003*, Korea, January 2003.

[9] Jaehoon Jeong, Jungsoo Park and Hyoungjun Kim, ”Name Directory Service based on MAODV and Multicast DNS for IPv6 MANET”, *VTC 2004-Fall*, Los Angeles, CA, USA, September 26-29, 2004.

[10] H. Zhou, M. Mutka, and L. Ni, “IP Address Handoff in the MANET,” *Proceedings of IEEE INFOCOM 2004*, March 2004.

[11] Benjie Chen and Robert Morris, ”L+: Scalable Landmark Routing and Address Lookup for Multi-hop Wireless Networks,” MIT LCS Technical Report 837, March, 2002.

[12] Jakob Eriksson, Srikanth V. Krishnamurthy, Michalis Faloutsos, “Peer-Net: Pushing Peer-to-Peer Down the Stack”, *International Peer-To-Peer Symposium (IPTPS 2003)*, Berkeley, Feb 2003.

[13] C.E. Perkins and E.M. Royer, ”Ad-Hoc On-Demand Distance Vector Routing,” in *Proceedings of IEEE WMCSA’99*, New Orleans, LA, Feb. 1999, pp. 90-100.

[14] D. B. Johnson and D. A. Maltz, ”Dynamic Source Routing in Ad Hoc Wireless Networks”, In *Mobile Computing*, edited by T. Imielinski and H. Korth, Section 5, Kluwer Publishing Company, 1996, pp. 153-181.

[15] B. Bellur and R. G. Ogier, “A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks,” in *Proc. IEEE INFOCOM ’99*, New York, March 1999.

[16] G. Pei, M. Gerla and X. Hong, ”LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility,” in *Proceedings of IEEE/ACM MobiHOC 2000*, Boston, MA, Aug. 2000, pp. 11-18.

[17] V. Fuller, T. Li, J. Yu, K. Varadhan, ”Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy”, RFC 1519, Sept., 1993.

[18] Xiaoyan Hong and Mario Gerla, ”Dynamic Group Discovery and Routing in Ad Hoc Networks,” *Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop (Med-hoc-Net 2002)*, Sardegna, Italy, Sept. 2002.

[19] C. R. Lin, and M. Gerla, ”Adaptive Clustering for Mobile Networks,” *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 7, Sep. 1997, pp. 1265-1275.

[20] A. Amis, R. Prakash, D. Huynh and T. Vuong, ”Max-Min D-Cluster Formation in Wireless Ad Hoc Networks”, in *Proceedings of INFOCOM 2000*, Tel Aviv, Israel, March 2000.

[21] M. Gerla, T.J. Kwon and G. Pei, ”On Demand Routing in Large Ad Hoc Wireless Networks with Passive Clustering,” *Proceedings of IEEE WCNC 2000*, Chicago, IL, Sep. 2000.

[22] Kaixin Xu and Mario Gerla, ”A Heterogeneous Routing Protocol Based on a New Stable Clustering Scheme,” *IEEE MILCOM 2002*, Anaheim, CA, Oct. 2002.

[23] M. Takai, L. Bajaj, R, Ahuja, R. Bagrodia and M. Gerla, ”GloMoSim: A Scalable Network Simulation Environment”, *Technical report 990027*, UCLA, Computer Science, 1999.

[24] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, ”A Group Mobility Model for Ad Hoc Wireless Networks”, in *Proceedings of ACM/IEEE MSWiM’99*, Seattle, WA, Aug. 1999, pp.53-60.