KELOP: Distributed Key-Value Lookup in Wireless Ad Hoc Networks

Shahid Bashir, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{*shahid,bli*}*@eecg.toronto.edu*

***Abstract*— It is critical to discover and utilized shared services and resources in wireless ad hoc networks. While wireline networks can use the Domain Name System (DNS) to perform such key-value lookups, designing scalable key-value lookup protocols with high success rates and low message overhead in the dynamic topologies of wireless ad hoc networks presents a non-trivial challenge. In this paper, we present *KELOP*, a *Key-Value Lookup Protocol* for wireless ad hoc networks. KELOP is a fully distributed best-effort protocol that relies only on the local information stored at each node to locate the *closest estimates* of the target. This strategy results in remarkably low control-message overhead. Simulation results show that, in addition to low message overhead, KELOP is able to provide lookup success rate close to 100% in most cases.**

I. Introduction

It is critical to utilize and discover shared resources and services efficiently in wireless ad hoc networks with dynamic topologies. A generic key-value lookup protocol, once designed, is able to support such resource discovery without changes. The key-value lookup problem may be described as follows. Given a key, locate the node that hosts the corresponding value.

In ad hoc networks, since very limited global information can be available at each node, key-value lookups significantly depend on the following question: if the value corresponding to a particular key is not known yet, from where can this information be obtained? In Domain Name System (DNS), for example, this question is answered by forwarding the request to the next server in the hierarchy of DNS servers [1]. However, the dynamic topology and the lack of base infrastructure in wireless ad hoc networks pose special challenges not encountered in wireline networks. In mobile ad hoc networks, such an hierarchy is expensive to maintain.

Since ad hoc networks are peer to peer in nature, the list of the node identifiers present in the network is valuable to achieve successful key-value lookups. Once such information is available, a key can be hashed to a node identifier in a known set. However, the list of nodes is dynamically changing in ad hoc networks. To make matters worse, frequent information exchanges to update the node list is not feasible because of bandwidth constraints.

In this paper, we identify that the *information cached by the routing protocols* is one of the best candidates for the target node lookups in wireless ad hoc networks. Using this information, we seek to find a known node having an identifier that is *closest* to the target node, rather than finding an *exact* target node. Based on these concepts, we propose KELOP, a fully distributed key-value lookup protocol in wireless ad hoc networks. The objective of KELOP is to achieve high key-value lookup success rates with a low message overhead. We present KELOP in details, and perform simulation studies to evaluate the associated success rate and message overhead. Although KELOP is independent of any specific routing protocols, the analysis and simulations in this paper are carried out using the route cache of the Dynamic Source Routing (DSR) protocol [2].

The remainder of the paper is organized as follows. We begin the presentation of the KELOP protocol by first analyzing the cached information from a routing protocol in Sec. II, followed by the details of the KELOP protocol in Sec. III. We evaluate the performance of KELOP in Sec. IV. Finally, we conclude the paper with a survey of the related work in Sec. V and concluding remarks in Sec. VI.

II. KELOP: Cache Information Analysis

In this section, we analyze the local cache information at each node for the purpose of target node lookup corresponding to a given key. As we have discussed, the most important information for this purpose is the list of nodes present in the network, referred to as *node information* in the remainder of the paper. To analyze node information, we calculate the update rate λ at which a node n' can update its route cache with the ID of another node n. $1/\lambda$ is the inter-update time. If the update rate is higher, with a higher probability n' knows the availability of n correctly.

We assume the following setup. The node information flow in the network is caused by the normal data packet flows. Packet generation at each node is a random Poisson process with a constant rate α packets per time unit t_u. Each packet is forwarded to a destination node selected at random with a uniform probability distribution.

A. Analysis of one-dimensional networks

For one-dimensional networks, we assume a setup such as the one shown in Fig. 1. The transmission range is fixed and is the same as the inter-node distance. Therefore, only adjacent nodes can hear the transmission by a node. With these assumptions in a one-dimensional formulation of Fig. 1, we get the following result:

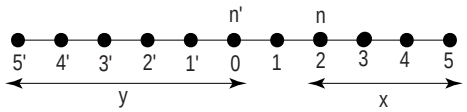

Fig. 1. A one-dimensional network with 11 nodes ($Y = 6$, $X = 4$).

Theorem 1: [1]In a one-dimensional node array such as the one shown in Fig. 1, with a fixed transmission range of one hop, and random packet generation by each node with a rate of α per time unit, the rate λ at which a node n' can update its route cache with the ID of another node n is the following:

$$\lambda = \alpha X(2Y+1)/N \tag{1}$$

where X and Y are the number of nodes in segments x and y respectively as shown in Fig. 1.

Theorem 1 can be analyzed for the two aspects we mentioned in the beginning of this section. First: The information update rate and therefore the correct information available at any node depends on the packet generation rate (α) and the total number of nodes in the network (N). Second: The rate at which n' can update the ID of n in its route cache is highly dependent on X and Y, and in turn on the positions of n and n' in the network. When both n and n' are at the center, X and Y are both of the order of $O(N)$, and the update rate is of the order of $O(N)$. When n and n' are at extreme ends, X and Y both are of the order of $O(1)$ and the update rate is of the order of $O(1/N)$.

B. Analysis of two-dimensional networks

In a two-dimensional network a packet flow can follow any one of the several possible paths from the source to the destination. However, for simplicity, we assume a straight line shortest path between a source and a destination. Furthermore, we assume the following:

- The total network area A is of rectangular shape with a uniform node density ρ. The total number of nodes, N, is ρA.
- The node density is high enough such that the number of nodes in any given area segment a_i can be approximated to $a_i\rho$ without any error.
- All nodes have a fixed transmission range r.

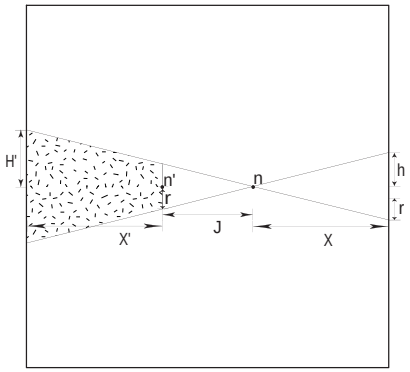

Fig. 2. Flow groups 1 and 2 in a two-dimensional network

Fig. 2 illustrates a two-dimensional network with nodes n and n' marked. The entire rectangular area contains nodes with uniform density ρ according to the assumption. In the following discussion, we calculate the three flow groups for the two-dimensional network of Fig. 2.

The first group consists of all the flows originated by n that can be overheard by the node n'. It is clear from Fig. 2 that n overhears all flows originated from n and directed towards the dotted area. The dotted area is equal to $(H'(X'+J)-rJ)$ which can be simplified to $X'r(2+X'/J)$, using the fact that $H' = r(X'+J)/J$. The number of nodes in the area $X'r(2+X'/J)$ is $\rho X'r(2+X'/J)$. Since the rate at which packets are being generated at n is α and a destination is randomly selected from the total N nodes, the flow rate of the first group, λ_{2D_1}, can be written as $\alpha\rho X'r(2+X'/J)/N$.

[1]Proof is excluded due to space constraints and is available upon request.

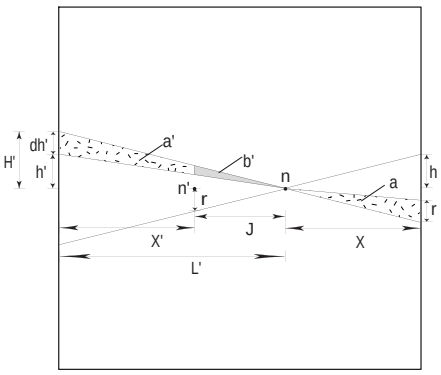

Fig. 3. Flow group 3 in a two-dimensional network

The second group consists of all the flows destined to n that can be overheard by the node n'. That is, the packet flows from the nodes in the dotted area to n. The rate of packet flow from any given node to a specific node (n in this case) is α/N. Therefore, the total flow rate for the second group, λ_{2D_2}, is (number of nodes in the dotted area)$\times\alpha/N$, which is the same as λ_{2D_1}. Therefore, the first two flow rates can be written as the following:

$$\lambda_{2D\_1} = \lambda_{2D\_2} = \alpha\rho X'r(2+X'/J)/N \tag{2}$$

A subset of third flow group is shown in Fig. 3; originated by the nodes in area a' and destined to the nodes in area a. All such flows, having n on the path, are overheard by n'. The flow rate from a single node in the area a' to the nodes in area a is $\alpha\rho a/N$. Therefore, the flow rate from the nodes in area a' to the nodes in area a can be written as $\rho a' \times \alpha\rho a/N$. Therefore, the flow rate $d\lambda_{2D\_3}$ corresponding to the incremental area shown in Fig. 3 can be written as:

$$d\lambda_{2D\_3} = 2(\rho a')(\alpha\rho a/N) \tag{3}$$

Area a' in Fig. 3, is equal to $L'dh'/2 - J(J/L')dh'/2$. Similarly, area a can be written as $Xr/2$. By putting values of a and a', introducing fraction c, integrating dh' over $2H'$, and using the facts $H' = rL'/J$ and $L' = X' + J$, the rate of the third flow group can be written as the following:

$$\lambda_{2D\_3} = c\alpha\rho^2(1/N)(Xr^2/J)(X'^2 + X'J) \tag{4}$$

The total rate (λ_{2D}) at which the route cache of n' is updated with the information of node n is the sum of three flow rates calculated above for the two-dimensional case. That is,

$$\lambda_{2D} = c\alpha\rho^2(1/N)(Xr^2/J)(X'^2 + X'J) + 2\alpha\rho X'r(2+X'/J)/N \tag{5}$$

Eq. (5) illustrates that node information in a two-dimensional network depends on node density, packet flow rate in the network, the total number of nodes, and the transmission range.

For the analysis of this result with respect to node position, we use the property $A \propto N$, which directly follows from the assumption that $\rho = N/A$ is a constant. When both n' and n are in the center of the network, X' and X are of the order of $O(\sqrt{A})$, and J is of the order of $O(1)$. With these parameters

and with α, ρ, r and c constants, λ_{2D} in Eq. (5) results in the order of $O(\sqrt(N))$. Intuitively, it can be visualized as follows: The major contribution is from the flows in the one-dimensional lines, with small or fixed widths, from $O(\sqrt{N})$ nodes to $O(\sqrt{N})$ nodes integrated over $O(\sqrt{A}) = O(\sqrt{N})$, giving us $O(N\sqrt{N})$ flows. The probability of each flow, however, decreases with the rate $O(N)$, giving us the effective update rate of the order of $O(\sqrt{N})$. In another scenario, when both n' and n are on the opposite ends of the network, X' and X are of the order of $O(1)$ and J is of the order of $O(\sqrt{A})$. In this case, λ_{2D} comes out to be of the order of $O(1/N)$.

From the above analysis, it is clear that center nodes have high availability of node information than the nodes on the boundaries, since such nodes have more packets to forward and are more active in the network. To increase the possibility of utilizing the information possessed by these nodes, KELOP uses the route cache information of all *forwarding nodes*.

III. KELOP: THE PROTOCOL

In this section, we present and explain KELOP. KELOP is composed of two modules, one for key-value advertisements and the other for lookups (i.e., querying the value, given a key). Figures 4 and 5 show the KELOP advertisement and lookup protocols respectively. Fig. 6 shows the helper functions for finding the nearest node and the next target (*next_ID*) from the known node IDs. For the next target and the nearest node selections, node IDs are calculated in a cyclic fashion. That is, if the ID space is 1–1000, the smallest ID greater than 1000 is 1.

Following is the explanation of KELOP advertisement and lookup protocols.

1) KELOP Advertisements: The advertising node hashes the key to a target ID (called *hashed_target*) in a predefined ID space, using a globally known hashing function. Subsequently, the advertising node finds the greatest ID less than *hashed_target* from the known node IDs in its route cache. This greatest ID is called *current_target*. If the advertising node's own ID is not the same as *current_target*, the packet is transmitted to the next hop aimed for the destination, *current_target*.

An intermediate node, upon receiving the advertisement packet for forwarding, performs the following two actions before forwarding. First, it adds (or refreshes) the key-value pair in its key-value table. Second, it checks its own route cache for a better *current_target*.

The value of *current_target* is updated to reflect the new value if a better *current_target* is found such that: previous *current_target* $<$ new *current_target* $<$ *hashed_target*. Utilizing the information stored in the route caches of the forwarding nodes helps improve the estimate of an available target.

When the advertisement packet reaches the target node (*current_target*), the target node adds (or refreshes) the key-value pair in its table. Moreover, the target node tries to further advertise the pair to a next available target *next_ID*, until the number of advertisement *attempts* reach the maximum value K. *next_ID* is the smallest known ID that is greater than *hashed_target* (Fig. 6). After finding the value of *next_ID*, *hashed_target* is set to *next_ID* for the next advertisement. When the number of attempts reach the value of K, advertisement is done.

2) KELOP Lookups: Before initiating the query, the requesting node (i.e. a *requester*) checks its own *key_value_table* for the key (Fig. 5). If the key is not found, the requesting node calculates *hashed_target* and *current_target*, and forwards the packet in the same way as explained for the advertisement module.

An intermediate node, upon receiving the query packet for forwarding, performs the following steps before forwarding the packet. First, if the key is found in its *key_value_table*, it replies back to the requester with the key-value pair and the query is done. Second, if the key is not found in its *key_value_table*, the forwarding node tries to find a better guess to the value of *current_target* and forwards the packet to the next hop destined to the new *current_target*.

If the query reaches the target node (*current_target*), and remains unsuccessful, the target node forwards the query to a new target, *next_ID*, in a next lookup attempt. When the maximum number of attempts (K) have been made, and a query is still unsuccessful, a failure response is generated. $K = 3$ is used in our simulations for advertisements as well as lookups. When the value of K is high for advertisements, it results in an almost proportional increase in message overhead. For lookups, however, a higher value of K means only a slight increase in message overhead because the query is done as soon as the key-value pair is found.

IV. PERFORMANCE EVALUATION

KELOP is designed to maximize the service discovery success rate with low protocol message overhead. To test how well KELOP achieves its goal, we simulate KELOP for the lookup success rate and the corresponding message overhead. Following is the simulation setup and the discussion of results.

A. Simulation setup

We compare the simulation results of three cases: the ideal case, KELOP, and the worst case. In the *ideal* case, each node knows the shortest path to every other node. For a lookup, given a key, the requester directly contacts the corresponding host. Since the host ID is known to the user, no advertisement is required and the lookup success rate is 100%.

To avoid the route request flooding by DSR, *KELOP* forwards packets to only the nodes that are known to be in the network. In the *worst* case, available links and node information are the same as in KELOP. The main difference of the worst case from KELOP is the following: advertising and querying nodes do not use the *nearest target* strategy. In the worst case, if a hashed target is not present in the network, or not known to the advertising or querying nodes, lookup fails.

The simulation is composed of two components. The first component consists of the simulation of a mobile ad hoc network with regular packet flows for $450s$. NS2 is used for simulating various scenarios with 25 to 200 nodes with increments of 25 (8 scenarios). The corresponding deployment area in each case is selected such that the node density remains constant (i.e., 1 node per $7200m^2$). The maximum speed is 10m/s and pause time is 100s. During this regular packet generation and flow, route cache builds up. This route cache information is then used to evaluate KELOP in the second component.

KELOP Advertisement:

adv_packet < *sender, receiver, key, value, hashed_target, current_target, attempts* >
max number of *attempts* = K
node running the algorithm is *current_node*

Packet Initiation:(at the advertising node)
  hashed_target ← hash(*key*)
  key_value_table = (*key,value*) ∪ *key_value_table*
  current_target ← findNearestNode(*hashed_target*)
  if (*current_target* ≠ *current_node*)
    forward *adv_packet* to the next hop for *current_target*

Packet Forwarding:(at the forwarding node)
  key_value_table = (*key,value*) ∪ *key_value_table*
  if (*current_node* = *hashed_target*) OR (*current_node* = *current_target*)
    if (*attempts* < K)
      increment *attempts*
      hashed_target ← findNextTarget(*hashed_target*)
    else
      done
  current_target ← findNearestNode(*hashed_target*)
  forward *adv_packet* to the next hop for *current_target*

Fig. 4. KELOP advertisement

KELOP Lookup:

query_packet < *sender, receiver, key, hashed_target, current_target, attempts, requester* >
response_packet < *sender, receiver, key, value, is_successful, requester* >
max number of *attempts* = K
node running the algorithm is *current_node*

Packet Initiation:(at the requesting node)
  if (*key* ∈ *key_value_table*)
    return (*key,value*); done //value is found in the requester's cache
  hashed_target ← hash(*key*)
  current_target ← findNearestNode(*hashed_target*)
  if (*current_target* ≠ *current_node*)
    forward *query_packet* to the next hop for *current_target*
  else
    return *lookup_failure* //no packet forwarded

Packet Forwarding:(at the forwarding node)
  if (*key* ∈ *key_value_table*)
    is_successful ← True
    send *response_packet* to *requester*; done
  if (*current_node* = *hashed_target*) OR (*current_node* = *current_target*)
    if (*attempts* < K)
      increment *attempts*
      hashed_target ← findNextTarget(*hashed_target*)
    else
      is_successful ← false
      send *response_packet* to *requester*
  current_target ← findNearestNode(*hashed_target*)
  forward *query_packet* to the next hop for *current_target*

Fig. 5. KELOP lookup

Finding the nearest node and the next target:

ID_type findNextTarget(*hashed_target*)
  next_ID ← (the smallest known node_ID > *hashed_target*)
  return *next_ID*

findNearestNode(*hashed_target*)
  return (the greatest known node_ID < *hashed_target*)

Fig. 6. Finding the nearest node and next target

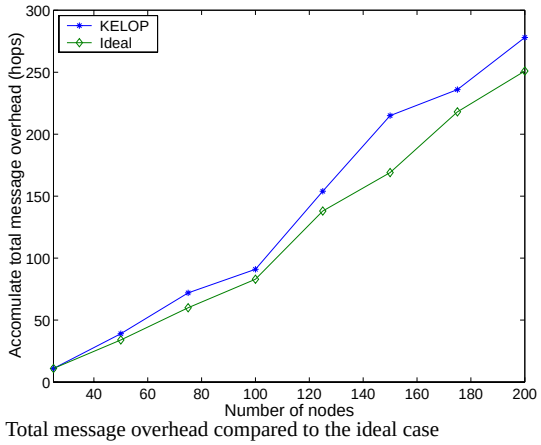

Fig. 7. Total message overhead compared to the ideal case

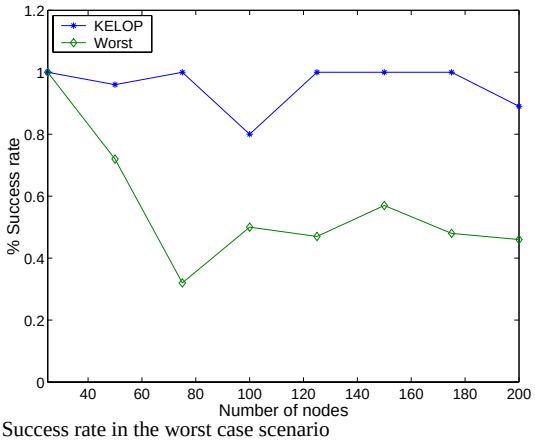

Fig. 9. Success rate in the worst case scenario

B. Simulation results

We first compare the total message overhead introduced by KELOP with the one introduced by the ideal case. Fig. 7 shows the comparison of the two. For KELOP, the total overhead is the sum of the service advertisement overhead and the lookup overhead. In the ideal case, the overhead only involves the messages for lookups. The total overhead incurred by KELOP is very close to that of the ideal case as expected from the conservative approach of KELOP. An interesting implication of advertisements is that the query succeeds in a very few hops. This phenomenon results in efficient lookups and compensates for the message overhead incurred for advertisements.

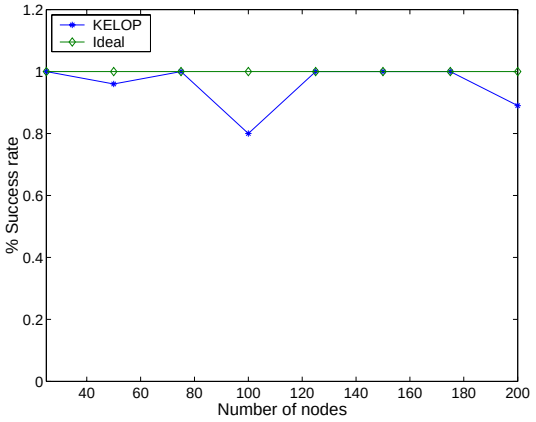

Fig. 8. Success rate compared to the ideal case

After observing that KELOP is conservative in terms of message overhead, in Fig. 8 we compare the success rate of KELOP with the ideal case (success rate of 100%) in the absence of overhearing. Only three of the eight simulated scenarios fall below the ideal 100%. Since KELOP highly depends on the current route cache contents of any node, the deviation in success rate is expected.

The worst case success rate has a downward trend with the increase of the number of nodes (Fig. 9). When the number of nodes increases and the deployment area becomes larger, a node can have only a limited information of the far nodes. Therefore, the success rate is very low.

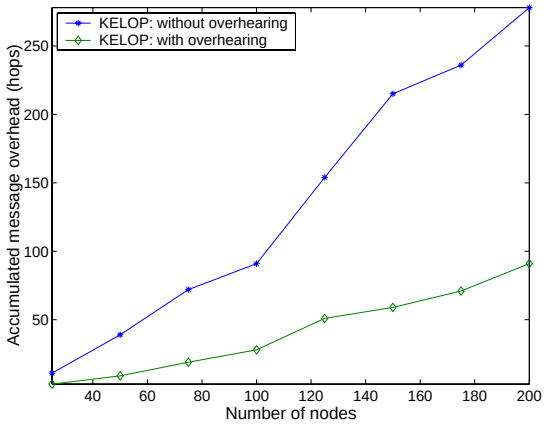

Fig. 10. Message overhead with overhearing enabled

Fig. 10 and 11 show the effects of overhearing on success rate and message overhead respectively. Because of overhearing, with high probability, queries are answered from within a few hops from the requesting node. Therefore, lookup message overhead decreases and a high lookup success rate is achieved. In Fig. 11, for all the simulated network sizes, the success rate achieved is 100%. Clearly, KELOP adapts nicely to take full advantage of the useful overheard key-value information.

Finally, in Fig. 12 we observe the effect of a lower transmission range. With the lower transmission range, the overhearing effect is reduced. When the transmission range decreases from $250m$ to $200m$, success rate decreases from 100% in some of the scenarios. Among the 8 simulated scenarios, 3 scenarios have success rate below 100%, minimum being 94% in the 100 node

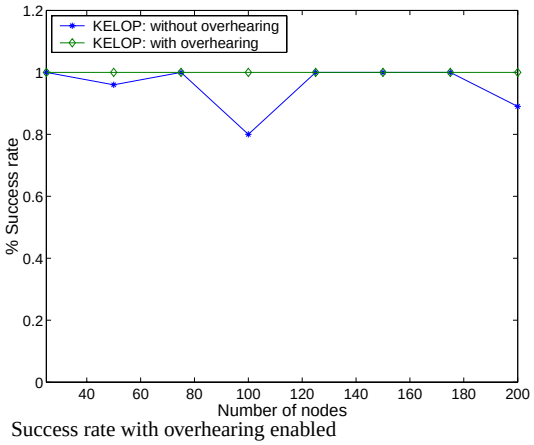

Fig. 11. Success rate with overhearing enabled

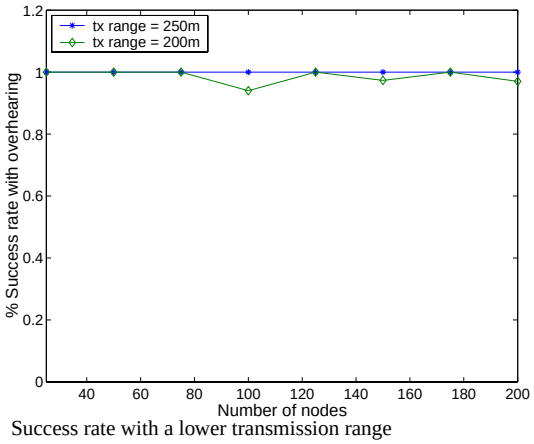

Fig. 12. Success rate with a lower transmission range

scenario.

The above discussion of the simulation results shows that the ideal case, using the global network information, is able to provide 100% lookup success rate with low message overhead. A simple strategy that does not use KELOP, i.e. the worst case, is left far behind the ideal case in terms of success rate. However, KELOP tracks the performance of the ideal case closely by achieving almost the ideal success rate with the realistically available DSR route cache information.

V. Related Work

Most of the research work related to ad hoc networks concentrates on routing [3], [2], [4], [5], routing performance evaluations [6], [7], [8], location management and resource availability. In this context, resource discovery is a relatively new research area. However, there is a good amount of closely related research work on resource discovery in wired peer-to-peer networks.

In wired peer-to-peer networks, several protocols such as Gnutella [9], Freenet [10] and Chord [11] facilitate resource sharing. Creating and maintaining structures like Chord [11], is too expensive, since each point to point packet flow in wired networks is in fact a multi-hop end-to-end packet flow in wireless ad hoc networks. Furthermore, most of the key-value lookup strategies used in wired peer to peer networks concentrate on exact key-value lookups. Although feasible in wired peer to peer networks, seeking an exact key-value lookup can be very unsuccessful in wireless ad hoc networks due to high probability of target node un-availability. Therefore, KELOP, using the best effort strategy, tries to find a *nearest node* instead of the exact target node. Furthermore, designed specifically for wireless ad hoc networks, KELOP uses overhearing and re-evaluates its target on *per-hop* basis when a query or advertisement packet is forwarded.

Some of the recent research work has focused on the service availability in mobile ad hoc networks. Wang *et al.* [12] have proposed protocols for guaranteed service coverage in partitionable mobile ad hoc networks. Once the services are available, to facilitate the utilization, the discovery protocols such as KELOP can help users in discovering these services. In a closely related work on service discovery in ad hoc networks, Kozat *et al.* [13] form a virtual backbone to facilitate the distribution of service information. KELOP, however, advertises service information and provides successful service discovery without needing any virtual backbone and incurring the associated control message overhead.

VI. Concluding Remarks

Taking up the challenge of providing an effective and efficient key-value lookup protocol for mobile ad hoc networks, this paper presents and evaluates KELOP. KELOP is a fully distributed key-value lookup protocol and relies only on the local route cache information. Furthermore, instead of trying to locate the exact target node, KELOP's strategy is to work with the best estimate of the target node. To the best of our knowledge, there exists no key-value lookup protocol for ad hoc networks based only on the local information. The simulation results show that KELOP is able to achieve high success rates, close to 100% in most cases. KELOP's conservative approach in terms of message overhead and adaptability to take full advantage of the available information make it a suitable key-value lookup protocol for mobile ad hoc networks.

References

[1] P. Mockapetris, "RFC-1034 Domain Names - Concepts and Facilities," *Network Working Group*, 1987.

[2] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing (ed. T. Imielinski and H. Korth)*, 1996.

[3] C. Perkins and E. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 99)*, 1999.

[4] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing $(DSDV)$ for Mobile Computers," in *Proceedings of ACM SIGCOMM*, 1994.

[5] Y. B. KO and N. H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 98)*, 1998, pp. 66–75.

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[7] S. R. Das, C. E. Perkins, and E. E. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks," in *Proceedings of IEEE INFOCOM*, 2000.

[8] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[9] Gnutella, "http://www.gnutella.com/," .

[10] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Workshop on Design Issues in Anonymity and Unobservability*, 2000, pp. 46–66.

[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM*, 2001.

[12] K. H. Wang and B. Li, "Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad-hoc Networks," in *Proceedings of the IEEE INFOCOM*, 2002.

[13] U. C. Kozat and L. Tassiulas, "Network Layer Support for Service Discovery in Mobile Ad Hoc Networks," in *Proceedings of IEEE INFOCOM*, 2003.