k-Anycast Routing Schemes for Mobile Ad Hoc Networks

Bing Wu and Jie Wu
Dept. of Computer Science and Engineering
Florida Atlantic University
bwu@fau.edu, jie@cse.fau.edu

***Abstract*—** *Anycast is a communication paradigm that was first introduced to the suit of routing protocols in IPv6 networks. In anycast, a packet is intended to be delivered to one of the nearest group hosts. k-anycast, however, is proposed to deliver a packet to any threshold k members of a set of hosts. In this paper, we propose three k-anycast routing schemes for mobile ad hoc networks. Our research work is motivated by the distributed key management services using threshold cryptography in mobile ad hoc networks in which the certification authority's functionality is distributed to any k servers. However, security is not the main focus of this paper. Our goal is to reduce the routing control messages and network delay to reach any k servers. The first scheme is called controlled flooding. The increase of flooding radius is based on the number of responses instead of increasing radius linearly or exponentially. The second scheme, called component-based scheme I, is to form multiple components such that each component has at least k members. We can treat each component as a virtual server as in anycast, thus, we simplify the k-anycast routing problem into an anycast routing problem. For the highly dynamic network environment, we introduce the third scheme, called component-based scheme II, in which the membership a component maintains is relaxed to be less than k. The performances of the proposed schemes are evaluated through simulations.*

Keywords: *Anycast, k-anycast, mobile ad hoc networks, routing, simulation.*

I. INTRODUCTION

Network routing provides a means to find a path between a source node and a destination node. The routing protocols can be roughly classified into unicast, broadcast, multicast, and anycast. Figure 1 illustrates different types of communication services. Figure 1 (a) shows unicast where the message is sent to a particular receiver D (also called *one-to-one* model). In broadcast (*one-to-all*), as shown in Figure 1 (b), the message is sent to *all* network hosts, namely hosts A, B, and C. Figure 1 (c) illustrates multicast (*one-to-many*), where the message is sent to all members of a particular group, for instance, host A and host C. In anycast, which is also referred to as a *one-to-any* model, the message is sent to any member out of a group, either A1, A2, or A3, preferably the nearest member, as shown in Figure 1 (d). Anycast is a communication paradigm that was first introduced to the suit of routing protocols in IPv6 networks [8]. In k-anycast as proposed in this paper, however, the message is sent to any k members out of a particular group, as opposed to anycast where there is only one receiver, as illustrated in Figure 1 (e). The cardinality of each group is at least k. For instance, when $k = 2$, {A1, A2}, {A1, A3}, {A2, A3}, and {A1, A2, A3} are the possibilities. Unicast and broadcast are two special cases of multicast with the number of members equal to 1 and n, respectively. Similarly, anycast is a special case of k-anycast with $k = 1$.

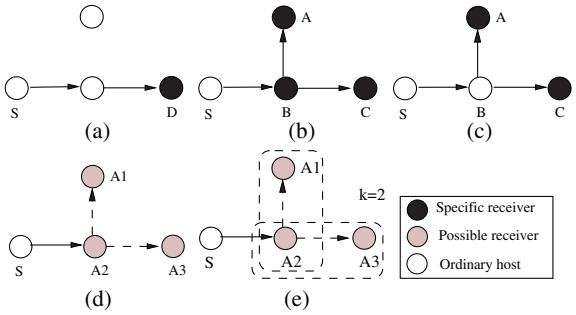

Fig. 1. Different routing services.

It is a common practice to deploy replicas of service providers in networks under certain circumstances. The potential advantages of this scheme are fault tolerance, load balancing, security, and improvement in system performance. For instance, server replication provides load balance by deploying multiple copies of servers and sharing client load across the replicas. Such schemes are used in domain name systems (DNS), mirrored web sites, and *ftp* servers. In this scenario, a user can locate any one server among the replica and not necessarily be concerned about which one actually provides the service. In mobile ad hoc networks (MANETs), anycast can provide easier service in a dynamic and distributed environment. For instance, in a situation where the network is partitioned because of host mobility or function failure, the host can still get service within the local component.

Further, in a network configuration where any threshold number of servers can provide particular services, a user needs to locate any k servers out of a server group. In this case, a k-anycast scenario appears. Here, we systematically study k-anycast in MANETs. It should adapt to the characteristics of the mobile and ad hoc environment. While unicast and multicast routing in MANETs has been extensively studied in recent literature, the research on anycast routing in MANETs has only recently been introduced. Most of the anycast routing protocols proposed for MANETs are extensions to the current unicast routing protocols [5][6], such as AODV, DSR, and TORA. To the best of our knowledge, this is the first literature on k-anycast routing in MANETs, though the term appears in

1-4244-0054-6/06/$20.00

[11]. There are possible network configurations where there are a group of hosts with different capabilities, yet a threshold number of hosts can provide equivalent functionalities and efficient k-anycast protocols are necessary to realize such needs. For instance, in a distributed public key management framework using threshold cryptography [12][13], a user can "assemble" a key from shares of any k servers. In fact, this assembly process can be done through a *k-any-gathering* process, where different shares are collected through the *reverse searching tree* built from k-anycast.

In this paper, we propose three k-anycast routing schemes. The first scheme is called controlled flooding. The increase in flooding radius is based on the number of responses and not merely increased linearly or exponentially. The second scheme is to form multiple components such that each component has at least k members. The third scheme also maintains components, however, each component can have number of members less than threshold k. In the second approach, we can treat each component as a virtual server as in anycast, thus, we simplify the k-anycast routing problem into an anycast routing problem. In the third scheme, service from multiple components can be aggregated instead of forcing each component to maintain k members, thus, it could reduce the invocation of the costly component recovery procedures, such as component merging. The major contributions of our research work are the following:

1) We introduce the k-anycast concept and its applications in MANETs.
2) We propose three k-anycast routing schemes in MANETs.
3) We present detailed operations of the k-anycast protocols.
4) We evaluate the performance of the proposed routing schemes in both static and dynamic network environments.

This paper is organized as follows: Section II reviews related work. Details of the k-anycast routing protocols are described in Section III. In Section IV, we present the experiment results. We conclude the paper and discuss the possible future work in Section V.

II. Related Work

A. *Anycast*

The notion of anycast was introduced by Partridge, Mendez, and Milliken in RFC 1546 in 1993 [3][8]. It was designed for anycast service within the IP Internet. Since then, providing anycast service to be used in IPv6 has been studied mainly in wired networks. Most of the research focused on anycast addressing, router selection, and server load balancing in wired networks. Jia [9] proposed an integrated anycast algorithm. The basic idea of this approach is that some routers execute multiple path routing (MPR) and the other routers execute single path routing (SPR). The assignment of routers to determine whether to execute SPR or MPR is based on the network topology and servers' load. However, such determination of routers in a distributed and dynamic network environment is not applicable without substantial modifications. Several extensions of anycast routing can be found in [10].

Recently, anycast was introduced in MANETs [2][4][7]. Although it is promising for anycast protocols to provide services in the dynamic, resource constrained, and distributed environment, anycast protocols are still at the relatively early stage of study in wired networks, and much less work has been done in MANETs. Park and Macker described anycast routing for mobile services [2]. Their idea is to extend the existing unicast protocols for the anycast service, such as extensions from link-state routing, distance vector routing, or link-reversal routing.

Likewise, Wang, Zheng, and Jia proposed an AODV-based anycast routing protocol in MANETs [6]. Similar to the unicast protocol AODV, whenever a new RREQ message is received, the RREP message is replied if there is a match of the anycast address, or there is a path in the routing table cache. The path with the least hop count could be selected in case there are multiple replies (routes). Wang, Zheng, Leung, and Jia [5] also proposed an extension of DSR-based anycast protocols in MANETs. Similar to the unicast DSR routing protocol, a node replies with the anycast route discovery message and includes its own address when there is a match of the anycast address or it has a route to the group address. While extension from existing unicast routing protocols to support anycast is straightforward, it does not, however, take advantage of the semantics of anycast for bandwidth efficiency.

B. *Multicast*

Many MANET applications need the mobile nodes to work together closely as a group to perform a task. Multicast provides a useful means for group communications. Some scalable multicast protocols have been proposed [14][15]. Although the multicast service can be achieved based on unicast, it is not efficient and could cause much unnecessary network traffic. The traditional multicast protocols can be classified into tree-based (such as DVMRP [16] and MAODV [17]) and mesh-based approaches (such as ODMRP [15] and FGMP [18]). In the former, all the routes form a tree infrastructure with the source node as the root, thus there is only one single path between every pair of sender and receiver. In the latter, a mesh infrastructure is maintained, thus, more than one path could exist between each sender and receiver pair.

Recently, some efforts have been made in the design of multicast routing protocols in MANETs for better performance by using overlay networks or a backbone to constrain the spread of state information. These protocols can be classified into the overlay-based, backbone-based, stateless, and hierarchical approach [1]. In overlay multicast, a virtual infrastructure is built on top of the underlying physical links among group members. The virtual network deals with the multicast functionalities while the underlying physical links provide a best-effort unicast datagram service. The backbone-based multicast protocols construct the state information by selecting some nodes to form a virtual backbone and by including state information being held in the backbone nodes. For the stateless multicast protocol, the routing information is not required to

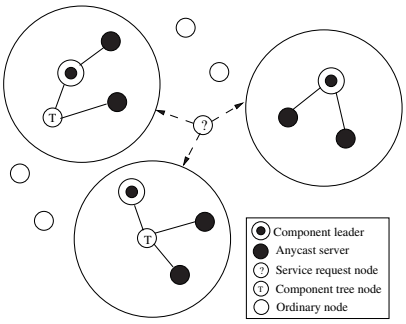

Fig. 2. Illustration of component architecture and k-anycast model.

be kept at forwarding nodes to further reduce overhead. Some of those protocols can be found in [19][20][21].

C. *Our approach*

In this paper, we introduce another type of group communication - k-anycast for MANETs, in which an application (or task) needs the collaboration of any k servers to complete. We introduce three k-anycast routing schemes for MANETs. A simple and obvious solution is flooding. In this approach, whenever a node needs to contact any k servers out of a server group, it floods the request. This can be done through the traditional linear or exponential expanding ring searching approach to limit the scope of flooding. As a baseline approach, we first introduce a controlled flooding scheme by adjusting the flooding scope according to the number of responses for a request. In addition, we also propose two advanced multi-component schemes. The basic idea of the first advanced scheme is to form multiple components with each component maintaining at least k members. Whenever a node demands service, it can contact any one of these components. In the second advanced scheme, components may have less than k members maintained. During service request, the service from multiple components could be aggregated instead of forcing each component to maintain k members, thus, it could reduce the cost for the invocation of costly component merging procedures. We present the detailed operations of the k-anycast routing in next section.

III. k-ANYCAST ROUTING SCHEMES

A. *k-anycast model*

We define a MANET as a graph $G = (V, E)$, where V is the set of all mobile nodes and E is the set of all edges between pairs of nodes. The routing request of k-anycast is to find the paths from a source to at least k destination nodes out of a group A, with $A \subseteq V$. Here, $k \leq |A|$, and $|A|$ denotes the cardinality of the set A. For instance, if there are 100 network nodes, and among them there are 40 server nodes, then, for $k = 5$, a node needs to contact any 5 servers out of the 40 servers and get the full service in combination from 5 servers.

B. *Overview of k-anycast schemes*

The basic idea of the first approach is quite straightforward. Whenever a node needs to contact any k servers out of a server group, it floods the request to the network with a limited TTL. The TTL keeps increasing until a node receives responses from at least k servers. The traditional approach for increasing the TTL is either linearly or exponentially. However, the traditional approach does not take the number of responses that a node has received (its current state) into account. Obviously, this approach can be improved in many ways. Here, we first introduce a simple extension in which a node predicts its future searching radius when necessary based on the number of responses it received. Obviously, in this scheme a node does not need the maintenance of any network structure.

The basic step of our second and third approaches is to initiate and maintain multiple components in a distributed way. To form a component, a server decides to act as a leader and sends out inviting messages (*join_component*) with its identity as the identifier of the component. The inviting messages propagate until the TTL is down to 0. The replies from servers propagate back to the leader. These processes repeat until the number of replies reaches k, which is a predetermined threshold value. The component formation phase is the same for both schemes. The difference between the second and third schemes is the component maintenance strategy. For the component-based scheme I, each component needs to maintain at least k members. Whenever affiliated members drop below k, certain procedures (such as sending inviting message, reconnecting broken link, or even merging components) are invoked. But for the component-based scheme II, components could have less than k members maintained and the service requesting nodes could aggregate service from multiple components, thus, no costly recovery procedures would be invoked.

Here, we use the term *component* instead of the term *cluster*. Although there are some similarities between the structure of a component and a cluster, there are fundamental differences. First, in our k-anycast protocols, a component leader is randomly selected in a distributed way, while in the clustering scheme, a clusterhead is elected through either an identifier-based or a connection-based process. There is a communication delay and overhead for the election process. Second, members of the component can be more than 1 hop away from the component leader, and the distance between members can be more than 2 hops away from each other, while cluster members are at most 2 hops away from each other and 1 hop away from the clusterhead.

From Figure 2 we can see that each component has a leader, chosen from the servers. Some non-server nodes might be needed to forward the inviting and replying message, and so become part of the component tree. For the component-based scheme I, since each component maintains at least k members, an ordinary node outside all components can query any component (any one of the three components in Figure 2). Each k-component can be treated as a virtual server as in the anycast model. Any existing anycast routing protocol proposed in MANETs can be utilized, or a traditional expanding ring searching scheme can be applied. A node inside component can query through the component tree. For the component-based scheme II, since each component could maintain less than threshold k members, an ordinary node

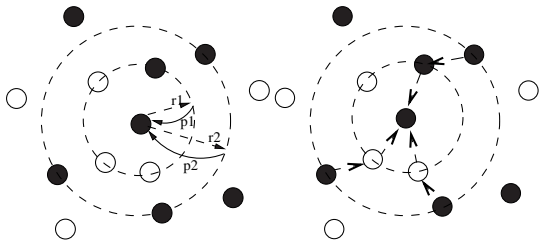

Fig. 3. Illustration of the controlled flooding k-anycast scheme.

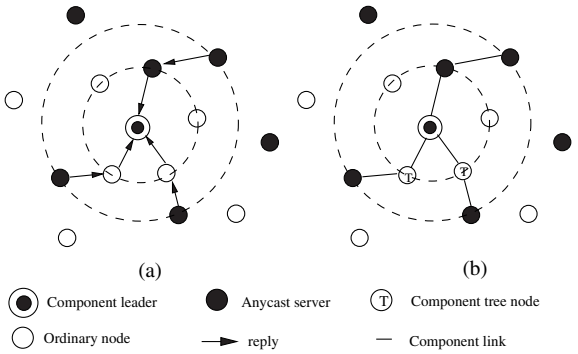

Fig. 4. Illustration of component initialization.

outside all components can query some of the components and aggregate responses from them. In subsequent discussion, we refer to the component-based scheme I as Scheme I and the component-based scheme II as Scheme II and use them interchangeably.

C. *Controlled flooding k-anycast routing scheme*

The first k-anycast routing scheme that we propose is a controlled flooding scheme. Whenever a node needs to contact any k servers out of a server group, it floods the request to the network with a limited TTL. When a node does not have enough responses for the flooded request it will increase the next TTL based on the current number of responses (p_i) it receives assuming $p_i < k$. For simplicity, we assume that nodes are uniformly distributed in the network area, then we can estimate the next expected TTL (r_{i+1}) as below,

$$\frac{\pi r_i^2}{p_i} = \frac{\pi r_{i+1}^2}{p_{i+1}} \Rightarrow r_{i+1} = \sqrt{\frac{p_{i+1}}{p_i}}\, r_i$$

Figure 3 illustrates the controlled flooding procedure. In this scheme a node does not need to maintain any network structure. While this scheme is simple and easy to implement, however, it produces relativly high network searching overhead.

D. *Component-based k-anycast routing scheme I*

Our second approach for k-anycast routing is to form and maintain multiple components in networks. The design of this component-based k-anycast routing scheme needs to answer the following basic questions. 1) How to initiate the component. 2) How to maintain the component. 3) How to perform k-anycast routing. 4) What is the advantage of this component-based scheme. Our description of the second scheme follows the order of answering above questions.

*1) **Component initialization**:* The basic strategy is that some servers decide to act as leaders locally, and each server should act as a leader alternatively from a long term view of the operation. The advantage is that it avoids the communication overhead and delay of the leader election process. A percentage is defined as ℓ ($0 < \ell < 1$). Initially, every server i generates a random number α_i between 0 and 1 and a threshold value ℓ. If $\alpha_i \leq \ell$, this server will become a component *leader*. Thus, there are about $\ell|A|$ component leaders in the network, where A is the server group and ℓ could be set so that $\ell|A| = \frac{|A|}{k} \Rightarrow \ell = \frac{1}{k}$. Note that $\ell|A|$ determines only the initial number of components. If this number is set to be too large, it will be reduced through a component merge process. If this number is set to be too small, more components will be formed dynamically after a time out.

Once a server decides to act as a component leader, it issues message *joinComponent* to invite servers to join its group. The component leader uses its identity as the component identifier. For simplicity, we use the notation *joinComponent_i* to indicate the inviting message sent by component leader i, and *memberCount_i* is the total affiliated servers in component i. The inviting message includes the number of members so far, the number of hops from the leader, the TTL of the message, and the sequence number. Initially the number of members affiliated is 1, the number of hops is 0, and the TTL is set to 2. The sequence number is to eliminate duplicate packets and routing loops.

The non-leader servers wait for the inviting message. They will reply to the first inviting message and ignore the subsequent messages. The TTL value decreases by 1 when the message is passed through. The message will be dropped if it is duplicated, or TTL is down to 0. The leader ignores inviting messages from other components. The non-server nodes need to forward the inviting and replying messages and become the component tree members of multiple components if they are on the path. If a server does not receive any inviting messages within the predetermined time, it will decide to serve as a component leader and send out inviting messages.

After a certain period of time, the leader will get some replies from servers. The leader counts the number of members $memberCount\_i$. If $memberCount\_i \geq k$, it notifies all members of the total number of affiliations. However, if $memberCount\_i < k$, the leader will expand the searching ring. The increase of search radius is bounded by the network diameter and may follow the same scheme as the controlled flooding that we described above. In case after the maximum number of tries, or search radius reaches network diameter, if $memberCount\_i < k$, a merge procedure needs to be invoked. The merge procedure is described later.

*2) **Component maintenance**:* As we know, in MANETs nodes could move around, switch on and off at will and possibly cease to function. All these factors make the network topology dynamic. Thereafter, member servers of a component could move away, a new server could join in, and the forwarding tree nodes in a component could switch to non-

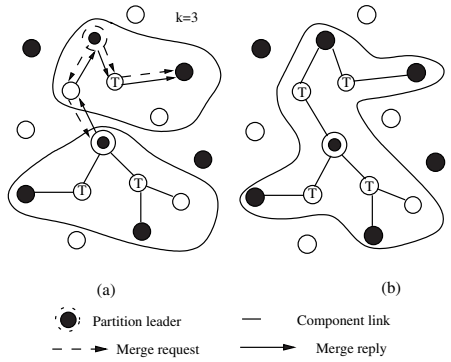

Fig. 5. Illustration of the component merge procedure.

forwarding component tree members. Thus, certain procedures are required to maintain the structure of a component.

The leader of a component sends out *refreshMember* messages periodically along the component tree structure instead of broadcasting and all member servers of the component reply to the message through the component tree links. We can use the notation *refreshMember_i* to indicate a member-refresh message sent by component leader i. This periodic member-refresh message is used for a leader to count members which are still in the component. It also helps member servers to detect their affiliation status. If a member server or a new server has not heard the periodical messages within the predetermined time, it will decide to serve as component leader and send out inviting messages the same way as in the initialization phase. Based on the replies, if the $memberCount\_i < k$, the leader knows that the component is partitioned, and need to invoke a merge procedure.

As we described above, when a component does not invite enough members or a component is partitioned and causes the number of component members to be less than k for a period of time, the leader of the component will issue a request to merge with neighboring component(s). Here, each single server that does not belong to any component can be treated as a component during the merge process. The merge request is broadcast with TTL set to the current maximum hops of the component and could increase if necessary as the approach described in the controlled flooding scheme. The increasing TTL is bounded by the network diameter. A merge request is replied by other component members. Once a component is involved in the merge process, it should not respond to another request. Based on replies received, the requesting leader can choose the component from which the combined members are more than k, otherwise, the largest component is selected. When there are multiple components available, the requesting leader can use the ID to break the tie (such as the smallest ID), or choose the nearest one, etc. The requesting leader will decide on one component and send a confirmation message back. The confirmation message should be forwarded to the leader of the other side. Meanwhile, an agreement must be made to decide the new leader of the merged component and notify all members. A simple solution is that the leader with the smallest ID becomes the new leader; of course, other options exist. Thereafter, two components are merged together and a new component is formed. This merge process may need to repeat until the total number of member servers is at least k. The maximum search range is still constrained by the network diameter. Figure 5 illustrates the operation of merging components. Figure 5 (a) shows one component leader sends out a merge request. Figure 5 (b) shows the component architecture after the merge procedure.

Note that as we mentioned in the component initialization and maintenance phases, servers that do not receive inviting messages or the periodical member refresh messages for a predetermined time will decide to act as leaders. Thus, in sparse networks, there could be more leaders than what is initially expected. However, through the component merge procedure, some leaders will change their status from leader to regular server. Thus, the total number of leaders will decrease. On the other hand, because of the merge procedure, the size of some components might keep increasing. However, some servers will switch to leaders because of node mobility, and then the size of the component will shrink. Therefore, the total number of components in the network will be balanced at $\frac{|A|}{k}$ and the member servers of a component will be maintained to be at least k dynamically.

3) k-anycast routing in Scheme I: Multiple components are formed and maintained in the network and each has at least k server members. Thus, every component can provide equivalent service. Therefore, each component can be regarded as an anycast server as in anycast protocols. Any existing anycast routing scheme could be utilized, or a simple expanding ring search can be made to locate any one of those components. Figure 6 (a) illustrates the component-based k-anycast routing model of Scheme I. When a node outside any components requests service, it can broadcast a request if no path has been discovered before. When a service request message arrives at a component, the component member who receives the request can send a reply message back to the requesting node, meanwhile, it could forward the request to the component leader. If the requesting node receives multiple replies it could select a component based on the response time, distance, or the number of members, etc. Inside a component, messages can flow through the tree structure efficiently.

E. Component-based k-anycast routing scheme II

Our third approach for k-anycast routing is similar to the second scheme as described above. In both component-based scheme I and scheme II, multiple components are formed and maintained in the network. In Scheme I, each component leader tries to maintain k members through a sequence of maintenance procedures which could be very costly when the topology is highly dynamic. In contrast, in Scheme II, a component leader may not necessarily maintain k members. Though the component leader selection is the same and the initialization phase is also similar with minor difference, the strategy of maintaining components between Scheme I and Scheme II is different. Therefore, in the description of Scheme II, we only give an outline of the initialization phase, and present details of component maintenance.

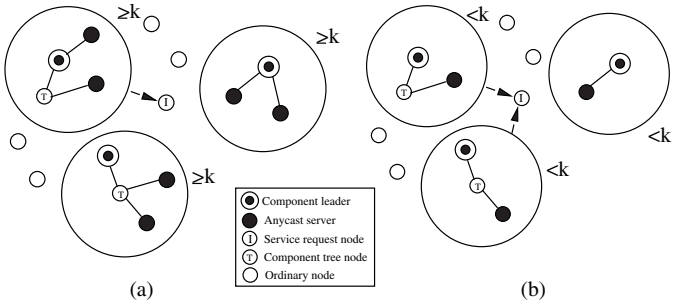

Fig. 6. Illustration of k-anycast routing model of (a) Scheme I and (b) Scheme II.

*1) **Component initialization**:* The initialization phase of component-based scheme II is quite similar to Scheme I. A predetermined number of servers will act as leaders and broadcast inviting messages. The TTL field of the inviting message can be set as described in the controlled flooding scheme. The maximum TTL is bounded by a predetermined value or the network diameter. Some of the major differences between Scheme I and Scheme II in this phase are: 1) If a component leader cannot invite enough servers to join its group, no merge procedure will be invoked in Scheme II. 2) In Scheme II, a server that has not received any inviting messages will not decide to be a leader, instead, it will move around and wait for incoming messages (the inviting messages in the initialization phase, or the periodical member reenforce messages in the maintenance phase). Therefore, no additional servers will function as component leaders during these phases.

*2) **Components maintenance strategy**:* In the component-based scheme I, relatively costly component maintenance procedures are required in order to make sure that each component has maintained at least k members. Obviously, when the mobility and the frequency of link breakage is very high, the maintenance of component structure is very costly. In Scheme II, we try to alleviate this situation. In Scheme II, a component leader periodically sends out *MemberReinforce* messages. The periodic message is broadcast with TTL set to the maximum hops of the current component. This message helps member servers to detect their affiliation status and also functions as an inviting message for a new server or a "disconnected" server. Some of the major differences between Scheme I and Scheme II in this phase are: 1) In Scheme II there is no component merge procedure being invoked even though total member servers of a component is less than threshold k. The *memberCount* will be updated to the current number of members affiliated, which could be much less than threshold k. 2) Unlike Scheme I, a "disconnected" server which has not received the periodical *MemberReinforce* messages will not decide to act as a leader, instead, it will move around and remain in the non-server status until it receives one or more incoming *MemberReinforce* messages from some components.

*3) **k-anycast routing in Scheme II**:* When a node outside any component requests service, it broadcasts a request if no path has been discovered before and waits for replies propagating back. In Scheme I, every component can provide equivalent service, thus, any existing anycast routing scheme could be utilized, or a simple expanding ring searching can be made to locate anyone of those components. In Scheme II, however, since the number of replies from one component could be less than k, which is not enough to complete a task (we have assumed that a task can only be completed by at least threshold k servers), the requesting node needs to collect replies from multiple components. If replies from multiple components are less than k, the TTL field of the service request message will have to be increased until at least threshold k replies are received. When there are multiple choices, the service request node could choose the least number of components and use ID as a general rule to break ties. Figure 6 (b) illustrates the component-based k-anycast routing model of Scheme II.

F. Advantages of component-based schemes

The component-based k-anycast routing schemes are based on underlying multiple components which are maintained as described above. Obviously, the maintenance of multiple components presents overhead and delays. However, from a long term point of view, the amortized cost of component maintenance is relatively low and is more efficient than the flooding scheme when service rate is high since it significantly reduces the redundance of flooding messages. Second, it is quite simple and much more efficient for servers to coordinate with each other within the component rather than searching for nodes scattered across the whole network. For example, in the distributed trust management scheme deployed in the network, servers need to work together to provide services, i.e. updating certificates. The relatively small size of components has obvious advantages for server cooperation. Third, maintaining multiple small components is more reliable and requires less maintenance overhead than maintaining one server group as described in [12]. Depending on the mobility of nodes, the larger the size of the component, the higher the frequency of link breakage and the invoking of costly maintenance procedures.

G. Summary

We described the controlled flooding scheme for the k-anycast service first where no network components were maintained. Then, we described the component-based Scheme I approach. The components were initiated by the leaders locally. The components were maintained by periodic beacons from the leader. The membership was dynamic because of the mobility of servers. The partitioned components could be repaired or merged. Since the k-components were maintained in the network, we turned the k-anycast routing problem into a simple version of the anycast routing problem in such a way that service requests within the component could be handled easily while service requests outside the component were handled by querying any one of those well-maintained components. To adapt to the highly dynamic network environment, we introduced a lightweight version of the component-based scheme, Scheme II, in which each component did not necessarily maintain a threshold k members. A service

request might be served by combining replies from multiple components when no neighboring component alone was able to serve the task.

IV. EXPERIMENTS

In this section, we give the simulation environment first and then present the experimental results. We also analyze the performance of different k-anycast schemes that we proposed.

A. Simulation environment

The simulation was implemented in Matlab. The simulation was conducted in a 100×100 2-D free-space by randomly allocating a given number of nodes in the range from 50 to 200. We assume every node has fixed transmission range $r = 20$. Two nodes are directly connected if their distance is within each other's transmission range. We conduct experiments in both static and dynamic network environments. In the dynamic environment, ρ is defined as the probability of movement for each host (ρ is 0.5 in our simulation). For each host in an update interval, a random number in $[0...1]$, is associated with each node. If the number is less than or equal to ρ, the corresponding host moves within the range of l units (l is 5 in our simulation) in any direction, but of course should be in the 100×100 2-D free-space. If the number is more than ρ, it represents that the corresponding host remains stable in the corresponding interval. Since each host has the same transmission radius, the network can be modeled as an undirected graph. In the experiments, we compared the cost of the flooding-based schemes (controlled flooding and exponential flooding) and the component-based schemes. For the component-based schemes, we evaluated the multiple component-based scheme I and scheme II with the one component scheme in which all servers form a single component.

B. Cost measurement

In the experiments, the cost is measured by the number of messages. For the component scheme the cost includes the component formation cost c_F, the component maintenance cost c_M, and the searching cost c_S. c_F is the average message cost during the component initialization phase, which includes leader inviting messages, intermediate relay messages, and server join messages. c_M is the average message cost per round during the component maintenance phase, which includes periodical member refresh messages, merge request and reply messages, and relay messages. c_S is the average message cost per service request during the service request phase, which includes requesting, replying, and inside and outside component relaying messages. For the flooding scheme, the only cost is the searching cost c'_S, which includes messages sent by requesters, intermediate relay, and reply messages. We compared the amortized cost for different schemes. For the component scheme, although the cost for component initialization and periodical maintenance is very high, the frequency of performing these tasks is relatively low compared to the service request rate. For instance, the cost of a one-time component initialization phase over a long term is relatively small. In the experiments, we defined that the frequency of periodical component maintenance was f_α, and the frequency of service request was f_β. So, for the component scheme the total cost over a period T is:

$$c_T = c_F + f_\alpha.c_M + f_\beta.c_S$$

and for the flooding scheme, the total cost is given by,

$$c'_T = f_\beta.c'_S$$

Obviously, with lower maintenance frequency f_α and higher service rate f_β, the component scheme could beat the flooding scheme in message cost.

$$c_F + f_\alpha.c_M + f_\beta.c_S \leq f_\beta.c'_S$$

if $c_F \ll f_\beta.c_S$,

$$f_\alpha.c_M + f_\beta.c_S \leq f_\beta.c'_S$$

That is,

$$\frac{f_\alpha}{f_\beta} \leq \frac{c'_S - c_S}{c_M}$$

We also conducted experiments with different threshold value k with $k = 5, 8, \text{and } 10$. The threshold k could affect the $c_F, c_M, \text{and } c_S$ in component schemes as well as the c'_S in flooding schemes. In the experiments, we assumed the server rate p (percentage of nodes that are servers) was 30%. We first show the experimental results for the static network environment, then follow by the experimental results of the dynamic network settings as described in above experiment section.

C. Result analysis

Figures 7 to Figures 8 show the results of the first experiment under static network settings.

Figures 7 (a), (b) (c), and (d) show the average cost of different flooding-based and component-based schemes when the threshold value k is 8, the request frequency f_β is 20, and the maintenance frequency f_α is $0.5, 1, 2, \text{and } 3$, respectively. The cost is measured by the amortized number of messages.

Figure 7 (a) shows the results for different schemes with $f_\alpha = 0.5$. The cost decreases as the number of nodes increase for both flooding schemes because of the increasing node density; however, the cost increases for the one component scheme, and the cost for multi-component scheme does not change significantly with the increasing number of nodes. The difference of the cost between the flooding schemes also decreases as the number of nodes increases. Figure 7 (a) also shows the cost of flooding schemes is higher than component-based schemes when the number of nodes is less than about 165 and the multi-component scheme has the least cost among all schemes.

Figure 7 (b) shows the results for different schemes with $f_\alpha = 1$. The flooding schemes have higher cost than the one component scheme when the number of nodes is less than 140. The cost of the multi-component scheme increases slowly with the increase of nodes. As f_α increases from 0.5 to 1, the

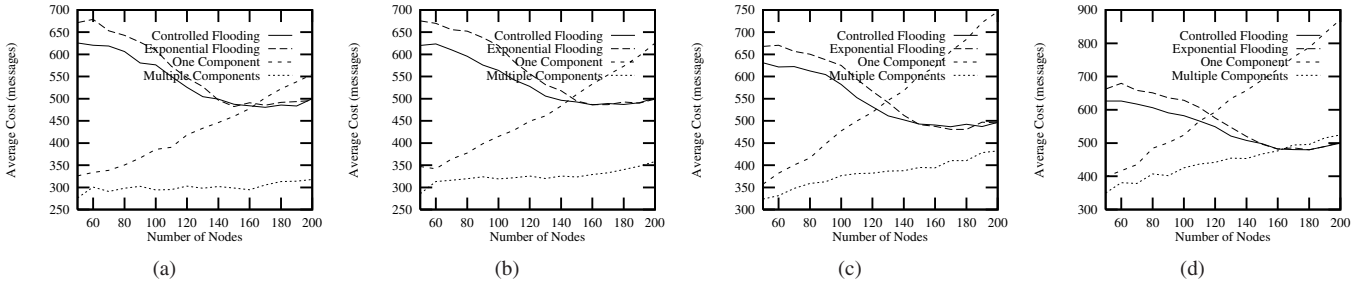

Fig. 7. Average cost in static settings when $k = 8$, $f_\beta = 20$, and (a) $f_\alpha = 0.5$, (b) $f_\alpha = 1$, (c) $f_\alpha = 2$, (d) $f_\alpha = 3$.

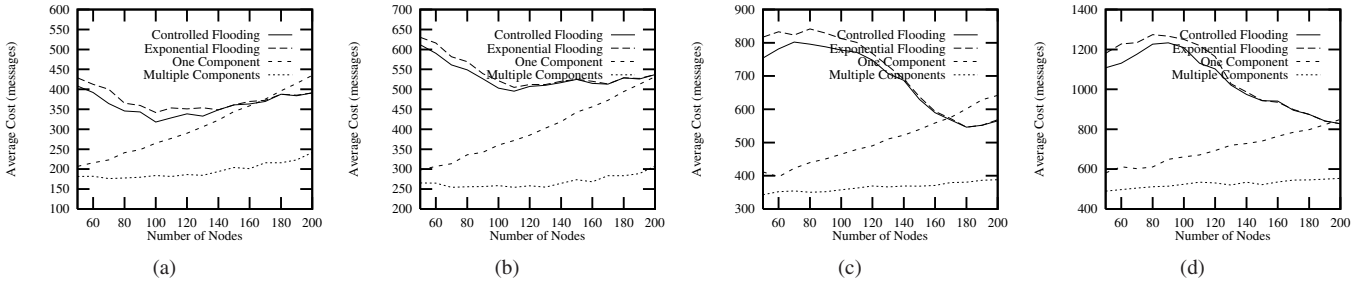

Fig. 8. Average cost in static settings when $f_\alpha = 0.5$ and (a) $k = 5$, $f_\beta = 20$, (b) $k = 5$, $f_\beta = 30$, (c) $k = 10$, $f_\beta = 20$, (d) $k = 10$, $f_\beta = 30$.

cost for the component-based schemes increases. For instance, when the number of nodes equals 200, the cost for the multi-component scheme is 550 when $f_\alpha = 0.5$, and is 625 when $f_\alpha = 1$. However, for the flooding-based schemes the cost remains unchanged because of no cost for any maintenance procedure.

Figures 7 (c) and (d) show similar results with maintenance frequency $f_\alpha = 2$ and 3, respectively. The cost for component-based schemes increases as f_α increases. From Figure 7 (d) we can see that when $f_\alpha = 3$ the flooding-based schemes outperform the one component scheme and the multi-component scheme when the number of nodes in the network is 120 and 160, respectively. In summary, from Figures 7 (a) to (d) we can see that with the increase of component maintenance frequency f_α the cost for component-based schemes increases, and the flooding-based schemes outperform the component-based schemes when the number of nodes in the network is large enough.

Figures 8 (a) and (b) show the average cost of different schemes when the threshold value k is 5, the maintenance frequency f_α is 0.5, and the request frequency f_β is 20 and 30, respectively. Figure 8 (a) shows that the cost decreases as the number of nodes increase for both flooding schemes when the number of nodes is less than 100. The cost increases slightly as the number of nodes grows more than 100; however, the cost of one component scheme increases, and the cost of multi-component scheme does not change significantly with the increasing number of nodes. When the number of nodes is more than 140, the cost of both flooding schemes is almost the same. Figure 8 (a) also shows the cost of flooding schemes is higher than the component-based schemes when the number of nodes is less than about 160 and the multi-component scheme has the least cost among all schemes. When the number of nodes is more than 160, the cost of the one component scheme is higher than the flooding schemes. Figure 8 (b) shows the results with $f_\beta = 30$ (f_α remains 0.5). From the figure we can see that the cost of all schemes increases with the increasing request frequency. The cost of flooding schemes (both controlled and exponential flooding) increases significantly compared with $f_\beta = 20$.

Figures 8 (c) and (d) show the results when the threshold value k is 10, the maintenance frequency f_α remains 0.5, and the request frequency f_β is 20 and 30, respectively. Figure 8 (c) shows that the cost decreases as the number of nodes increase for both flooding schemes; however, the cost increases for the one component scheme, and the cost for the multi-component scheme does not change significantly with the increasing number of nodes. Figure 8 (c) also shows the cost of flooding schemes is higher than the component-based schemes when the number of nodes is less than about 170 and the multi-component scheme has the least cost among all schemes. Figure 8 (d) shows the results with the request frequency $f_\beta = 30$ and the maintenance frequency $f_\alpha = 0.5$. The cost of flooding schemes (both controlled and exponential flooding) increases significantly compared with $f_\beta = 20$. In summary, from Figure 8 (a) to (d) we can see that with the increase of request frequency f_β the cost for both flooding-based and component-based schemes increases, and the multi-component scheme has least message cost among all, meanwhile, the one component scheme is also better than the flooding schemes when f_β is 30.

The experimental results of component-based schemes under dynamic network settings are shown in Figure 9 and Figure 10.

Figures 9 (a), (b), (c), and (d) show the average cost of the component-based schemes when the threshold value k is 8,

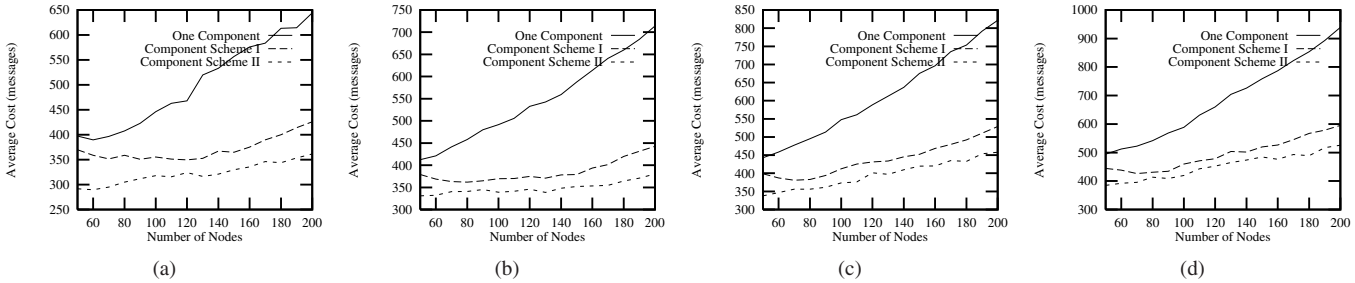

Fig. 9. Average cost in dynamic environment when $k = 8$, $f_\beta = 20$, and (a) $f_\alpha = 0.5$, (b) $f_\alpha = 1$, (c) $f_\alpha = 2$, (d) $f_\alpha = 3$.

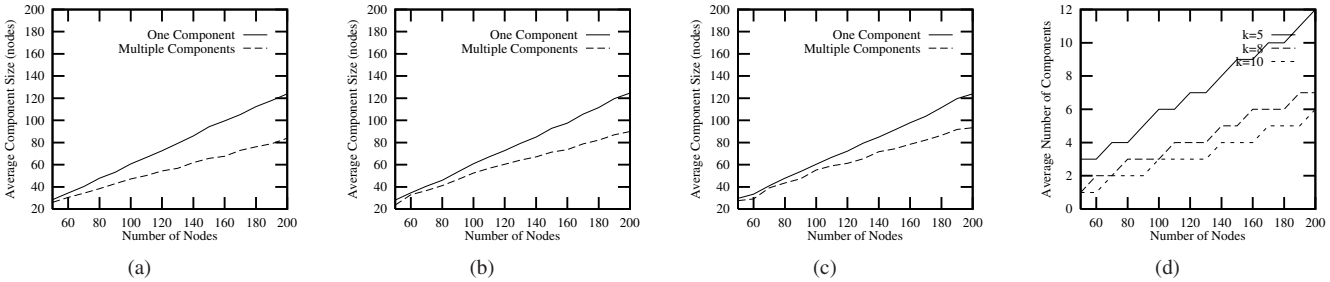

Fig. 10. Average component size in dynamic environment when (a) $k = 5$, (b) $k = 8$, (c) $k = 10$, and (d) average number of components when $k = 5, 8,$ and 10.

the request frequency f_β is 20, and the maintenance frequency f_α is $0.5, 1, 2,$ and 3, respectively. Figure 9 (a) shows that the cost increases for the one component scheme as well as the multiple component scheme I and scheme II. However, the cost of one component scheme increases faster than the multiple component schemes. The cost of Scheme I and Scheme II does not change significantly with the increasing number of nodes. Compared with the static network settings, the cost of component schemes in the dynamic network settings is higher, meanwhile, the cost of one component scheme increases more than the multiple component schemes (Scheme I and Scheme II). For instance, under the dynamic network settings, when the number of network nodes is 100, the cost of the one component scheme increases about 18% and for the multiple component scheme, the cost increases about 14%. Figure 9 (a) also shows the cost of Scheme I is higher than Scheme II with increasing number of network nodes. Scheme II has the least cost among all components schemes.

Figure 9 (b) shows the results for component-based schemes with $f_\alpha = 1$. The one component scheme has higher cost than Scheme I and Scheme II with the increasing number of nodes. The cost of the multi-component scheme increases slower than the one component scheme. For instance, when the number of network nodes is 100, the cost for the one component scheme in static and dynamic network settings is 410 and 480, respectively, which increases about 17%. For the multiple component scheme, the cost increases about 12%. Figures 9 (c) and (d) show similar results with maintenance frequency $f_\alpha = 2$ and 3, respectively. The cost of all component-based schemes increases as f_α increases. From Figure 9 (d), we can see that when $f_\alpha = 3$ Scheme II outperforms Scheme I while both multiple component schemes outperform the one component scheme. In summary, from Figures 9 (a) to (d) we can see that with the increase of component maintenance frequency f_α the cost of component-based schemes increases. Meanwhile, compared with the cost in the static network environment, the cost in the dynamic network environment is higher and the component Scheme II outperforms other component schemes.

Figures 10 (a), (b), and (c) show the average component size for the component-based schemes when threshold value k is 5, 8, and 10, respectively. Here, the component size includes the number of servers and the non-server component tree nodes. The component tree nodes connect servers (includes component leader) inside the component(s). From the figures we can see that the component size increases in accordance with the increase of network nodes. Figure 10 (a) shows the average component size of multiple component schemes is about 15% to 32% less than the one component scheme when k is 5. Figures 10 (b) and (c) show similar results when $k = 8$ and $k = 10$, respectively. From Figures 10 (a), (b), and (c) we can see that as the threshold value increases (k from 5 to 10), the component size increases while the one component scheme remains unchanged.

Figure 10 (d) shows the average number of components. The actual number of components maintained depends on the number of servers, the threshold value k, and the network topology. Figure 10 (d) shows the average number of components when the threshold is fixed, which is $k = 5, 8,$ and 10. Obviously, the smaller the threshold value is, the larger the average number of components. Meanwhile, as the number of network nodes increases, the component size increases. For instance, when total number of nodes is 100, there are 30 servers when server rate p is 30%. The number of components

should be at most 6 when $k = 5$, about 3 when $k = 8$, and at most 3 when $k = 10$.

D. *Result summary*

The results of the experiments can be summarized below:

- With increase of maintenance rate f_α (fixed f_β), message cost of component schemes increases. The flooding schemes outperform the component schemes at larger f_α.
- The increase of service request rate f_β (fixed f_α) causes the cost of all schemes increase. The component schemes outperform the flooding schemes at larger f_β.
- The increase of threshold k causes an increase of message cost of all schemes significantly; however, it does not affect one scheme more than the other.
- With increase of network density, the cost of component scheme increases while the cost of flooding schemes decreases. The cost of one component scheme increases more quickly than the multiple component schemes. The controlled flooding scheme has less cost than the exponential flooding scheme in a sparse network.
- In the dynamic network environment, the component-based schemes have higher cost than in the static environment. The cost increases from about 14% to 18% for the one component scheme and about 11% to 14% for the multiple component schemes under different settings, such as threshold values, maintenance frequency, and service request frequency. Scheme II has the least cost and outperforms other component-based schemes.
- With increase of threshold (fixed server rate), component size increases while the number of components decreases in component schemes.

V. Conclusion

In this paper, we propose the notion of k-anycast, give three k-anycast schemes, and describe the operational phases. In the first approach, controlled flooding scheme, the increase of flooding radius is predicted based on the number of responses for a request. In the second approach, multiple components with at least k members are maintained by the component leader's periodic beacons. The membership of components can be adjusted according to changes of network topology. Detailed procedures for handling possible component partitions are also presented. In the third scheme, which is also a component-based scheme, however, the number of members a component must maintain is relaxed to be less than k. The service responses from multiple components can be aggregated instead of forcing each component to maintain k members. The simulation results show that the multiple component schemes are more reliable, and requires less maintenance overhead because of reduced component size compared to the one-component scheme. However, there is a higher cost associated with maintaining the component structure compared to the flooding schemes. The amortized cost of component-based schemes is relatively low as the results show. An effort for component-based schemes has been made to better adapt to the dynamic network environment and to reduce the component maintenance cost. In our future research, we will extend the component-based k-anycast routing protocols and integrate them with a distributed trust management service in MANETs. In addition, some more in depth simulations are needed, especially in a dynamic environment.

Acknowledgment

This work was supported in part by NSF grants CCR 0329741, CNS 0422762, CNS 0434533, ANI 0073736, EIA 0130806, and the Federal Earmark Project on Secure Telecommunication Networks.

References

[1] C. Perkins, Ad Hoc Networking. Addison-Wesley, 2001.

[2] V. Park and J. Macker, Anycast Routing for Mobile Services. *Proc. of the 33rd Annual Conference on Information Sciences and Systems (CISS 99)*, 1999.

[3] S. Weber and L. Cheng, A Survey of Anycast in IPv6 Networks. *IEEE Communications Magazine*, pp. 127-132, 2004.

[4] G. Peng, J. Yang, and C. Gao, ARDSR: An Anycast Routing Protocol for Mobile Ad Hoc Network. *Proc. of IEEE 6th CAS Symp. on Emerging Technologies: Mobile and Wireless Communication*, pp. 505-508, 2004.

[5] J. Wang, Y. Zheng, C. Leung, and W. Jia, A-DSR: A DSR-Based Anycast Protocol for IPv6 Flow in Mobile Ad Hoc Networks. *Proc. of IEEE Vehicular Technology Conference: Symposium on Data Base Management in Wireless Network Environments*, 2003.

[6] J. Wang, Y. Zheng, and W. Jia, An AODV-Based Anycast Protocol in Mobile Ad Hoc Networks. *Proc. of the 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol 1-3, pp. 221-225, 2003.

[7] V. Park and J. Macker, Anycast Routing for Mobile Networking. *Proc. of the MILCOM'99*, 1999.

[8] C. Partridge, T. Mendez, and W. Milliken, Host Anycasting Service. Internet Draft, RFC 1546, 1993.

[9] W. Jia, Integrated Routing Algorithms for Anycast Messages. *IEEE Communications Magazine*, pp. 48-53, January 2000.

[10] D. Xuan, W. Jia, W. Zhao, and H. Zhu, A Routing Protocol for Anycast Messages. *IEEE Transactions on Parallel and Distributed Systems*, vol. 11 no. 6, pp. 571-588, June 2000.

[11] W. Wang, X. Li, and O. Frieder, k-Anycast Game in Selfish Networks. *Proc. of the International Conference On Computer Communications and Networks (ICCCN 2004)*, pp. 289-294, 2004.

[12] B. Wu, J. Wu, E. Fernandez, M. Ilyas, and S. Magliveras, Secure and Efficient Key Management in Mobile Ad Hoc Wireless Networks. Accepted to appear in *Journal of Network and Computer Applications (JNCA)*.

[13] H. Luo and S. Lu, URSA: Ubiquitous and Robust Access Control for Mobile Ad-Hoc Networks. *IEEE/ACM Transactions On Networking*, vol. 12, no. 6, pp. 1049-1063, 2004.

[14] S. Bae, S. Lee, W. Su, and M. Gerla, The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocol in Multihop Wireless Networks. *IEEE Network*, January/February 2000.

[15] S. Lee, W. Su, and M, Gerla, On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks. *Internet Draft*, draft-ietf-manet-odmrp-02.txt, January 2000.

[16] T. Pusateri, Distance Vector Routing Protocol. *in draft-ietf-idmr-dvmrp-v3-07*, 1998.

[17] E. Royer and C. Perkins, Multicast Operation of The Ad-hoc On-demand Distance Vector Routing Protocol. *Mobile Computing and Networking*, pp. 207-218, 1999.

[18] C. Chiang, M. Gerla, and L. Zhang, Forwarding Group Multicast Protocol (FGMP) for Multihop Mobile Wireless Networks. *Cluster Computing*, pp. 187-196, 1998.

[19] J. Xie, R. Talpade, A. Mccauley, and M. Liu, AMRoute: Ad Hoc Multicast Routing Protocol. *ACM Mobile Networks and Applicaions*, vol. 7, no. 6, 2002.

[20] C. Jaikaeo and C. Shen, Adaptive Backbone-based Multicast for Ad hoc Networks. *Proc. of IEEE International Conference on Communications (ICC 2002)*, NY, April 2002.

[21] L. Ji and M. Corson, Differential Destination Multicast - A MANET Multicast Routing Protocol for Small Groups. *Proc. of INFOCOM 2001*, pp. 1192-1202, 2001.