International Journal of Electronics Communication and Computer Engineering
Volume 4, Issue (6) NCRTCST-2013, ISSN 2249–071X

National Conference on Recent Trends in Computer Science and Technology (NCRTCST)-2013

Stateless Multicast & Multipath DSR in Adhoc Networks

N. Sandeep Chaitanya
Department of Computer Science,
CMRCET, Hyderabad

K. Vijaya Kumar
Department of Computer Science,
CMRCET, Hyderabad

P. Sruthi
Department of Computer Science,
CMRCET, Hyderabad

T.V. Suresh Kumar
Department of Electronics & Communications
Vijaya Engineering College, Khammam

SP Santhosh
Department of Computer Science,
HMITS

***Abstract* – Ad hoc networks are useful for providing communication support where no fixed infrastructure exists or the deployment of a fixed infrastructure is not economically profitable, and movement of communicating parties are allowed. Due to the dynamic topology, developing better routing protocol became a challenging task. The Dynamic Source Routing (DSR) protocol is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. Multi-path routing protocol is one of the approaches used to have less overhead, better bandwidth cost by distributing the load among a set of paths . However, due to interfering in the channels of the paths, multi-path increases the end to end delay and do not work well in highly congested networks. In this paper we propose a congestion aware multi-path Dynamic Source Routing Protocol. We generate a set of disjoint multi-paths and handle the problem of end to end delay using the correlation factor measurement. A set of path selection criteria have been used to improve the end to end delay. We also proposed Multicast routing protocols typically rely on the a priori creation of a multicast tree (or mesh), which requires the individual nodes to maintain state information. In dynamic networks with bursty traffic, where long periods of silence are expected between the bursts of data, this multicast state maintenance adds a large amount of communication, processing, and memory overhead for no benefit to the application. Thus, we have developed a stateless receiver-based multicast (RBMulticast) protocol that simply uses a list of the multicast members' (e.g., sinks') addresses, embedded in packet headers, to enable receivers to decide the best way to forward the multicast traffic. This protocol, called Receiver-Based Multicast, exploits the knowledge of the geographic locations of the nodes to remove the need for costly state maintenance (e.g., tree/mesh/neighbor table maintenance), making it ideally suited for multicasting in dynamic networks.**

***Keywords* – RB Multicast, Adhoc Networks, Multiupath DSR.**

I. INTRODUCTION

A very attractive and promising category of wireless networks that has emerged is based on an Ad Hoc topology; these networks are called Wireless Ad Hoc Networks. The term wireless network implies a computer network in which the communication links are wireless. The term Ad Hoc comes from the fact that there is no fixed infrastructure for forwarding/ routing the packets. We shows a simple MANET. The circles indicate communication ranges of individual nodes. In the real-world, this boundary is never likely to be a perfect circle and the links in fact can even be unidirectional in many cases – node 'A' can reach node 'B' on link 1 but node 'B' may not be able to use this link to reach node 'A'. This can happen due to the signal strengths of the two transmitters being unequal or can even be based on the transmission path. In Ad Hoc networks, each node is willing to forward data to other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity. Several routing protocol have been developed, those protocols can be grouped in two main classes: proactive and reactive protocols. Proactive algorithms employ classical routing strategies such as distance-vector routing (e.g., DSDV) or link-state routing (e.g., OLSR and TBRPF). They maintain routing information about the available paths in the network even if these paths are not currently used. The main drawback of these approaches is that the maintenance of unused paths may occupy a significant part of the available bandwidth if the topology of the network changes frequently. In response to this observation, reactive routing protocols were developed (e.g., DSR, TORA, and AODV). Reactive routing protocols maintain only the routes that are currently in use

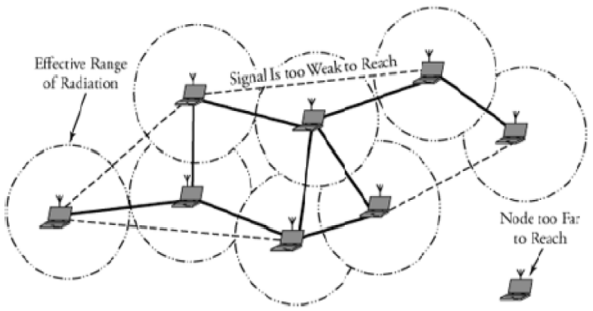

Fig.1. A Typical MANET

thereby reducing the burden on the network when only a small subset of all available routes is in use at any time. However, they still have some inherent limitations. First, since routes are only maintained while in use, it is typically required to perform a route discovery before packets can be exchanged between communication peers. This leads to a delay for the first packet to to be transmitted. Second, even though route maintenance for reactive algorithms is restricted to the routes currently in

use, it may still generate a significant amount of network traffic when the topology of the network changes frequently. Finally, packets en route to the destination are likely to be lost if the route to the destination changes.[6] In this paper, we propose an on demand approach to search for maximum number of node disjoint paths, then we have to find the correlation between the paths and how much each path is congested. Load distribute between paths based on the congestion metric calculated and result collected. In Identity-Based Cryptography (IBC), the public key of each user is derived from his identity which could be an arbitrary string. Each user needs to obtain from a server called private key generator (PKG) his private key for his identity. To encrypt a message for a user, only his identity and the PKG's public key are needed and no public key certification is needed. The PKG can be seen as an admission agent for an ad hoc group. When used in mobile ad hoc networks (MANETs), IBC has clear advantage over standard public key schemes in that others can send a message to an authenticated user without interaction or pre-arrangement (assuming the PKG's public key is universally known) — that is, non-interactive session key setup — which is a desired property when only a unidirectional channel exists between two nodes or accessing the certification authority (CA) is impossible; in standard public key schemes, users need to obtain the public key certificate from the recipient or the server. However, the main problem of adopting IBC in MANETs is that a centralized server is needed as the PKG, which violates the self-organization nature of a MANET. Nodes in a MANET usually belong to different users, implying difficulty in finding a trusted server to issue user private keys. The PKG task must be distributed among all users. We give a protocol for this, thus increasing the usability of IBC for MANETS. When users have no prior trust established, it would be tempting to use group key agreement to obtain a group key as the initial trust. However, the key agreement process needs to be conducted again whenever members join or leave the group and this rekeying process is not efficient in MANET and in most cases cannot survive or tolerate its changing topology. On the other hand, pre-computing all the group keys using key agreement is not feasible due to the huge key storage requirement. Distributing the public key certification task among users has been considered in [4]. Through the application of Feldman's verifiable secret sharing scheme, a construction for sharing the task of the IBC-PKG among all users is given. More specifically, the main contribution of this article is that a distributed PKG implementation for Boneh- Franklin's IBE [3] is presented, which allows the function of a trusted private key generator (needed for IBC) to be securely distributed among all the participating nodes in a MANET.

II. Dynamic Source Routing

The DSR protocol consists of two mechanisms: Route Discovery and Route Maintenance. Route Discovery is the mechanism by which a node **S** wishing to send a packet to a destination **D** obtains a source route to **D**. To perform a Route Discovery, the source node **S** broadcasts a ROUTE REQUEST packet that is flooded through the network in a controlled manner and is answered by a ROUTE REPLY packet from either the destination node or another node that knows a route to the destination. To reduce the cost of Route Discovery, each node maintains a cache of source routes it has learned or overheard, which it uses to limit the frequency and propagation of ROUTE REQUESTs. Route Maintenance is the mechanism by which a packet's sender **S** detects if the network topology has changed such that it can no longer use its route to the destination **D** because two nodes listed in the route have moved out of range of each other. When Route Maintenance indicates a source route is broken, **S** is notified with a ROUTE ERROR packet. The sender **S** can then attempt to use any other route to **D** already in its cache or can invoke Route Discovery again to find a new route.

III. Load Balancing Algorithm

At Source node: Initially source node does not have the location information of destination node so it broadcasts route request (RREQ) message for the route discovery. The route request message carries the source ID, destination ID, and a path vector which contains the relaying node ID, and amount of the traffic the relaying node has delivered. This RREQ is again forwarded by the neighboring nodes till the destination node has been reached; this mechanism is known as flooding. After broadcasting route request message Source node waits for route reply packets till predefined amount of time. Once this amount of time has expired after broadcasting route request message and no route reply packet has received the source node again broadcasts the route request message. Once the source node receives route reply packet it comes to know that a route has been build and starts transmitting data packets via received route.

At Intermediate node: When the intermediate node receives this RREQ packets it first checks its routing table weather its any of the neighboring node is active for that instant of time if yes then it drops the route request packet, else adds its node ID, load information in the path vector of the route request packet and again re broadcast it. When intermediate node receives route reply packet it first updates its routing table by making flag bit 1 (set) with the TTL value listed in the route reply packet along with the cache information and then uni cast this route reply packet and waits for the data packets to arrive. When the intermediate node receives data packets it first replaces the TTL value by the $ttl amount which is defined as the same

value of TTL listed in the route request packet received earlier. Once this $ttl time is out the flag bit is reset. When intermediate node relays a data packet, then for every data packet being relayed it updates its routing table by the respective load information and $ttl value being defined.

At the destination node: Once the destination node receives the route request message it first records the TTL value defined in the route request packet for that respective route and waits for predefined amount of time to collect other route request messages after collecting the various route's information, the destination node then chooses the best node whose path load is least among all the available paths. The destination node then simply swaps the path vector of the chosen route and after attaching the calculated $ttl value in the time field of route reply packet sends the route reply message and simultaneously set its flag bit. When intermediate node receives this route reply packets, it first set its activity flag i.e. *Aflag* = 1 then after recording the information in its route cache, unicast this route reply packet to the next hop node defined in the path vector of the route reply packet. When the source node receive this unicast route reply (RREP) packet it comes to know that a route has been build and then first set its flag bit and then starts data transmission. When any of the intermediate relaying node moves away from the transmission range of its upstream neighbor due to mobility, its upstream neighbor informs the source node by sending the route error (RERR) message. The source node upon receiving RERR message again floods the RREQ message to obtain the optimal route and process of route construction is repeated.[2][3][4]

Finding Dis-joint Multipaths

The broadcasting of RREQ is similar to the ordinary DSR, but instead of replying from the intermediate node cache, the RREQ will be stored in the cache table if the request is seen before, otherwise, it will be broadcast. When a RREP is generated, Redirection field is set to true if the generated path is a disjoint path with respect to the stored paths in the destination node otherwise, the field is set to false. When intermediate node receives the RREP, it redirects it to the next node in the RREP path if redirection field is set to true, otherwise, it will search for the shortest path from the cache table whose first hop is different from the first hop of the RREP path.

Correlation Factor Calculation

Since the proposed algorithm may generate some paths with length longer than the one used by the ordinary DSR and because of sharing channels between paths, the end to end delay is high comparing with DSR. Several criteria have been considered in selecting paths when attempting to send data. First criteria in selecting path is node disjoint paths, second criteria is to calculate the *correlation factor* (CF) between paths. CF is defined as the number of links connecting two paths. To find the CF, we have to find the neighbor nodes for each node in each path , then use this attribute to find number of links relating each node in the current path to the rest of paths. Figure 2 shows two paths with 7-related links.

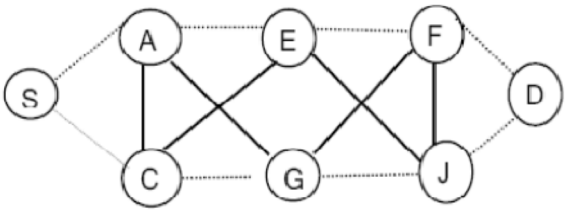

Fig.2. Two disjoint paths

The lower the CF is, the better the end to end delay is. The following equation is used to find the CF between two paths: If p1, p2 are two paths, then

CF = No. of links between p1, p2 / hop Count[p1] * hop Count[p2].

The third criteria is select paths with length less than or equal to a number defined by the user. Also the difference between shortest path and alternative ones must be small. Because if the difference is large this leads to having many unordered packet received at destination and need time to order them which increase queuing delay. Before source node start transmitting data packet, it needs to apply the third criteria then apply the CF equation. We must choose a CF threshold such that the final paths used to send data packet are related to each other by not more than the CF threshold chosen.

IV. RB MULTICAST

RB Multicast is a receiver-based cross-layer protocol that performs multicast routing based on receiver-based geographic unicast protocols such as XLM [10]. The receiver based unicast only needs the sender node's location and the final destination node's location, which are provided in the MAC packet, to decide the next hop along the route. We assume that the "void" (hole) problem in geographic routing is solved implicitly, for example, using the righthanded rule as in GPSR [23]. Throughout this paper, we will assume that the multicast members are stationary, such as multiple stationary sinks in WSNs or stationary roadside access points in vehicular ad hoc networks. The intermediate nodes can be either static or mobile. Although mobile intermediate nodes result in route breaks in conventional multicast protocols, since no multicast tree or mesh is used in RB Multicast, mobile intermediate nodes are supported at no additional cost in RB Multicast. Mobile destinations (multicast members) create a challenging problem for multicast protocols, and its solution is out of the scope of this paper.

Overview

Nodes in RBMulticast create what we call multicast regions" centered around themselves. There are several ways to create these regions. However, we use a quadrants approach due to its simplicity and good performance,

where each multicast region corresponds to one quadrant of the network, for a grid centered at the node. When a user initiates a request to send (RTS) a packet to a multicast group, data are passed down to the RBMulticast module in the protocol stack. Once the RBMulticast module

gets this packet, it retrieves the group list from its group table, assigns the group nodes to the multicast regions based on their locations, and using these locations, calculates a "virtual node" location for each multicast region. RBMulticast replicates the packet for each multicast region that contains one or more multicast members and appends a header consisting of a list of destination nodes (multicast members) in that region, Time to Live (TTL) value, and a checksum value. The destination of a replicated packet is the "virtual node" of the corresponding multicast region, which can be determined in several ways e.g., as the geometric mean of the locations of all the multicast members in that multicast region. In the end, all packets for all multicast regions are inserted in the

MAC queue, and are then broadcasted to the neighborhood. The node closest to the virtual node (within the available relay nodes as determined by receiver-based contention at the MAC layer) will take responsibility for forwarding the packet. The procedure for transmitting packets is summarized in pseudocode in Algorithm 1.

Algorithm 1.
RBMulticast Send
Require: Packet output from upper layer
Ensure: Packets inserted to MAC queue
1: Get group list N from group table
2: for node n in group list N do
3: for multicast region r in 4 quadrants regions R do
4: if n 2 r then
5: Add n into r:list
6: end if
7: end for
8: end for
9: for r 2 R do
10: if r:list is non-empty then
11: Duplicate a new packet p
12: Add RBMulticast header (TTL, checksum,
r.list) to p
13: Insert p to MAC queue
14: end if
15: end for

When a node receives a multicast packet, RBMulticast first examines the checksum in the packet header, and drops the packet if any corruption exists in the packet. It also drops the packet if it is not in the forwarding zone. The forwarding zone is the area within the radio range of the sender that has a smaller distance to the destination than the sender-destination distance.

After a node receives a multicast packet, it then retrieves the destination node list from the RBMulticast packet header. If this node is inside the destination list, it removes itself from the list and passes a copy of the packet to the upper layers in the protocol stack. RBMulticast then checks the TTL value and drops the packet if the TTL is lower than 0. Finally, if there still remain nodes in the destination list, multicast regions and virtual nodes are recalculated, and new packets are generated if required. The packets (one per multicast region that contains multicast members) are then inserted in the MAC queue for transmission. The procedure executed after receiving packets is summarized in pseudocode in Algorithm 2.

Algorithm 2.
RBMulticast Receive
Require: Packet input from lower layer
Ensure: Forwarded packets inserted to MAC queue
1: Calculate checksum. Drop packet if error detected
2: Drop packet if not in Forwarding zone
3: Get destination list D from packet header
4: for node d in destination list D do
5: if I am d then
6: Duplicate the packet and input to upper layer
7: Remove d from list D
8: end if
9: end for
10: if TTL in header ¼ 0 then
11: Drop the packet
12: return
13: end if
14: for d 2 D do
15: for multicast region r in 4 quadrants regions R do
16: if d 2 r then
17: Add d into r:list
18: end if
19: end for
20: end for
21: for r 2 R do
22: if r:list is non-empty then
23: Duplicate a new packet p
24: Add RBMulticast header (TTL _ 1,
checksum; r:list) to p
25: Insert p to MAC queue
26: end if
27: end for

Fig. 3 gives an example of how RBMulticast is employed. The two multicast regions, the southwest and northwest quadrants, contain only one multicast member each, and thus a packet is sent directly to these multicast destinations.

The northeast multicast region has three multicast members, and thus a single packet is sent to the virtual node located at the geometric mean of the locations of the multicast members (dotted circle with label 3 in the figure). The southeast multicast region has no multicast members, and hence no packet is transmitted into this region. Once a packet sent toward a virtual node reaches an intermediate node for which the multicast members are

no longer in the same multicast region, the node will split off packets to each of the multicast regions accordingly.

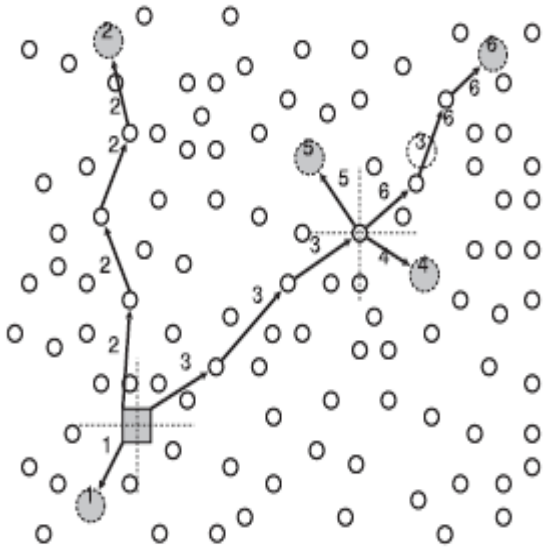

Fig.3. Example showing how RBMulticast delivers multicast packets. The source node is the square node. Multicast members are shaded circles, and virtual nodes are dotted circles. Because every destination node will become a virtual node at the end, they are all shown with dotted circles. The number on the side of the lines indicates the destination of that packet.

RBMulticast Header

The goal of a stateless approach is to keep intermediate nodes from having to store any data for routing and medium access. This is possible only if all information required to multicast a packet is carried along with the packet. The question is how much information the multicast packet needs to carry for successful delivery to all multicast members. Fig. 4 shows the structure of an RBMulticast header. The first byte Protocol ID is for protocol identity in the protocol stack [24]. TTL provides a maximum time, in hop number, that a packet should last in the network. Type Of Service (TOS) indicates four kinds of packets in RBMulticast, which are “data,” “join,” “leave,” and “update” packets. The update packets are used in group management and periodic group list updates. Destination List Length (DLL) indicates how many nodes are in the node list, and thus will determine the length of the header. The RBMulticast header size is not fixed since the destination list length is variable. Source Address is the address of the source node, which equals the RBMulticast group ID of this packet, and Destination List Address stores the locations of the DLL destination nodes. The RBMulticast group ID is not actually needed in this protocol since all the multicast members are included in the packet header. Because we assume a receiver-based MAC layer, the next hop is determined by a joint decision among potential receivers. Hence, the RBMulticast header does not need to carry any state for routing the packet. However, we still need to decide when the packet must be split off to different destinations. This is usually implied by tree branches in tree based multicast approaches. Because of the location information assumption, wecan use multicast regions to decide when packets must be split off without any tree structure. A packet will be split off to each multicast region if multicast members exist in these regions. Therefore, a destination list is the only requirement for multicast packet delivery: this destination list must be carried inside the packet header. As with any multicast protocol that uses a destination list, the packet header length will increase linearly with the number of destination nodes. The maximum number of multicast members allowed in a group is restricted by the packet size. For packets in the IEEE 802.15.4 standard of Wireless Sensor Networks, the maximum packet size is 128 bytes, and hence the maximum number of nodes in the destination list is around 50, which is sufficient for practical purposes.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Protocol ID								Time to Live							
Type of Service								Destination List Length							
Check Sum															
Source Address															
Destination List Address 1															

Fig.4. Packet header of the RBMulticast protocol

V. CONCLUSION

In this paper we introduced the use of multi-path DSR. The resultant protocol will generate set of highly disjoint paths and calculate the correlation factor between the path to decrease the end to end delay. Its shown than after handling the correlation factor end to end delay improved and overhead as well. The proposed protocol will create a network with better performance, less number of packets dropped, and throughput will be increased compared to conventional DSR. In future we will handle the problem of congestion to have better packet delivery as well as better overhead and end to end delay. We also presented a new stateless multicast protocol for ad hoc networks called Receiver-Based Multicast. RBMulticast uses geographic location information to route multicast packets, where nodes divide the network into geographic “multicast regions” and split off packets depending on the locations of the multicast members. RBMulticast stores a destination list inside the packet header; this destination list provides information on all multicast members to which this packet is targeted. Thus, there is no need for a multicast tree and therefore no tree state is stored at the intermediate nodes. RBMulticast also utilizes a receiver-based MAC layer to further reduce the complexity of routing packets.

REFERENCES

[1] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. 1996 IEEE Symposium on Security and Privacy*, pp. 164–173.

[2] D. Boneh and M. Franklin, "Efficient generation of shared RSA keys," *J. ACM*, vol. 48, no. 4, pp. 702–722, July 2001.

[3] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. 2001 Advances in Cryptology*, pp. 213–229, 2001.

[4] S. Capkun, L. Butty´an, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 2, no. 1, pp. 52–64, Jan-Mar 2003.

[5] A. C-F. Chan, "Distributed symmetric key management for mobile ad hoc networks," in *Proc. IEEE INFOCOM 2004*, pp. 2414–2424.

[6] A. C-F. Chan, "Cryptographic key management revisited," Ph.D. thesis, University of Toronto, 2005.

[7] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 1987 IEEE Symposium on Foundations on Computer Science*, pp. 427–437.

[8] R. Gangishetti, M. Choudary Gorantla, M. D. Das, and A. Saxena, "Threshold key issuing in identity-based cryptosystems," *Computer Standards & Interfaces*, vol. 29, no. 2, pp. 260–264, 2007.

[9] V. Goyal, "Reducing trust in the PKG in identity based cryptosystems," in *Proc. 2007 Advances in Cryptology*, pp. 430–447.

[10] C.-H. Feng and W.B. Heinzelman, "RBMulticast: Receiver Based Multicast for Wireless Sensor Networks," IEEE Wireless Comm. And etworking Conf. (WCNC '09) Apr. 2009.

[11] Akyildiz, M. Vuran, and O. Akan, "A Cross-Layer Protocol for ireless Sensor Networks," Proc. Conf. Information Science and Systems (CISS '06), Mar. 2006.

[12] .-J. Lee, M. Gerla, and C.-C. Chiang, "On-Demand Multicast Routing Protocol," Proc. IEEE Wireless Comm. and Networking Conf. (WCNC' 99), vol. 3, pp. 1298-1302, 1999.

[13] Garcia-Luna-Aceves and E. Madruga, "A Multicast Routing Protocol for Ad-Hoc Networks," Proc. IEEE INFOCOM, vol. 2, pp. 784-792, Mar. 1999.

[14] R. Vaishampayan and J. Garcia-Luna-Aceves, "Efficient and Robust Multicast Routing in Mobile Ad HOC Networks," Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems, pp. 304-313, Oct. 2004.

[15] E.M. Royer and C.E. Perkins, "Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol," Proc. ACM MobiCom, pp. 207-218, 1999.

[16] J.G. Jetcheva and D.B. Johnson, "Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks," MobiHoc '01: Proc. Second ACM Int'l Symp. Mobile Ad Hoc Networking and Computing, pp. 33-44, 2001.

[17] C. Wu and Y. Tay, "AMRIS: A Multicast Protocol for Ad Hoc Wireless Networks," Proc. IEEE Military Comm. Conf. (MILCOM '99), vol. 1, pp. 25-29, 1999.

[18] K. Chen and K. Nahrstedt, "Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in Manet," Proc. IEEE INFOCOM, vol. 3, pp. 1180-1189, 2002.