Tree based Xcast Routing and Flooding based Group Forming for Ad hoc Networks

Fumihito Kan* Nobuo Kawaguchi†

Abstract

Multicast is becoming increasingly important for applications such as IP telephony and video-conferencing because of its ability to efficiently send data to a group of destinations. However, it has the scalability issues for supporting a very large number of distinct multicast groups. To overcome these issues, Xcast has been proposed as an alternative solution. In this paper, we consider the situations in which a small number of nodes wish to communicate with each other in Ad hoc Networks. Taking advantage of the ability of Xcast for supporting a large number of small groups, we propose a new Xcast-like routing scheme named TXR along with a completely new group membership administration scheme, FGF. With these schemes the users can easily get/provide information from/to other users efficiently even in a temporary or an emergency situation in which neither network infrastructure is provided nor stable network topology guaranteed.

1 Introduction

In this paper, we propose a new scheme, Tree based Xcast Routing (TXR), based on Xcast (Explicit Multi-unicast)[1] which would work more efficiently than Ad hoc multicast(e.g., MAODV[3], AMRIS[4], ODMRP[5], Amroute[6]) when applied to a large number of small groups rather than a small number of huge groups. With this new scheme, users can easily get/provide information from/to other people in Ad hoc Network. Like using multicast, any one of members could send information to the others, and the exactly same information will be distributed efficiently to all other members joining the group.

*Graduate School of Information Science, Nagoya Univ., Japan

†Graduate School of Engineering, Nagoya Univ., Japan

When users wish to begin a group session in an Ad hoc Network, they start their laptops with IPv4 addresses auto configured or IPv6 link local addresses setup. We do not assume any multihop protocol working on this Ad hoc network though. Therefore each node could only communicate with its direct neighbors, but not any indirect neighbor.

To communicate with other nodes including indirect neighbors and to begin a group session, users start our TXR application on their laptops. To be a source node providing information to other users, the application first performs our new membership administration algorithm, FGF, and then uses TXR to send information to others. To get information from a source node, the application simply waits for *session advertisements* and joins the session. We show these processes in more details in the following sections.

Since the format of information could be diverse (e.g., text, file, video etc.), this scheme could be used to many applications. We introduce a couple of services which will be contributed by our application in the later chapter.

This paper is organized into eight sections. In the next section, we compare IP multicast[2] and Xcast broadly and give several reasons to adapt Xcast instead of IP multicast for the situation we assume. The third section focuses on the main topic of this paper, our proposal TXR along with FGF. In the fourth section, we presents an example scenario. In fifth section, we introduce some services which will be fairly contributed by our application. And we conclude the paper with a summary and future work.

2 Motivation

Multicast is becoming increasingly important for applications such as IP telephony and video-conferencing because of its ability to efficiently send data to a group of destinations. However, while traditional IP multicast schemes

are scalable for huge multicast groups (e.g., the audio and video multicasting of a presentation to all employees in a corporate intranet), they have scalability issues for a very large number of distinct, especially temporary multicast groups (e.g., videoconference involving 3 or 4 parties) because the number of addresses assigned for multicast is limited and some efforts are required to create a new multicast group.

Xcast solving those issues and complementing the existing schemes is more suitable for a very large number of small multicast sessions[1]. This is achieved by explicitly encoding the list of destinations in the data packets and keeping track of the destinations in the multicast channel that it wants to send packets to, instead of using a multicast group address as a logical identifier. More precisely, the source encodes the list of destinations in the Xcast header, puts the nearest one in the destination address, and then sends the packet to a router. Each router along the way parses the header, bundles up destinations which have same next-hop, and forwards a packet with an appropriate Xcast header to each of the next hops. When there is only one destination left in the destination list, the Xcast packet can be converted into a standard unicast packet. Also, Xcast does not need for a special multicast routing protocol such as PIM.

In Ad hoc Networks, although during recent years many multicast protocols have been designed specifically for Mobile Ad hoc Networks(e.g., MAODV[3], AMRIS[4], ODMRP[5], Amroute[6]), they may become less efficient and more expensive to function as well, when applied to use with small and sparsely distributed groups because of their scalability issues mentioned above[7].

Therefore it seems more suitable to use Xcasting approaches instead of IP multicating ones when applied to use with a huge number of small groups both in wired networks and wireless ones. Yet the current Xcast only supports wired networks. In this paper, we adapted the basic concept of the current Xcast and added several new features in it to make it work more efficiently on Ad hoc Networks.

3 Group Forming and Tree based Xcast Routing for Ad-hoc Network

3.1 Broad Concepts

As we mentioned in the previous sections, it seems suitable to use Xcasting approaches when applied to use with a huge number of small groups both in wired networks and wireless ones. Yet the current Xcast only supports wired networks. In this paper, we adapted the basic concept of the current Xcast and added several new features in it to make it work more efficiently on Ad hoc Networks. Although we consider mainly building an application working on Ad hoc Network in this paper, the concept can be also applied to lower layers.

The new scheme we propose consists of two main parts. For the first part, Flooding based Group Forming (FGF) is designed as an alternative solution against "xcgroup", the current group membership administration scheme in Xcast. This absolutely new scheme is more suitable than "xcgroup" when applied to Ad hoc Networks, because of its flexibility and independence of need to a central server. For the second part, Tree based Xcast Routing (TXR) is proposed for providing a Xcast-like routing scheme. Unlike current Xcast routing, in TXR the source node constructs a *Tree Table* based on the network topology and other nodes use the *Tree Table* to forward the packets.

In the next subsection, we explain FGF, and we describe TXR in the "Tree based Xcast Routing" subsection. Note that to make the mechanism more clear we gave two different names to member administration mechanism and routing mechanism respectively. However it may be more suitable to consider FGF as a part of TXR scheme. And we will define a common header format for both of them.

3.2 Membership administration

To provide group session services, we must figure out how to deal with membership administration at first. That is, to provide information to other nodes, first the provider must know who are joining the group and how to reach them. In this subsection, we propose a new scheme named FGF to solve the membership administration problem in Ad hoc Emergency Networks.

In Xcast, the sender must know the addresses of the other group members to which the packets should be sent. In current Xcast implementation, a scheme called

"xcgroup" has been used to provide membership administration. In this scheme, every node contacts with a central server to get membership information (e.i., the addresses of the other nodes which belong to same group session). However, in an Ad hoc networks, it seems unsuitable to have a central server for membership administration. Firstly, since the server wouldn't have a static name, nodes in the network do not know the server address when joining the network. Although the nodes could broadcast the server searching packets, it seems inefficient that every nodes broadcast packets to all of the other nodes in the network to find a central server. Secondly, because the network topology can change easily anytime, connectivity with the server is not always guaranteed. Thirdly, and the most important, to have a server locate in every Ad hoc network is against the principle of this kind of networks. Therefore we need a new scheme which may efficiently control group membership without a central server.

Aiming at solving the previous issues, we propose a new scheme, Flooding based Group Forming (FGF). With FGF, nodes in an Ad hoc networks can easily find and join certain group sessions without the aid of group administration servers. We explain the general concept here, and will give a clear example later.

Basically, as the name Flooding based Group Forming implies, the source node[1], firstly floods a *session advertisement* [2] (Instead of using multicast address as identifier, we use text type group session name, which could be "Food-Info" for example). As the session advertisement floods the network, every nodes receives this advertisement, and adds their addresses into the packet no matter whether they want to join the group or not. This is because even if they are not joining the group they construct the route between other nodes and the source node. The nodes which wish to join the session group set the *Member Flag* in the packet to "1", otherwise, leave it unchanged. As the advertisement passes around the network, the addresses in the packet grow up and eventually up to maximum which will be determined by network topology, *TTL*(Time To Live), and other elements, and then the last node sends the packet which contains all of the addresses along the path back to the source node. The source node receives the return packets and looks them up to see who is joining the group. Besides TTL, we define *TIME OUT* to be the alternative determination of the timing to return the session advertisement. Also, we define *Sequence Number* to avoid loop problem. Also, to guarantee the uniqueness of the session name, we do not allow any session with the name same as the existing ones. We will explain the design of FGF in more details and present a clear example in later chapters.

[1]The initiator, the node having information to send to others

[2]The advertisement to inform others of the new group session name

3.3 Tree based Xcast Routing

In this subsection we give a solution to deal with the routing problem(e.g., how to reach the nodes joining the group session), which is named Tree based Xcast Routing (TXR). By using our proposal nodes don't have routing information (e.g., routing table) themselves. Routing process is done by our tree based routing method.

After flooding *session advertisement* as described in the previous subsection, the initial node gets the whole tree structure of joining nodes. The initial node then constructs a *Tree Table* based on the tree structure. And the nodes sends the packets with the *Tree Table* inside the headers. When other nodes receive the packets, they look at the *Tree Table* to decide how to forward the packet to other nodes.

Although we could define the tree structure as simpler form (e.g., combinations of lists or a bitmap graph), we want to curtail as much as possible the calculations occurred on the nodes along the path which behave like routers. To do so, we define our *Tree Table* which makes the calculations occurred on the nodes trivial.

Now, we give an example for our *Tree Table*. Suppose the tree structure to be Fig. 1.

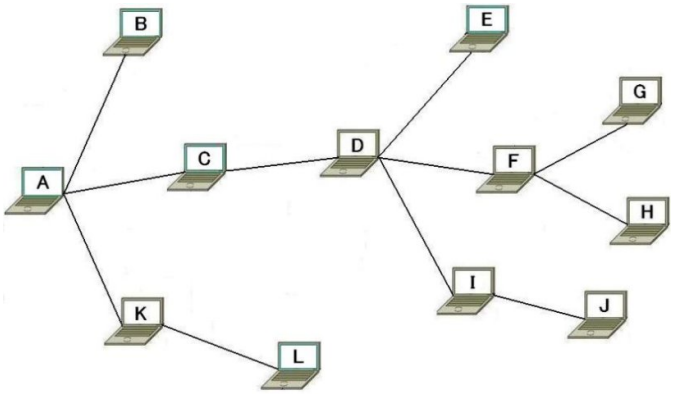

Figure 1: Example for Tree Table

In Fig. 1 to make it easier to see we assign each node an alphabet in depth-first order. And our *Tree Table* should look like Fig. 2. To construct this *Tree Table*, first, we arrange the addresses of all nodes in depth-first order and assign each of them an ID number, A is 1, B is 2, and so on.

In Fig. 1 A has three subsets which are {B}, {C to J}, and {K to L}. Converting the subsets to ID number, we get {2}, {3 to 10} and {11 to 12}. To make the table smaller, we can represents them as {2 3 11 LAST[3]}. B is a leaf, so we assign ϕ for it. C has only one subset which can be represented as {4 11}. Likewise, the other nodes can be represented by same concept. Therefore, our *Tree Table* would look like Fig. 2.

ID	Address	Sub Nodes
1	A	2 3 11 LAST
2	B	ϕ
3	C	4 11
4	D	5 6 9 11
5	E	ϕ
6	F	7 8 9
7	G	ϕ
8	H	ϕ
9	I	10 11
10	J	ϕ
11	K	12 LAST
12	L	ϕ

Figure 2: Tree Table for the example

Now, we explain how a node calculates the *Tree Table*. For example, after receiving the *Tree Table*, node D only looks at its line which is {5 6 9 11}. With this information, D knows that its subsets are {E}, {F,G,H}, and {I,J}, so it modifies the *Receiver Flag* and sends three packets to E, F and I respectively. This calculation is trivial enough.

Note that the receivers who get the *Tree Table* can also use it to send packets without flooding an advertisement themselves. Yet the tree structure represented by it may not be the best structure for them.

[3]We define LAST as (the total nodes number + 1), which is 12+1=13 in this example

3.4 TXR Header

Considering the previous requirements, our TXR header[4] would look like Fig. 3. Note that this figure is just a broad outline. We don't specify concrete design for its header here. Therefore the figure is not drawn in real scale.

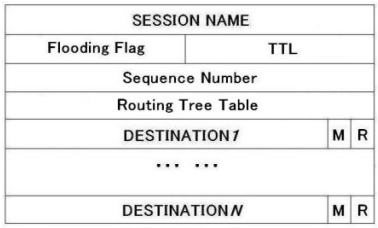

Figure 3: TXR Header

SESSION NAME. The name for a certain session (e.g., Food-Info, Water-Info).

Flooding Flag. The flag to distinguish flooding messages(session advertisements) from ordinary TXR packets. "1" is for flooding messages, "0" otherwise.

TTL:Time To Live. This integer will be decreased as the packet passes each node. In FGF mechanism, once it reaches to zero, the last node has to send the packet back to the source node.

Sequence Number. We define *Sequence Number* to solve loop problem. This number is defined by the address of the source node and a sequential number. If packets with the same source addresses and sequential numbers have arrived, we see this arose because they get there via different routes. In this case, the receiver simply discards the duplicated ones. If there has arrived a packet with the same source address but a smaller sequential number compared with the previous one, we see this arose from delay.

Routing Tree Table. This block could be seen to be one of the main parts in our source routing mechanism. We represent the whole tree structure of the network topology as *Tree Table*.

DESTINATION N[5]. The destination blocks. The number of destinations will be determined by network topology, *TTL* and *TIME OUT*. We define *TIME OUT*

[4]Note that although we gave two different names to member administration mechanism and routing mechanism respectively, FGF can be considered as part of TXR scheme. And we define a common header format for both of them.

[5]An integer

in the end of this subsection, because it does not appear in the TXR header.

M:Member Flag. This bit denotes whether the destination is a member of the session or just a node along the path. If the bit is set to be "1", the destination is a member of the session group, and the packets will be received by the Xcast application running on the node. Otherwise it's only a node we need to route the packet to other nodes.

R:Receiver Flag. This bit denotes whether the node has probably received the packet. For those who has probably received the packet the sender sets its corresponding bit to "0" from "1". We use the word "probably" here because the sender does not really know if the receiver actually received the packet (e.g., the packet could be lost during the transmission).

This header format can apply to both IPv4 and IPv6. Note that with IPv6 we could define *DESTINATION* as 112 bits instead of 128. Because in an Ad hoc networks all nodes can use link local addresses, which are defined as FE80:..., we can omit FE80 to make the packet shorter. The extra bits can be used for other sakes.

Also, we define *TIME OUT* to be the alternative determination of the timing to return the session advertisement. To minimize network traffic the nodes not joining the session do not send back the advertisements even if *TTL* is exhausted. However, if the last node does not send back the advertisement, all of the joining nodes along the path will be overlooked. Also, any advertisement packets send by a node could be lost during the transmission, and therefore causes overlooking problem. We define *TIME OUT* to overcome these problems. After sending an advertisement, each node starts its timer and waits for the returning advertisement packet. If no packet returns before the timer exceeds, considering no more nodes joining the session or the packet has been lost during the transmission, the node sends the advertisement back to the source node. We define *TIME OUT* as:

$2d(TTL + 1)$ where d could be changed (e.g., 10msec)

4 An Example Scenario

Let us present a conceivable scenario in this section. Suppose a severe earthquake struck an area, and network infrastructure has been severely damaged. Yet people in the area still could connect to Ad hoc network with their laptops. The laptops can use auto configured addresses in IPv4, or link local addresses in IPv6. We do not assume any multihop protocol working on this Ad hoc network. Therefore any node could only communicate with their direct neighbors, but not any indirect neighbors.

To communicate with other nodes including indirect neighbors, users start our TXR application on their laptops. To provide information to other users, the application performs FGF algorithm periodically to get member information, and then uses TXR to send information to others. To get information from an initiator, the application simply waits for session advertisements and joins the session. We show these processes in more details in the following paragraphs.

In this example, suppose that user on node A wants to create a new group session named "Food-Info", and provides information concerned with foods to other nodes (Fig. 4). Here we assume that node A though G can only communicate with the direct neighbors connected by lines. And the colored nodes are the nodes which are interested in joining the session. To simplify the issue, we also assume *TTL* to be 3.

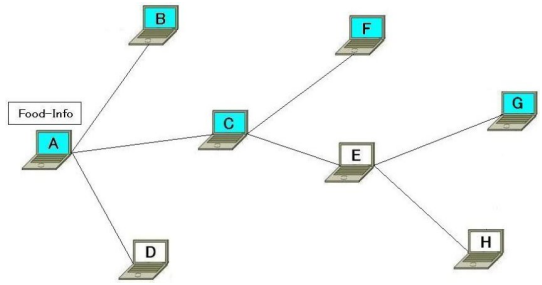

Figure 4: An example

4.1 FGF

To be an initiator sending information to other members, the user on node A starts our TXR application. And node A must know who are going to join the session. To do so, once the application starts, it performs the FGF algorithm as following. Note that in Fig. 5, we omitted some details in header such as *TTL*, which is set to be 3 in this example, and *Routing Information*(Routing tree table), which has no value yet at this point. In Fig. 5, the blocks in packet headers denote *SESSION NAME*, *DESTINATIONS* along with their *Member Flags* and *Receiver Flags*.

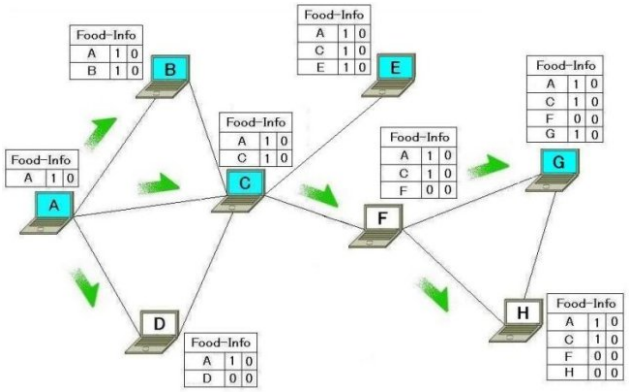

Figure 5: Flooding based Group Forming

(0). Suppose that all nodes in this network whether want to join sessions or are willing to forward the packets. They start our application waiting for session advertisements.

(1). To create a session(e.g., "Food-Info") and provide information to others, node A floods a session advertisement. Trying to find the nodes interested in joining "Food-Info" session, at first node A sends three identical packets with "Food-Info" as *SESSION NAME* to B, C and D respectively. This packet does not carry any payload. The *Member Flag* for A is set to 1, because obviously node A is one of the members of the session.

(2-1). Node B receives the packet, and decides to join the session. So B adds its address and sets its *Member Flag* to "1". Since the packet comes from A, node B does not forward the same packet to A. Node B forwards the packet to the other neighbor, C.

(2-2). Node C receives the packet from A, and takes the same action as B. Node C tries to send the packet to neighbors except A. If at this point no packet comes from node B or node C, C forwards the packet to B, C and F. Otherwise, if C receives a packet with same *Sequence Number* from B after receiving it from A, C simply discards the duplicated packet and sends nothing back to B. As B waits for the advertisement back from C, the *TIME OUT* in B expires eventually. So B sends the advertisement back to the initiator A.

(2-3). Node D receives the packet, but decides not to join the session. So D adds its address and sets its *Member Flag* to 0, and forwards the packet to node C. Different from B, D does not start its Timer to wait for return packet, because not being a member D won't send the advertisement back to A in any case.

(3). Node C forwards the packet to node E and F, and starts its Timer waiting for return packet. Joining the session, node E modifies the packet and sends it back to node A, because node E is a LEAF. Deciding not to join the session, F adds its address in the packet and sets *Member Flag* to 0. After modifying the packet, F forwards two identical packets to G and H.

(4-1). To join the session, node G adds its address in the packet and sets *Member Flag* to 1. Because TTL is 3 in this example, node G has to send back the packet to initial node A along the path, G, F, C and A.

(4-2). Deciding not to join the session, node H simply discards the advertisement and does nothing because TTL has expired.

(5). Finally, the packet got back to node A, and A knows that B, C, E and G want to join the session. And by examining the DESTINATIONS, node A knows the whole tree structure for this session which can be used in the stage of source routing.

(6). To accommodate the mobility of Ad hoc networks, node A floods the *session advertisement* at regular intervals, and gets updated tree structure for the current network topology.

4.2 TXR

After performing FGF, node A knows the members who are joining the "Food-Info" session. Also, by receiving returned advertisement packets node A knows the whole tree structure which looks like Fig. 6.

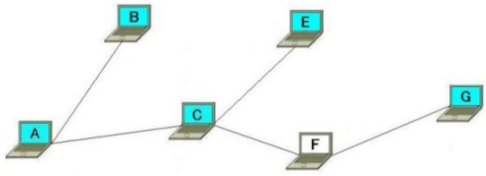

Figure 6: Tree Structure

Knowing the tree structure, node A creates the *Tree Table* in our style, which looks like Fig. 7.

Now, all node A has to do is sending TXR packets with the *Tree Table* to its direct neighbors. And the nodes along the path will transfer the packets to its destinations

1	A	2	3	LAST
2	B	ϕ		
3	C	4	5	LAST
4	E	ϕ		
5	F	6	LAST	
6	G	ϕ		

Figure 7: Tree Table in the TXR header

properly by using the *Tree Table*. As we mentioned before, with our *Tree Table* the calculations in every nodes will be trivial. We show how the packet passes through the network in the following (Fig. 8).

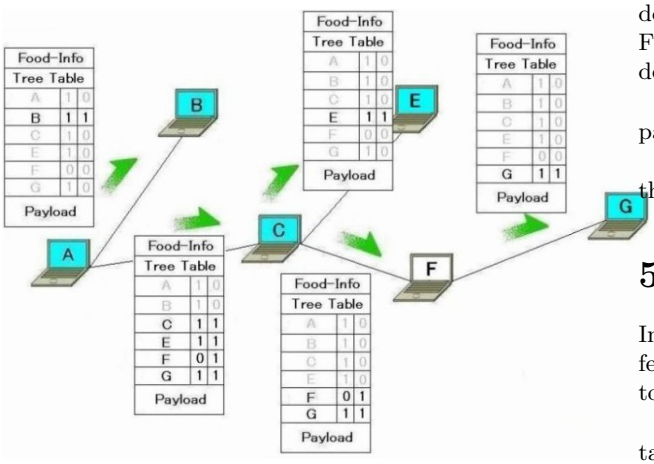

Figure 8: Delivery of TXR packets

(1). Trying to send packets to B, C, E and G, node A creates two packets with same payloads and slightly different headers, which look like Fig. 9 respectively. Except the *Receiver Flag*, the packets are almost same.

(2-1). Looking at the *Tree Table*, node B knows that B itself is the last destination, so B simply receives the packet without forwarding to others.

(2-2). Receiving the packet from A, node C examines the *Tree Table* and knows it has two subsets which are {E} and {F,G}. So it creates two packets, one for node E, one for node F, with different *Sending Flags*.

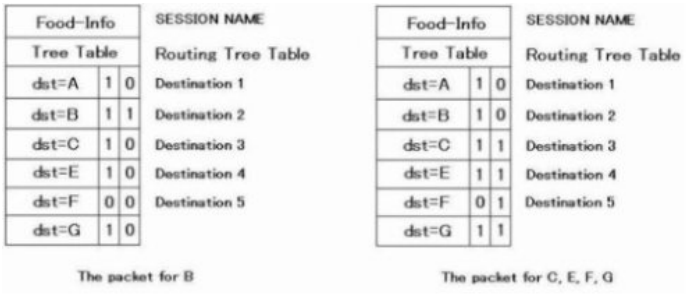

Figure 9: TXR Header

(3-1). As node B, node E simply receives the packet from node C.

(3-2.). Not being the member in the session, node F doesn't transfer the packet to its application. Still node F is responsible for transferring the packet to the final destination G.

(4). As node B and E, node G simply receives the packet from node F.

(*). Each receiver could be a source node as well with the *Tree Table* in the headers.

5 Services Using TXR

In this section we introduce some services in which the features of our proposing application displays its ability to the full.

Users with cell phones implemented our application can talk to multiple persons joining the same group in the area simultaneously, probably without any charge. One of the members initiates the session by sending session advertisement. Other members receive the advertisement and join the group. In this case, there must be some kinds of member authentication mechanisms to guarantee the joining members are welcome. The easiest way to do this is probably the authentication by other nodes which have already joined the group. That is, without permission by the initiator or other members any node can not join the group.

Suppose some users with their own cell phones or laptops want to arrange to meet at somewhere in a town. In a crowded place it might be hard to find each other. Of course, they could call each other. However with our application users can talk to multiple persons simultane-

ously, like transceivers.

Our proposal perfectly contributes to this case because there could be large number of groups composed of small number of members in a crowded place.

Also, any store can be initiators providing information or advertisements to anyone in the area. For example, a Chinese restaurant called "Shanghai" can send a session advertisement named "restaurant-chinese-shanghai", and anyone interested in receiving the information can join the group and receive information from the restaurant such as the location of the restaurant, today's special menu, etc. In addition, the customers can share their opinions and advice for the restaurant. In places where there locate many stores, our application can fairly contribute to the service. To support large number of customers the stores could establish multiple session with same information.

6 Summary and Future works

In this paper, we proposed a new Xcast-like routing scheme named TXR along with a completely new group membership administration mechanism, FGF. With these schemes users can easily, efficiently get/provide information from/to other users efficiently even in a temporary network or an emergency situations.

We introduced a broad design for TXR which contains FGF mechanism as well, so that both membership administration function and source routing function are packed into a single format. The nodes except the source node do not have routing information themselves. Instead, the source node sends the packets with specially designed tree tables which help other nodes route the packets. Additionally, with our special tree tables the calculation required for each node will be trivial.

We presented TXR along with FGF for application layer in this paper. However, the concept can be applied to lower layers. We will define TXR routing protocol in the near future.

In this paper, we proposed that the initiator should flood the *session advertisement* at regular intervals. However, we expect to find a way to avoid unnecessary flooding. Therefore, the initiator floods the *session advertisement* only once. We could use the ordinary TXR packets to accommodate the changes of network topology and membership without flooding.

Receivers who have received the *Tree Table* can also use it to send packets. However the tree structure represented by it may not be the best structure for them. We expect to find some ways to solve these problems.

References

[1] *Explicit Multicast (Xcast) Basic Specification, Internet Draft*, draft-ooms-xcast-basic-spec-09.txt, (December 2005), work in progress.

[2] S. Deering, *Host Extensions for IP Multicasting*, RFC 1112, (August 1989).

[3] T. Kunz and E. Cheng, *Multicasting in Ad-Hoc Networks: Comparing MAODV and ODMRP*

[4] C. Wu, Y. Tay and C. Toh, *Ad hoc multicast routing protocol utilizing increasing id-numbers (amris) functional specification*, Intenet Draft, draft-ietf-manet-amris-spec-00.txt, (November 1998).

[5] S. Lee, W. Su and M. Gerla, *On-demand multicast routing protocol(ODMRP)*, Internet Draft, draft-ietf-manet-odmrp-02.txt, (June 1999), work in progress.

[6] E. Bommaiah, A. McAuley and R. Talpade, *Amroute: ad hoc multicast routing protocol, Internet Draft*, draft-talpade-manetamroute-00.txt, (February 1999), work in progress.

[7] L.Ji and M.Corson, *Explicit Multicasting for Mobile Ad Hoc Networks*, (2003).

[8] J.J. Garcia-Luna-Aceves and E. Madruga, *A multicast routing protocol for ad-hoc networks*, in: Proceedings of IEEE INFOCOM'99, (March 1999), pp. 784-792.

[9] J. Jercheva, Y. Hu, D.Maltz and D. Johnson, *A simple protocol for multicast and broadcast in mobile ad hoc networks*, Internet Draft, draftietf-manet-simple-mbcast-01.txt, (July 2002), work in progress.

[10] L. Ji and M. Corson, *Light-weight adaptive multicast*, in: Proceedings of IEEE GLOBECOM'98 (November 1998). work in progress.

[11] T. Ozaki, J. Kim and T. Suda, *Bandwidth-efficient multicast routing protocol for ad hoc networks*, in: Proceedings of IEEE ICCCN'99, (October1999).

[12] E. Royer and C. Perkins, *Multicast using ad hoc on-demand distance vector routing*, in: Proceedings of ACM/IEEE MOBICOM'99, (1999).