EFFICIENT MULTICASTING FOR MULTI-CHANNEL MULTI-INTERFACE WIRELESS MESH NETWORKS

Sung-Hwa Lim*, Cheolgi Kim*, Young-Bae Ko†, and Nitin H. Vaidya*

* Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
Email: {sunghwa, cheolgi, nhv} @illinois.edu
† Dept. of Information & Computer Engineering, Ajou University, Suwon, Korea
Email: youngko@ajou.ac.kr

ABSTRACT

Multicasting can be an useful service in wireless mesh networks (WMNs), which have gained significant acceptance in recent years due to their potentials of providing a low-cost wireless backhaul service to mobile clients. Many applications in WMNs require efficient and reliable multicast communication, i.e., with high delivery ratio but with less overhead, among a group of recipients. However, in spite of its significance, there has been little work on providing such a multicast service in multi-channel mesh networks. Traditional multicasting protocols for wireless multi-hop networks mostly assume that all nodes (equipped with a single interface) collaborate on the same channel. The single-channel assumption is not always true for WMNs that often provide the nodes with multiple interfaces for the purpose of substantial performance enhancement. In multi-channel/interface mesh environments, the same multicast data needs to be sent multiple times by a sender node if its neighboring nodes operate on different channels. In this paper, we try to tackle this challenging issue of how to design a multicast protocol more suitable for multi-interface and multi-channel WMNs. Our multicasting protocol builds multicast paths while inviting multicast members, and allocates the same channel to each of neighboring members in a bottom up manner. This mechanism can reduce message overheads and delivery delays while guaranteeing successful message deliveries. For the performance evaluation, we have implemented the proposed scheme on a multi-channel/interface mesh network test-bed with two IEEE 802.11a cards per node.

I. INTRODUCTION

Multi-Channel Multi-Interface (MCMI) wireless mesh networks have emerged as a new paradigm in multi-hop wireless networks. They enhance network-wide throughput by parallelizing packet forwarding on multiple channels. In MCMI WMNs, neighboring nodes try not to share channel to allow simultaneous packet transmissions; when two neighboring nodes occupy the same channel, only one of them can communicates at a time.

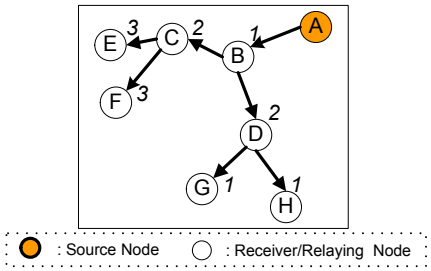

Fig. 1. An example of hidden terminal problem on a multicast tree constructed by MCM algorithm

Wireless communication is helpful for performing multicasting due to the broadcast nature of the wireless channel. However, note that this potential benefit is lost when the neighbors of a node are tuned to different channels. Especially, if the dominant communications in a network is multicast, as in military command networks, then use of multiple channels may potentially be even harmful to performance.

To cope with this problem, recent research has yielded top-down and centralized multicasting schemes initiated by the group owner (source node) [1,5,6]. Here, the source node of a multicast group builds a minimal path tree from the source (root) to all group members (leaves). Then, it assigns the same channel on all child nodes of a one-hop parent node. We instead take a bottom-up and distributed approach to construct multicasting tree and to assign channels. Bottom-up approach scales better in a network environment where various communications co-exist, and can require less control overheads. Also, previous research has mostly focused on tree building from the source node to multicast client nodes, and has not always provided the other procedures required for practical multicasting (e.g., session creation/close and advertisement, member management, etc.). Moreover, the past protocols are typically not implemented on real networks, and only evaluated on simulation tools.

Our multicasting protocol builds multicast paths while inviting multicast members, and allocates same channel to each of neighboring members of a parent in a bottom up manner. While building paths, intermediate relaying

The work described in this paper was funded in part by The Boeing Research & Technology, and coordinated with Dr. Jae H. Kim (Boeing PM).

978-1-4244-5239-2/09/$26.00 ©2009 IEEE

nodes are chosen based not only by path length but also by link quality. We have implemented our protocol on a real MCMI test-bed named Net-X [3]. We have shown that our scheme can reduce message overheads and delivery delays while improving successful message delivery.

II. RELATED WORKS

Although multicasting can be a useful service, there is not much work on multicasting for MCMI WMNs. In [1], the most closely related work to ours, Zeng et al. proposed a multi-channel multicast (MCM) scheme that consists of a sub-optimal multicast tree construction algorithm and a channel allocation algorithm. They find a minimal set of relaying nodes after BFS (breadth first search), and build the tree which spans every member node of the multicast group from the source node. After that, they assign an identical channel to sibling nodes that have the same parent on the tree. This strategy can reduce the number of links from a source to all group members with small co-channel interference. Cheng et al. proposed a greedy algorithm to reduce *total channel conflicts* (the number of interfering links) [6]. Their algorithm iteratively selects a route from the source to a receiver node, and assigns channels on all nodes in the route, preferring routes that have low total channel conflicts at each iteration.

Nguyen et al. proposed a new channel assignment scheme to reduce channel conflicts by relieving *hidden channel problem,* from which MCM algorithm [1] may suffer [5]. The hidden channel problem, similar to the "hidden terminal problem" in multi-hop wireless networks, occurs when two nodes two-hops away from each other try to tune on the same channel. Fig.1 shows an example of this hidden channel problem when a multicast tree is constructed. We assume that the numbers written close by the node figures are the assigned channels of each node. Multicast data are transmitted from node A (source) to all nodes using each of their one-hop child's channels (e.g., node B sends out packets only on channel 2). As both nodes B and G are tuned on channel 1 for receiving, the nodes A and D will transmit packets on channel 1. Then, the signals sent from both nodes A and D will conflict with each other at node B if they transmit at the same time. Therefore, node B may not receive correct multicast packets from node A. To reduce these channel conflicts, Nguyen et al. employ not only orthogonal channels but also partially overlapped channels for the channel assignment. To reduce the adjacent channel interference caused by employing partially overlapped channels, they also proposed a channel separation algorithm by considering interference between all channels.

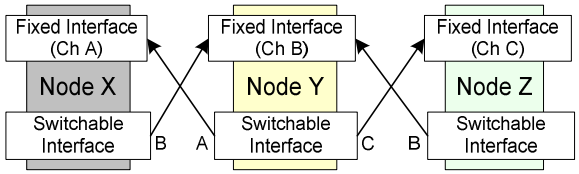

Fig. 2 A simple illustration of our network model (3 nodes, each of which has 2 wireless interfaces)

As pointed out earlier, these previous schemes for MCMI WMNs employ top-down and centralized methods. Thus, building multicast trees and assigning distinct channels on every node require the source (or a coordinator) node to know all the information about network topology. Moreover, some channel assignment forced by source node may conflict with the current channel assignment strategy in the target WMN. For example, the Net-X test-bed employs balanced channel allocation strategy, which tries to tune distinct channels on every node within two-hop range to decrease co-channel interference for unicast transmission [2]. Therefore, we suggest bottom-up distributed channel and parent selection scheme that also considers current link quality.

III. NETWORK MODEL

Our MCMI WMN model, similar to the one used in [2,3,7], consists of several wireless mesh nodes, each of which is equipped with two wireless interfaces. One of these interfaces is used primarily for receiving data, and is referred to as *Fixed Interface* and the channel tuned on the fixed interface is called *Fixed Channel*. A node can receive data only on its fixed channel. The other interface is named *Switchable Interface*. If a node wants to transmit data to another node, it tunes its switchable interface to the fixed channel of the destination node before transmitting the data. Of course, if two neighboring nodes share the same fixed channel, then the transmitter sends data on its fixed interface. Fig.2 shows an example of data communication in our network model assuming that there are three nodes X, Y, Z and the fixed channel of each node is A, B, and C, respectively. However, if Node Z is using channel B as the fixed channel instead of channel C, then Node B will use its fixed interface for data transmission to Node Z.

Nodes exchange their fixed channel information by periodically broadcasting "Hello Messages" on all possible channels. By exchanging hello messages, every node can also check current link quality from each of its neighbors. Each node keeps a tab of the successfully received hello messages in last 64*H seconds where H is the hello interval while maintaining 64-bitmaps for each of its neighboring node. A node can estimate the link quality to another

node by using both backward and forward link delivery probability. Backward link delivery probability for a neighboring node can be estimated by this mechanism. By enclosing each of these estimated values of all neighbors while sending hello message, every node comes to know the forward link delivery probability for each of its neighboring nodes. Each node also includes its neighbor list, fixed channel of each neighbor, and link quality between them in the hello message. Therefore, all nodes are able to have current information about their one-hop and two-hop neighbors. To reduce co-channel interference, our network system employs a balanced channel algorithm [2] in which the fixed channel of a node may be changed over time if there are many neighboring nodes in two-hop range that are using the same fixed channel.

Like other MCMI WMNs, the same broadcast data need to be sent multiple times by a sender on all channels. For multicasting, the source node disseminates multicast data only on the channels that are stored in the multicast table. Channels and interface information are filled in the multicast table with respect to the multicast groups.

G=(V,E) denotes the network graph where *V* represents a node and *E* represents a directed link between two nodes. If $u, v \in V$, and link $(u,v) \in E$ exists, then node *u* can transmit data to node *v* directly without going through other nodes. We define *Q(u,v)* as a delivery probability from *u* to *v* when $(u,v) \in E$ exists, which ranges from 0 to 1. The term "delivery probability" and "link quality" will be used interchangeably in this paper. We can estimate current link quality by exploiting received hello messages without extra network overhead. We define Th_{MCn} as a minimal delivery probability for a reliable direct link in multicast session *n*. If $Q(u,v) \geq Th_{MCn}$, and $Q(v,u) \geq Th_{MCn}$ then we say that nodes *u* and *v* are one-hop neighbors of each other in multicast session *n*. Th_{MCn} value may be proportional to the required minimal success ratio of data delivery for the multicasting session *n*. In this paper, we also use the term "multicast session" and "multicast group" interchangeably.

IV. MULTICASTING FOR MULTI-CHANNEL MULTI-INTERFACE WMN

A source node initiates a multicast group and disseminates an *advertisement message* to every node in the network. This message is supposed to be relayed only by some designated intermediate nodes, which are chosen among the current sender's one-hop neighbors. A multicast tree is constructed while the advertisement message is being disseminated hop by hop. Thus, on receiving the advertisement message and attempting to join the group, a node replies a *join message* back to the node (parent node) that has sent the advertisement message to itself. Now, when receiving the join message, the recipient node sets the sender node as one of its child nodes and sends a *reply message* to that sender node. Some channel adjustment information is enclosed in the reply message to let newly joined child node have the same fixed channel as other child nodes have. When the source node sends multicast data, it is relayed to all group members only through designated relaying nodes. Table 1 describes major control and data messages for our multicasting scheme. Every multicast message includes multicast session ID, source address, and sender's address by default.

Table 1. Description of key messages for multicasting

Message	Featured Contents	Description
JOIN_ADV	List of next relaying nodes	Broadcast message originated by source node to advertise the session.
JOIN_REQ	The address of the node that originated this message	Unicast message sent by a node who wants to join the session.
JOIN_RPL	the channel adjusting information for switching channel	Unicast message sent by a source or a relaying node in response to JOIN_REQ
MCAST _DATA	Data	Multicast data originated by the source node

For the relaying of MCAST_DATA to receiver nodes, intermediate nodes can participate in the multicast session in one of following states.

- *Coordinator Candidate*: A node enters this state if its IP address matches one of relaying node addresses enclosed in the received JOIN_ADV.
- *Multichannel Coordinator*: An intermediate node that relays MCAST_DATA enters this state. Only a *Coordinator Candidate* node can enter this state after receiving JOIN_REQ.

4.1 Group Management and Channel Adjustment

4.1.1 Join Advertisement and Relaying Procedure

After a source node initiates a multicast group, it disseminates JOIN_ADV on all possible channels to advertise the multicast session. The source node finds out the minimal set of relaying nodes among its one-hop neighbors that can span all of its two-hop neighbors, by using a minimal relaying node search algorithm. Then, the set (list) is enclosed in the JOIN_ADV message. Only the nodes in this list process the message; others discard the message.

Algorithm 1. Link-quality based minimal relaying node search algorithm

INPUT: p_u: the parent node of node u
$N1_u$: a set of u's all 1-hop neighbors
$N2_u$: a set of u's all 2-hop neighbors
OUTPUT: MPR_u : the set of relay nodes from node u

$N1_u = N1_u - \{p_u\}$; // exclude p_u
$N1_u = N1_u - N1_{p_u}$; // exclude p_u's one-hop neighbor
$N2_u = \bigcup_{i=1}^{|N1_u|} N1_{k_i} - \{u\} - \{p_u\}$ where $\forall k_i \in N1_u$;
WHILE ($N1_u$ is not empty and $N2_u$ is not empty) **DO**
$S_k = \left\{ k_i \middle| \underset{k_i \in N2_u \text{ for } 1 \le i \le |N2_u|}{\operatorname{argmin}} \left| P_{k_i} = N1_{k_i} \cap N1_u \right| \right\}$;
$S_{P_k} = \bigcup_{i=1}^{|S_k|} P_{k_i}$ where $\forall k_i \in S_k$;
$S_l = \left\{ l_i \middle| \underset{l_i \in S_{P_k} \text{ for } 1 \le i \le |S_{P_k}|}{\operatorname{argmax}} \left| N1_{l_i} \cap N2_u \right| \right\}$;
$C = \{ i \in S_l \mid Q(i,u) \ge Th_{MCn} \text{ and } Q(u,i) \ge Th_{MCn} \}$;
$l_{max} = \underset{l_i \in C \text{ for } 1 \le i \le |C|}{\operatorname{argmax}} Q(u,l_i)$;
$MPR_u = MPR_u \cup \{l_{max}\}$;
$N2_u = N2_u - N1_{l_i}$;
$N1_u = N1_u - \{l_{max}\}$;
DONE

A link-quality based minimal relaying set searching algorithm is described above in Alg.1. Because finding out the minimal relaying set is a variation of the set-cover problem, which is NP-complete, we have employed the approximate algorithm proposed in [1,8], and modified it by adding a link quality checking procedure (4th and 5th line in *while* loop). Main idea is as follows: At first, the algorithm finds two-hop neighbors who have the smallest number of potential parent nodes in one-hop neighborhood of current node (u in the algorithm). Then, every parent (i.e., a one-hop neighbor of current node u) of these nodes is a relay candidate. Among the candidates, the algorithm finds a set of nodes who have the smallest number of potential child nodes among u's 2 hop neighbors. Finally, among these nodes, the algorithm finds a relaying node (one-hop neighbor of u) who has the best link-quality from the current node u. Then, the selected relay node and its child nodes are removed from the one-hop list and the two-hop list of u. These steps are repeated until either one-hop neighbor list or two-hop neighbor list become empty.

Fig.3 shows an example of the procedure for JOIN_ADV originated by a source node (node 0). Node 0 disseminates JOIN_ADV with a relaying node list including node 2. On receiving the message, node 2 becomes a coordinator candidate and forwards the message with new relaying node list (node 6 and node 8).

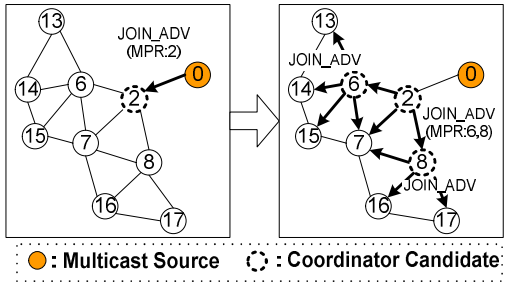

Fig. 3. An example of JOIN_ADV messaging

On receiving the message, nodes 6 and node 8 become coordinator candidates and forward it, and so on. In our example, node 7 will store node 2, node 6 and node 8 in its parent candidate list for the multicast session.

4.1.2 Member Joining and Channel Adjustment

After receiving JOIN_ADV, if the node wants to join the group, it selects its parent among parent candidate list. A node who currently shows best link quality from this node becomes the parent. We denote a node by u, and the parent candidate set of node u by L_u, the current link quality from x to y by $Q(x,y)$. Then, we can find out the parent node p_u as Eq.1. Upon the parent has been chosen, the node sends JOIN_REQ to the parent node by unicasting.

$$C = \{ i \in L_u \mid Q(i,u) \ge Th_{MCn} \text{ and } Q(u,i) \ge Th_{MCn} \}$$
$$p_u = \underset{l_i \in C \text{ for } 1 \le i \le |C|}{\operatorname{argmax}} Q(u,l_i) \quad (1)$$

On receiving JOIN_REQ, if the node is in *coordinator candidate* state, then it forwards this message to its parent that is also selected from its parent candidate list using Eq.1. Then, the node changes its state to *multichannel coordinator*, and stores the requesting node's address in its child list.

To respond to JOIN_REQ, the node sends JOIN_RPL to the requesting node. For efficient multicasting, the channel adjustment is performed by *a coordinator candidate* or *multichannel coordinator* node. The channel adjusting information (new fixed channel of the child node) is determined by *the channel adjustment algorithm* as shown in Alg.2, and enclosed in JOIN_REQ. If the node who has sent JOIN_REQ is the first coming child of current node (parent), then the adjustment information is set to as the child's current fixed channel unless the fixed channel is same to its parent's. If the fixed channels are same, the parent node selects other one from its channel list to avoid channel conflict. If the node who has sent JOIN_REQ is not the first child, then the channel information in JOIN_RPL will be set as a dominant fixed channel that most of nodes in the child list of the parent node are using.

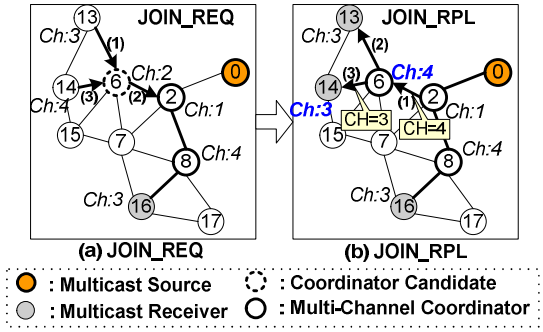

Fig. 4. Member joining and channel adjustment

Algorithm 2. Decision of channel adjustment information

INPUT: *curr*: current parent node, *req*: join requesting node
CH_{curr}: Fixed channel of node *curr*
CH_{req}: Fixed channel of node *req*
C_List_{curr}: child list of node *curr*
CH_List_{curr}: channel list of node *curr*
OUTPUT: CH_{ADJ} : Fixed channel to be stored in the channel adjustment information

IF (C_List_{curr} is empty)
  IF (CH_{req}==CH_{curr})
    CH_{ADJ}= one of channels in CH_List_{curr} except CH_{req} ;
  ELSE CH_{ADJ}=CH_{req} ;
ELSE CH_{ADJ}= the dominant fixed channel used in C_List_{curr} ;
Insert node *req* into C_List_{curr} ;

On receiving JOIN_RPL, a node changes its fixed channel according to the channel adjustment information in the message, and locks its fixed channel not to change during the multicast session. Fig.4 shows an example of member joining and channel adjustment procedure when node 13 and node 14 try to join the multicast group. "*Ch:x*" written by a node means its current fixed channel, the number by a link is the logical sequence of message events. In Fig.4(a), node 13 sends JOIN_REQ to node 6. Because node 6 is still in *coordinator candidate* state and the child list is empty, it forwards the message to node 2 who is its parent. In response, node 14 sends JOIN_REQ to node 6. In Fig.4(b), after that, node 2 sends JOIN_RPL with channel adjusting information (CH=4) to node 6 because it already has node 8 as a child whose fixed channel is 4. Then, node 6 switches its fixed channel to channel 4, and sends JOIN_RPL to node 14 with channel adjusting information (CH=3) to make node 14 switch its fixed channel to channel 3 that is being used by node 13.

4.1.3 Member Disjoin

A receiver node sends DISJOIN_REQ message to its current parent. After sending DISJOIN_REQ, the node unlocks its channel lock and removes all relevant data structures. On receiving DISJOIN_REQ from its child, the *multichannel coordinator* node removes the node in its child list.

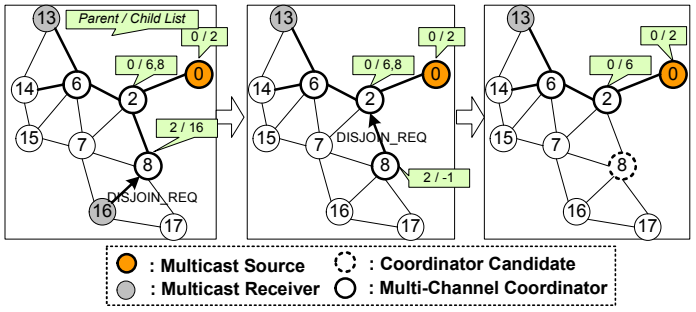

Fig.5. Member disjoining procedure

If a *multichannel coordinator* node's child list becomes empty, then it sends DISJOIN_REQ to its parent node, resigns from *mulitichannel coordinator* state, and releases all relevant data structures. Fig.5 shows an example of member disjoining procedure when node 16 is about to disjoin the group. Node 16 sends DISJOIN_REQ to node 8 who is its parent and a multichannel coordinator. Because the child list of node 8 became empty, node 8 sends DISJOIN_REQ to node 2 who is its parent. Finally, node 2 removes node 8 from its child list.

4.2 Multicast Data Forwarding

The source node sends out MCAST_DATA only on the channel that its one-hop child nodes are listening on. On receiving the data, only multichannel coordinators forward them in the same manner. With these sophisticated procedures, message overheads can be minimized.

4.3. Reliable Multicasting

We also address the side effects of channel assignment in multicast group initialization. In practical MCMI WMNs, a so-called *link fluctuation problem* caused by the fluctuating link quality may seriously degrade the reliability of multicasting. In Fig.6(a), the wireless link from node 0 to node 3 is considered as a one-hop route for a multicasting session while initializing the group. We assume that node 0 is a source and node 3 is a client for a multicast session. If node 3 receives a child-probing message from node 0 for constructing a multicast tree, then node 3 may set node 0 as its one-hop parent. Then, node 3 will receive multicast data from node 0 but not from node 1 or 2. However, if the link quality between node 0 and 3 becomes poor for some reason, then most of the multicast data sent by node 0 will not be delivered to node 3 as shown in Fig.6(b).

Dynamic channel allocation used in MCMI WMNs may be one of the major reasons of the link fluctuation. Although we usually assume that every channel has equal link quality, it may not be true in practice because physical characteristics of each channel can be different [11].

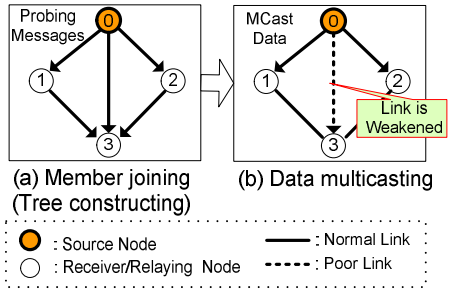

Fig. 6. Link fluctuation problem

That is to say, a wireless link between two neighboring nodes may be disconnected or weakened after changing their operating channels. To cope with these problems, we let every member node keep not one parent but several parent candidates for a multicast tree during tree construction. After that, the node will receive multicast data from the chosen parent node in the candidate list who has the best link quality to the node.

V. IMPLEMENTATION AND EVALUATION

5.1 Experimental Environments

Our MCMI WMN test-bed consists of 18 Soekris net4521 nodes deployed on the 4-th floor of CSL building in University of Illinois. Fig.7 shows a screen shot of our monitoring tool displaying the state of deployed nodes. Each node has a 133MHz CPU and two Atheros 802.11a interfaces as a fixed and switchable interface. We implemented our multicasting scheme on top of IEEE 802.11a protocol and Linux 2.4.6 kernel. Though 802.11a has 12 orthogonal channels in 5 GHz band, we use four channels (namely, channels 40, 52, 64, and 149) to avoid adjacent channel interferences. We set the wireless interfaces to operate at 6Mbps fixed mode. Slightly modified madwifi driver is used for the wireless device driver. The two interfaces are abstracted as one by using bonding driver. The software architecture is borrowed from Net-X system [3] and modified.

We evaluate the performance of our proposed algorithm (MMCA) by comparing to MMNCA and ACM, which are described below:

- **All Channel based Multicasting (ACM):** MCAST_DATA originated by a source node are forwarded only by the chosen intermediate nodes (relay nodes) in the multi-hop environment. These relay nodes are determined by minimal multiple relaying algorithms. Data transmission in the relaying or source node is done by broadcasting on every channel the node has.

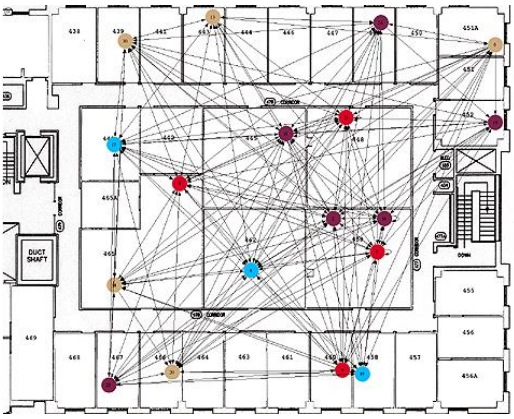
Fig. 7. Deployed nodes for the experiments

- **MCMI-based Multicasting with no Channel Adjustment (MMNCA):** Determining of relay nodes and relaying of MCAST_DATA are same as in ACM. Multicast data from the relaying or source node are sent out only on the channels being used by one-hop neighbors who are multicast receivers or relay nodes.
- **MCMI-based Multicasting with Channel Adjustment (MMCA):** Determining of relay nodes and relaying/transmitting of MCAST_DATA are same to those of MMNCA. The relaying or source node tries to unify fixed channels of its one-hop child nodes by sending JOIN_RPL with channel adjusting information to them.

Performance metrics for our evaluation are as follows:

- **Network traffic:** sum of multicast traffic every node sends during an experiment
- **Delivery ratio:** the average ratio of successful delivery of MCAST_DATA each multicasting receiver receives during an experiment
- **Average goodput:** the average amount of non-redundant MCAST_DATA each multicasting receiver receives during an experiment.

5.2 Results and Analysis

We assume that the number of multicast group and its source node is one. The size of a MCAST_DATA packet is 1024 bytes. Hello messages are disseminated by every node at every 4 seconds. The minimal link quality for a reliable direct link (Th_{MCn}) is set to 0.96. Experiments for each scenario are repeated by 30 times.

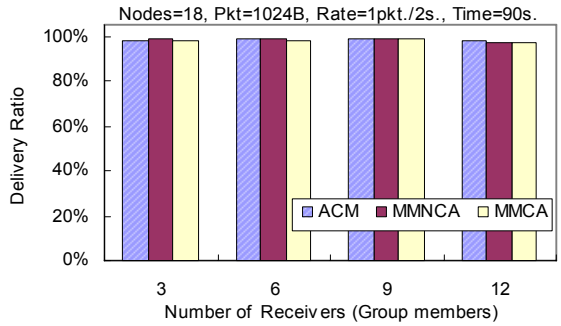

Fig. 8 Data delivery success ratio

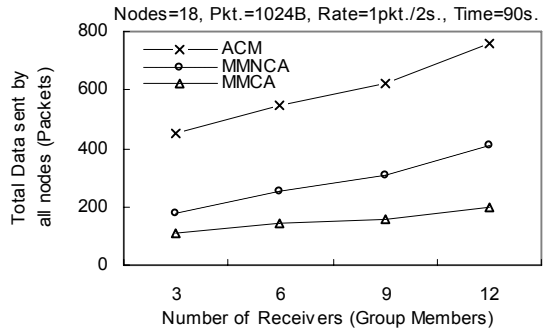

Fig. 9. Total network traffic transmitted by all nodes

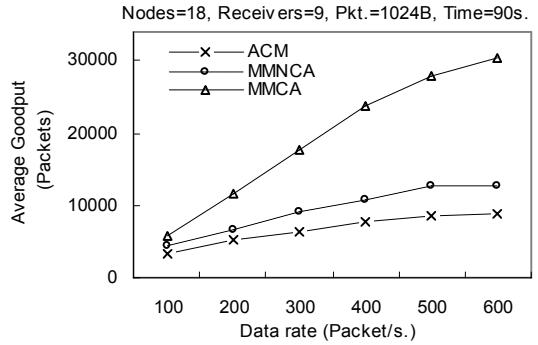

Fig. 10. Average goodput per receiver

Fig.8 shows the delivery success ratio for MCAST_DATA packets sent by the source node at a low rate (1 packet every 2 seconds) by varying number of multicast receivers (group members) during 90 seconds. As we can see, our scheme (MMCA) shows similar delivery success ratio compared with others. We also measured total network traffic sent by all nodes for the multicast. Total 46080 bytes of MCAST_DATA are sent out from the source node. As we can see in Fig.9, channel adjustment of MMCA can effectively reduces network traffic overheads compared to other schemes.

We have measured the goodput of each scheme during 90 seconds by varying the MCAST_DATA transmission rate from 100 packets/sec. to 600 packets/sec. when number of receivers is 9. As we can see in Fig.10, MMCA shows more than double the goodput than other schemes for load over 400 packet/sec. Delays for switching channels on the switchable interface to send out data on all possible channels may incur buffer overflow that decreases goodput in ACM. Moreover, broadcasting on all possible channels may incur co-channel interference to MCAST_DATA transmissions of nearby nodes. MMNCA may also be suffered from the same problem if relaying or source nodes have more than one child.

VI. CONCLUSION

In this paper, we try to tackle the challenging issue of designing a multicast protocol suitable for MCMI WMNs. Our multicasting protocol builds multicast paths while inviting multicast members, and allocates same channel to each of neighboring members with bottom-up and distributed approach. This mechanism reduces message overhead while improving successful message deliveries. We have implemented the proposed scheme on a real multi-channel mesh network test-bed (Net-X), and have shown that our proposed scheme performs well.

REFERENCES

[1] G. Zeng, B. Wang, et al., "Multicast Algorithms for Multi-Channel Wireless Mesh Networks," *IEEE ICNP*, Oct. 2007.

[2] P. Kyasanur and N. H. Vaidya, "Routing and link layer protocol for multi-channel multi-interface ad-hoc wireless networks," *ACM SIGMOBILE MC2R*, vol.10, pp.31-43, Jan. 2006.

[3] C. Chereddi, P. Kyasanur, and N. H. Vaidya, "Design and Implementation of a Multi-Channel Multi-Interface Network," *REALMAN*, May 2006.

[4] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel WMN," *IEEE INFOCOM*, 2005.

[5] H. Nguyen and U. Nguyen, "Channel assignment for multicast in multi-channel multi-radio wireless mesh networks," Wireless Communications and Mobile Computing, Published Online, Oct. 2008.

[6] H. Cheng and S.Yang, "Joint Multicast Routing and Channel Assignment in Multiradio Multichannel Wireless Mesh Networks Using Simulated Annealing," LNCS, vol.5361, 2008.

[7] C. Kim, Y.-B. Ko and N. H. Vaidya, "Link-State Routing Protocol for Multi-Channel Multi-Interface Wireless Networks," *MILCOM*, Nov. 2008.

[8] J. Macker, "Simplified Multicast Forwarding for MANET," Internet draft, draft-ietf-manet-smf-08.txt, 2008.

[9] J. Xie, R. Talpade, A. Mcauley, and M. Liu, "AMRoute : Ad Hoc Multicast Routing Protocol," Mobile Networks and Applications, vol.7, pp. 429-439, 2002.

[10] Y. Zhao, L. Xu, M. Shi, "On-Demand Multicast Routing Protocol with Multipoint Relay (ODMRP-MPR) in Mobile Ad-Hoc Network," *ICCT*, April 2003.

[11] V. Raman, "Dealing with Adjacent Channel Interference Effects in Multichannel, Multi-interface Wireless Networks," Master's Thesis, University of Illinois, 2008.