Channel Allocation Strategies for Interference-Free Multicast in Multi-Channel Multi-Radio Wireless Mesh Networks

Wen-Lin Yang[1] and Wan-Ting Hong[1]
[1] Dept. Computer Science and Information Engineering, National University of Tainan
Tainan, Taiwan
[e-mail: *wlyang@mail.nutn.edu.tw*, *roof770904@hotmail.com*]
*Corresponding author: Wen-Lin Yang

Received August 29, 2011; revised November 30, 2011; accepted January 30, 2012; published February 28, 2012

Abstract

Given a video stream delivering system deployed on a multicast tree, which is embedded in a multi-channel multi-radio wireless mesh network, our problem is concerned about how to allocate interference-free channels to tree links and maximize the number of serviced mesh clients at the same time. In this paper, we propose a channel allocation heuristic algorithm based on best-first search and backtracking techniques. The experimental results show that our BFB based CA algorithm outperforms previous methods such as DFS and BFS based CA methods. This superiority is due to the backtracking technique used in BFB approach. It allows previous channel-allocated links to have feasibility to select the other eligible channels when no conflict-free channel can be found for the current link during the CA process. In addition to that, we also propose a tree refinement method to enhance the quality of channel-allocated trees by adding uncovered destinations at the cost of deletion of some covered destinations. Our aim of this refinement is to increase the number of serviced mesh clients. According to our simulation results, it is proved to be an effective method for improving multicast trees produced by BFB, BFS and DFS CA algorithms.

Keywords: Channel Allocation, interference free, multicast, best first, backtracking, wireless mesh networks.

A preliminary version of this paper appeared in IEEE ICMT 2011, July 26-28, Hangzhou, China. This version includes a new refinement procedure for improving solution quality. This research was supported by a research grant from the National Science Council, Taiwan, ROC [NSC-98-2221-E-024-004-MY2].

DOI: 10.3837/tiis. 2012.02.011

1. Introduction

The wireless mesh network (WMN) has recently become a promising technology for the next-generation broadband wireless networks. A WMN consists of a number of mesh routers and mesh clients [1]. The mesh routers equipped with at least one network interface are stationary and form a multi-hop wireless backbone. Some special mesh routers act as gateway nodes which connect Internet with wired lines. The other mesh routers can only access Internet by multi-hop communications through the gatway. Except for acting as a router, a mesh router can also act as an access point (AP) by which neighboring mobile mesh clients can access backbond with one-hop communications through AP. Some commerical WMNs are deployed and operated with this architecture [2][3].

However, to make WMNs popular and indispensable, it is still required to prove that killer applications such as net TVs and interactive online games can be deployed and operated on WMNs in a commercial way. Therefore, we consider the following scenario: assuming that there is only one gateway node with video/audio servers in a WMN, in which a number of mesh clients attached with different mesh nodes may simultaneously send out a request to ask for receiving a streaming TV program. The program is assumed to be sent out from the video server only. A number of mesh clients in different mesh nodes may then subscribe to watch a live MLB baseball streaming at the same time.

Obviously, it is better to implement this video streaming system in a multicast tree structure to save bandwidth. It is well known that the performance of a WMN is greatly affected by interference between links with overlapping channels in vicinity areas. If multiple radio interfaces can be available in each mesh router, we may bind interfaces with partially overlapping channels to minimize the interference and increase network capacity [5][6]. That is, a multi-channel multi-radio (MCMR) WMN is required for constructing a multicast tree to implement our video streaming system described above. In most published references [7][8], most multicasting problems were studied based on how to minimize the maximum interference or the overall interference. Based on view points of mesh clients, we focus on how to maximize the number of mesh clients that can successfully receive their subscribed video stream in a multicast session. Since the more clients receive streaming service, the more profit we can make. This optimization goal is more reasonable and attractable for content providers and mesh networks owners.

In addition, since high transmission rate is required by video streams to provide high-resolution programs, concurrent transmissions on all the links in a multicast tree should be interference free. By putting these two goals together, a problem named maximum-revenue and delay-constrained multicast (*MRDCM*) is proposed in reference [4]. This problem is solved in two steps. First, a load-oriented multicast tree is determined. Second, a channel allocation (CA) algorithm is then used to allocate interference-free channels to tree links. Because the channel resource is limited, some links which can not have channel allocated are then deleted. The final channel-allocated multicast trees satisfy two goals described above. A number of heuristics proposed in reference [4] are developed based on this approach.

The CA algorithm is an important step for solving *MRDCM* problem and has been proved to be an NP-complete problem [9]. In reference [4], two primitive heuristics (load-oriented BFS and load-oriented DFS) are presented for it. These two approaches are developed based on the number of mesh clients in mesh routers and common searching paradigms such as depth-first search (DFS) and breadth-first search (BFS). In this paper, we re-investigate the CA problem

and propose a new and efficient method referred as the best-first and backtracking (BFB) algorithm. With the backtracking technique, BFB allows previous channel-allocated links to have feasibility to select the other eligible channels when no conflict-free can be found for the current link during the CA process. As a result, we may expect the BFB based CA algorithm consistently outperforms DFS and BFS based CA algorithms.

Since the final channel-allocated multicast tree is a partial multicast tree, it may not cover all the destinations. Hence, the tree is not optimal and can be improved even further. Therefore, we also propose a tree refinement procedure to improve the partial multicast tree by adding uncovered destinations at the expense of deletion of some covered destinations. Our aim of this refinement is to increase the number of serviced mesh clients.

Given a set of multicast trees constructed by the load-based multi-channel multicast (LMCM) method proposed in our previous work [4], we provide simulation experiments in section 6 to compare the performance of BFB with DFS and BFS based approaches. At the same time, we also give simulation results to show that the tree refinement technique is an effective way for improving the quality of solutions found by BFB, DFS and BFS algorithms.

The rest of the paper is organized as follows. Section 2 reviews past work related CA in MCMR WMNs. Section 3 gives a system model and definitions for the CA problem studied here. Section 4 describes BFB based CA algorithm. Section 5 gives a refinement algorithm for improving the channel-allocated multicast trees. Our experimental results are shown in section 6. Finally, we conclude our work in section 7.

2. Related Work

The CA problem of minimzing the interference caused by frequency overlapping channels is the core problem of multicast routing implemented on MCMR WMNs. Most CA algorithms published for multicast focus on the issue about minimizing interference while network connectivity does not degrade. In general, CA problem for multicast in MCMR WMNs can be classified into three categories: (1) routing first, CA second [10][11][12][13][14][15]; (2) CA first, routing second [16][17][18]; (3) joint design of CA and routing [19][20].

LCA and MCM algorithms proposed in references [10][11] belongs to the first category. The LCA contains two steps. First, the BFS traversal is used to build a multicast tree where each node is equipped with two interfaces and assigned a level value. Second, for nodes in level i, they are assigned with two channels: $i-1$ and i, where channel $i-1$ is used for receiving packets from parent node and channel is used to broadcast packets to its child nodes. The scheme is simple but the co-channel and adjacent channel interference from nearby links could be very serious. Therefore, the authors in [11] proposed a more intelligent algorithm named MCM. In MCM, a multicast tree with minimum relay nodes and minimum hop counts is constructed first. Then, a heuristic CA scheme is used to assign partially overlapping channels to the sending interface of a node by minimizing interference derived from neighboring relay nodes. However, this approach suffers from the hidden channel problem which can be eliminated by a CA algorithm named minimum interference multi-channel multi-radio multicast (M4) proposed in [14][15]. In M4, the authors assume that nodes that are three or more hops away from each other do not interfer. This assumption is not practical since it is very easy to have a situation that the interference range is larger than two hops. The wireless broadcast advantage (WBA) [23] is considered in both MCM and M4 to save the number of transmissions. The WBA is defined as all neighboring nodes within transmission range of a sending node can receive a data packet in one transmission from the sending node. For references [16][17] in the second category, the authors proposed the multi-channel

minimum number of transmissions (MCMNT) algorithm to construct multicast trees for given channel-allocated networks. The goal of MCMNT is to minimize the total number of transmissions required to deliver a data packet from source to all destinations. However, no discussion on interference is given and only orthogonal channels can be used in their random CA scheme. Also in the second category, Lim et al. [18] proposed a distributed and bottom-up approach to construct multicast trees with the objective that the number of channels which interferes with each other within two-hop distances is minimized. Nevertheless, this approach requires channel switching which may degrade the performance.

In contrast to previous approaches, the study in the third category is concerned about how to solve both multicast routing and channel allocation problems jointly. In our previous work [19], by taking both multicast routing and channel assignment into account simultaneously, a heuristic algorithm named load-oriented routing and channel assignment (LORCA) based on wireless broadcast feature is developed for constructing multicast trees. The simulation results show that LORCA is better than the other methods which are routing first and then followed by CA. However, only orthogonal channels are available in this work. In reference [20], the authors provide a mathematical model for the cross-optimization of multicast tree construction with minimal interference. The optimal solution is obtained by solving the binary integer programming (BIP) formulation of the model.

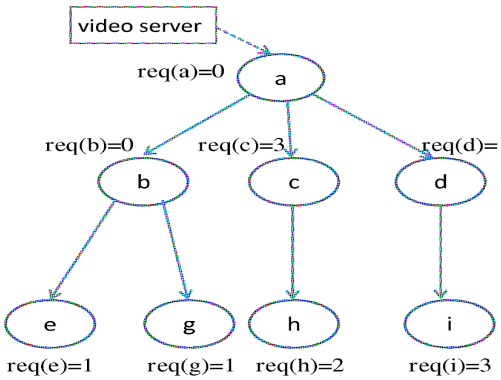

Fig. 1. A multicast tree

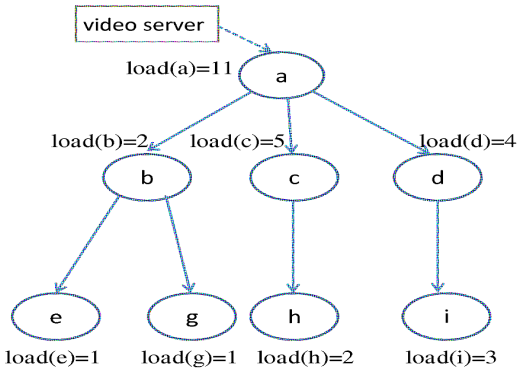

Fig. 2. A node-weighted multicast tree

3. Problem Definitions and Assumptions

Given a multicast tree $T=(V,E)$ embedded in a MCMR WMN, as shown in the **Fig. 1**, it is assumed that a video-stream delivering system is deployed in the gateway node a (the root) to provide continuous video/audio streams, and the other internal and leaf nodes are the mesh routers. For each node $u \in V$, it is equipped with multiple radios and a weight $req(u)$. Here, the value of $req(u)$ represents the number of mesh clients serviced by node u. We also define a destination set $M = \{ u \mid u \in V \text{ and } req(u) > 0 \}$. A destination is defined to be a node where at least one mesh client under it subscribes to receive a video/audio stream provided by the multimedia server. Hence, a mesh client is also called a subscriber in this paper.

Assumed that at least two network interfaces are equipped with each node, and the transmission range of each node is the same. The problem we study here is to find a way to bind each interface to a channel so that multiple streams can be concurrently transmitted among all the links without any interference. That is, our problem is required to allocate interference-free channels to tree links so that the nodes in M can be totally covered.

Let $\overline{T}=(\overline{V},\overline{E})$ be the resulting multicast tree after CA is performed and $\overline{T} \subseteq T$. Since the channel resource is very limited, it is possible that $M \not\subset \overline{V}$. Therefore, our goal is then set to find a set of interference-free channels for links so that the following equation is maximized:

$$Gain = \sum_{v \in M \&\& v \in \overline{V}} req(v) \tag{A}$$

$Gain$ is the total number of mesh clients who can be serviced by $\overline{T}$. The maximum $Gain$ is obtained when $M \subseteq \overline{V}$. To obtain an interference-free multicast tree, we may simply allocate orthogonal channels to links and then delete the unallocated links. However, the number of orthogonal channels is so few that the total number of clients covered by $\overline{T}$ is also very few. For example, there are only three orthogonal channels in IEEE 802.11b/g standard. Hence, in this study both partially overlapping and orthogonal channels are utilized for CA. Based on interference factor discussed in next paragraph, we will show how to assign non-orthogonal channels to links to achieve multiple interference-free transmissions.

Given a transmission range R and an interference range IR, an interference factor (IF) defined as IR/R by Zeng [10] to measure the interference strength between two channel-allocated nodes. Suppose channel $c1$ and $c2$ are allocated to two neighboring nodes, a term named channel separation (CS) is defined as $|c1-c2|$. In reference [10], a method is described to measure IF based on different CS. The relationship between IF and CS is greatly affected by the transmission rate at the physical layer. For transmission rate of 11Mbps in $IEEE$ 802.11b standard, the relationship between IF and CS is given in **Table 1**.

Table 1. Channel separation vs interference factor when transmission rate is 11Mbps

Channel separation (CS)	0	1	2	3	4	5
Interference factor (IF)	2.0	1.2	0.7	0.5	0.2	0.0

In the CA algorithm of MCM, IF is used to minimize the overall interference from the neighborhood of a node. In this study, since our goal is to create an interference-free multicast system, we use CS to guide CA process as the distance δ between two nodes is known. For

example, if $\delta = 1.1R$, the value of CS must be at least 2 to avoid interference between two nodes. In $IEEE$ 802.11b system, if channel $c_1 = 2$ is allocated to one node, then the feasible channels that can be allocated to another node would be $c_2 = 4, 5, ..., 11$. For the case of $\delta \geq 2R$, two nodes does not interfere with each other with the same number of channel. The above relationship is established based on the following assumption: all the nodes have the same R and IR.

With the distance between two links is fixed, we can assign both orthogonal and partially overlapping channels to them so that the channel separation is large enough to support interference-free concurrent transmissions. Since all the channels are available, the value of $Gain$ computed is expected to be much better than the value of $Gain$ computed by a method with orthogonal channels only.

4. Heuristics for Channel Allocation

In this section, we first introduce a data structure called interference matrix IM. It is used to record the relative location between any two links and the least channel separation. The implmentation details of IM will be discussed in next sub-section. With the help of IM, we can easily determine whether two links are interfering each other and then find conflict-free channels which can be allocated to them.

4.1 Interference Matrix Definition

Given a multicast tree, we are required to compute an auxiliary data structure called interference matrix IM based on Table I before we can allocate channels to the links. Definition of IM is given in **Fig. 3**. For each entry $IM(l_i, l_j) = (\alpha, \beta)$, α denotes the interference type for links l_i and l_j, and $\beta = |channel(l_i) - channel(l_j)|$ which denotes the least number of channel separation to avoid interference between two links. The value of α can be 1, 2, 3. The value of β can be 0 ~ 5.

The interference range (IR) is assumed to be $2R$ where R is the transmission distance between any two nodes. Here, we assume transmission power is the same for all the nodes. The distance between two links, l_i and l_j, is denoted as $\delta_{i,j}$ which is the minimum distance between any two terminal nodes of links l_i and l_j. When $\delta_{i,j} = 0$, two tree links must be adjacent or consecutive. The two situations are distinguished by α value that can be 1 or 2. If they are sibling links, they may share one common channel by taking advantage of wireless broadcast nature. Hence, $IM(l_i, l_j) = (1, 0)$ is shown in line (1) in **Fig. 3**. If they are consecutive links, only non-overlapping channels can be allocated to them. That is, the number of channel separation, β, must be at least 5. Hence, $IM(l_i, l_j) = (2, 5)$ is shown in line (2). As for line (3) ~ (8), α is 3 which represents the third type of relationship of two links. Based on the value of $\delta_{i,j}$, the value of β is then determined to achieve a goal which is that the concurrent transmissions of l_i and l_j are interference-free. In particular, when $\delta_{i,j} \geq 2R$, links l_i and l_j can simultaneously transmit data without any interference. Hence, the channel separation β is 0.

For two links l_i and l_j, $IM(l_i, l_j)$ is defined as follows:

(1) $\delta_{i,j} = 0$, l_i and l_j are sibling links, $IM(l_i, l_j) = (1, 0)$;

(2) $\delta_{i,j} = 0$, l_i and l_j are consecutive links, $IM(l_i, l_j) = (2, 5)$;

(3) $0 < \delta_{i,j} < 0.2R$, $IM(l_i, l_j) = (3, 5)$;

(4) $0.2R \le \delta_{i,j} < 0.5R$, $IM(l_i, l_j) = (3,4)$;

(5) $0.5R \le \delta_{i,j} < 0.7R$, $IM(l_i, l_j) = (3, 3)$;

(6) $0.7R \le \delta_{i,j} < 1.2R$, $IM(l_i, l_j) = (3, 2)$;

(7) $1.2R \le \delta_{i,j} < 2.0R$, $IM(l_i, l_j) = (3, 1)$;

(8) $2.0R \le \delta_{i,j}$, $IM(l_i, l_j) = (3, 0)$;

Fig. 3. Interference matrix IM definition.

4.2 Load-oriented BFS and Load-oriented DFS CA algorithms

Traditionally, there are two approaches for allocating channels to links of multicast trees embedded in MCMR WMNs. One is the breadth-first search (BFS), and another one is the depth-first search (DFS). Based on these two searching paradigms, two CA algorithms, load-oriented BFS and load-oriented DFS, are presented in our previous work [4]. The time complexity of both BFS and DFS is $O(|V|^2 \log(|V|))$ [4].

4.3 Best-First and Backtracking (BFB) based CA algorithms

In this section, a new CA algorithm named BFB is presented. The BFB algorithm can allocate an interference-free channel to a link. If no such channel can be found, a backtracking procedure is used to adjust channels continuously that have been allocated to the interfering links so that a feasible and conflict-free channel can be found for the current link. Hence, our BFB based CA algorithm consists of two parts: best-first search based procedure and backtracking procedure. In fact, the backtracking procedure can also be combined with DFS or BFS methods to improve performance.

4.3.1 Best-First Search Strategy

Although the searching strategy of load-oriented BFS CA algorithm is guided by the *load* value associated with each node, the sequence of links selected for channel allocation is generally in a level-by-level fashion. For load-oriented DFS CA algorithm, the sequence of next link selected is different. Nevertheless, as the load-oriented BFS method, the searching sequence is also fixed in advance.

In this section, we introduce a new method based on best-first searching paradigm. The CA procedure starts from the root s. Among all the direct neighbor nodes of the root, a node u with the highest *load* value is selected and the link $(s \to u)$ is then selected for channel allocation. The channel-allocated multicast tree $\overline{T} = (\overline{V}, \overline{E})$ currently covers two nodes, s and u, and one link $(s \to u)$. In next iteration, all uncovered neighbor nodes of $\overline{V}$ are considered, and the best link which can lead to a neighbor node with the highest *load* value is selected for channel allocation. The idea behind this method is to allocate channels to links with higher *load* values as early as possible. Hence, the searching sequence is not fixed and may be very

different for different multicast trees. The BFB-based CA algorithm shown in **Fig. 4** is designed based on this idea.

In **Fig. 4**, the *load* value of each node is defined and computed recursively in a bottom-up manner in line (1) ~ (2). For example, *load* values of the multicast tree shown in **Fig. 1** are computed and shown in **Fig. 2**. In line (3) ~ (5) in **Fig. 4**, a number of data structures are initialized. Among them, interference matrix *IM* is constructed based on the discussion given in **Fig. 3** and will be used for checking interference between nodes. Moreover, Linked-List L is set up for storing channel-allocated links and conflict-free channels that can be allocated to them. Linked-List *L* is designed for the *backtracking* procedure. It will be discussed in detail in next sub-section.

The best-first searching process which is guided by the *load* value is started at line (6) by initializing a priority queue to store all links incident from the root "s". Then, in the "while loop" at line (7), the best directed link $l_a = (u \to w)$ is selected and removed from Q if the node w has the largest *load* value. That is, the node with the largest *load* value has the highest priority to be selected for allocating channels. This selection criterion is called best-first strategy. In line 10, we add all the links incident from w into Q. For example, in **Fig. 2**, $Q = \{(a,c),(a,d),(a,b)\}$ before entering the while loop. Since $load(c) = 5$ is the largest load, link (a,c) has the highest priority to be selected for CA. As a result, $Q = \{(a,d),(a,b),(c,h)\}$. In the next step, link (a,d) will be the next link selected for CA since $load(d) = 4$ is the maximum load.

In line (12)~(14), we start a process to find a conflict-free channel for the selected link l_a. This process contains three steps as follows:

(1) Check if WBA can be applied.

(2) *Find_Free_Channel* procedure is used to find all the interference-free channels for l_a.

(3) The *backtracking* procedure is used to find all the interference-free channels for l_a.

If none of the above steps can allocate a channel to l_a, l_a is then deleted. The implementation and cooperation of the above three CA procedures will be discussed in detail in next section.

Input: A multicast tree $T=(V,E)$ and a set of channel $\{0,1,\ldots,c-1\}$;
(1) Let $load(v)=\sum_{u\in Y_v} req(u)$, where Y_v is a sub-tree rooted at v, $\forall v\in V$;
(2) Recursively compute the $load$ value of each node in a bottom-up manner;
(3) Initialize $channel[l]=-1$, $\forall l\in E$;
(4) Construct interference matrix IM for any two links $l_i,l_j\in E$;
(5) Define a Linked-List L where each node $\lambda\in L$ stores a link l and a candidate list Γ_l; initialize $L=\phi$;
(6) At the root s, let priority queue $Q=\{(s,v)\,|\,(s,v)\ is\ a\ directed\ link\ incident\ from\ s\}$;
(7) While ($Q\neq\phi$) {
(8) Select a link $l_a=(u,w)$ where $load(w)$ is the maximum value for all $w\in Q$;
(9) $Q=Q-\{(u,w)\}$;
(10) $Q=Q+\{(w,v)\,|\,(w,v)\ is\ a\ directed\ link\ incident\ from\ w\}$;
(11) $L=L+l_a$; $\Gamma_{l_a}=\phi$;
(12) If ($IM(l_a,l_b)=(1,0)$ && ($channel[l_b]\geq 0$)) { $channel[l_a]=channel[l_b]$;}
(13) Else if ($Find\_free\_channel(l_a,\Gamma_{l_a})>0$) {
(14) $channel[l_a]$ = the first element of Γ_{l_a}; }
(15) Else if ($Backtracking(l_a,\Gamma_{l_a})>0$){
(16) $channel[l_a]$ = the first element of Γ_{l_a}; }
(17) }
(18) Delete a link $l=(u,w)$ and the sub-tree rooted at w if $channel[l]=-1$;
(19) Recursively remove a leaf node w if $req(w)=0$;
Output: a channel-allocated multicast tree $\overline{T}=(\overline{V},\overline{E})$;

Fig. 4. Best-first and backtracking based CA algorithm.

4.3.2 Backtracking Procedure for Channel Allocation

Before the backtracking procedure starts, a linked-list must be initialized to store links and candidate lists associated with them. At the line (5) in **Fig. 4**, a linked-list L is used to store a link l_a with a candidate list Γ_{l_a}. The elements in Γ_{l_a} are the interference-free channels that can be allocated link l_a. One example for the linked-list L is shown in **Fig. 6**. At line (12), we start to do CA for the current link l_a.

To do CA for a link l_a, the first thing is to look up the interference matrix IM to find a channel-allocated sibling link l_b. If it exists, a channel allocated to l_b can be shared with l_a based on the wireless broadcast advantage. Otherwise, the *Find_Free_Channel* procedure shown in **Fig. 5** is used to find all the conflict-free channels which are stored in candidate list Γ_{l_a}. The channel of l_a is then set to be the first element of Γ_{l_a}.

Based on these candidate lists, we may have flexibility to change the channels for the allocated links so that at least one channel can be found for the current link. For example, in **Fig. 6**, we assume that no channel can be found for link (B,E) and it is interfering with link

(B,F), we may change the current channel associated with link (B,F) from 8 to 9 or 10 so that a conflict-free channel can be found for link (B,E). It is called backtracking strategy.

When it fails to allocate a conflict-free channel to a link l_a by *Find_Free_Channel* procedure, the backtracking procedure in **Fig. 7** is then invoked. The backtracking procedure works as follows. We starts to search for the first node μ in L where μ contains a link l_b which is interfering with link l_a and has no common node with l_a. In the "for loop" at line (6) in **Fig. 7**, all the feasible channels stored at Γ_{l_b} are enumerated. For each enumeration, we try to build up the candidate list Γ_{l_a}. The "for" loop stops when a non-empty Γ_{l_a} is found and returned. Otherwise, a next μ node is tested. There are at most *len* nodes in L examined before we declare that it fails to find a feasible channel for link l_a. In that case, link l_a is deleted from the multicast tree.

4.3.3 Time Complexity

The time complexity of *Find_Free_Channel* procedure is $O(c*|E|)$ where c is the number of channels. As for *Backtracking* procedure in **Fig. 7**, it also requires $O(len*c*|E|)$ steps to finish. According to our simulation results shown in **Fig. 15**, performance ratio converges when *b* value is less than four. It means that "while loop" at line 3 in **Fig. 7** executes in at most three steps before *Backtracking* procedure stops. That is, *len* can be a small constant. Hence, the time complexity of *Backtracking* procedure would be $O(c*|E|)$.

For BFB based CA algorithm shown in **Fig. 4**, the construction of interference matrix takes $O(\|E\|^2)$ steps at line 4 and the "while loop" at line 7 takes at most $O(c*|E|^2)$ steps. In total, the time complexity of BFB algorithm is bounded by $O(|E|^2)$ at the worst case since c is a constant for different wireless communication standard.

Since the time complexity of both load-oriented DFS and load-oriented BFS is $O(|V|^2 \log(|V|))$, the BFB method is more efficient than these two methods when spare networks are considered where the number of links is close to number of nodes. However, for highly dense networks like complete graphs, the BFS method would be slower than these two methods.

```
Find_Free_Channel ( l_a , Γ_{l_a} ) {
(1)  For ( c = 0 ; c < max_channel ; c++ ) {
(2)      flag_test = true ;
(3)      For ( ∀ l ∈ T && ( k = channel [l] ) ≥ 0 ) {
(4)          r = |c − k| ;
(5)          If ( r < β value stored in IM[l_a , l] ){ flag_test = false ; break;}
(6)      }
(7)      If ( flag_test ) { Γ_{l_a} = Γ_{l_a} ∪ {c} ;}
(8)  }
(9)  Return (the number of elements stored in Γ_{l_a} );
}
```

Fig. 5. Find_Free_Channel procedure

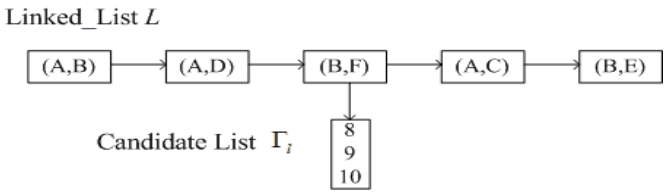

Fig. 6. Linked-List L

Backtracking (l_a){
(1) Searching Linked-List L from the head to find the first node μ containing l_a such that α value of $IM[l_a, l]$ is 3;
(2) Let len be the length of Linked-List L ; set $b = 0$;
(3) While ($b < len$) {
(4) If ($\Gamma_{l_b} \neq \phi$ && (α value of $IM[l_a, l_b] == 3$)) {
(5) m = number of channels stored in Γ_{l_b} ;
(6) For ($i = 0$; $i < m$; $i++$) {
(7) $channel[l_b] = \Gamma_{l_b}[i]$;
(8) If (($k = Find\_free\_channel(l_a, \Gamma_{l_a})$) > 0) {return k ; }
(9) } }
(10) $b++$;
(11) Try next node in L , i.e. $\mu = \mu + 1$;
(12) }
(13) return -1 ;

Fig. 7. Backtracking procedure

5. A Multicast Tree Refinement Algorithm

The channel-allocated multicast trees determined by the CA algorithms, such as DFS, BFS and BFB heuristics, may cover only a part of destinations. There are still some destinations which are not included in the channel-allocated multicast tree $\overline{T}$. Hence, we introduce a new algorithm called *Tree_Refined* in **Fig. 8** to improve the *Gain* value of $\overline{T}$ by including those uncovered nodes. However, this addition may cause some covered nodes being deleted from the tree to avoid interference. Therefore, how to trade off these nodes to obtain higher *Gain* value becomes the goal of *Tree_Refined* algorithm presented in this section. To achieve that goal, there are two sub-problems must be solved first. The first one is how to determine a path that can connect an uncovered node to $\overline{T}$. Another one is how to allocate channels to the path so that no interference is invoked in $\overline{T}$.

For each uncovered node ω , a path P which can connect ω to a node α in $\overline{T}$ is determined based on the channel-unallocated tree T . However, only path P with length no more than 3 links is considered. It is because that a longer path may cause more serious interference with the links currently in $\overline{T}$. A large amount of links may be then deleted from $\overline{T}$ when path P is

added in $\overline{T}$. As a result, it is unlikely to generate a new $\overline{T}$ with higher *Gain* value. Line (6) in **Fig. 8** is implemented with the above idea.

In **Fig. 8**, we assume that channels based on IEEE 802.11b/g standard are used for CA. The available channels are c1, c2,….c11. Among them, c1, c6, c11 are orthogonal channels. From line (7) to line (17) in **Fig. 8**, three different CA strategies are implemented. When $\|P\|$=3, channels allocated to three consecutive links must be non-overlapping. That is, channel separation must be at least 5. Hence, only three orthogonal channels, channel 1, 6, 11, can be used. In total, there are 6 different combinations for allocating these orthogonal channels to three links. For similar reason, non-overlapping channels are also required when $\|P\| = 2$. For simplicy, only channel separation equals 5 is considered. In total, there are 11 pairs of CA are considered: {c1, c6}, {c2, c7},…..{c11, c5}. As for $\|P\| = 1$, the channels can be c1 ~ c11.

For each CA scheme for path P, the procedure *Addin_Path* given in **Fig. 9** is called at line (23) to reconstruct $\overline{T}$ by adding channel-allocated path $\overline{P}$ and removing those interfering links which are generated by this addition. This reconstruction involves deletion of unnecessary leaf-nodes and recomputation of interference matrix IM. The implementation details are described in **Fig 9**. At line (26) in **Fig. 8**, the best tree is kept if the current path addition can lead to a tree with better *Gain* value.

The time complexity of *Tree_Refined* algorithm is bounded by the running time of *Addin_Path* procedure. Since the construction of interference matrix takes $O(\|E\|^2)$ steps and the number of links in P is a constant, the nested loop at line (3) ~ (11) in **Fig. 9** takes $O(\|E\|^3)$ steps at the worst case. That is, the running time of *Addin_Path* procedure is $O(\|E\|^3)$. Therefore, for the *Tree_Refined* algorithm, its running time is bounded by $O(\|M\| * \|E\|^3)$ at the worst case, where $\|M\|$ is the number of destinations.

Input: A multicast tree $T=(V,E)$, a destination set M and its channel-allocated tree $\overline{T}=(\overline{V},\overline{E})$.

(1) Construct interference matrix IM for any two links $l_i, l_j \in \overline{E}$;

(2) Let $\Lambda=\{\omega \mid \text{destination node } \omega \in M \text{ and } \omega \notin \overline{V}\}$;

(3) Sort all elements in Λ into a decreasing order based on their "req" values;

(4) Let $Tree_{best}=\overline{T}$, $Gain_{best}=Gain(\overline{T})$;

(5) For each uncovered destination node $\omega \in \Lambda$ {

(6) Find the path $P=(\alpha \to \beta \to \gamma \to \omega)$ in T where $\alpha \in \overline{T}$, $\beta,\gamma,\omega \notin \overline{T}$ and $0<\|P\| \le 3$, $\alpha,\omega \neq \phi$;

(7) If ($\|P\|==3$) {

(8) Allocate orthogonal channels to three consecutive links;

(9) There are 6 different channel assignments for P;

(10) Let $\Theta_{ca}=\{$ 6 different channel assignments found above$\}$;

(11) }

(12) Else if ($\|P\|==2$) {

(13) Allocate two non-overlapping channels (i.e. $\beta=5$) to two consecutive links;

(14) There are 11 different channel assignments for P;

(15) Let $\Theta_{ca}=\{$11 different channel assignments found above$\}$;

(16) }

(17) Else if ($\|P\|==1$) {

(18) Channel from 0 to 10, there are 11 different channel assignments for P;

(19) Let $\Theta_{ca}=\{$11 different channel assignments found above $\}$;

(20) }

(21) For each channel assignment $\lambda \in \Theta_{ca}$ for P {

(22) $\overline{P}$ = Allocate channels to P based on λ;

(23) Ψ = $Addin\_Path$ ($\overline{T}$, $\overline{P}$);

(24) Let Ψ_{best} be Ψ with the largest $Gain(\Psi)$;

(25) }

(26) If ($Gain(\Psi_{best}) > Gain_{best}$) {$Tree_{best}=\Psi_{best}$; $\overline{T}=\Psi_{best}$;}

(27) }

Output: $Tree_{best}$ and $Gain_{best}$.

Fig. 8. Tree_Refined Algorithm

```
Addin_Path (T̄, P̄) {
(1)  T̄ = T̄ + P̄ ;
(2)  Rebuild the interference matrix IM for T̄ ;
(3)  For each link l ∈ P̄ {
(4)    For each link η ∈ T̄ {
(5)      If ((η ≠ l) and (|ch(l) − ch(η)| > β value of IM[l, η])) {
(6)        T̄ = T̄ − {η} ; }
(7)    }
(8)    Remove leaf-node u from T̄ if req(u) = 0 ;
(9)    Rebuild the interference matrix IM for T̄ ;
(10) }
(11) Repeat the above nested loop until no interfering link being found;
(12) Return T̄ ;}
}
```

Fig. 9. Addin_Path procedure

6. Experimental Results

To evaluate the performance of our BFB and *Tree_Refined* CA algorithms, a number of experiments are designed in this section to compare them with previously proposed methods: load-oriented DFS and load-oriented BFS CA algorithms. The performance comparisons are carried out based on the following metrics: number of serviced clients, throughputs and end-to-end delay. The first set of data about the number of serviced clients is obtained by a set of Java programs, which are developed, based on the aforementioned CA algorithms. As for the other sets of data, they are all measured by Qualnet simulator 4.5 [21].

All the above data is obtained based on a set of multicast trees generated for our simulations. These multicast trees are constructed by the following procedure. First, we generate a random WMN with n nodes on a 100×100 grid using the $MAX-DPA$ algorithm proposed in reference [22]. Second, we set a node to be the gateway and then randomly select a given number of nodes to be the destinations for multicasting. Third, the number of mesh clients is randomly set for each destination. The settings of other simulation parameters are shown in **Table 2**.

Table 2. Simulation settings

simulation parameter	value
transmission range	10 units
interference range	20 units
wireless technology	IEEE 802.11b
antenna	omni-directional
network size	30, 60, 100 nodes
maximum degree of a node	7
# of interfaces per node	≥ 2
destination ratio (# of destinations / # of total nodes)	10% ~ 50%

# of mesh clients per destination	1~5

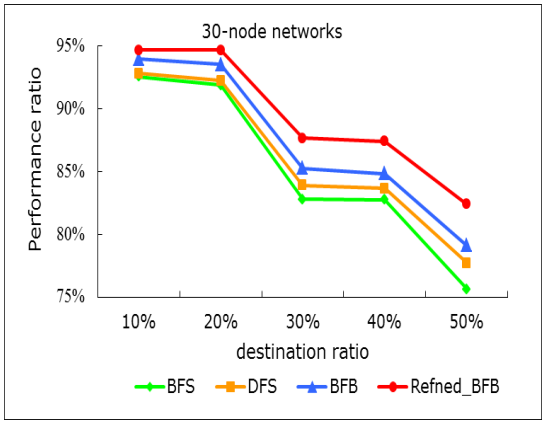

Fig. 10. Performance for 30-node networks

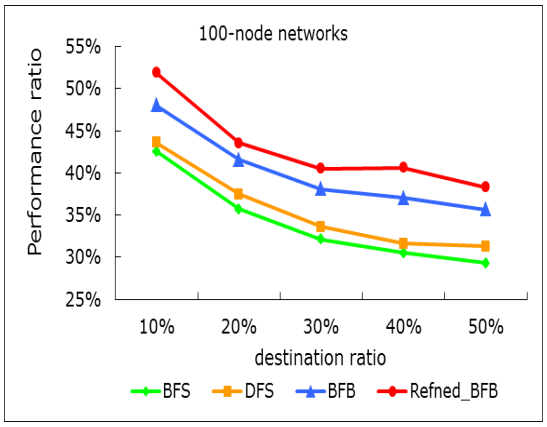

Fig. 11. Performance for 100-node networks

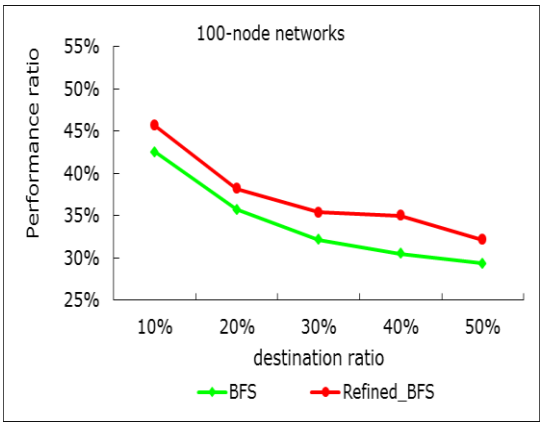

Fig. 12.. BFS vs Refined_BFS

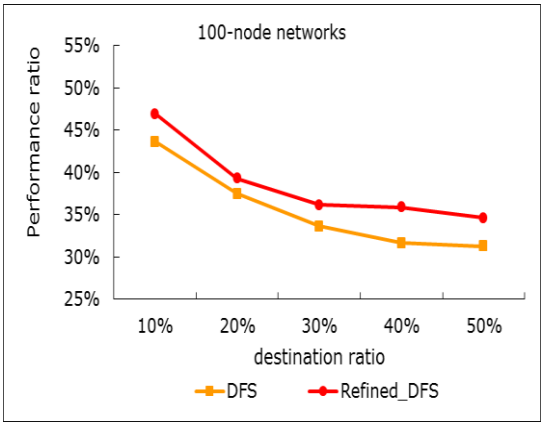

Fig. 13.. DFS vs Refined_DFS

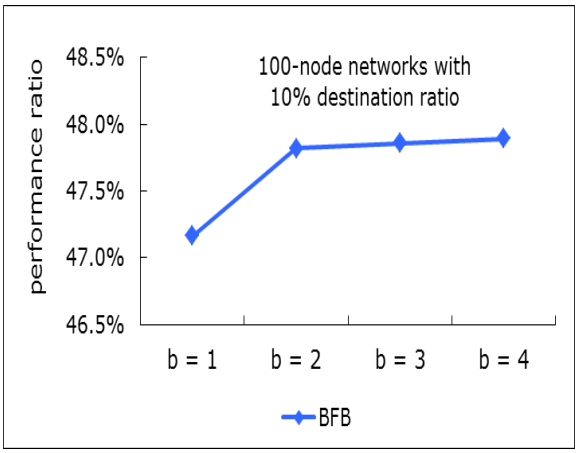

Fig. 14. b value vs performance ratio

6.1 Number of serviced clients

For measuring the performance of tested algorithms, a metric term named "performance ratio" θ is defined as follows:

$$\theta = \frac{Gain}{\Pi} \times 100\% \quad \text{(B)}$$

where $Gain$ is total number of serviced clients which is defined in equation (A) in section 3 and Π is the number of total clients initially assigned to the multicast group in the given network. Note that Π is an upper bound for the optimal value. That is, the optimal value of CA for a given multicast tree must be no more than Π. Since CA is a NP-complete problem, it is only possible to have optimal solutions for very small-size networks. Hence, we use Π to substitute for the optimal value for performance evaluations. Each data shown in **Fig. 10**, **Fig. 11**, **Fig. 12**, **Fig. 13**, and **Fig. 14** is an average value taken on 1000 runs of simulations.

(1) Performance of BFB based CA algorithm

According to the data shown in **Fig. 10** and **Fig. 11**, θ value decreases as the value of destination ratio increases for all the tested algorithms. In particular, when destination ratio increases from 10% to 50% for 30-node networks, θ value decreases from 95% to 80% for BFB. This phenomenon is because that it becomes harder to allocate conflict-free channels to tree links when the number of destinations increases.

The experimental results also show that BFB-based CA algorithm consistently outperforms load-oriented BFS and load-oriented DFS CA algorithms for various destination ratios in different sizes of networks. In addition, the superiority of BFB-based method becomes more obvious and significant when network size increases. That is, BFB-based CA method is much more suitable for large-size multicast trees. This superiority is due to the backtracking technique used in BFB approach. It allows previous channel-allocated links to have feasibility to select the other eligible channels when no conflict-free channel can be found for the current link during the CA process. In fact, besides the best-first searching strategy, the backtracking technique can also be used to enhance the performances of DFS or BFS based CA method.

(2) Optimal CA algorithm

Since CA is an NP-complete problem, it is unlikely to design an optimal polynomial-time algorithm for it. For a given multicast tree, we may allocate each link with all the possible channels. As a result, a large amount of different CA schemes are generated for evaluation. After removing interfering links, each channel-allocated multicast tree is then computed for $Gain$ value, which is the optimal solution for our problem. Since this is an exhaustive-search based method, it is only feasible for multicast trees with small number of nodes. According to our experiments, optimal CA solutions can be consistently found in a reasonable time only for multicast trees with no more than 12 nodes. In addition, our experiments also show that BFB-based CA algorithm can find optimal solutions for more than 90% of simulated multicast trees.

(3) Impact of Tree_Refined Algorithm

In this set of experiments, we study the impact on the performance ratio when the tree_refined algorithm is applied to BFS, DFS and BFB based algorithms. Based on data shown in **Fig. 10** and **Fig. 11**, the channel-allocated trees found by BFB and then processed by tree_refined procedure are consistently better than those found by the other methods. As for BFS and DFS based methods, the data in **Fig. 12** and **Fig. 13**, also shows that tree_refined algorithm can effectively improve the quality of trees constructed by them for 100-node networks. The same conclusion holds for other sizes of networks.

(4) Simulation of *b* value

For the backtracking algorithm shown in **Fig. 7**, its time complexity is determined by the number of "while loop" executed at line 3. Although the loop may take $O(|E|)$ steps at the worst case where $|E|$ is the length of linked list, the simulation results in **Fig. 14** show that the performance ratio is not improved any more if the b value is greater than three for 100-node networks. The same results are also obtained for different sizes of networks. Hence, the number of "while loop" in **Fig. 7** executed can be a small constant.

6.2 Average throughput

The data in **Fig. 15** shows the average throughput simulated by Qualnet for 20 random multicast trees constructed by BFS, DFS, BFB and refined BFB. Since the number of bits received per second is proportional to the number of destinations in the tree, the throughput increases as destination ratio increases. Based on similar reason, the throughput obtained from refined BFB is certainly be the highest since the number of destinations in the trees found by refined BFB is the largest. Hence, the descending order of throughput in **Fig. 15** is: refined_BFB > BFB > DFS > BFS. The same order is also shown in **Fig. 11**.

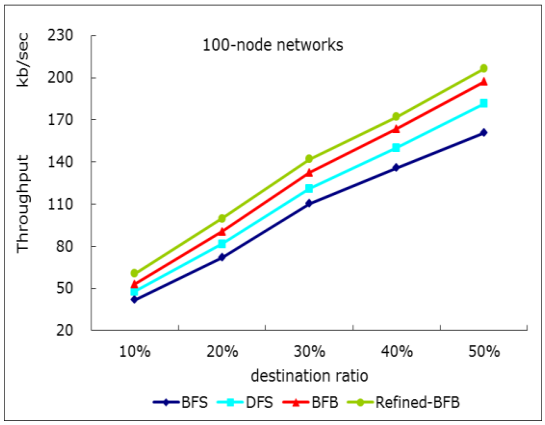

Fig. 15. Throughput for 100-node networks

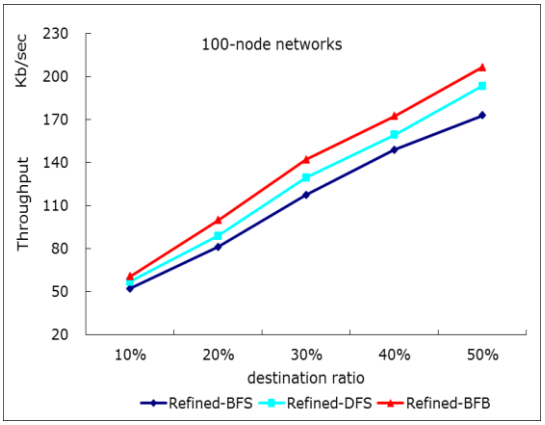

Fig. 16. Tree_Refined for 100-node networks

In **Fig. 16**, we compare the throughput for three kinds of refined multicast trees. They are first constructed by BFS, DFS and BFB methods respectively and then processed by the *Tree_Refined* algorithm. The results indicate that the refinement algorithm works well with all CA methods studied in this work.

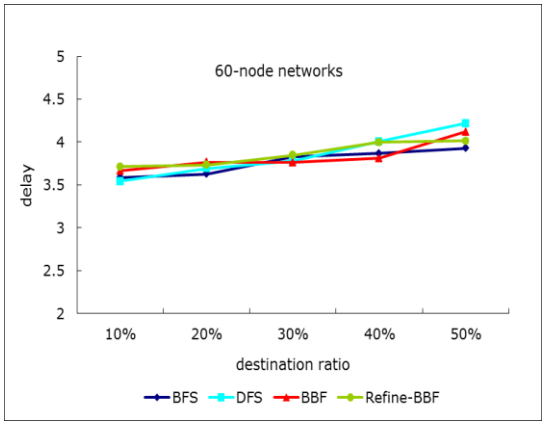

Fig. 17. Delay for 60-node networks

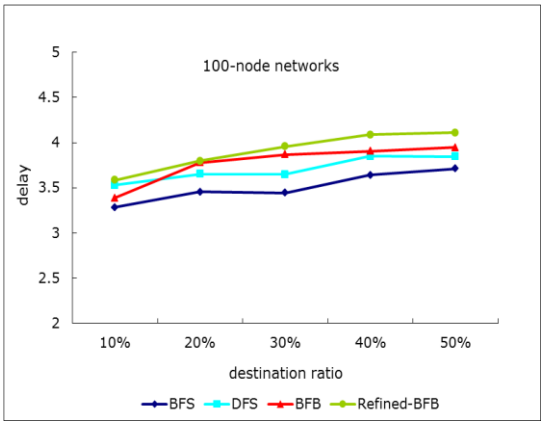

Fig. 18. Delay for 100-node networks

6.3 Average Delay

In this section, a set of simulations is set up for computing the end-to-end delay of multicast trees. The end-to-end delay of a multicast tree is defined to be the delay of the longest path among all the paths from gateway to destinations. For 60-node networks in **Fig. 17**, there is no much difference on the delay for all methods when destination ratio is no more than 40%. However, when destination ratio is 50%, the tree found by DFS contains the longest delay. It is because nodes in higher levels are examined in higher priority by DFS strategy. However, this feature does not exist for 100-node networks in **Fig. 18**. In this case, interference is so serious that paths can not grow too long in large-size networks such as 100-node networks. Therefore, the refined multicast trees built by refined_BFB tend to have a longer delay in **Fig. 18**. It is reasonable because an extra path with no more than three links is constructed and appended to the multicast tree when an uncovered destination is added into the tree. The extra path leads a longer delay.

7. Conclusion

In this paper, we present a best-first and backtracking based algorithm for allocating channels to links of a multicast tree. The objective of this work is to create an interference-free multicast tree and maximize the number of serviced mesh clients. The experimental results show that our BFB based CA algorithm outperforms previous methods such as DFS and BFS based CA methods. In addition to that, we also propose a tree refinement method which has been proved to be an effective method for improving the quality of channel-allocated multicast trees determined by BFB, BFS and DFS based algorithms. In the future work, the CA strategies studied in this paper will be applied to the approach of cross-layer design of multicast routing and channel allocation for multicasting in MCMR WMNs.

References

[1] I.F. Akyildiz, X. Wang and W. Wang, "Wireless mesh networks: a survey," *Computer Networks*, vol.47, pp.445-487, 2005. Article(CrossRef Link)

[2] EchoStream Commercial Mesh Network, http://www.inovonics.com/commercial-mesh-network.aspx

[3] Inovonics, http://www.inovonics.com/default.aspx.

[4] Wen-Lin Yang, Chi-Chou Kao and Cheng-Huang Tung, "Heuristic algorithms for constructing interference free and delay-constrained multicast trees for wireless mesh networks," *KSII Transactions on Internet and Information Systems*, vol.5, no.2, pp.269-286, Feb.2011. Article(CrossRef Link)

[5] P. Bal, A. Adya, J. Padhye and A. Wolman, "Reconsidering wireless systems with multiple radios," *ACM SIGCOMM Computer Communications Review*, vol.34, no.5, pp.39-46, 2004. Article(CrossRef Link)

[6] A. Mishra, E. Rozner, S. Banerjee and W. Arbaugh, "Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage," in *Proc. of Internet Measurement Conference*, pp.311-316, 2005. Article(CrossRef Link)

[7] Y. Ding, Y Huang, G. Zeng and L. Xiao, "Channel assignment with partially overlapping channels in wireless mesh networks," in *Proc. of WICON*, 2008. Article(CrossRef Link)

[8] Y. Ding and L. Xiao, "Channel allocation in multi-channel wireless mesh networks," *Computer Communications*, vol.34, pp.803-815, 2011. Article(CrossRef Link)

[9] X. Mao, X. Y. Li and S. K. Makki, "Static channel assignment for multi-radio multi-channel multi-hop wireless networks," in *Proc. of WTASA* 2007. Article(CrossRef Link)

[10] G. Zeng, B. Wang, Y. Ding, L. Xiao and M. Mutka, "Multicast algorithms for multi-channel wireless mesh networks," in *Proc. of International Conf. on Network Protocols*, 2007. Article(CrossRef Link)
[11] G. Zeng, B. Wang, Y. Ding, L. Xiao and M. Mutka, "Efficient multicast algorithms for multi-channel wireless mesh networks," *IEEE Transactions on Parallel and Distributed Systems*, vol.21, no.1, pp.86-99, 2010. Article(CrossRef Link)
[12] R. Daves, J. Padhye and B. Zill, "Routing in the multi-radio, multi-hop wireless mesh networks," in *Proc. of ACM MobiCom*, 2004. Article(CrossRef Link)
[13] A.P. Subramaniam, H. Gupta and S. R. Das, "Minimum interference channel assignment in multiradio wireless mesh networks," *IEEE Transactions on Mobile Computing*, vol.7, no.12, 2008. Article(CrossRef Link)
[14] H.L. Nguyen and U.T. Nguyen, "Channel assignment for multicast in multi-channel multi-radio wireless mesh networks," *Wireless Communications and Mobile Computing*, vol.9, pp.557-571, Apr.2009. Article(CrossRef Link)
[15] H.L. Nguyen and U.T. Nguyen, "Minimum interference channel assignment for multicast in Multi-Radio wireless mesh networks," in *Proc. of International Conference on Wireless Communications and Mobile Computing*, pp.626-631, 2008. Article(CrossRef Link)
[16] H.L. Nguyen and U.T. Nguyen, "Bandwidth efficient multicast routing in multi-channel multi-radio wireless mesh networks," in *Proc. of IEEE International Conference on Ultra Modern Telecommunications & Workshop*, 2009. Article(CrossRef Link)
[17] H.L. Nguyen and U.T. Nguyen, "Algorithms for bandwidth efficient multicast routing in Multi-channel Multi-radio wireless mesh networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)* , pp.1107-1112, 2011. Article(CrossRef Link)
[18] S. H. Lim, C. Kim, Y. B. Ko and N. H. Vaidya, "An efficient multicasting for Multi-Channel Multi-Interface wireless mesh networks," in *Proc. of IEEE Military Communications Conference,* 2009. Article(CrossRef Link)
[19] Wen-Lin Yang, "A maximum-revenue multicast routing problem on wireless mesh networks," in *Proc. of International Conference on Information Networking*, 2010. Article(CrossRef Link)
[20] M. Jahanshahi, M. Dehghan and M. R. Meybodi, "A mathematical formulation for joint channel assignment and multicast routing in multi-channel multi-radio wireless mesh networks," *Journal of Networks and Computer Applications*, vol.34, no.6, pp.1869-1882, 2011. Article(CrossRef Link)
[21] Qualnet simulator, http://www.scalable-networks.com/
[22] F. A. Ont, I. Stojmenovic and H. YaniKomeroglu, "Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks," *Pervasive and Mobile Computing*, vol.4, pp.597-615, 2008. Article(CrossRef Link)
[23] J. E. Wieselthier and G. D. Nguyen, "Algorithms for energy-efficient multicasting in static ad hoc wireless networks," *Mobile Networks and Applications*, vol.6, pp.251–263, 2001. Article(CrossRef Link)

Wen-Lin Yang received the PhD Degree in Computer Science from the Pennsylvania State University, University Park, USA, 1993. He is currently a Professor at the Department of Computer Science and Information Engineering in the National University of Tainan, Tainan, Taiwan. His primary research interests include routing protocols, quality of service, ad hoc networks and distributed computing.

Wan-Ting Hong was born in Taiwan, R.O.C., in 1988. She received her BS degree from the Department of Computer Science and Information Engineering, National University of Tainan, Taiwan, in 2010. Currently, she is a master student in the Department of Computer Science and Information Engineering, National University of Tainan, Taiwan. Her research interests include wireless networks and multicast routing.