A Distributed Bandwidth Reservation Algorithm for QoS Routing in TDMA-based Mobile Ad Hoc Networks

Wenjian Shao, Victor O.K. Li
Dept. of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong, China
{wjshao, vli@eee.hku.hk}

King Sun Chan
Dept. of Electrical and Computer Engineering
Curtin University of Technology
Perth, WA6845, Australia
chank@ece.curtin.edu.au

***Abstract*— In this paper, we propose a distributed bandwidth reservation algorithm for QoS routing in mobile ad hoc networks (MANET). The bandwidth resource is organized into TDMA time slots. Previous methods either rely on global information for time slot reservation, which is impractical, or suffer from the shortcut collision problem. In our method, both the hidden terminal problem and the shortcut collision problem are eliminated. Furthermore, our method is performed in a distributed manner, thus has a low computational complexity. Simulation results show that the reserved bandwidth of our algorithm is close to the upper bound.**

I. Introduction

Wireless communication systems are increasingly being used to support multimedia services. These real-time services demand Quality of Service (QoS) mechanisms to guarantee the bandwidth, delay, and delay jitter, etc. In mobile ad hoc networks (MANET), there are some QoS routing proposals [1] [2] [3] for supporting real-time services. Generally speaking, these QoS routing protocols consist of two parts: bandwidth management (including bandwidth calculation and reservation), and path set up.

Bandwidth management relies on the channel access schemes used in the MAC layer. They can be divided into two categories: contention-based random access and controlled access . CSMA/CA used in 802.11 is a typical contention-based channel multiple access scheme. It is simple and easy to implement. However, because each node contends to access the channel independently, it is difficult to satisfy the end-to-end QoS requirements. So, the QoS support in contention-based ad hoc networks is only soft QoS support [1], and will not always guarantee the service level. In controlled access MAC, resources are allocated in terms of time slots, frequencies or spreading codes. The contention-based approach is suitable for best-effort data services, while the controlled access approach is more suitable for the environments that need QoS guarantees.

Among controlled access protocols, TDMA is the most commonly used in ad hoc networks. In TDMA, a scheduling method is needed to assign the time slots to mobile nodes in networks. In [4], a centralized TDMA scheduling method uses graph theory to maximize the system throughput and avoid collisions. However, it relies on global network information which makes it unsuitable in networks with topology changes. To resolve this problem, topology-transparent algorithms are proposed in [5] and [6]. In [7] and [8], distributed scheduling methods are proposed. In [7] and [8], each channel is divided into two parts: a signalling control part and a data part. Mobile nodes negotiate with its neighbors to reserve slots in the control part and transmit data in the data part. These methods do not provide QoS routing. QoS routing needs not only bandwidth reservation for a link but also bandwidth calculation for an end-to-end routing path. In [2], a CDMA over TDMA approach, the author proposes path bandwidth calculation and reservation for ad hoc networks with multiple channels. Another algorithm is proposed for measuring the available bandwidth of a given path in single-channel ad hoc networks in [3]. However, we will explain in this paper that the algorithm in [3] is inaccurate in some circumstances.

In this paper, we discover that QoS routing may cause a collision which we called the shortcut collision problem. To resolve the shortcut collision problem and the hidden terminal problem, We propose a distributed bandwidth calculation and reservation algorithm for an end-to-end path in TDMA-based ad hoc networks. Bandwidth calculation is initiated by the source node. Every node along the path calculates the end-to-end bandwidth from the source node to itself. Bandwidth reservation is initiated by the destination node to reserve time slots for each link in the reverse direction. We also assume we can exchange signalling messages between neighbors either through a sub-channel, as in [7] and [8], or through a separate dedicated channel. This rest of this paper is organized as follows. In Section II, we describe the system model. In Section III, the shortcut collision problem is discussed. In Section IV, we present the bandwidth reservation algorithm. Numerical results are provided in Section V, and Section VI is the conclusion.

II. System Model

The notations in this paper are summarized as follows.

S: the set of all time slots.

T_k: the set of time slots being used by node k for transmission.

R_k: the set of time slots currently used for reception at k.

TS_k^t: the set of time slots available for transmission at k.

TS_k^r: the set of time slots available for reception at node k.

0-7803-8924-7/05/$20.00 (c)2005 IEEE.

In order to avoid collision at node k's neighboring nodes, TS_k^t can not be those time slots which are currently used by its neighbors for reception. TS_k^r should exclude those time slots used by its neighbors for sending. In addition, node k can not transmit and receive simultaneously. It is straightforward to obtain that

$$TS_k^t = S - T_k - R_k - \cup_{i \in \mathcal{N}_k} R_i \quad (1)$$

$$TS_k^r = S - T_k - R_k - \cup_{i \in \mathcal{N}_k} T_i \quad (2)$$

where $\mathcal{N}_k$ is the set of neighbors of node k.

We denote the path from node $n_m$to n_0 as: $n_m \rightarrow n_{m-1} \rightarrow \cdots \rightarrow n_1 \rightarrow n_0$, where n_m is the source and n_0 is the destination. This path involves $m+1$ nodes and includes m links. We call these $m+1$ nodes and m links the nodes and links belonging to the path from n_m to n_0. We use $n_i \rightarrow n_{i-1}$ to represent the link from node n_i to node n_{i-1}.

Let $TS_{i,j}$ be the set of time slots which are available for node i to deliver packets to its neighboring node j over link $n_i \rightarrow n_j$. $TS_{i,j}$ should be the intersection of the set in which node i can send and the set in which node j can receive.

$$TS_{i,j} = TS_i^t \cap TS_j^r \quad (3)$$

We denote $PTS_{i,i-1}$ as the set of time slots that is available for delivering packets over link $n_i \rightarrow n_{i-1}$ without colliding with time slots on other links. The difference between $TS_{i,i-1}$ and $PTS_{i,i-1}$ is that $TS_{i,i-1}$ includes the time slots that may collide with time slots over neighboring links while $PTS_{i,i-1}$ does not. Obviously, $PTS_{i,i-1} \subseteq TS_{i,i-1}$. The goal of a bandwidth reservation algorithm is to calculate the proper PTS for each link along the path so as to resolve all collisions and achieve optimal end-to-end bandwidth.

We will explain the collision between the links of a path in the next section. Here we define a new notation called the link distance for analyzing the collision problem. For two links $n_{i+1} \rightarrow n_i$ and $n_{j+1} \rightarrow n_j$ belonging to a path, the link distance is the distance in terms of hops between the two transmitting nodes n_{i+1} and n_{j+1} along this path.

III. Shortcut Collision Problem

According to [2] and [3], the time slots used by a node should not collide with those of neighboring nodes. Furthermore, to avoid the hidden terminal problem, any three consecutive nodes along a path should use different time slots. However, such a condition is in fact not sufficient, since it does not consider the scenario of the shortcut collision problem. Fig. 1 shows an example illustrating this problem.

Based on the algorithm in [3], two time slots can be allocated along the path $n_5 \rightarrow n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \rightarrow n_0$. Note that path $n_5 \rightarrow n_4 \rightarrow n_1 \rightarrow n_0$ is not selected since there is no bandwidth available for link $n_4 \rightarrow n_1$. In fact, only one time slot is available for the path from n_5 to n_0. From Fig. 1, we see that n_5 will use time slots $(0,1)$ to transmit to n_4, and n_1 also uses $(0,1)$ to transmit to n_0. Unfortunately, since n_4 and n_1 are neighbors, time slots 0 and 1 cannot be used by the two nodes at the same time. In other words, time slots 0 and 1 should be split between link $n_5 \rightarrow n_4$ and link $n_1 \rightarrow n_0$. Therefore, there is actually only one time slot available over the whole path. The

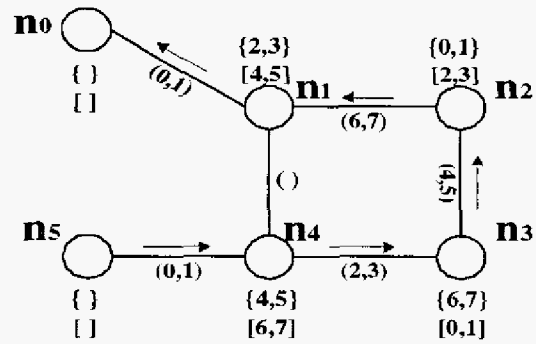

Fig. 1. An example of the shortcut collision problem. Assume the frame size is 8 time slots, labelled from 0 to 7. The numbers in {} represent the slots used by nodes for transmitting; the numbers in [] represent the slots used by nodes for receiving; the numbers in () are the available slots of the associated link calculated by formulae (1), (2) and (3).

algorithm in [3] does not consider the conflict caused by the shortcut path $n_4 \rightarrow n_1$, and is thus inaccurate in bandwidth calculation.

The reason for this inaccuracy is that in the algorithm in [3], one implied assumption is that two nodes more than or equal to two hops away along the path are not neighbors, which is true in the shortest path routing. However, in QoS routing, the path may not be the shortest path since the shortest path may not satisfy the QoS requirements. So, above assumption is untenable in general. For example, in Fig. 1, we do not choose the shortest path $n_5 \rightarrow n_4 \rightarrow n_1 \rightarrow n_0$ with the consideration of bandwidth requirement. In the selected path $n_5 \rightarrow n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \rightarrow n_0$, n_4 and n_1 are three hops away along the path, but they are neighbors.

The link $n_4 \rightarrow n_1$ does not belong to the path $n_5 \rightarrow n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \rightarrow n_0$. However, it is a shortcut between n_4 and n_1. It actually shortens the distance between some links belonging to the original path and adds a potential collision. We name this type of link a shortcut link and call the new collision a shortcut collision and the problem caused by it as the shortcut collision problem.

We observe that shortcut collision occurs between two links which satisfy the following two conditions:

1) These two links are connected through the shortcut link.
2) The distance between these two links is larger than two hops along the original path and exactly equal to two hops through the shortcut.

For example, in Fig. 1, link $n_5 \rightarrow n_4$ and $n_1 \rightarrow n_0$ are contending links. The distance between $n_5 \rightarrow n_4$ and $n_1 \rightarrow n_0$ is four hops through path $n_5 \rightarrow n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \rightarrow n_0$ but two hops through shortcut $n_5 \rightarrow n_4 \rightarrow n_1 \rightarrow n_0$.

Let us analyze the cause of the shortcut collision phenomenon. Assume n_i and n_j are two adjacent nodes with distance larger than two hops along the path, link $n_i \rightarrow n_j$ is a shortcut link. Scheduling some time slots for link $n_{i+1} \rightarrow n_i$ will increase the size of R_i because i is the receiver. This will decrease the size of TS_j^t because i is the neighbor of j and TS_j^t is calculated according to (1). The decrease in the size of TS_j^t will affect the scheduling of link $n_j \rightarrow n_{j-1}$, because $TS_{j,j-1}$ is decided by (3). Thus, link $n_{i+1} \rightarrow n_i$ and link $n_j \rightarrow n_{j-1}$

are contending links and the distance is two hops through the shortcut. The increase in the size of R_i will not affect the size of TS_j^T according to (2). Since TS_j^T is not affected, the scheduling of link $n_{j+1} \rightarrow n_j$ is not affected. So $n_{i+1} \rightarrow n_i$ and link $n_{j+1} \rightarrow n_j$ are not contending links and the distance is three through the shortcut.

To avoid the hidden terminal collision and the shortcut collision, We conclude the sufficient conditions for no collision at link $n_i \rightarrow n_{i-1}$ are:

$$\begin{cases} PTS_{i+2,i+1} \cap PTS_{i+1,i} \cap PTS_{i,i-1} = 0 \\ PTS_{i,i-1} \cap PTS_{i-1,i-2} \cap PTS_{i-2,i-3} = 0 \end{cases} \quad (4)$$

$$PTS_{i,i-1} \cap (\cup_{j \in P_i} PTS_{j+1,j}) \cap (\cup_{k \in Q_{i-1}} PTS_{k,k-1}) = 0 \quad (5)$$

where P_i is the set of neighbors of node n_i, and Q_{i-1} is the set of neighbors of node n_{i-1}.

(4) is used to resolve the hidden terminal problem and (5) is introduced to resolve the shortcut collision problem. To schedule collision-free slots for a given path, we need to reserve and allocate appropriate PTS for each link belonging to the path. The PTS for each link must comply with (4) and (5). In the next section, we will discuss how to reserve and allocate appropriate PTS.

IV. Bandwidth Reservation Algorithm

In this section, we discuss the algorithms to reserve appropriate time slots for each link to avoid all potential contentions between the links and maximizing the end-to-end bandwidth. The bandwidth reservation algorithm consists of two parts. The first part calculates the collision-free bandwidth for each link of the path from source to destination, thus getting the end-to-end bandwidth of the path. The second part reserves time slots for each link in the reverse direction from the destination to the source.

First we discuss the bandwidth calculation problem. The end-to-end bandwidth of a path is decided by the bandwidth of the bottleneck link. Let PBW be the end-to-end bandwidth of a path.

$$PBW = min|PTS_{i,i-1}|, \quad i = m, m-1, \cdots, 1 \quad (6)$$

The key of the bandwidth calculation is to maximize the capacity of a path. So, the bandwidth calculation problem is actually a max-min fairness scheduling problem. Max-min fairness scheduling has been discussed in [9], which is concerned with the contention and scheduling problem between end-to-end flows. In this paper, we consider the contention and scheduling problem between links within an end-to-end path.

To study the contention problem, we introduce the link contention graph $G = (N, A)$ of a path, where N is the set of vertices of the graph and represents the links belonging to a path, A is the set of edges of the graph and represents the contention existing between two links on the routing path. The contention graph for the path in Fig. 1 is presented in Fig. 2. Note that each vertex represents a link rather than a node. If two links are connected by an edge, it means there is potential contention between them. If two potential contending links

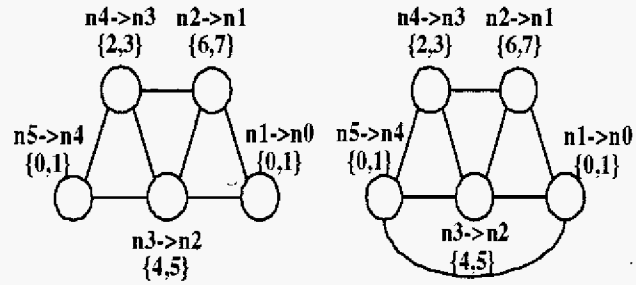

Fig. 2. Link contention graphs. The numbers within { } denote the available time slots, namely $TS_{i,i-1}$, for link $i \rightarrow i-1$. The left figure is the contention graph considered by the algorithm in [3], while the right figure is the contention graph considered by our algorithm.

share common slots, it means they contend for those common slots. Comparing the two parts of the contention graph, we include the shortcut collision by introducing an edge between the vertices labelled $n_5 \rightarrow n_4$ and $n_1 \rightarrow n_0$. So, the shortcut collision problem can be resolved.

Now, our task is to schedule and reserve time slots in the link contention graph to eliminate all collisions between links and achieve max-min fairness, thus, maximizing the bandwidth of the bottleneck link.

There are two solutions: one is centralized using global information, another is distributed computation by each node. Even if we do not consider other defects of a centralized solution, it has been proven in [9] that it is an NP-hard problem to schedule and reserve bandwidth in a global contention graph. So, we offer a distributed, iterative bandwidth calculation heuristic in this paper.

The basic idea for our algorithm is to iteratively decompose the contention graph into local cliques vertex by vertex from the source to the destination. Then contentions are resolved in each local clique. Since there is no global contention information, a downstream node is in charge of discovering the contentions, and tries to allocates time slots among contending links in a fair manner. Then the allocation results are passed to the next node for the next round of iteration. Repeating this procedure vertex by vertex toward the destination, all contentions are resolved. The steps of allocation and collision resolution for the path in Fig. 1 are shown in Fig. 3.

The bandwidth reservation algorithm for the path from n_m to n_0 is listed as:

Part 1 Bandwidth calculation algorithm (from source to destination)

When node n_{i-1} receives a bandwidth calculation request from an upstream node n_i:

Init: Get all upstream links' computed PTSs carried in the request: $\{PTS_{m,m-1}, PTS_{m-1,m-2}, \cdots, PTS_{i+3,i+2}\}$. Get the nearest two upstream links' TS carried in the request: $\{TS_{i+2,i+1}, TS_{i+1,i}\}$. Identify all upstream links that cause shortcut collisions with link $n_i \rightarrow n_{i-1}$. Let L_{usc} be the set of these links and let PTS_{usc} be the set of these links' PTSs;

Do

Case 1: $i = m$ // special case for one-hop paths

$PTS_{m,m-1} = TS_{m,m-1}$

Case 2: $i = m - 1$ // special case for two-hop paths

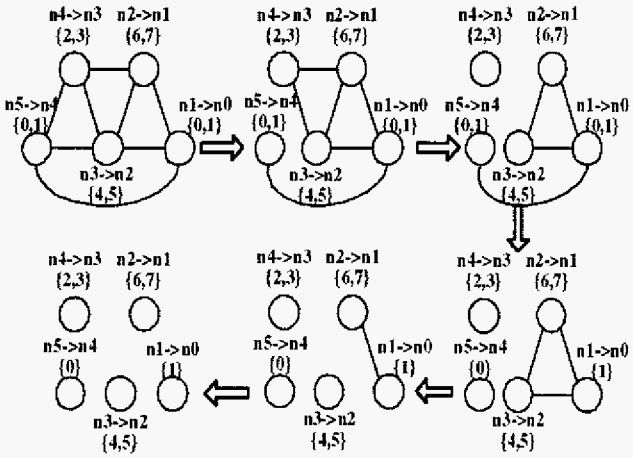

Fig. 3. Steps for bandwidth allocation and collision resolution.

$(PTS_{m,m-1}, PTS_{m-1,m-2}) = FA2(TS_{m,m-1}, TS_{m-1,m-2})$
Case 3: // normal iteration
Foreach PTS in PTS_{usc}
$(PTS, TS_{i,i-1}) = FA2(PTS, TS_{i,i-1})$
$PTS_{i+2,i+1} = FA3(TS_{i+2,i+1}, TS_{i+1,i}, TS_{i,i-1})$
$TS_{i+1,i} = TS_{i+1,i} - PTS_{i+2,i+1}$
$TS_{i,i-1} = TS_{i,i-1} - PTS_{i+2,i+1}$
If $i = 1$ // special case for the last hop
$(PTS_{2,1}, PTS_{1,0}) = FA2(TS_{2,1}, TS_{1,0})$
End
where $FA2$ (two-point fairness algorithm) allocates time slots fairly between two links and $FA3$ (three-point fairness algorithm) allocates time slots fairly among three contending links. Details of these two algorithms are given in appendix.
Part 2 Bandwidth reservation algorithm (from destination back to source)
When node n_i receives a bandwidth reservation request for k time slots from a downstream node n_{i-1}:
Init: Get $PTS_{i,i-1}$ carried in the request;
Do
$V = PTS_{i,i-1} - TS_i^r - TS_{i-1}^t$
If $|V| \geq k$
Reserve k time slots in set V for link $n_i \rightarrow n_{i-1}$
Else
Reserve all the time slots in set V for link $n_i \rightarrow n_{i-1}$
$k' = k - |V|$; $V' = PTS_{i,i-1} - V$
Reserve k' time slots in V' for link $n_i \rightarrow n_{i-1}$
Update T_i, R_i, TS_i^t and TS_i^r
End

Now, we give some explanation to the second part of the bandwidth reservation algorithm. After the first part of our algorithm, we get the PTS of each link and the end-to-end bandwidth, PBW, of the path. Assume k time slots are to be reserved for a path and k is less than or equal to the PBW of the path. k time slots are reserved for each link belonging to the path from the destination back to the source. Since we know the end-to-end bandwidth of the path and each link's collision-free time slots, each link selects k time slots from its PTS for this reservation. By carefully choosing these k slots, we try to minimize the impact of this reservation on the transmission and reception abilities of the link's start point and end point. The goal is to increase the available resources for further resource reservation on other paths, hence improving the system utilization.

V. Numerical Results

We study the performance of the bandwidth reservation algorithm by computer simulation. In our simulation, we fix the number of time slots, S, to 32. The end-to-end path consists of m links from n_m to n_0. The availability of each slot at a link is decided by an iid Bernoulli trial with probability P_a. The average number of available time slots at a node is thus $S \cdot P_a$. The number of shortcut collisions of an end-to-end path is n. Each of these n shortcut collisions is selected randomly between two nodes. In order to distinguish the shortcut collision from hidden terminal collision, the distance between two nodes is larger than two hops. By adjusting the value of parameters P_a, m and n, we can study the performance of the algorithm in different situations.

We need an upper bound to evaluate the efficiency of our algorithm. We get the upper bound through enumeration. The enumeration method calculates all possible allocation schemes and chooses the best one from them.

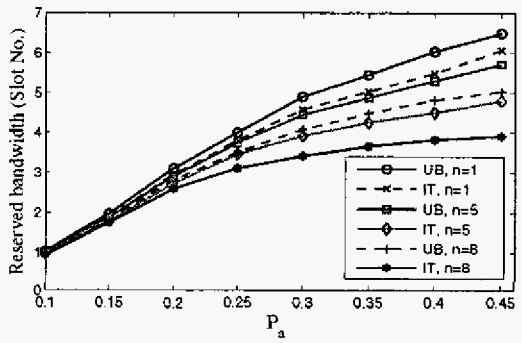

Fig. 4. Reserved bandwidth comparison between our algorithm and the enumeration method.

In Fig. 4, we fix the length of the paths to 8 hops and vary the shortcut collision number to observe the influence of shortcut collision. We compare the bandwidth reserved by our iterative algorithm (IT) with that obtained from enumeration (UB). From the results in Fig. 4, we can see that our method has small degradations from the enumeration method. However, the enumeration method has a complexity of $O((S \cdot P_a)^6)$, which is much more complex than our algorithm and is not practical. We also find that the reserved bandwidth of both the enumeration method and our algorithm declines when the number of shortcut collision increases. This is obviously due to the influence of shortcut collision on the bandwidth reservation. So, the routing protocol should try to select the path with less shortcut collisions.

In Fig. 5, we fix the shortcut collision number to 1 and change the length of paths to see the influence of the path length on the reservation results. We find that for longer paths, the reserved bandwidth is smaller. So, we prefer to select the shorter path in

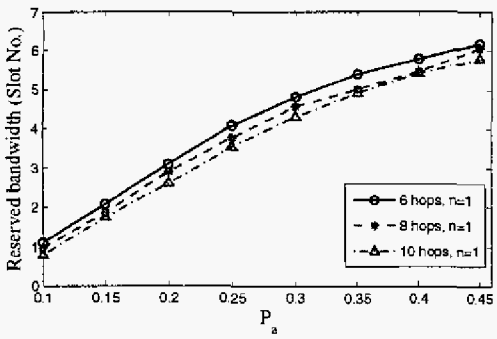

Fig. 5. Reserved bandwidth comparison among paths with different length.

routing. However, the impact of the path length is insignificant, compared with that of the shortcut collision.

VI. Conclusion

We propose a distributed end-to-end bandwidth allocation algorithm for TDMA-based MANET. Our algorithm can resolve the shortcut collision problem as well as the hidden terminal problem. Its distributed nature makes it efficient to compute. Simulation results also show the influence of the shortcut collision and the path length on the bandwidth reservation. As we have pointed out, bandwidth management is only one part of QoS routing. We also need a routing protocol to set up the end-to-end path. Based on the algorithm proposed in this paper, we are working on a new QoS routing protocol. The new protocol has two phases. First, the source tries to find a path with sufficient bandwidth. When the eligible path is found, the destination initiates bandwidth reservation to reserve bandwidth for the path.

Acknowledgments

This research is supported in part by a Croucher Senior Research Fellowship, the Croucher Foundation.

Appendix

A. Two-point fairness algorithm

Input: $INPUT_1, INPUT_2$
Output: $OUTPUT_1, OUTPUT_2$
Do
$U_{12} = INPUT_1 \cup INPUT_2$
$I_{12} = INPUT_1 \cap INPUT_2$
$E_1 = INPUT_1 \cap \overline{INPUT_2}$
$E_2 = INPUT_2 \cap \overline{INPUT_1}$
$average = \lfloor |U_{12}|/2 \rfloor$
If $|E_1| \geq average$
$OUTPUT_1 = E_1$
Else
$\delta = average - |E_1|$; move δ items from I_{12} to E_1
$OUTPUT_1 = E_1$
End *If*
$OUTPUT_2 = U_{12} \cap \overline{OUTPUT_1}$
End

B. Three-point fairness algorithm

Input: $INPUT_1, INPUT_2, INPUT_3$
Output: $OUTPUT$
Do
$U_{123} = INPUT_1 \cup INPUT_2 \cup INPUT_3$
$I_{123} = INPUT_1 \cap INPUT_2 \cap INPUT_3$
$I_{12} = INPUT_1 \cap INPUT_2 \cap \overline{INPUT_3}$
$I_{13} = INPUT_1 \cap INPUT_3 \cap \overline{INPUT_2}$
$I_{23} = INPUT_2 \cap INPUT_3 \cap \overline{INPUT_1}$
$E_1 = INPUT_1 \cap \overline{INPUT_2} \cap \overline{INPUT_3}$
$E_2 = INPUT_2 \cap \overline{INPUT_1} \cap \overline{INPUT_3}$
$E_3 = INPUT_3 \cap \overline{INPUT_1} \cap \overline{INPUT_2}$
$average = \lfloor |U_{123}|/3 \rfloor$
If $|E_1| \geq average$
$OUTPUT = E_1$
Else
$\delta = average - |E_1|$
While $\delta > 0$
$S_2 = |E_2| + |I_{12}|$; $S_3 = |E_3| + |I_{13}|$
If $S_2 \geq S_3$
try to move one item from I_{12} to E_1
If fails, try to move one item from I_{123} to E_1
If fails, try to move one item from I_{13} to E_1
If fails, jump out of While loop
Else
try to move one item from I_{13} to E_1
If fails, try to move one item from I_{123} to E_1
If fails, try to move one item from I_{12} to E_1
If fails, jump out of While loop
End *If*
$\delta = \delta - 1$
End *While*
$OUTPUT = E_1$
End *If*
End

References

[1] S.-B. Lee, G.-S. Ahn, X. Zhang, A. T. Campbell, "INSIGNIA: an IP-based quality of service framework for mobile ad hoc networks," *Journal of Parallel and Distributed Computing*, vol. 60, pp. 374-406, Apr. 2000.

[2] C. R. Lin, "Admission control in time-slotted multihop mobile networks," *IEEE JSAC*, vol. 19, no. 10, pp. 1974-1983, Oct. 2001.

[3] C. Zhu and M. Corson, "QoS routing for mobile ad hoc networks," *Proc. IEEE INFOCOM'02*, New York, USA, June 2002, pp. 958-967.

[4] R. Nelson and L. Kleinrock, "Spatial TDMA: A collision free multihop channel access protocol," *IEEE Trans. Commun.*, vol. 33, pp. 934-944, Sept. 1985.

[5] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Trans. Networking*, vol. 2, pp. 23-29, Feb. 1994.

[6] J. Ju and V. O.K. Li, "An optimal topology-transparent scheduling method in multihop packet radio networks," *IEEE/ACM Trans. Networking*, vol. 6, pp. 298-306, June 1998.

[7] I. Cidon and M. Sidi, "Distributed assignment algorithm for multihop packet radio networks," *IEEE Trans. Comput.*, vol. 38, pp. 739-746, Oct. 1989.

[8] L.C. Pond and V. O.K. Li, "A distributed time-slot assignment protocol for mobile multi-hop broadcast radio networks," *Proc. IEEE MILCOM'89*, Boston, USA, Oct. 1989, pp. 70-74.

[9] X. L. Huang and B. Bensaou, "On max-min fairness and scheduling in wireless ad-hoc networks: analytical framework and implementation," *Proc. ACM MobiHoc'01*, California, USA, Oct. 2001, pp. 221-231.