A Class-Based Queueing Service for IEEE 802.11e Wireless Networks

Chyouhwa Chen, Huei-Wen Ferng, Jun-Chuan Chen, Hao-Lun Chin, and David Shiung†

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology Taipei 106, Taiwan
E-mail: hwferng@mail.ntust.edu.tw

†Novatek Microelectronics Corp.
2F, No. 13, Innovation Road I, Science-Based Industrial Park, Hsin-Chu, 300, Taiwan

Abstract— **In order to fairly access a shared wireless channel and allocate network resources, resource management mechanisms are required for controlling link-sharing in wireless local area networks (WLANs), which are expected to support real-time and multimedia applications. We then propose an extended Class-Based Queueing (CBQ) service in this paper based on controlled hierarchical link-sharing model with QoS-aware MAC protocol of IEEE 802.11e to support sharing of the link bandwidth among different users in a distributed manner. Dynamically adjusting parameters in IEEE 802.11e, e.g., AIFS, $\mathrm{CW}_{\mathrm{min}}$, and $\mathrm{CW}_{\mathrm{max}}$, which are used to control the probability to access the channel, our scheme is able to achieve fair channel sharing. Using a simulation approach, we demonstrate the fairness improvement of our scheme is good.**

I. Introduction

Currently, WLANs are targeted to provide not only data transmission, but also multimedia applications, e.g., video, audio etc. For multimedia applications, most of them require real-time services. To support real-time services, IEEE 802.11 Task Group E proposed a QoS-aware MAC protocol called 802.11e which defines eight priority types. Its purpose is to let packets with higher priorities get the channel easier than those with lower priorities. Thus, one can set a higher priority to the real-time traffic to satisfy its QoS requirements in throughput or delay bound etc. However, the priority setting may results in the starvation problem for users with lower priorities. Therefore, we propose a controlled link-sharing mechanism, which is similar to CBQ proposed in [3], to guarantee that each service can get necessary channel allocation and to prevent traffic with lower priority from starvation. There are some previous papers in the literature have addressed this issue, e.g., [1], [4], [5], [8], [9], and [10]. Kanodia et al. [6], [7] proposed the distributed priority scheduling which modifies the RTS/CTS mechanism of 802.11 MAC. In [4], Fragouli et al. studied a bandwidth sharing policy and achieved two goals: (i) to coordinate multiple users in sharing and (ii) to have best utilization. Bhagwat et al. [2] proposed the Channel-State Dependent Packet Scheduling (CSDPS). They thought that the channel utilization can be promoted if the number of links with worst QoS can be reduced. As for the purpose of our paper, we want to enable channel allocation for each service more reasonable. Since distributing the process to coordinate the channel sharing gets better efficiency than the centralized one. Thus, we propose a distributed frame structure for CBQ in IEEE 802.11 to support channel sharing.

The remainder of the paper is organized as follows. Section II reviews the CBQ and the distributed priority scheduling. Section III describes the extended CBQ mechanism. In Section IV, we provides the numerical experiments and discussions. Finally, Section V concludes the paper and outlines the possible future work.

II. Review of CBQ and Distributed Priority Scheduling

A. The CBQ Mechanism

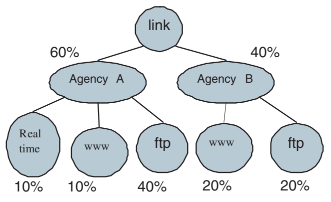


Fig. 1. A hierarchical link-sharing structure.

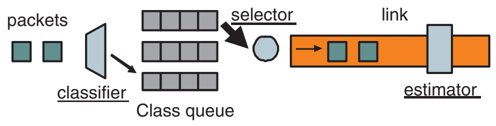


Fig. 2. The CBQ mechanism.

Let us first give a brief description on CBQ. Fig. 1 shows a hierarchical link-sharing structure in which the lowest level classes are called leaves, the highest level class is named as root, and the others are called interior classes. Each class records its desirable percentage of channel allocation. we assume that there is a corresponding device, such as router or gateway to implement the following two things: (i) To guarantee channel allocation for each class. (ii) To reallocate channels not used.

To fulfill the goal of CBQ, some modules, i.e., classifier, estimator and selector are required and shown in Fig. 2. A classifier classifies packets, assigns them class-IDs, and stores them into the class queue. The function of an estimator is to figure out the traffic flow in a certain period for each class. Then the information is used to estimate whether enough channels or not have been assigned. Using the information from the estimator, a selector has to decide which class queue is allowed to send packets so that each class satisfies its bandwidth requirement. According to [3], a selector should implement a general scheduler and a link-sharing scheduler to achieve the goal of CBQ resource management. The general scheduler is to be used to schedule the class queues if the allocated bandwidth for each class can meet the requirement. Otherwise, the link-sharing scheduler is used to adjust the transmission rates. For the sake of convenience, let us define the following states.

- Over-limit: it stands for more data than a specified value during a period has been received by a class.
- Under-limit: it stands for less data than a specified value during a period has been received by a class.
- At-limit: it represents a state that is neither over-limit nor under-limit.
- Unsatisfied: it means that a class belongs to the under-limit class and the backlog exists.
- Satisfied: it means that a class is not unsatisfied.
- Regulated: it says that a class is scheduled by link-sharing scheduler.
- Unregulated: it says that the transmission rate of a class is not limit by any scheduler.

Using the above states, the three functions of CBQ are stated as follows: (i) an estimator can measure the limit status for each class; (ii) it can decide whether the class is satisfied or unsatisfied. (iii) if a class is unsatisfied, CBQ will limit the traffic flow of other classes to satisfy the unsatisfied class.

B. Distributed Priority Scheduling

In order to provide the QoS service in a random access wireless network, we should solve the issue of medium access and packet scheduling. That is, we need to determine which wireless terminal (WT) to get the channel and it can meet its QoS requirement. Kanodia et al. proposed the distributed priority scheduling which modifies the RTS/CTS exchange in IEEE 802.11. Thus, each WT needs to know the priority of every package before sending. By this way, WTs can negotiate with each other to fulfill the QoS. But how to let all WTs know the transmission order of other neighboring WTs. A four-way handshaking should be adopted in IEEE 802.11. Before data transmitting, the WT uses RTS/CTS exchange to avoid interference from other WTs and adds the priority and the order of the current head-of-line packet into the RTS/CTS exchange. For other WTs, which would not like send data, over hear the channel to obtain the priority/order information; then decide their order of the head-of-line packet. Thus, a table containing priorities of head-of-line packets can be established. Finally, all WTs know when to transmit their packets so that the QoS requirements are satisfied.

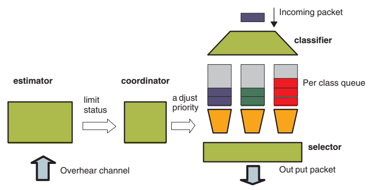


Fig. 3. The extended CBQ mechanism.

```
Classifier
Begin
        get a packet P from upper layer
        set i = P.classid
        insert P into class_queue[i]
end
```

Fig. 4. The pseudo code of a classifier.

```
Selector
BEGIN
        FOR i = 0 TO count(class_queue)
                IF class_queue[i].backoff_timer is expired THEN
                        IF class_queue[i].length > 0 THEN
                                SET RTS.backog_flag = ture
                                CALL 802_11_four_way_   hand_shake_procedure()
                        END IF
           END IF
        NEXT i
END
```

Fig. 5. The pseudo code of a selector.

III. The Extended CBQ Mechanism

In this paper, we implement CBQ within every wireless terminal in a distributed manner. Except original eight class queues, we add four modules to QoS-aware MAC protocol of IEEE 802.11e. The four modules are $classifier$, $selector$, $estimator$, and $coordinator$, respectively which are shown in Fig. 3. We now detail these four modules as follows:

A. Classifier

A $classifier$ is used to classify arriving packets to appropriate classes which will be added to the header of every packet. According to the class-ID, packets are put into each class queue. In Fig. 4, the pseudo code of a classifier is shown.

B. Selector

A $selector$ is used to determine which packet can be sent from one of the eight class queues. In the wireless network, the selected packet needs to coordinate with other wireless terminals because of the shared media. Thus, a selector can use the randomized backoff timer to avoid collision. That is to say, the eight class queues have their own independent backoff timers. Once a backoff timer of the class queue expires, then the packet in that queue can be sent. Fig. 5 shows the pseudo code for the selector.

```
Estimator
BEGIN
        overhear a RTS/CTS packet P from channel
        SET i = P.classid
        IF flag[i] < (now + P.duration) THEN
                SET class_accumulative_traffic[] += ((now + P.duration)       – flag[i])
                SET flag[i] = (no  w + P.duration)
                IF P.backlog_flag is   true  THEN
                        SET class_backlog[i] += 1
                END IF
        END IF
END
```

Fig. 6. The pseudo code of an estimator.

C. Estimator

An *estimator* is used to estimate the bandwidth utilization for each class. We can add the class-ID of the packet to RTS/CTS and record the packet transmission time. Thus, we can exchange the bandwidth usage information from other wireless terminals. Here, a class backlog array is used to record the backlog of the class queue in each wireless terminal. The backlog information then can be used by a coordinator to determine whether the class is satisfied or not. Fig. 6 shows the pseudo code of an estimator.

D. Coordinator

The function of a *coordinator* uses the differential service and the bandwidth usage information measured by the estimator to dynamically adjust the priority for each class. The work of the coordinator can be split into two parts. For the first part, a coordinator should realize whether the allocated channels are enough or not for a class. If a class does not obtain enough channels, then the coordinator starts the second part work to coordinate priorities of other classes to fulfill the bandwidth requirement for each class. Let us describe these in more details in the following.

1) Determination of satisfaction: In order to determine a class is satisfied or unsatisfied, we need to know whether the backlog for a class queue occurs. If the class is satisfied, then one of the following conditions occurs: (i) The class is overlimit; (ii) The class is underlimit and there is no backlogs in its class queue. Therefore, the class is underlimit and its class queue has backlogs only when the class is unsatisfied.

2) Channel-sharing guideline for the flat structure: Considering only one level and two classes in a WLAN, namely, class A and class B. Thus, we want to develop a scheme to fulfill priorities of classes so that the bandwidth allocation is achieved, say, 40% for class A and 60% for class B. After the coordinator receives the status from the estimator, it decides whether to upgrade or degrade the priority for a specific class by the following rules: (i) a class priority should be upgraded if the class is unsatisfied; (ii) a class priority should be degraded if the class is at its overlimit and there exist other unsatisfied classes. We adjust the priority for each class so that it can satisfy the expected CBQ bandwidth. The key point is that as long as differences of priorities between two classes are enough, then it can satisfy the CBQ requirement. Our target is to reach the channel sharing and the best channel efficiency simultaneously. Fig. 7 shows the pseudo code pertinent to how the coordinator works. The coordinator obtains a survey from

```
Coordinator
BEGIN
WHEN estimator.timer expired THEN
SET BW = sum(all class_accumulative_traffic[])
FOR  i = 0 TO count(class_queue[])
  SET class_actual_BW_per   centage[i]=class_accumulative_traffic[i] / BW
        IF class_ actual _ BW_percentage[i]>class_target_BW_percentage[i] THEN
        SET class_status[i] =overlimitT
        ELSE
                SET class_status = underlimit
        END IF
NEXT i
FOR i = 0 to count(class_queue[])
        IF class_status[i]   is UNDERLIMIT AND class_backlog[i] > 0 THEN
                IF class_priority[i] is not highest priority THEN
                        increase class_priority[i]
                        SET adj_flag =   true
                END IF
                SET unsatisfied_flag = true
        END IF
NEXT i
FOR i = 0 TO count(class_queue[])
        IF  unsatisfied  _fla g_ is true THEN
                IF class_status[i] is overlimit AND adj_falg is not true THEN
                        decrase class_priority[i]
                END IF
        END IF
NEXT i
reset estimator.timer
clear  value of class_accumulative_traffic[]
END
```

Fig. 7. The pseudo code of a coordinator.

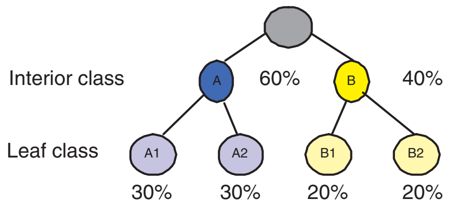


Fig. 8. A hierarchical channel-sharing structure example.

the estimator. Then the coordinator searches for unsatisfied classes to increase their priorities. If a specific class is at its highest level, the coordinate searches for overlimit classes to lower their priorities to yield bandwidth for other classes.

3) Channel-sharing guideline for the hierarchical structure: The hierarchical channel-sharing structure is shown in Fig. 8. Assume that each leaf class shares enough traffic flow. First, we ignore interior classes, i.e., assume the hierarchical structure is just one level. The the guideline for the flat structure in the previous subsection can be applied. Sequentially, take interior classes into consideration. If every single leaf at a different level satisfies the expected bandwidth sharing, then the hierarchical channel-sharing structure will satisfy the expected requirement. If a specific leaf class does not produce enough traffic flow for its requested bandwidth, then the extra bandwidth will be allocated to other leaf classes. However, the interior classes maybe are unsatisfied. In order to enable all of interior classes to reach the satisfied state, we temporarily ignore the leaf classes. Now, the hierarchical channel-sharing structure may be simplified into the single level channel-sharing structure. Thus, the priority of each interior class can be adjusted according to the guideline for the flat structure.

Shown in Fig. 9 is an example. If a leaf class A1 ceases data transmission, then the extra 30% bandwidth will be shared by other three classes. Hence, each of these classes obtains extra 10% bandwidth and they will be in satisfied state. However, the interior level class A still exists backlogs in class A2

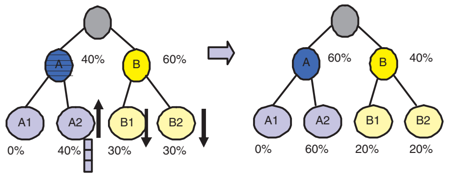
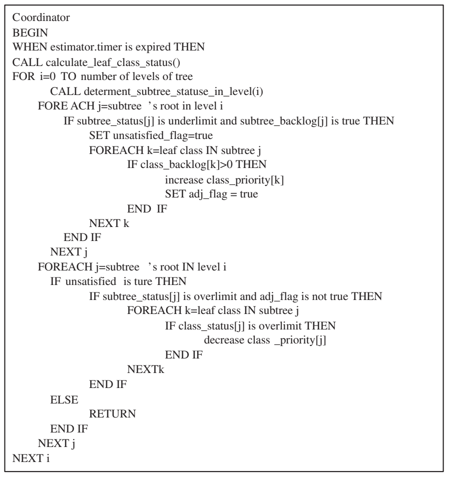


Fig. 9. An example of adjusting priority for hierarchical channel-sharing guideline.



```
Coordinator
BEGIN
WHEN estimator.timer is expired THEN
CALL calculate_leaf_class_status()
FOR i=0 TO number of levels of tree
      CALL determent_subtree_statuse_in_level(i)
    FORE ACH j=subtree  ’s root in level i
        IF subtree_status[j] is underlimit and subtree_backlog[j] is true THEN
            SET unsatisfied_flag=true
            FOREACH k=leaf class IN subtree j
                  IF class_backlog[k]>0 THEN
                        increase class_priority[k]
                        SET adj_flag = true
                  END  IF
            NEXT k
        END IF
      NEXT j
    FOREACH j=subtree  ’s root IN level i
      IF unsatisfied  is ture THEN
            IF subtree_status[j] is overlimit and adj_flag is not true THEN
                  FOREACH k=leaf class IN subtree j
                        IF class_status[j] is overlimit THEN
                              decrease class _priority[j]
                        END IF
                  NEXTk
            END IF
      ELSE
            RETURN
      END IF
    NEXT j
NEXT i
```

Fig. 10. The pseudo code of a hierarchical coordinator.

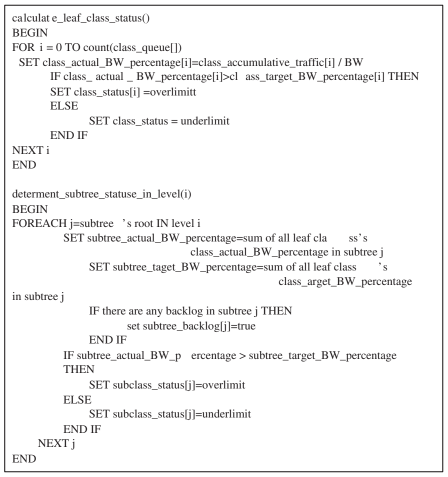
and the expected bandwidth is now reduced to 40% which is less than the expected value of 60%. This causes class A to be unsatisfied. According to the channel-sharing guideline mentioned above, one should upgrade the class priority of class A2 which belongs to class A, or degrade the class priority of class B1 and B2 which are in over-limit state in class B. After a complete cycle of priority adjustment, interior class A would eventually obtain extra 20% bandwidth. Fig. 10 and Fig. 11 are the pseudo codes for the procedure.

IV. Simulation and Numerical Experiments

In this section, we use ns-2 simulator to simulate the proposed mechanism. We set two CBQ channel-sharing structures, i.e., flat channel-sharing structure and hierarchical link-sharing structure to validate the wireless-CBQ efficiency.

A. Flat channel-sharing structure

We assume that a wireless LAN has bandwidth of 11 Mbps and there are three wireless terminals. The detailed configuration is shown in Fig. 12. At the beginning of the simulation, the Flow1 transmits data. Flow2 transmits data 10 seconds later. The transmitting rate is not limited as long as the MAC layer of this wireless terminal can transmit data. A packet of 1100 bytes is generated and transmitted to the target terminal. From Fig. 13, we find that each class is allocated fairly the expected bandwidth and all of these flows of the same class, such as Flow2 and Flow3 fairly share the bandwidth of class2 . There is no any effect on the bandwidth allocation of class1. Fig. 14 shows the average ratio of the



```
calculat e_leaf_class_status()
BEGIN
FOR i = 0 TO count(class_queue[])
  SET class_actual_BW_percentage[i]=class_accumulative_traffic[i] / BW
          IF class_ actual _ BW_percentage[i]>cl  ass_target_BW_percentage[i] THEN
          SET class_status[i] =overlimitt
          ELSE
                SET class_status = underlimit
          END IF
NEXT i
END

determent_subtree_statuse_in_level(i)
BEGIN
FOREACH j=subtree  ’s root IN level i
            SET subtree_actual_BW_percentage=sum of all leaf cla     ss’s
                                    class_actual_BW_percentage in subtree j
                SET subtree_taget_BW_percentage=sum of all leaf class     ’s
                                                class_arget_BW_percentage
in subtree j
                IF there are any backlog in subtree j THEN
                      set subtree_backlog[j]=true
                END IF
            IF subtree_actual_BW_p  ercentage > subtree_target_BW_percentage
            THEN
                SET subclass_status[j]=overlimit
            ELSE
                SET subclass_status[j]=underlimit
            END IF
      NEXT j
END
```

Fig. 11. The subroutine of a hierarchical coordinator.

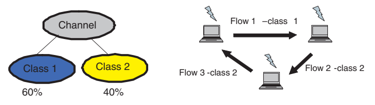


Fig. 12. The flat channel-sharing structure.

allocated bandwidth for each class in 1 second. We find that the usage of bandwidth follows the expectation of Fig. 12.

B. Hierarchical channel-sharing structure

In order to implement the hierarchical channel-sharing structure mechanism (Shown as Fig. 15.), we use four wireless terminals and each wireless terminal dynamically starts to transmit and stops transmitting the data flow. Fig. 16 illustrates the channel allocation in wireless LAN according to the hierarchical channel-sharing structure as expected. From Fig. 17, we find that regardless the leaf class is in starting or stoping state, interior classes can be always allocated the expected bandwidth. Besides, due to class B ceases transmission during simulation time 25-30 seconds, class A need not compete with other classes. Thus, this situation is the same as the 802.11.

V. Conclusions and Future Work

In this paper, we use different sizes of AIFS, CW_{min}, and CW_{max} to change the priority for obtaining the channel to control traffic flow. We guarantee enough bandwidth for each class to satisfy the service request for each class. If some class

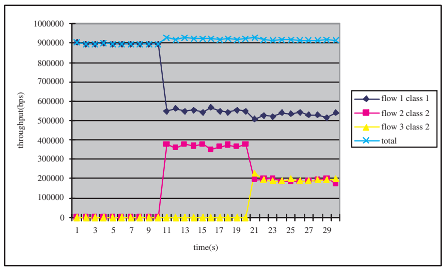


Fig. 13. The throughput of each flow.

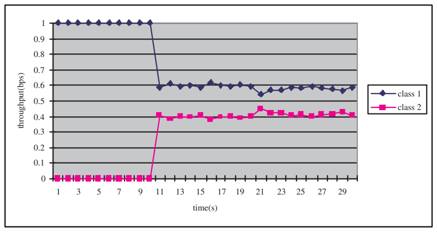


Fig. 14. The bandwidth in percentage of each class.

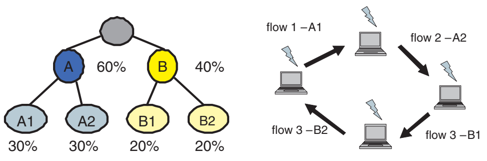


Fig. 15. The hierarchical channel-sharing structure.

over uses the bandwidth, our extended CBQ mechanism will adjust the class flow and discard some packets. Once the class is aware of the discarded packets, it will be noticed by the upper layer and then sends a request to degrade the rate of data transmission. From experimental results, we demonstrate that our mechanism reaches the goal of CBQ bandwidth allocation request in WLANs.

However, it can not reach the goal in multi-hop WLANs. This problem can be solved by dividing the transmission scope of the physical network into a number of overlaying single-hop WLANs and deserves to be studied in the future.

References

[1] I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11,", in *Proc. IEEE INFOCOM'01*, pp. 209–218, 2001.

[2] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," in *Proc. IEEE INFOCOM* '96, pp. 1133–1140, 1996.

[3] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE Trans. Networking*, vol. 3, pp. 365–386, 1995.

[4] C. Fragouli, V. Sivaraman, and M. Srivastava, "Controlled multimedia wireless link sharing via enhanced class-based queueing with handel-state-dependent packet scheduling," in *Proc. IEEE INFOCOM '98*, pp. 572–577, 1998.

[5] P. Garg, R. Doshi, R. Greene, M. Baker, M. Malek, and X. Cheng, "Using IEEE 802.11e MAC for QoS over wireless," in *Proc. IEEE IPCCC '03*, pp. 537–542, 2003.

[6] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Distributed multi-hop scheduling and medium access with delay and throughput constraints," in *Proc. ACM MOBICOM '01*, pp. 200–209, 2001.

[7] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Distributed priority scheduling and medium access in ad hoc networks," *Wireless Networks*, vol.8, no. 5, pp. 455–466, 2002.

[8] A. Lindgren, A. Almquist, and O. Schelen, "Quality of service schemes for IEEE 802.11," in *Proc. IWQoS '01*, pp. 6–8, 2001.

[9] A. Lindgren, A. Almquist, and O. Schelen, "Evaluation of quality of service schemes for IEEE 802.11 wireless LANs," in *Proc. IEEE LCN '01*, pp. 15–16, 2001.

[10] L. Romdhani, Q. Ni, and T. Turletti, "AEDCF: enhanced service differentiation for IEEE 802.11 wireless ad hoc networks," INRIA Research Report No. 4544, 2002.

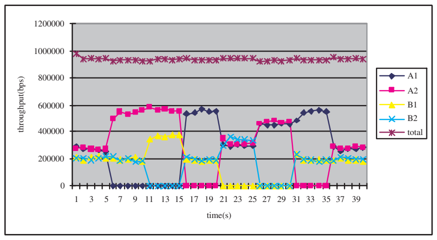


Fig. 16. The throughput of each leaf class.

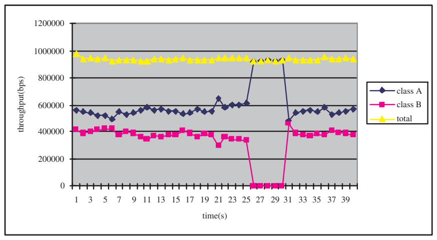


Fig. 17. The throughput of each interior class.