Chapter 2

Enhancement of PCF scheduling

2.1 Background and literature work

As wireless technologies have prevalently deployed, portable devices have become ubiquitous. Meanwhile, multimedia applications, such as voice over IP (VoIP), video on demand (VOD), are blooming. As widely known, wireless environment does not have as high bandwidth as wired network. Besides, radio conditions fluctuate with user mobility and location. The issue of deriving the same quality of service (QoS) in wireless environment has drawn lots of research efforts.

IEEE 802.11 standard is popularly accepted. It provides two mechanisms for data transmission: *Distributed Coordination Function (DCF)* and *Point Coordination Function (PCF)*. Following is the amendment, *IEEE 802.11e*, to enhance the QoS provision in 802.11. Similar to 802.11, 802.11e provides two medium access protocols: *Enhanced Distributed Channel Access (EDCA)* and *HCF Controlled Channel Access (HCCA)*. The former one is for decentralized access, while the latter needs a *Hybrid Coordinator (HC)* as the main controller. $EDCA$ slightly supplements DCF for delay-bounded traffics [3]. However, $EDCA$ could only provide statistical differentiation instead of fully guaranteeing the requirement of privileged streams. Besides, *EDCA*'s performance will degrade dramatically when the load

is heavy [4]. The reason is that $EDCA$ is a distributed scheme, every user contend for the medium independently; when the traffic is heavy, lots of the resource is wasted on contention instead of real data transmission. Compared with $EDCA$, polling-based scheme could not only increase overall throughput performance but also meet the demands of real-time traffics [3] [4] [5].

How to determine the polling sequence in a centralized method is nothing trivial. Scheduling is especially a big challenge since the coordinator (access point) has limited knowledge about every individual user. Scheduling has attracted a great deal of work in both wired and wireless network. *Generalized Processor Sharing (GPS)* [6] provides absolute fairness among all flows. Nonetheless, its assumption is that the server could process all backlogged queues at the same time, and the server will serve each queue in an infinitesimal amount every time. Although this scheme could provide absolute fairness among all queues, it could not be realized in real life because a packet is indivisible. *Weighted Fair Queuing (WFQ)* [7] approximates the GPS policy and is proved to achieve not only the long-term fairness, but also the bounded delay. Unlike GPS, though, WFQ is feasible and practical, and it is implemented in many wired routers and gateways for packet dispatching. The deficiency of WFQ is high computational complexity resulting from its complex equations. *Start-time Fair Queuing (SFQ)* [8] reduces the computational complexity of WFQ with minor modification.

The prerequisite of wired fair queueing is an error-free environment, in which either all the stations can transmit or none of them can all the time. This hypothesis is undoubtedly true in wired network but often impractical in wireless environment. While wired scheduling is mature, the debates over wireless scheduling have never ceased due to the instable wireless link. Some wireless users may not receive full bandwidth or even could not transmit due to error-prone channels. Various wireless fair scheduling algorithms have been developed. *Idealized Wireless Fair Queuing (IWFQ)* [9] claims that the most suffering stream has the highest priority when its radio recovers, which guarantees lagging flows would catch up other leading ones immediately after its channel becomes clear again. *Channel-condition*

Independent Fair Queuing (CIF-Q) [10] swaps slots between leading streams and lagging ones, while *Server Based Fairness Approach (SBFA)* [11] allocates a part of resource for suffering users. The literature work mentioned above only takes bandwidth fairness into consideration. However, the compensation for lost bandwidth is meaningless for delay-sensitive traffics. Besides, these well-known algorithms only apply to downlink (access point to wireless stations) scheduling. They could not operate in uplink (wireless stations to acess point) scheduling.

Some researchers proposed completely different polling mechanisms, such as *super poll* [12] and $multipoll$ [13]. Although these schemes could benefit in overall throughput, they need significant modification to *IEEE 802.11*. In the absence of an explicit scheduling method in IEEE 802.11, an effective scheduling algorithm plays a decisive role. [14] and [15] enhanced *Deficit Round Robin (DRR)* [16] for polling list maintenance. However, DRR could lead to long traffic latency, which is vital for real-time traffics. [17] and [18] classify different traffics during association process. The prioritization would influence the polling sequence. Estimation of queue length was devised in [19]. However, it could not prohibit ill-behaved users from debilitating others' performance. [20] only focused on voice traffic by exploiting talkspurt-silence characteristic for scheduling.

The wired scheduling mechanisms could not apply directly to the wireless environment because of the instability of the wireless radio. Although there is numerous literature work on wireless scheduling, most of them dealt with downlink problems, and seldom discussed about the uplink transmission. The uplink scheduling in a centralized environment has more challenges than a downlink one since the central coordinator has little information about current states of every single wireless station. The proposed method is for uplink traffic (mobile node to access point) scheduling in a centralized situation. Because AP has limited information about each individual station, the polling sequence is hard to determine. The objective of the proposed method is to recommand a fair scheduling algorithm, which is compatible with *IEEE 802.11*. No drastic modification is needed for the scheduling method. The proposed algorithm is flexible in view of various multimedia applications nowadays.

2.2 The proposed solution

802.11 PCF scheme has been introduced in the previous chapter. It is polling-based. In PCF, the central controller is called PC, which is usually embeded in the AP. PC will query every station in turn to check whether a wireless station has frames to send. A station can not transmit unless it is polled by PC. The polled station will transmit a frame after receiving the poll from PC. If the polled station has nothing to transmit while receiving a poll, it will send a *Null Acknowledgement* to PC instead. *IEEE 802.11 PCF* does not define the scheduling algorithm in PC. However, how to determine the polling sequence is the most critical factor in QoS. *Round Robin (RR)* is the simplest scheduling method and is also easy to implement. RR is fair only if all the packet size is the same, which is not true for most of today's network applications. Therefore, RR could not guarantee the bandwidth fairness among all the users.

Observing the behavior in *802.11 PCF*, all wireless stations could be divided into three categories: *backlogged state*, *bad channel state*, and *unbacklogged state*. A wireless station is in *backlogged state* if it replies a normal data frame when receiving a poll from PC. On the contrary, if a station transmits a *Null Acknowledgement* in response of a poll, then it belongs to *unbacklogged state*. That means that it has no buffered frames in queue. Because of the instability of wireless radio or the user mobility, the ongoing transmission may suffer from poor channel condition. A station may not receive a poll because the poll frame is lost; or the replied messages, such as a data frame and null acknowledgement from mobile stations, or even the acknowledgement from the PC, fail to arrive in the middle of the process. After sending out a poll, the PC will query the next candidate station on the polling list if it does not receive a response from the polled station within the $PIFS$ interval. According to *802.11 PCF*, if the data frame from a wireless station is not in turn acknowledged, a station could not retransmit until it is polled by PC again, or it could retransmit during the *Contention Period (CP)*. So for an uplink traffic, a station is classified as the *bad channel state* if the PC does not accept any response within $PIFS$ interval after sending out a poll frame. According to

the interactions in PCF operations, the PC could divide all the wireless stations into three different states: *backlogged state*, *unbacklogged state*, and the *bad channel state*. The PC could use the result in the process to determine the polling sequence next time.

The proposed method is based on the well-known *Weighted Fair Queuing (WFQ)* algorithm. WFQ stems from *Fair Queueing* [7], which hypothesizes a service discipline in which the transmission is in a *bit-by-bit* style. It also defines $R(t)$ as the number of rounds in the round-robin fashion during time t. $N_{ac}(t)$ denotes the number of *active flows* at time t, in which an active flow means a flow having buffered frames in queue. So

$$\frac{\partial R}{\partial t} = \frac{u}{N_{ac}(t)} \text{ , where } u \text{ is the outgoing linespeed of the gateway.} \tag{2.1}$$

The equation above indicates the relation between round during t, $R(t)$, and the service speed, u. The server will fairly dispatch its available resource (outgoing bandwidth) to current active flows. For convience, the bandwidth u is assumed as 1; that is, the server could only handle 1 bit in a unit time. We define *Head Of Line (HOL)* as the first packet in the buffered queue. If a packet with packet length P becomes HOL packet at time t_0, it will finish after P rounds since each round the server only serves one bit. Then we could get

$$R(t) = R(t_0) + P \text{ , } t \text{ is the packet finish time.} \tag{2.2}$$

Let t_i^α denote the arrival time of the i_{th} packet of flow α, S_i^α represent the number of rounds when the packet starts being served, and F_i^α be the number of rounds when the packet finishes. Then the following relation is formed:

$$\begin{cases} F_i^\alpha = S_i^\alpha + P \\ S_i^\alpha = \max\{F_{i-1}^\alpha,\ R(t_i^\alpha)\} \end{cases} \tag{2.3}$$

The first formula in the upper equation just replaces $R(t_0)$ and $R(t)$ in Eq. 2.2 with S_i^α and F_i^α respectively. The rounds the server travels when the previous packet departures, F_{i-1}^α, may be the rounds of the current packet when becoming HOL. If there is no packet

buffered in queue when packet i arrives in queue α, then its S_i^α is the number of rounds now. So the maximum value of F_{i-1}^α and $R(t_i^\alpha)$ determines S_i^α.

The *bit-by-bit* fashion is not practical because all transmission is *packet-by-packet* in real world. In order to become feasible, the author emulated the rules in a *packet-by-packet* view. From Eq. 2.3, we can get that an HOL packet will leave first among other active queues if its F_i^α is the smallest. Besides, the parameters S_i^α and F_i^α could be determined as soon as the packet arrives in queue. So in [7], the author proposed that an HOL packet whose F_i^α is the smallest will be served first.As the packet finished, the next one who will be picked up is that of the smallest F_i^α value. Although the F_i^α of a newly arriving packet may be smaller than the currently served one. But a preemptive scheme does not match the current transmission situation.

The discrepancy bound of the non-preemptive *packet-by-packet* emulated fair queueing algorithm has been proved to be within P_{MAX} bits, where P_{MAX} is the maximum packet size in bit among all the packets. That means, the total bit sent by any of the two queues at any given time will not differ too much, since there exists a bound, P_{MAX}.

The *Fair Queueing* algorithm is futher extended as *Weight Fair Queueing (WFQ)*, which simply assigns each flow a "weight". During each round, the server should give different quantities of service according to the assigned weight of each queue. The formula is thus rewritten as follows:

$$\begin{cases} S(p_f^j) = \max\{v(A(p_f^j)),\ F(p_f^{j-1})\} \\ F(p_f^j) = S(p_f^j) + \frac{l_f^j}{\phi_f} \end{cases} \text{, where } \frac{dv}{dt} = \frac{C}{\sum_{j \in \mathbf{B(t)}} \phi_j} \tag{2.4}$$

In the Eq. 2.4, p_f^j means the j_{th} packet of flow f. $S(p_f^j)$ is called the *start tag* of p_f^j, while $F(p_f^j)$ is the *finish tag*. $A(p_f^j)$ is the arrival time of packet p_f^j, and the *virtual function*, $v(t)$ equals the $R(t)$ function mentioned in Eq. 2.3. l_f^j is the packet length of p_f^j. The weight of flow f is a constant value, ϕ_f. The equation between the *virtual function*, $v(t)$, and the

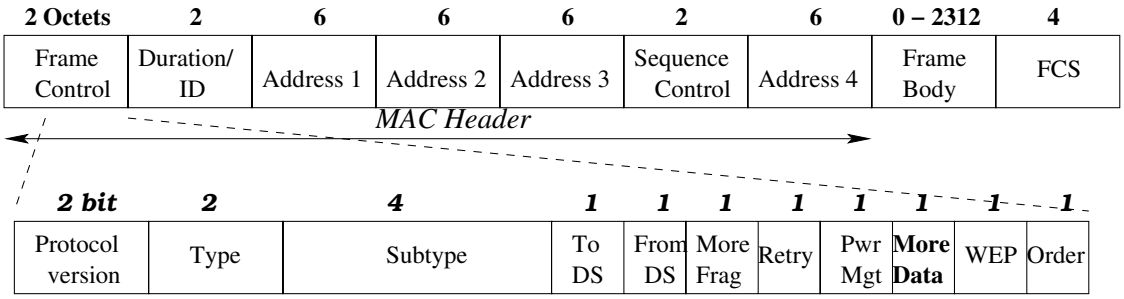

Figure 2.1: The "More Data" field in MAC frame control field

server capacity, C, is the server will provide $\sum_{j \in \mathbf{B(t)}} \phi_j$ bit of service during each round. $B(t)$ represents the set of flows in "active state" (having buffered frames in queue) at time t.

WFQ provides a fair and practical scheduling algorithm, and is commercialized in network products, such as routers. Despite the success of WFQ, there are still some deficiencies in it. First, it is only suitable for downlink (from AP to wireless stations) traffic scheduling, but could not apply in uplink traffic dispatching. The reason could be identified by Eq. 2.4. For the uplink traffic scheduling, an AP has no idea about the exact packet arrival time, $A(p_f^j)$, of each single wireless station. The WFQ algorithm could not operate without the parameter. In addition, an AP could not know the actual number of active queues currently. Although the "More Data" field in the *Frame Control* field in *802.11 MAC frame format* could be used to notify the PC that there is still (or no) remaining frames buffered in queue, as shown in Fig. 2.1, it is possible that frames arrive later on after the message has been sent as no further data. Moreover, the WFQ does not consider the wireless characteristic – the radio situation will fluctuate with surroundings or the user mobility. Even though PC has assigned the transmission opportunity to a user, it may not be able to accomplish a sucessful delivery because of the poor channel condition. The packet loss rate and bit error rate are much higher in wireless environment compared with the wired network. That is why the QoS is more a challenge in wireless world than the wired environment, in which the medium condition could be assumed always in good quality, and the packet error rate could be neglected.

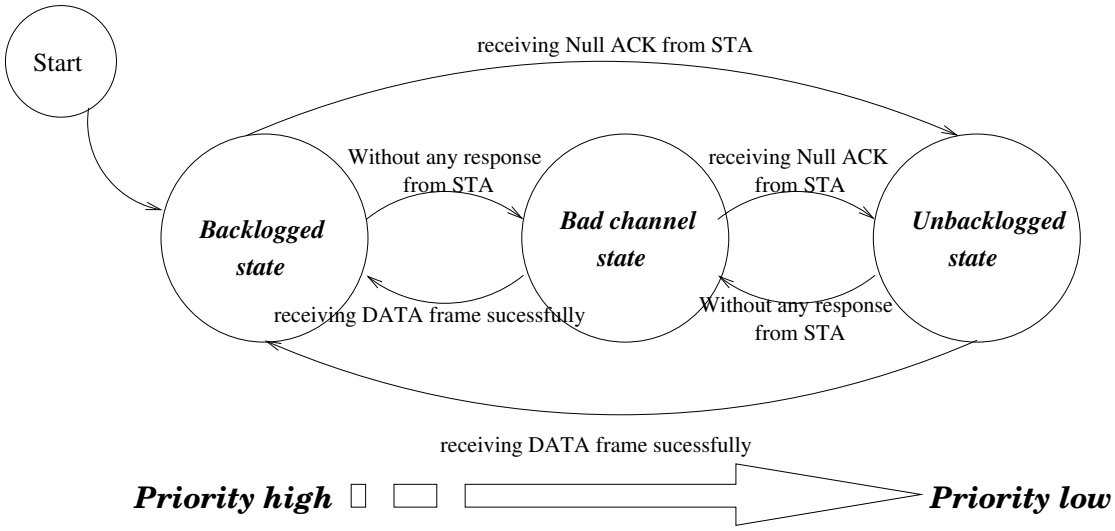

Figure 2.2: The state transition diagram of the three states: *backlogged*, *bad channel*, and *unbacklogged* states

In order to make up the insufficient parts in WFQ, we proposed the *Modified Weighted Fair Queueing (MWFQ)*. The weight of the flow f, ϕ_f in Eq. 2.4, is no longer a constant. It will vary according to the current state of a mobile user. We have mentioned earlier, according to the interaction between the PC and wireless stations in the polling process, we could divide all the stations into three categories: *backlogged state*, *bad channel state*, and *unbacklogged state*. In the three states, we argue that the *backlogged state* has the highest priority, the second one is the *bad channel state*, and the *unbacklogged state* is the most inferior one. When being polled by PC, a station who can reply a normal data frame should have the highest priority. The priority of a station in the *bad channel state* should be higher than one in the *unbacklogged state* because the radio condition could not be chosen or controlled. A station without packets to send should be polled again after a longer time, so the *unbacklogged state* occupies the lowest status. The state transition diagram of the three states is illustrated in Fig. 2.2.

The actual packet arrival time, $A(p_f^j)$, in Eq. 2.4 is unknown to the PC. In the proposed $MWFQ$, we use the mean packet rate as the value. When a wireless station enters a

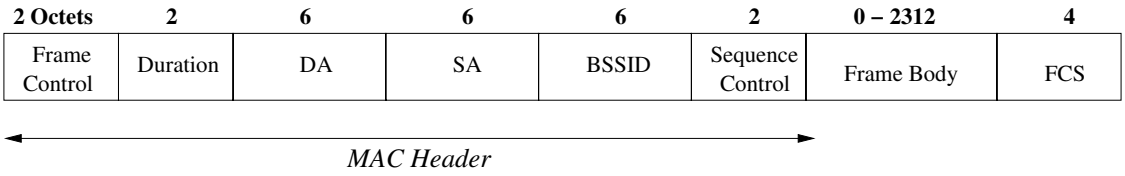

Figure 2.3: The format of 802.11 Association Request message

new radio coverage, it needs to do *authentication* and *association* procedures in the 802.11 standard. The *association request* message is as Fig. 2.3 shows. The format is actually the *Management frame*, and the practical content of *Association Request* is in the *Frame Body*. The *Association Request* information contains four parts in original design: *Capability Information*, *Listen Interval*, *SSID*, and *Supported rates*. Additional fields, called *Average Packet Data Rate* and *average packet size*, are needed in our method. The *Average Packet Data Rate* is used for notifying the average application data rate of the wireless station, while the *average packet size* is used for traffic packet size in byte. When a new flow is added, the station could send out the *Reassociation Request*, which is also a standard frame format in 802.11, to inform the PC. By the newly invented fields, *Average Packet Rate* and *average packet size*, the PC could gather all the traffic data rate and the packet size as well, and use them as the input parameters of each individual flow in the *WFQ* equations. To use the negotiated data rate as the one of the scheduling parameters could let traffics with higher data rate have more polled chances than those with lower rate. Although there might be some malicious or malfunctioned users exaggerating their data rate and trying to exhausting the whole network resources, the problems could be settled by the *Authentication* process or the billing mechanism, which is not covered in this thesis.

Besides the negotiation process and scheduling algorithm mentioned above, a new scheme is created in wireless stations in order to improve the radio utilization. The radio condition always fluctuates from time to time. The surroundings or the interference from other radio affect the quality of the wireless medium. In addition, the position of a wireless station will change over time, which makes the prediction of the current channel condition

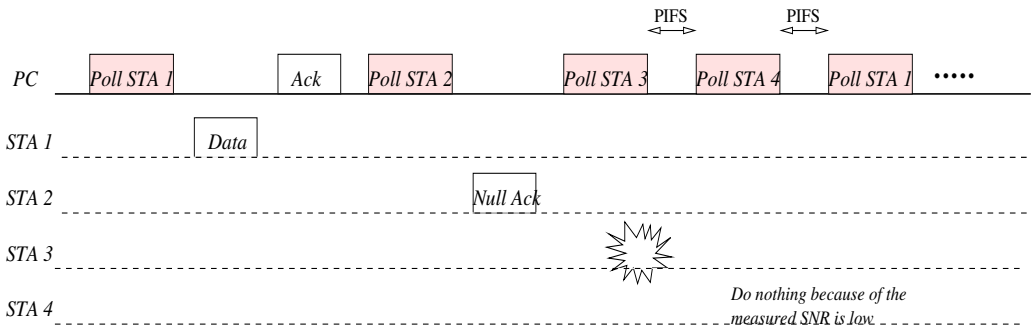

Figure 2.4: The process of MWFQ

more difficult. If a station is polled when suffering from a poor channel, then the resource is wasted since the transmission fails.

Signal-to-Noise Ratio (SNR) is the amplitude of the signal compared to the noise. SNR could be used as a basis for the current radio condition. The larger the SNR, the lower the error rate. The SNR value could be measured and calculated. *Bit Error Rate (BER)* is the number of erroneous bits divided by the total number of bits transmitted. Neglecting the error correction or detection, a relation between SNR and BER exists, which depends on the adopted modulation scheme. Higher SNR would lead to lower BER. The BER in turn determines whether the transmission will fail or not. Taking the advantage of the SNR and BER values, a station could identify its current radio condition. When receiving the poll from PC, a station suffering from a poor channel will relinquish its privilege to the next station on the polling list. Without hearing anything from the polled station, the timer in PC will expire after $PIFS$ interval. Then the PC will set the state of the polled station as *bad channel state*, and continue to query the next candidate on its polling list. Although a statoin with poor channel does not necessarily fail in transmission, it has "higher" risk in losing the packet. The mechanism could improve the medium utilization since risky stations would yield their transmission temporarily to the normal ones, and less bandwidth is wasted on failed transmission.

There are two important components in the $MWFQ$ mechanism,

1. *MWFQ scheduling module*: the module is embedded in the PC. It is an emulator, which runs the *bit-by-bit WFQ* algorithm. During the association process, the module could gather all the data rates of current flows, and these parameters are used as the packet inter-arrival time in the emulator. The next polled candidate is the one who is going to finish in the emulation. After polling a station, the module needs to modify the weight of the polled user according to the various states of wireless stations.

2. *Radio detection module*: the module is in a wireless station. It will decide whether to transmit or not regarding the current SNR statistics measured by the physical layer.

The overall operations of the proposed method, $MWFQ$, are illustrated in Fig. 2.4. Assuming there are four wireless stations (*STA 1, STA2, STA3, and STA4*) attaching to the PC, the states of all wireless stations are initialized as the *backlogged state* at the very beginning. For simplicity, all data rates of the four stations are supposed to be equal. The *MWFQ scheduling module* determines *STA 1* as the first candidate in the list. Accepting the poll, *STA 1* returns a normal data frame to PC, then the PC replies it with an *Acknowledgement*. The *scheduling module* does not need to modify the *STA 1* state since it still remains in the *backlogged state*. The next polled candidate is *STA 2*. Because of no buffered frames, *STA 2* answers a *Null Acknowledgement* in reply of the poll. The *MWFQ scheduling module* will then modify the *STA 2* state from *backlogged state* to *unbacklogged*, whose priority is the lowest. The third one is *STA 3*. The poll from PC to *STA 3* is lost due to the instable radio environment. The timer in the PC will expire after $PIFS$ interval, and it will continue to poll the next one. The succeeding station is *STA 4*. Despite receiving the poll correctly, the measured SNR is too low to transmit. The *Radio detection module* decides to be silent. The same as the last case, the PC will query the next candidate after $PIFS$. The state of *STA 4* will be changed as *bad channel*.

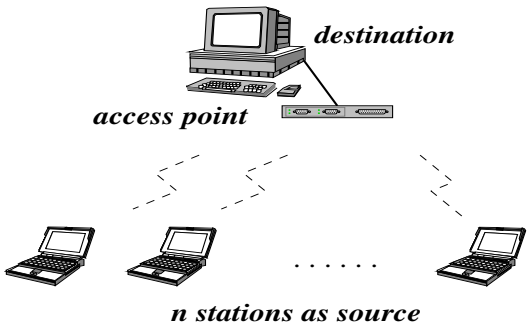

Figure 2.5: MWFQ simulation topology

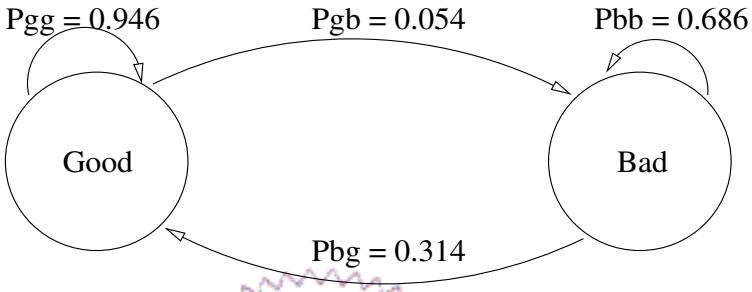

Figure 2.6: The error model

2.3 Simulation and Numerical Results

The simulation topology is shown in Fig. 2.5 . There is only one destination connecting to the AP via wired link. The destination acts as the sink receiving all the traffic flows sent by the wireless stations attaching to the AP. The wireless medium is *802.11b DSSS* with

Table 2.1: MWFQ simulation parameters

Parameter	Value
PHY	DSSS
bandwidth	2Mbps
slot time	20us
SIFS	10us
PIFS	30us

bandwidth of $2Mbps$. The related parameters are listed in Table. 2.1. In order to emulate the fluctuant radio link in the wireless environment, the two-state error model is adopted in the simulation [21]. As the Fig. 2.6 illusrates, the wireless link may be in the "good" or "bad" states. The transition probabilities, P_{gb}, P_{bg}, P_{gg}, and P_{bb} mark the probabilities that the radio link will transit from one state to the other. The association process is neglected, and every single experiment took 300 seconds of simulation time. Fig. 2.7 and Fig. 2.8 are the numerical results.

In Fig. 2.7, the traffic type is *constant bit rate (CBR)*. The data rate is $64kbps$ and $128kbps$ with the same packet size $400bytes$. The x-axis represents the number of mobile nodes between 2 to 10, while the y-axis stands for the packet queue delay in second. The packet queue delay means the total duration when the frame gets into the *MAC* layer until it is transmitted. The queue delay does not include the packet transmission time. Both curves of average queue delay with $64kbps$ are lower than those with $128kbps$. The average delay of $128kbps$ even surpasses $50ms$ when the number of mobile nodes becomes 9 no matter in both the proposed method and RR. Comparing the two delay curves of $64kbps$ with *Round Robin* and *Modified Weighted Fair Queueing*, the proposed method, $MWFQ$, could reduce the average delay by 36% to 40%. When the traffic load becomes twice as $128kbps$, the improvement is more obvious. The proposed method could lower the delay by 36% to 91%. In addition, when the load becomes intense or the number of mobile nodes gets larger, our proposed method, $MWFQ$, has more outstanding performance than RR.

The delay improvement is because a wireless station with a low SNR will give up its transmission opportunity in our proposed scheme. A station with poor channel has higher chances in failed transmission. Although the next candidate will not be polled until $PIFS$, the proposed method could still save the bandwidth compared with the original one. The $PIFS$ duration in the simulation settings is $30us$, in which a packet with length 60 bits could be transmitted. However, a station suffering from poor channel may occupy the medium much longer than $PIFS$ in the original scheme. Furthermore, the bandwidth is wasted since the station could not proceed successful packet delievery. According to the simulation

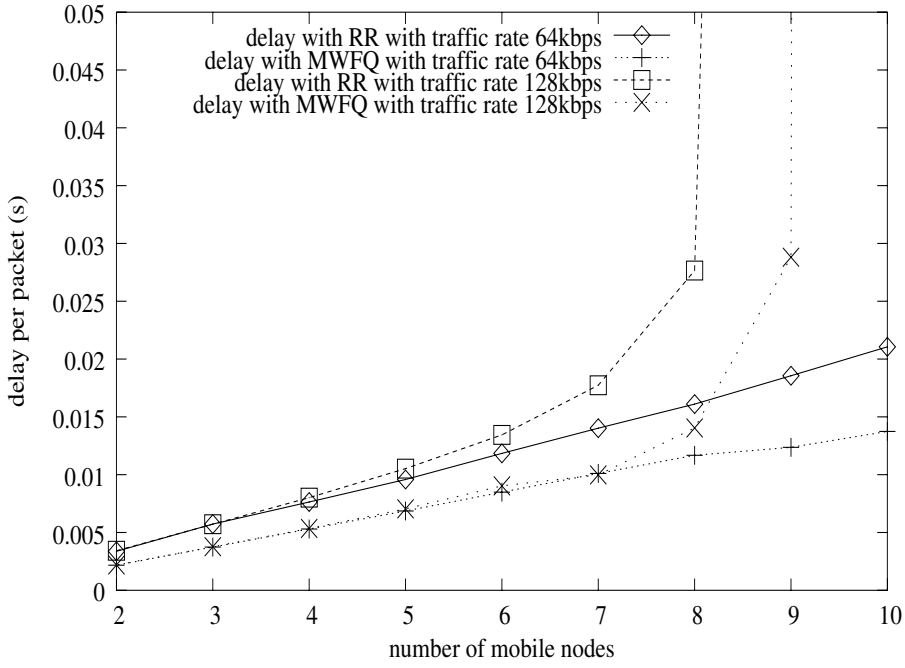

Figure 2.7: The delay results of *RR* and *MWFQ* with same packet size 400bytes

results, the modified scheme could not only benifit the average packet delay, but also the overall network performance. Compared with RR, the proposed algorithm could dispatch the bandwidth impartially even when the packet size is not the same.

2.4 Summary

PCF is a centralized method for medium dispatch in IEEE 802.11. The polling sequence determines the QoS each user perceives. However, the standard did not define the scheduling algorithm explicitly. Although a lot of scheduling methods have been proposed, none of them could be directly applied to the *PCF* scheduling. Most of the literature work focused on the downlink traffic (from the *AP* to the mobile user). For the downlink traffic, *AP* could easily gather all the scheduling information, such as queue length, packet arrival rate, or even the packet delay etc. On the contrary, *AP* is not able to acquire all the necessary information

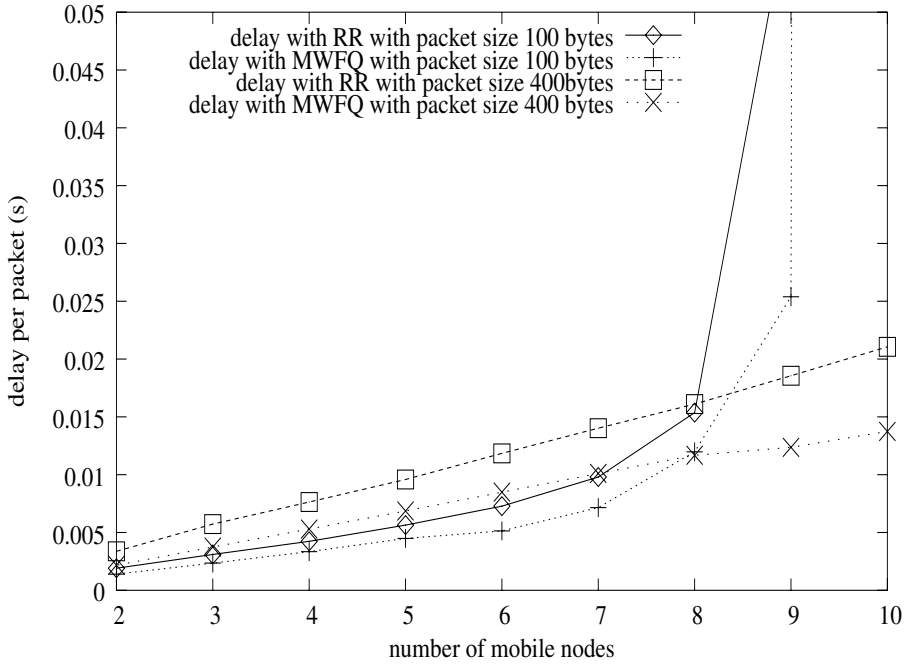

Figure 2.8: The delay results of *RR* and *MWFQ* with same traffic rate 64kbps

for the uplink traffic (from the mobile user to the *AP*). The scheduling for the uplink traffic is more a challenge than that for the downlink one. This work aims to devise a solution to enhance the performance of the *PCF* scheduling without huge modification of the IEEE 802.11 standard.

WFQ is a famous scheduling algorithm. It could distribute the resource according to each individual's weight. Besides, a bounded delay exists if the packet size is limited. The proposed method is based on WFQ. When entering a new radio coverage, a station needs to report its traffic data rate and packet size to PC during the association process. A *MWFQ scheduling module* runs the *WFQ* algorithm inside the PC according to the negotiated information. On the basis of the response, all stations could be classified into three different states: *backlogged*, *bad channel*, and *unbacklogged* states. A station is in the *backlogged* state if it sent a normal data frame in response of PC's poll. Provided that there is no reply from the polled user, the PC will continue asking the next candidate and the current user

will be labeled as *bad channel* state. The PC may receive the *null acknowledgement* which indicates the polled station has no frames to transmit. This kind of users are categorized as the *unbacklogged* state. Unlike the WFQ, though, the flow weights in $MWFQ$ will dynamically change according to each user's current state. Users in the *backlogged* state have the highest weight, while those in the *unbacklogged* state get the lowest priority. Higher weight leads to shorter polling delay in the $MWFQ$ algorithm.

In addition to the scheduling module in the PC, each mobile user needs to execute a *Radio Detection Module*. If the evaluated *SNR* is too weak, the station will give up the transmission opportunity even when being polled. Without hearing any response from the polled station, the PC will proceed its polling list after $PIFS$, and the station will be classified into the *bad channel* state. The scheme could yield better medium utilization because less bandwidth is wasted on the failed transmission. The simulation results also show the proposed method could better improve the average packet delay compared with RR scheme. $MWFQ$ could outperform the RR scheme even more as the traffic load continues growing.