# An Adaptive Weighted Fair Scheduling Algorithm for Wireless Networks

**Kuochen Wang, Cheng-Kai Yin, and Ae-Yun Liu**

Department of Computer and Information Science

National Chiao Tung University

1001 Ta Hsueh Rd., Hsinchu, Taiwan 300, R.O.C.

Phone: 886-3-5715900 FAX: 886-3-5721490

Email: kwang@cis.nctu.edu.tw

Corresponding Author : Kuochen Wang

**Abstract-** In this paper, we present an adaptive weighted fair queueing algorithm (AW-FQ) which can provide better QoS for error-prone flows and does not need to predict the channel quality of packet transmission in the wireless environment. Every flow is guaranteed to receive the minimal tolerable quality of its original service when experiencing channel errors. AW-FQ is suitable for applications that need different minimal QoS requirements. In AW-FQ, each flow is individually assigned a virtual flow which does not always occupy a fixed fraction of the bandwidth. The virtual flow can provide additional service for error-free flows and compensate the flow which it is assigned to. The amount of service degradation of a flow can be administratively controlled within a predefined tolerable ratio when the flow provides compensation. Simulation results show that our approach can guarantee throughput and fairness. Compared to the server based fairness approach (SBFA), AW-FQ has better queueing delay bound and does not have to assign a fixed flow which always occupies a fixed fraction of the bandwidth for providing compensation.

**Index Terms:** fair scheduling, quality of service, compensation, wireless network.

---

This work was supported by the National Science Council, Taiwan, ROC, under Grant NSC90-2213-E-009-162 and NSC 39095F.

# 1 Introduction

With the rapid growth of wireless networks in recent years, the demand for wireless applications is increasing. In the near future, multimedia services will be supported in 3G wireless systems. However, the wireless environment has high interference, high error rates, and limited bandwidth. Providing quality of service (QoS) for applications over the scarce and shared wireless medium is a great challenge. This is because that the shared radio channels have two key characteristics: (1) bursty channel errors and (2) location-dependent channel capacity and errors [1]. Some scheduling algorithms for wireless networks were designed to achieve the objective of maximizing the channel throughput by flows in bad channel state relinquishing their bandwidth to those in good channel state [1][2]. However, it would be improper that a flow receives no service while experiencing channel errors regardless of the type of applications. In this paper, we propose a wireless adaptive fair scheduling algorithm that has minimal bandwidth and throughput guarantees. Our scheduling algorithm can provide better QoS when flows experience channel errors.

# 2 Existing Methods

Channel State Independent Wireless Fair Queueing (CS-WFQ) [3][4] adopted Start-Time Fair Queueing (STFQ) [5] to support differentiated QoS requirements in the wireless environment. CS-WFQ is a solution that trades off a given level of system throughput to provide better fairness and QoS guarantees for bad channel flows [3]. However, in CS-WFQ, error-free flows are affected by flows which are in channel errors since the bandwidth share of error-free flows decreases. In addition, CS-WFQ did not provide compensation for these affected flows. Server Based Fairness Approach (SBFA) [6] provided a general framework for adapting existing wireline fair queueing algorithms to be used in wireless networks, though the fairness properties of these algorithms may not be satisfied in the wireless environment [7]. SBFA assumes that the base station which performs scheduling has the knowledge of whether or not to defer packet transmission of flows. If one flow has poor channel quality, the base station will defer the flow's packet transmission. To compensate lost service of lagging flows, SBFA reserves one or more special flows called *Long Term Fairness Servers* (LTFSs). SBFA satisfies the long term fairness and throughput bound

among all error-free flows. However, it did not provide short term fairness or throughput bound, and it provided very coarse worst case delay bound due to the HOL blocking of compensation [7]. CIF-Q (*Channel-condition Independent Fair Queueing*) considered location-dependent channel errors [2]. Although CIF-Q satisfied fair scheduling properties, it did not provide any service for flows that experience channel errors.

# 3 Design Approach

## 3.1 Overview of the Proposed Approach

Our approach (AW-FQ) does not need to predict the channel condition of the mobile host. If one flow perceives channel errors, the transmission rate of the flow is decreased to prevent further bandwidth being wasted. This flow can still receive a fraction of its original service to meet its minimal QoS requirement. For example, video transmission may still have sustainable audio quality when losing some bandwidth. When the flow becomes error-free, it will receive compensation to get back its *lost service* that was taken by the scheduler. We classify flows into *guaranteed flows* and *best-effort flows*. Each flow in AW-FQ is assigned a *virtual flow* for compensation. If a flow $i$ experiences channel errors, the virtual flow $VF_i$ is activated and begins to serve error-free best-effort flows. Once flow $i$ becomes error-free, $VF_i$ will turn to serve flow $i$ and begins to get the lost service back from other flows which $VF_i$ has served previously. Once the compensation for flow $i$ is complete, $VF_i$ will be deactivated. The virtual flow does not permanently occupy the weight which can be allocated to other flows.

## 3.2 Algorithm Description

AW-FQ is based on Start Time Fair Queueing (STFQ) [5], which was designed for wireline networks. The scheduling policy is to select a flow with a Head-Of-Line (HOL) packet which has the minimal virtual starting time to transmit. Each flow $i$ is assigned an original weight $W_i$, a current weight $W_i'$, and a *virtual flow* $VF_i$, respectively, as shown in Table 1. The current weight is equal to the original weight initially. The virtual flow is an abstract flow which does not actually store any packet. The scheduler treats the virtual flow the same as other backlogged flows. If a virtual flow is selected by the scheduler, it can serve

Table 1: Definition of flow parameters.

| Flow type | Parameters |
|---|---|
| Normal flow $i$ | Virtual starting time: $S_i$ |
| | Virtual finishing time: $F_i$ |
| | Original weight: $W_i$ |
| | Current weight: $W_i'$ |
| | Minimal ratio of original service: $\alpha_i$ |
| | Leading count: $lead_i$ |
| | Additional service count: $C_i$ |
| Virtual flow $i$ | Virtual starting time: $S_{VF_i}$ |
| | Virtual finishing time: $F_{VF_i}$ |
| | Current weight: $W_{VF_i}$ |
| | Compensation count from flow $j$: $CC_i^j$ |
| | Weight from flow $j$: $W_{VF_i}^j$ |

other flows by transmitting the HOL packet of the flow but charge the virtual flow itself. Initially, the weight $W_{VF_i}$ of $VF_i$ is zero and its virtual starting time $S_{VF_i}$ is set to infinity. Thus the virtual flow will not be selected by the scheduler at the beginning.

Figure 1 shows the overall operation procedure of AW-FQ. Initially, when flow $i$ with the smallest virtual starting time $S_i$ is selected by the scheduler and flow $i$ is not a virtual flow, its HOL packet is transmitted. If the transmission of the packet succeeds, the scheduler will continue to choose next flow to transmit. If the transmission fails, it means that flow $i$ has experienced channel errors. The *virtual flow activation procedure* is then activated. If flow $i$ does not receive any service from other flows or activate its virtual flow, the current weight $W_i'$ of flow $i$ is reduced according to the following equation:

$$W_i' = max\{1 - E_i,\ \alpha_i\} \cdot W_i \tag{1}$$

where $E_i$ is the packet error rate that flow $i$ has experienced and $\alpha_i$ is a predefined parameter. The parameter $\alpha_i$ $(0 < \alpha_i < 1)$ is used to control the minimal fraction of flow $i$'s weight and can be administratively set to provide the minimal tolerable bandwidth for flow $i$. That is, although we do not know the channel quality of the flow's next transmission, we just lower its weight accordingly for the next packet transmission to prevent further bandwidth being wasted.

After flow $i$ decreases its weight, it activates its virtual flow $VF_i$. The virtual flow $VF_i$ gains the weight which flow $i$ loses:

$$W_{VF_i} = W_i - W_i' \tag{2}$$

And the virtual starting time $S_{VF_i}$ of $VF_i$ is set as follows:

$$S_{VF_i} = max\{v(t),\ F_{VF_i}\} \tag{3}$$

where $v(t)$ is the system virtual time and $F_{VF_i}$ which is equal to zero initially, is the virtual finishing time of the virtual flow $VF_i$. Now the virtual flow $VF_i$ is activated and acts like other backlogged flows. $VF_i$ begins to share bandwidth with other flows. If flow $i$ has already activated its virtual flow $VF_i$ when experiencing channel errors, the weight of flow $i$ and $VF_i$ are updated accordingly.

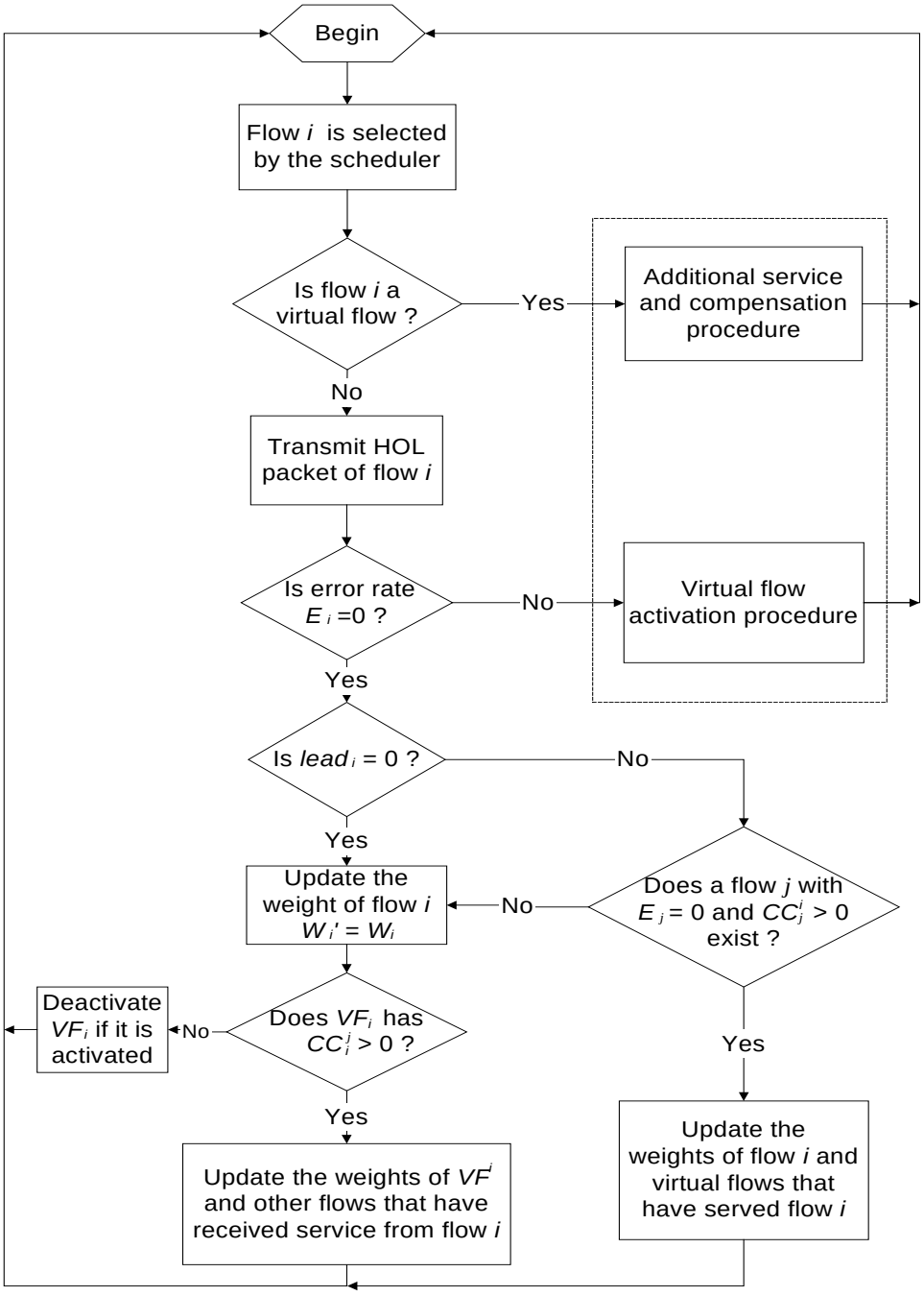



Figure 1: The overall operation procedure of AW-FQ.

When $VF_i$ is chosen by the scheduler, the *additional service and compensation procedure* is activated. It first examines if flow $i$ experiences channel errors. If the error rate $E_i$ of flow $i$ is not zero, it means that flow $i$ is still experiencing channel errors. Therefore, $VF_i$ will not serve flow $i$ to prevent unnecessary bandwidth being wasted. Instead, $VF_i$ serves other error-free best-effort flows by transmitting the HOL packet of the flow. We say that these error-free flows have received *additional service*. When flow $j$ receives additional service, it keeps a *leading count*, $lead_j$, to record how much additional service it has received. An active flow $j$ with $lead_j > 0$ is defined to be a *leading flow* and will give up its lead when the compensation begins. Besides, additional service is distributed among error-free best-effort flows according to *additional service count*, $C_j$, kept in every flow $j$ to achieve short term fairness. The values of $lead_j$ and $C_j$ are set to zero initially. If flow $j$ recoveries from channel errors, the value of $C_j$ must be reset to the maximal additional service count of flows in the system. When a flow $j$ receives additional service from the virtual flow $VF_i$, its $lead_j$ and $C_j$ are computed as follows:

$$lead_j = lead_j + l_j \tag{4}$$

$$C_j = C_j + l_j/W_j \tag{5}$$

where $l_j$ is the HOL packet's length of flow $j$ and $W_j$ is the weight of flow $j$. The virtual flow selects an error-free best-effort flow with a minimal $C_i$ to serve. Therefore, the amount of additional service received is in proportion to the flow's weight. This can ensure that a flow with a smaller weight will not actually receive more service than a flow with a larger weight. In this way, the short term fairness property is achieved. Since $VF_i$ serves flow $j$ using its own bandwidth, the virtual starting time $S_{VF_i}$ and the $CC_i^j$ of $VF_i$ are updated as follows:

$$F_{VF_i} = S_{VF_i} + \frac{l_j}{W_{VF_i}}$$

$$S_{VF_i} = max\{v(t),\ F_{VF_i}\}$$

$$CC_i^j = CC_i^j + l_j \tag{6}$$

where $l_j$ is the HOL packet's length of flow $j$ and $CC_i^j$ is the *compensation count*. That is, the virtual flow $VF_i$ uses its own bandwidth to serve flow $j$ and keeps the $CC_i^j$ to

record how much additional service has provided for flow $j$. $CC_i^j$ is kept only in the virtual flow and is initialized to zero. After flow $j$ is served by the virtual flow, the virtual starting time of flow $j$ still remains unchanged. However, its virtual finishing time $F_j$ is are updated as follows:

$$F_j = S_j + \frac{l_j}{W_j}$$

### 3.3 Compensation Model

When the channel quality of flow $i$ recovers, the *compensation procedure* for flow $i$ is activated. If the error rate $E_i$ of flow $i$ becomes zero and flow $i$ does not receive any additional service ($lead_i = 0$), its current weight $W_i'$ is set to the original weight $W_i$. The virtual flow $VF_i$ of flow $i$ will no longer to serve other flows but flow $i$ to provide *compensation service*. If there exists a virtual flow $VF_i$ with $CC_i^j > 0$ and flow $i$ is error-free, the virtual flow $VF_i$ will gain weight $W_{VF_i}^j$ from every leading flow $j$ ($lead_j > 0$) that $VF_i$ has served previously:

$$W_{VF_i}^j = (1 - \alpha_j) W_j \cdot \frac{W_i}{\sum_{k \in \delta} W_k} \ , \quad for\ all\ j \in \Delta \tag{7}$$

where $\Delta$ is a set of flows that $VF_i$ has served previously, and $\delta$ represents a set of flows that their assigned virtual flows have provided *additional service* for flow $j$ and require compensation from flow $j$ now. $\delta$ can be obtained by checking every error-free flow $k$ which its corresponding virtual flow has $CC_k^j > 0$. If $\delta$ is empty, it means that all flows which belonged to $\delta$ are experiencing channel errors and do not need compensation at this time. In this condition, if flow $j$ does not have channel errors, the weight of every flow $j$ in $\Delta$ remains unchanged ($W_j' = W_j$). Otherwise, if $\delta$ is not empty, the weight of every flow $j$ in $\Delta$ is updated as follows:

$$W_j' = \alpha_j \cdot W_j \tag{8}$$

That is, we take a fraction of weight from those *leading flows* that have received additional service from the virtual flow $VF_i$ previously to compensate flow $i$. Although leading flows contribute a fraction of weight for compensation, they still has a minimal fraction of service that each flow can sustain and is also bounded by the $\alpha_j$ parameter. In this way,

it can prevent service starvation and achieve *graceful degradation* for leading flows. In addition, a leading flow $j$ may receive additional service from multiple virtual flows. If a leading flow $j$ compensates multiple flows at the same time, flow $j$ distributes its weight $(1 - \alpha_j)W_j$ to multiple virtual flows in proportion to the original weights of the flows which these virtual flows are assigned to. That is, each $VF_i$ gains the weight contributed from flow $j$ in a ratio of $\frac{W_i}{\sum_{k \in \delta} W_k}$. This can ensure that a flow with a lower rate will not actually receive more service than a flow with a larger rate. The weight $W_{VF_i}$ of the virtual flow $VF_i$ is then obtained as follows:

$$W_{VF_i} = \sum_{j \in \Delta} W^j_{VF_i} \tag{9}$$

If $W_{VF_i}$ is zero, the virtual starting time of virtual flow $VF_i$ is set to infinity to deactivate it. Otherwise, the virtual flow $VF_i$ provides compensation service for flow $i$ using its updated weight $W_{VF_i}$. When $VF_i$ serves flow $i$, $CC^j_i$ is updated as follows:

$$CC^j_i = max\{\ 0,\ CC^j_i - l_i \cdot \frac{W^j_{VF_i}}{\sum_{k \in \Delta} W^k_{VF_i}}\}\ ,\quad for\ all\ j \in \Delta \tag{10}$$

where $l_i$ is the HOL packet's length of flow $i$. The compensation count for each flow $j$ $(CC^j_i)$ is decreased in proportion to the contributed weight for $VF_i$ when transmitting a packet of flow $i$. If $CC^j_i$ is equal to zero, it means that the compensation service which flow $j$ provides for flow $i$ is complete and the $W^j_{VF_i}$ of the virtual flow $VF_i$ is set to zero. Flow $j$ stops compensating flow $i$ and gets its contributed weight $W^j_{VF_i}$ back from $VF_i$. The virtual finishing time of flow $j$ is then computed using its updated weight. In addition, the current weight $W_{VF_i}$ of $VF_i$ is also updated according to equation (9). For each flow $j$, if $CC^j_i = 0$, the value of $W_{VF_i}$ will be zero and the virtual starting time of $VF_i$ is set to infinity. That is, flow $i$ gets all its lost service back so the compensation service for flow $i$ finishes. The virtual flow $VF_i$ is deactivated and will not be selected by the scheduler because its task of compensating flow $i$ is complete.

Since flow $j$ has provided its partial weight to compensate flow $i$, $VF_i$ must update its $lead_j$ in the same way with $CC^j_i$:

$$lead_j = max\{\ 0,\ lead_j - l_i \cdot \frac{W^j_{VF_i}}{\sum_{k \in \Delta} W^k_{VF_i}}\}\ ,\quad for\ all\ j \in \Delta \tag{11}$$

If $lead_j$ is greater than zero ($lead_j > 0$), it means that flow $j$ does not give up all its lead yet and must provide a fraction of its weight for compensation. If there is a leading flow $j$ ($lead_j > 0$), flow $j$ can not activate its virtual flow $VF_j$ when flow $j$ experiences channel errors except that there is no error-free flow which requires flow $j$ to compensate. This is because that flow $j$ should give up its lead before providing additional service. If there exists any error-free flow which its assigned virtual flow has served flow $j$ previously, flow $j$ must provide compensation and thus set its weight according to equations (7) and (8) even when flow $j$ experiences channel errors. If flow $j$ has activated its virtual flow $VF_j$ to compensate flow $j$ itself, the weight $W^i_{VF_j}$ which receives from flow $i$ is added to every flow $i$. The virtual finishing time of every flow $i$ is then computed using its updated weight.

# 4 Simulation and Evaluation

We use *ns*-2 [8] on Linpus Linux to simulate our approach.

## 4.1 Scenario 1: The Behavior of AW-FQ

Scenario 1 lasts for 36 seconds and there are five flows in this scenario: one 140 Kbps constant bit rate (CBR) flow, and four 320 Kbps FTP flows. Flow 4 and flow 5 start at the $0.1^{th}$ second and others start at the $0.0^{th}$ second. All flows are continuously backlogged during the simulation. Only flow 2 and flow 3 experience channel errors during time interval (12, 16). The pattern of channel errors is the location-dependent channel errors which smoothly increase packet error rate ranging from 10% to 80%. Table 2 shows the parameters of the five flows. The video or audio application is delay sensitive and requires low queueing delay. Therefore, the deadline of flow 1 is much smaller than that of others.

As shown in Figure 2, flow 1 is not affected by other flows and always gains its desired bandwidth. When channel errors occur during time interval (12, 16), flow 3 has higher bandwidth than flow 2 since the $\alpha$ value of flow 3 is larger than that of flow 2. This ensures that a flow with higher $\alpha$ can receive better service in the presence of channel errors. Thus AW-FQ has minimal bandwidth guarantee. The additional service is distributed among

Table 2: Parameters of the five flows used in scenario 1.

| | $W$ | $\alpha$ | $Deadline$ (sec) |
|---|---|---|---|
| flow 1 | 0.1 | 0.8 | 0.15 |
| flow 2 | 0.225 | 0.4 | 10 |
| flow 3 | 0.225 | 0.7 | 10 |
| flow 4 | 0.225 | 0.4 | 10 |
| flow 5 | 0.225 | 0.6 | 10 |

error-free best-effort flows in proportion to their transmission rates. Thus flow 4 and flow 5 receive the same amount of additional service. At the end of channel errors, flow 2 and flow 3 begin to receive compensation service from flow 4 and flow 5.

As illustrated in Figure 3, we can see that the same amount of additional service is received between flow 4 and flow 5 during time interval (12, 16). Therefore, the short term fairness property is achieved. During the compensation period for flow 2 and flow 3, flow 4 and flow 5 also achieve graceful degradation. That is, flow 4 and flow 5 receive a fraction of original service at the rate of the $\alpha$ value. After the compensation is complete, flow 4 and flow 5 can still receive their expected amount of service. In addition, flow 2 receives more compensation service than flow 3 after channel errors occur due to a lower $\alpha$ value. We can say that in the presence of channel errors, a higher $\alpha$ value which maintains a larger fraction of the original service is suitable for guaranteed flows and a lower $\alpha$ value which receives more compensation service is suitable for best-effort flows. Besides, AW-FQ has guaranteed throughput for error-free flows.

### 4.2 Scenario 2: In Comparison with SBFA

In scenario 2, there are five CBR flows in AW-FQ and SBFA. Each flow has a weight of 0.2. For SBFA, we create one LTFS and its weight is set to 0.2. For AW-FQ, the $\alpha$ parameters of these flows are shown in Table 3. Scenario 2 lasts for 140 seconds. Only flow 1 and flow 2 experience channel errors during the first 70 seconds. The wireless

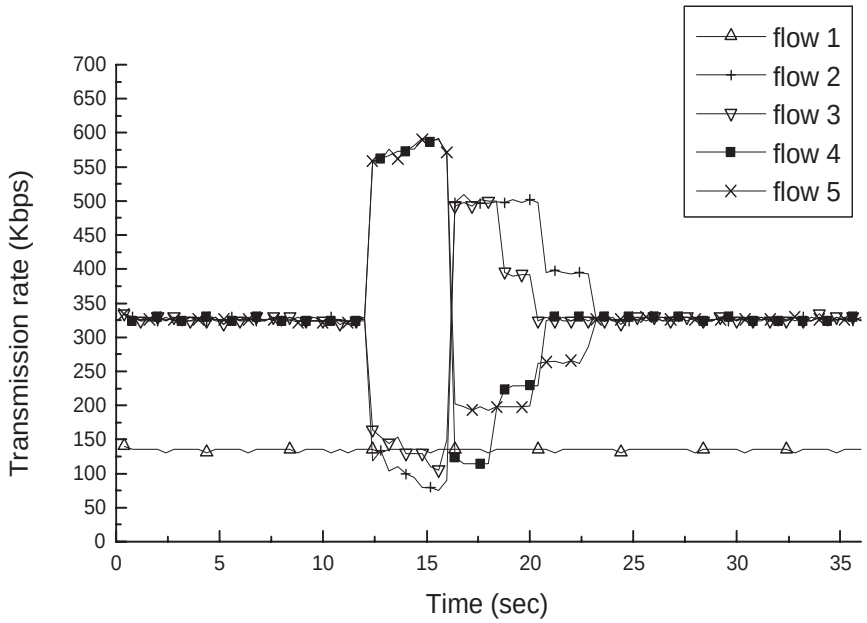



Figure 2: Transmission rate with location-dependent channel errors.

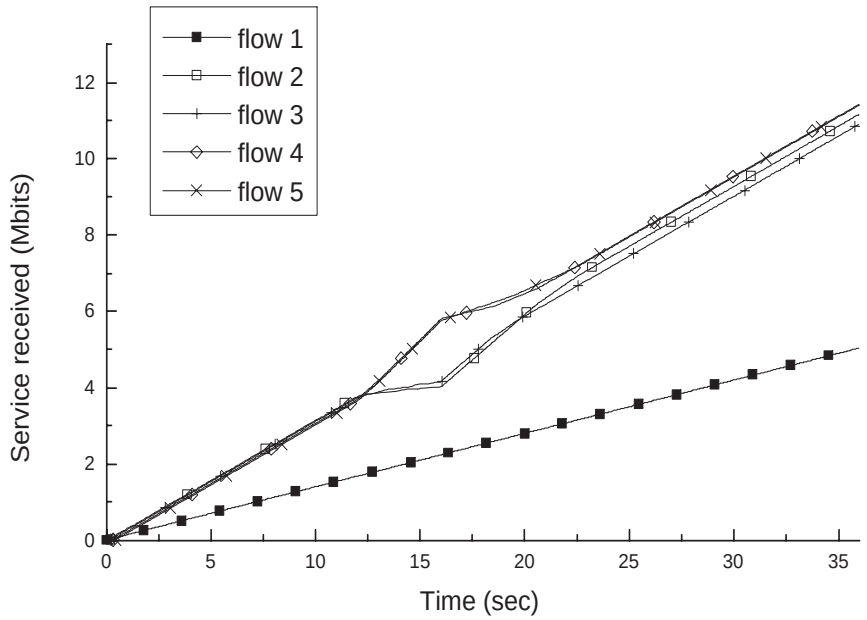
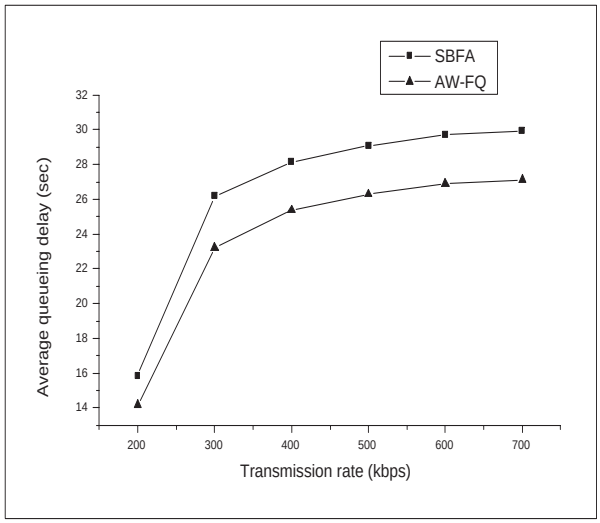
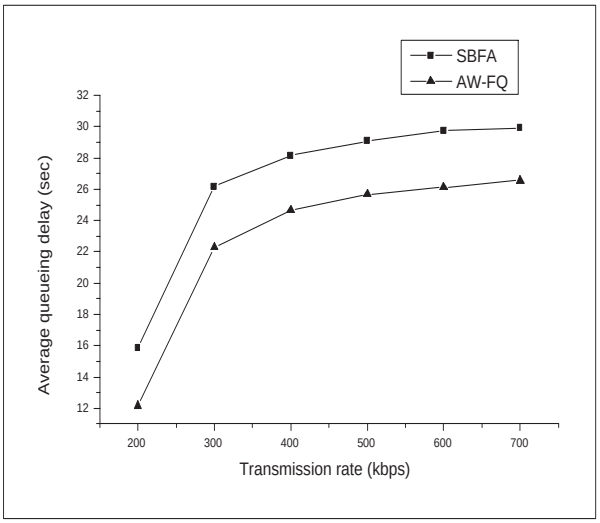


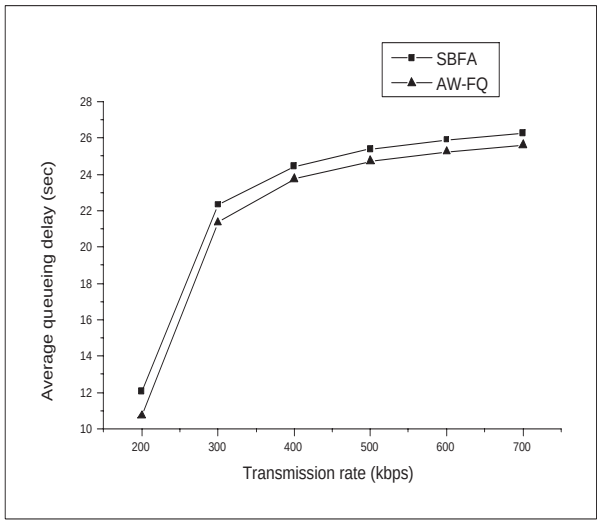

Figure 3: Service received when packet errors occur.



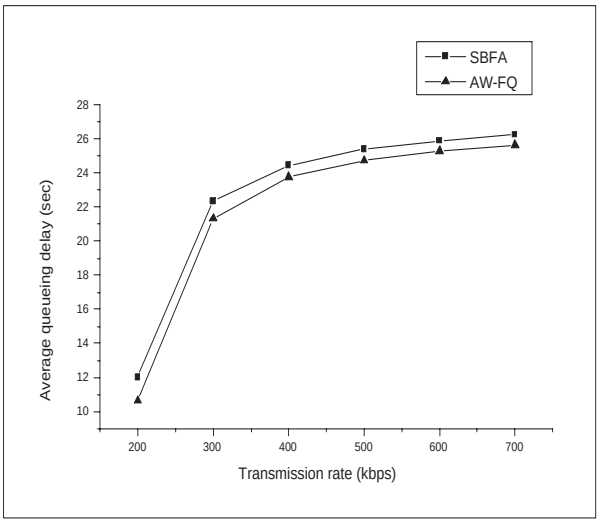

(a) Flow 1




(b) Flow 2




(c) Flow 3




(d) Flow 4

Figure 4: Average queueing delay.

Table 3: Parameters of the five flows used in scenario 2.

| | Traffic type | $W$ | $\alpha$ | Error period |
|---|---|---|---|---|
| flow 1 | CBR | 0.2 | 0.4 | (12, 16) |
| flow 2 | CBR | 0.2 | 0.8 | (12, 16) |
| flow 3 | CBR | 0.2 | 0.7 | None |
| flow 4 | CBR | 0.2 | 0.4 | None |
| flow 5 | CBR | 0.2 | 0.4 | None |

channel of flow 1 and flow 2 during the first 70 seconds evolves according to the two-state discrete Markov chain. Let $p_g$ be the probability that the next time slot is good given that the current time slot is in errors, and $p_e$ be the probability that the next time slot is in errors given that the current slot is good. Then, the steady state probability $P_G$ and $P_E$ of being in the good and bad states, respectively, are given by $P_G = \frac{p_g}{p_g+p_e}$ and $P_E = \frac{p_e}{p_g+p_e}$ [7]. Flow 1 and flow 2 have a steady state probability $P_G = 0.7$ with $p_g + p_e = 1$. The packet error rate increases 0.3 every time when the flow is in the bad state, and is set to zero when in the good state. We set different transmission rates for the five flows and compare the average queueing delay of each flow. As shown in Figure 4, AW-FQ provides rapid compensation for flow 1 and flow 2 that experience channel errors and has less impact on other error-free flows than SBFA.

## 5 Conclusions and Future Work

We have presented an adaptive weighted fair queueing algorithm (AW-FQ) for wireless networks. Unlike other alternative approaches which focus on maximizing system throughput, AW-FQ provides minimal throughput guarantee for flows in the presence of channel errors at the expense of limited system throughput. In addition, AW-FQ does not assume having full knowledge of current channel conditions since the error prediction algorithms used by other approaches are not realistic. Simulation results have shown that AW-FQ has

guaranteed throughput and fairness. Compared to the SBFA, AW-FQ has lower queueing delay and does not have to assign a fixed flow which occupies system bandwidth for compensation. AW-FQ provides better QoS for flows when experiencing channel errors and does not decrease the service amount of other error-free flows.

## References


[1] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *IEEE/ACM Transactions on Networking,* vol. 7, no. 4, pp. 473-489, Aug. 1999.

[2] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors," *Proc. IEEE INFOCOM'98,* vol. 3, pp. 1103-1111, Apr. 1998.

[3] P. Lin, B. Benssou, Q. L. Ding, and K. C. Chua, "CS-WFQ: A Wireless Fair Scheduling Algorithm for Error-Prone Wireless Channels," *Proc. 9th International Conference on Computer Communications and Networks,* pp. 276 -281, Aug. 2000.

[4] P. Lin, B. Benssou, Q. L. Ding, and K. C. Chua, "A Wireless Fair Scheduling Algorithm for Error-Prone Wireless Channels," *Proc. ACM MOBICOM'00,* pp. 11-20, Aug. 2000.

[5] P. Goyal, H. M. Vin, and H. Cheng,M "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *IEEE/ACM Trans. Networking,* vol. 5, pp. 690-704, Oct. 1997.

[6] P. Ramanathan and P. Agrawal, "Adapting Packet Fair Queueing Algorithms to Wireless Networks," *Proc. ACM/IEEE MOBICOM'98,* pp. 1-9, Oct. 1998.

[7] T. Nandagopal, S. Lu, and V. Bharghavan, "A Unified Architecture for the Design and Evaluation of Wireless Fair Queueing Algorithms," *Proc. ACM/IEEE MOBICOM'99,* pp. 132-142, Aug. 1999.

[8] Ns Manual, The VINT Projects, UC Berkeley, LBL, USI/ISI, and Xerox PARC, *http://www.isi.edu/nsnam/ns/*.