A Fair Scheduling Algorithm with Traffic Classification for Wireless Networks*

You-Chiun Wang, Shiang-Rung Ye, and Yu-Chee Tseng
Department of Computer Science and Information Engineering
National Chiao Tung University
Hsin-Chu, 30010, Taiwan
Email: {wangyc, shiarung, yctseng}@csie.nctu.edu.tw

Keywords: data communication, fair queuing, mobile communication system, scheduling, wireless network.

Abstract

Wireless channels are characterized by more serious bursty and location-dependent errors. Many packet scheduling algorithms have been proposed for wireless networks to guarantee fairness and delay bounds. However, most existing schemes do not consider the difference of traffic natures among packet flows. This will cause the *delay-weight coupling* problem. In particular, serious queuing delays may be incurred for real-time flows. To resolve this problem, we propose a *Traffic-Dependent wireless Fair Queuing (TD-FQ)* algorithm that takes traffic types of flows into consideration when scheduling packets. The proposed TD-FQ algorithm not only alleviates queuing delay of real-time flows, but also guarantees bounded delays and fairness for all flows.

1 Introduction

To meet QoS requirements, many packet scheduling algorithms [1, 2, 3, 4, 5, 6] have been proposed for wireline networks to guarantee fairness and delay bounds. However, it is not a trivial task to directly apply these algorithms to wireless domain. In particular, wireless channels are characterized by more serious bursty and location-dependent errors [7, 8]. Bursty errors may break a flow's continuous services, while location-dependent errors are likely to allow error-free flows to receive more services than they deserve, thus violating the fairness and delay bound properties.

To solve these problems, several wireless packet scheduling algorithms have been proposed [9, 10, 11, 12, 13, 14]. In *IWFQ (Idealized Wireless Fair Queuing)* [9], each packet is associated with a *finish tag*, which is computed according to the principles of *WFQ (Weight Fair Queuing)* [2]. The scheduler always selects the error-free packet with the smallest finish tag to serve. When a flow suffers from channel errors, all its packets will keep their old tags. Therefore, when the flow exits from errors, its packets are likely to have smaller finish tags, thus achieving the compensation purpose. In *CIF-Q (Channel-condition Independent Fair Queuing)* [10], fairness is achieved by transferring the time allocated to those error flows to those error-free flows. Later on, compensation services will be dispatched to the former proportional to their weights. However, as [13] shows, an inherent limitation of fluid fair queuing is that the delay observed by a flow is tightly coupled with the fraction of bandwidth given to that flow among all backlogged flows. Since the fraction is in turn coupled with the weight assigned to the flow, we call this the *delay-weight coupling problem*. Both IWFQ and CIF-Q may suffer from this problem.

In this work, we consider the fair scheduling problem in a wireless network whose input includes both real-time and non-real-time traffics. This problem is especially important with the recently emerging *multimedia services (MMS)* in next-generation wireless networks. Real-time applications are typically delay-sensitive. If wireless fair scheduling is supported without special consideration for real-time flows, the delay-weight dilemma would either hurt real-time flows or the system performance. Several wireless scheduling algorithms have been proposed to address this concern

*Y. C. Tseng's research is co-sponsored by the MOE Program for Promoting Academic Excellence of Universities, by NSC of Taiwan under grant numbers NSC92-2213-E009-076 and NSC92-2219-E009-013, by Computer and Communications Research Labs., ITRI, Taiwan, by Intel Inc., by the Institute for Information Industry and MOEA, R.O.C, under the Handheld Device Embedded System Software Technology Development Project, by the Lee and MTI Center of NCTU, and by Chung-Shan Institute of Science and Technology under contract number BC93B12P.

[11, 12, 13, 14]. However, they still suffer from certain weaknesses (refer to Section 2).

In this work, we propose a new algorithm called *Traffic-Dependent wireless Fair Queuing* (TD-FQ). Traffics arriving at a base station are mixed with real-time and non-real-time flows. TD-FQ is developed based on CIF-Q [10], but it adds extra mechanisms to reduce queuing delays of real-time flows by giving them higher priorities. Nevertheless, TD-FQ guarantees that the special treatment of real-time flows will not starve non-real-time flows. Thus, it still maintains fairness and bounded delays for all flows.

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 presents our TD-FQ algorithm. Section 4 formally proves several properties of TD-FQ. Simulation results are presented in Section 5. Conclusions are drawn in Section 6.

2 Related Work

In *SBFA (Server Based Fairness Approach)* [11], a fraction of bandwidth is reserved particularly for compensation purpose. A number of virtual servers called *LTFS (Long Term Fairness Servers)* are created for those flows that experienced errors. Then the reserved bandwidth will be used to compensate those LTFS flows. However, since the erroneous flows are compensated in a first-come-first-served manner, real-time lagging flows may still suffer from long queuing delay.

ELF (Effort-Limited Fair) [12] suggests to adjust each flow's weight in response to the error rate of that flow, up to a maximum defined by that flow's *power factor*. However, since the scheduler does not have immediate knowledge about the error rates of a flow, there could be some delay in adjusting its weight to respond to its channel and queue condition. Besides, when a real-time flow just exits from errors, it is emergent to deliver packets for the flow, or these packets may be dropped. Unfortunately, adjusting weights cannot guarantee higher priorities for such flows.

WFS (Wireless Fair Service) [13] assigns each flow i with a *rate weight* r_i and a *delay weight* Φ_i, and associates every packet p_i^k with a start tag $S(p_i^k)$ and a finish tag $F(p_i^k)$,

$$S(p_i^k) = max\{V(A(p_i^k)), S(p_i^{k-1}) + L_i^{k-1}/r_i\},$$
$$F(p_i^k) = S(p_i^k) + L_i^k/\Phi_i,$$

where L_i^k is the length of the kth packet of flow i, $A(p_i^k)$ is the arrival time of the packet, and $V(t)$ is the virtual time at time t. Essentially, flow i is drained into the scheduler according to the rate weight r_i, but served according to the delay weight Φ_i. The flow with the smallest finish tag will be picked by the scheduler. By introducing the delay weight, WFS decouples delay and bandwidth to a certain degree. However, since the computation of start tags is still based on rate weights, real-time flows may not get much benefit. Besides, WFS adopts a compensation mechanism based on a weighted round robin approach, where the lagging degree of a flow is used as its weight. Without distinguishing real-time and non-real-time flows, this algorithm may still cause serious queuing delays for real-time flows.

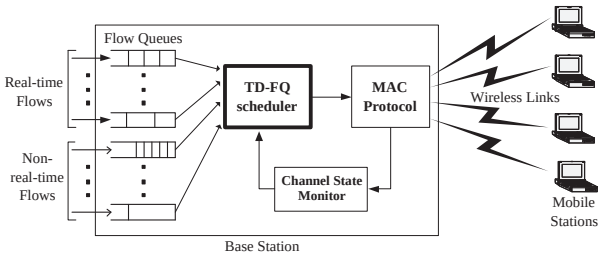

Figure 1: System architecture of TD-FQ.

Lee *et al* [14] classify flows into four groups: *poor*, *poorer*, *rich*, and *normal*. A flow is said *poor* if it receives less service than it expects. When a poor flow transmitting real-time traffic is about to drop packets due to long waiting time, this flow is changed to a *poorer* flow. When there are compensation services available, the poorer flows always have the highest priority to receive such services. However, this behavior may cause other poor flows to starve if there are many poorer flows.

3 The TD-FQ Algorithm

Below, we first introduce the system model and basic operations of TD-FQ, followed by some special designs of TD-FQ, including *graceful degradation*, *compensation*, and *lag redistribution*.

3.1 System Model

We consider a packet-cellular network as in Fig. 1. Packets arriving at a *base station (BS)* are classified into real-time traffic and non-real-time traffic and dispatched into different queues depending on their destination mobile stations. These traffic flows are sent to the *TD-FQ scheduler*, which is responsible for scheduling flows and transmitting their head-of-line (HOL) packets via the *MAC protocol*. The *Channel state monitor* provides information about the channel

state of each mobile station (there are different alternatives to achieve this, but this is out of the scope of this work). For simplicity, we assume that BS has immediate and accurate knowledge of each channel's state.

In this paper, we focus on the design of TD-FQ scheduler. Mobile stations may suffer from bursty and location-dependent channel errors. However, error periods are assumed to be sporadic and short relative to the whole lifetime of flows so that long-term unfairness would not happen.

3.2 Basic Operations

Following most fair queuing works, each flow i is assigned a *weight* r_i to represent the ideal fraction of bandwidth that the system commits to it. However, the real services received by flow i may not match exactly its assigned weight. So we maintain a *virtual time* v_i to record the nominal services received by it, and a *lagging level* lag_i to record its credits/debits. The former is to compete with other flows for services, while the latter is to arrange compensation services. The actual normalized service received by flow i is $v_i - lag_i/r_i$. Flow i is called *leading* if $lag_i < 0$, called *lagging* if $lag_i > 0$, and called *satisfied* if $lag_i = 0$. Further, depending on its queue content, a flow is called *backlogged* if its queue is nonempty, called *unbacklogged* if its queue is empty, and called *active* if it is backlogged or unbacklogged but leading. Note that TD-FQ will only choose active flows to serve. When an unbacklogged but leading flow (i.e., an active flow) is chosen, its service will actually be transferred to another flow for compensation purpose. Also, following the principle of CIF-Q, whenever a flow i transits from unbacklogged to backlogged, its virtual time v_i is set to $max\{v_i, min_{j\in A}\{v_j\}\}$, where A is the set of all active flows.

Fig. 2 outlines the scheduling policy of TD-FQ. TD-FQ follows the design principle of CIF-Q. First, the active flow i with the smallest virtual time v_i is selected. If flow i is backlogged and its channel condition is good, the HOL packet of flow i can be served if flow i is non-leading, in which case the service is called a *normal service (NS)*. Then we update the virtual time v_i as $(v_i + l_p/r_i)$, where l_p is the length of the packet. In case that flow i has to give up its service due to an empty queue or a bad channel condition, the service will become an *extra service (ES)*. On the other hand, if flow i is over-served (i.e., leading), the *Graceful Degradation Scheme* will be activated to check if flow i is still eligible for the service. If flow i has to give up its service, the service will be transferred to a *compensation service (CS)*. In both cases of CS and ES, the *Compensation Scheme* will be triggered, trying

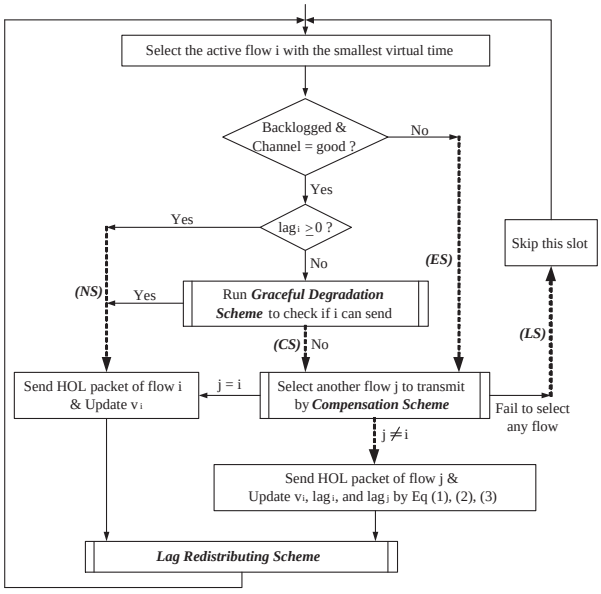

Figure 2: Scheduling policy of TD-FQ.

to select another flow j to serve. If the scheme fails to select any flow, this service is wasted, called a *lost service (LS)*. If the scheme still selects flow i to serve, then we update v_i and send its HOL packet. If a flow j $(\neq i)$ is selected, flow j's packet will be sent and the values of v_i, lag_i, and lag_j are updated as follows:

$$v_i = v_i + l_{p'}/r_i, \quad (1)$$

$$lag_i = lag_i + l_{p'}, \quad (2)$$

$$lag_j = lag_j - l_{p'}, \quad (3)$$

where p' is the packet being sent. Note that in this case we charge to flow i by increasing its virtual time, but credit (respectively, debit) to lag_i (respectively, lag_j) of flow i (respectively, j).

Whenever the scheduler serves the HOL packet of any flow i, it has to check the queue size of flow i. If it finds that flow i's queue is empty, it will invoke the *Lag Redistributing Scheme* to adjust flow i's lag, if necessary.

Below, we introduce the three schemes in TD-FQ. Table 1 summarizes notations used in TD-FQ.

3.3 Graceful Degradation Scheme

When a leading flow i is selected for service, the Graceful Degradation Scheme will be triggered to check its leading amount. Here we adopt the idea in CIF-Q to limit the amount of such services a leading flow may enjoy. The scheme in CIF-Q works as follows. A leading flow is allowed to receive an amount of extra service proportional to its normal ser-

Table 1: Summary of symbols used in TD-FQ.

Symbols	Definition
v_i	virtual time of flow i
lag_i	the credits/debits of flow i
r_i	weight of flow i
s_i	graceful degradation service index of flow i when $lag_i < 0$
α_R, α_N	graceful degradation ratios for real-time and non-real-time flows
δ	the threshold to distinguish seriously/moderately lagging flows
L_R, L_N, L_R^S,L_R^M, L_N^S, L_N^M	lagging flows (defined in CWC)
W_R, W_N, W_R^S,W_R^M, W_N^S, W_N^M	weights of lagging flows L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M, respectively
G_R, G_N, G_R^S,G_R^M, G_N^S, G_N^M	normalized amounts of ES/CS received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M, respectively
B	bound of differences of services (used in CWC)
c_i^S, c_i^M	normalized amounts of ES/CS received by flow i when $lag_i/r_i \geq \delta$ and $0 < lag_i/r_i < \delta$, respectively
f_i	normalized amount of ES received by flow i when $lag_i \leq 0$

vices. Specifically, when a flow i transits from lagging/satisfied to leading, we set up a parameter $s_i = \alpha \cdot v_i$, where α ($0 \leq \alpha \leq 1$) is a system-defined constant. Later on, flow i's virtual time will be increased every time it is selected by the scheduler (note that 'selected' does not mean that it is actually served). Let v_i' be flow i's current virtual time when it is selected. We will allow flow i to be served if $s_i \leq \alpha v_i'$. If so, s_i is updated as $s_i + l_p/r_i$, where l_p is the length of the packet. Intuitively, flow i can enjoy approximately $\alpha(v_i' - v_i)$ services, and this is called *graceful degradation*.

TD-FQ adopts the above idea. Further, to distinguish real-time from non-real-time flows, we substitute α by a parameter α_R for real-time flows, and by α_N for non-real-time flows. We set $\alpha_R > \alpha_N$ to distinguish their priorities.

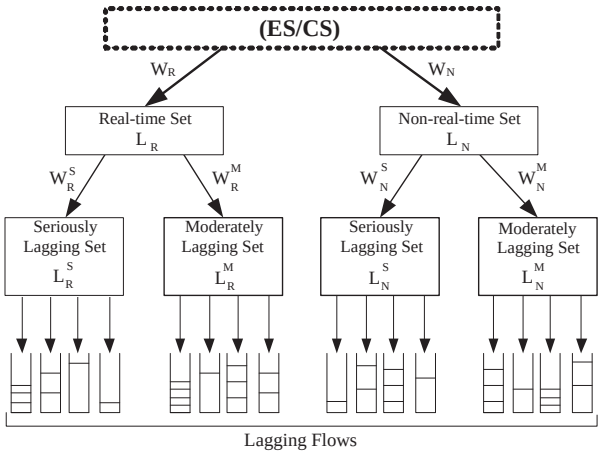

Figure 3: Structure of the class-based weight compensation (CWC) scheme.

3.4 Compensation Scheme

When the selected flow i has a bad channel or fails to satisfy the graceful degradation condition, the Compensation Scheme will be triggered (reflected by ES and CS in Fig. 2). In this case, lagging flows should always have a higher priority over non-lagging flows to receive such additional services. Section 3.4.1 discusses how to choose a lagging flow. Section 3.4.2 deals with the case when all lagging flows are experiencing error.

3.4.1 Dispatching ES and CS to Lagging Flows

The Compensation Scheme first tries to dispatch ES/CS to lagging flows. We propose a *class-based weight compensation (CWC)* mechanism, as illustrated in Fig. 3. CWC first divides lagging flows into a real-time set L_R and a non-real-time set L_N. These sets are each further divided into a seriously lagging set and a moderately lagging set. Individual flows are at the bottom. The concept of weight is used to dispatch services to these sets.

To dispatch ES/CS to L_R and L_N, we assign weights W_R and W_N to them, respectively. (Normally, we would set $W_R \geq W_N$.) Also, a variable G_R (respectively, G_N) is used to record the normalized ES/CS received by L_R (respectively, L_N). When both L_R and L_N have error-free flows, the service will be given to L_R if $G_R \leq G_N$, and to L_N otherwise. When only one of L_R and L_N has error-free flows, the service will be given to that one, independent of the values of G_R and G_N. When L_R receives the service, G_R is updated as

$$G_R = min\left\{G_R + \frac{l_p}{W_R}, \frac{B + G_N W_N}{W_R}\right\}, \qquad (4)$$

where l_p is the length of the transmitted packet, and B is a predefined value to bound the difference between

G_R and G_N. Similarly, when L_N receives the service, G_N is updated as

$$G_N = min\left\{G_N + \frac{l_p}{W_N}, \frac{B + G_R W_R}{W_N}\right\}. \quad (5)$$

Note that to avoid the cases of $G_R \gg G_N$ or $G_N \gg G_R$, which may cause L_R or L_N to starve when the other set recovers from error, we set up a bound $|G_R W_R - G_N W_N| \leq B$. This gives the second term in the righthand side of Eqs. (4) and (5).

The flows in L_R are further divided into a seriously lagging set L_R^S and a moderately lagging set L_R^M. We assign a real-time lagging flow i to L_R^S if $lag_i/r_i \geq \delta$, where δ is a predefined value. Otherwise, flow i is assigned to L_R^M. Similarly, the flows in L_N are divided into a seriously lagging set L_N^S and a moderately lagging set L_N^M. Again, services are dispatched to sets L_R^S, L_R^M, L_N^S, and L_N^M according their weights W_R^S, W_R^M, W_N^S, and W_N^M, respectively. To favor seriously lagging flows, we suggest that $W_R^S \geq W_R^M$ and $W_N^S \geq W_N^M$. Services are dispatched to these sets similar to the earlier case (i.e., the service distribution to L_R and L_N). We use G_R^S, G_R^M, G_N^S, and G_N^M to record the services received by these sets. Again a bound B is set to limit the differences between G_R^S and G_R^M and between G_N^S and G_N^M.

At the bottom of CWC are four groups of individual flows of the same properties (traffic types and lagging degrees). Here the scheduler dispatches ES/CS proportional to flows' weights. Specifically, for each flow i, we maintain two *compensation virtual times* c_i^S and c_i^M, which keep track of the normalized amount of ES/CS received by flow i when $lag_i/r_i \geq \delta$ and $0 < lag_i/r_i < \delta$, respectively. When the scheduler chooses the seriously lagging set (L_R^S or L_N^S), it selects the *error-free* flow i with the smallest c_i^S in the set to serve. Similarly, when the scheduler chooses the moderately lagging set (L_R^M or L_N^M), it selects the error-free flow i with the smallest c_i^M in the set to serve. When a lagging flow i receives such a service, its compensation virtual times are updated as

$$\begin{cases} c_i^S = c_i^S + l_p/r_i, & \text{if } lag_i/r_i \geq \delta \\ c_i^M = c_i^M + l_p/r_i, & \text{otherwise} \end{cases}.$$

When a flow i newly enters one of the sets L_R^S, L_R^M, L_N^S, and L_N^M or transits from one set to another, we have to assign its c_i^S or c_i^M as follows. If flow i is seriously lagging (i.e., $lag_i/r_i \geq \delta$), we set

$$c_i^S = \begin{cases} max\{c_i^S, c_{min}^{SR}\}, & \text{if flow } i \text{ is real-time} \\ max\{c_i^S, c_{min}^{SN}\}, & \text{if flow } i \text{ is non-real-time} \end{cases}.$$

Otherwise, we set

$$c_i^M = \begin{cases} max\{c_i^M, c_{min}^{MR}\}, & \text{if flow } i \text{ is real-time} \\ max\{c_i^M, c_{min}^{MN}\}, & \text{if flow } i \text{ is non-real-time} \end{cases},$$

where c_{min}^{SR} (respectively, c_{min}^{SN}) is the minimum value of c_j^S such that $j \in L_R^S$ (respectively, $j \in L_N^S$), and c_{min}^{MR} (respectively, c_{min}^{MN}) is the minimum value of c_j^M such that $j \in L_R^M$ (respectively, $j \in L_N^M$). One exception is when the set $L_R^S/L_N^S/L_R^M/L_N^M$ is empty, in which case $c_{min}^{SR}/c_{min}^{SN}/c_{min}^{MR}/c_{min}^{MN}$ is undefined. If so, we set $c_{min}^{SR}/c_{min}^{SN}/c_{min}^{MR}/c_{min}^{MN}$ to the value of c_j^S/c_j^M of the last flow j that left the set $L_R^S/L_N^S/L_R^M/L_N^M$.

The main contribution of CWC is that it compensates more services for real-time flows and for seriously lagging flows, thus alleviating these flows' queuing delays. Besides, CWC does not starve other lagging flows because these flows can still share a fraction of ES/CS.

3.4.2 Dispatching ES to Non-lagging Flows

If there is no lagging flow selected in the previous stage (due to errors), the service will be dispatched according to its original type. If the service comes from CS, it will be returned back to the originally selected flow. Otherwise, the (ES) service will be given to a non-lagging flow. Just like CIF-Q, TD-FQ also dispatches ES proportional to those non-lagging flows' weights. That is, each flow i is assigned with an *extra virtual time* f_i to keep track of the normalized amount of ES received by flow i when it is non-lagging ($lag_i \leq 0$). Whenever a backlogged flow i becomes error-free and non-lagging, f_i is set to

$$f_i = max\{f_i, min\{f_j \mid \text{flow } j \text{ is error-free, backlogged, non-lagging, and } j \neq i\}\}.$$

The scheduler selects the flow i with the smallest f_i value among all *error-free*, *backlogged*, and *non-lagging* flows to serve. When flow i receives the service, f_i is updated as $(f_i + l_p/r_i)$. An exception occurs when there is no selectable non-lagging flow, in which case this time slot will simply be wasted.

3.5 Lag Redistribution for Unbacklogged Flows

After a flow is served, if its queue state changes to unbacklogged and it is still lagging, we will distribute its credit to other flows that are in debet and reset its credit to zero. This is because the flow does not need the credit any more [15]. This is done by the Lag Redistribution Scheme.

The scheme examines the flow i that is actually served in this round. After the service, if flow i's queue becomes empty and $lag_i > 0$, we will give its credit to other flows in debet proportional to their weights, i.e.,

for each flow k such that $lag_k < 0$, we set

$$lag_k = lag_k + lag_i \times \frac{r_k}{\sum_{lag_m<0} r_m}.$$

Then we reset $lag_i = 0$. Our redistribution rule is slightly different from CIF-Q (where all flows, including lagging ones, will share the credit). We feel that it makes sense to give these credits to only those flows in need of services.

4 Theoretical Analyses

In this section, we analyze the fairness and delay properties of TD-FQ. Our proofs rely on the following assumptions: (i) $\alpha_R \geq \alpha_N$, (ii) $W_R \geq W_N$, (iii) $W_R^S \geq W_R^M$, (iv) $W_N^S \geq W_N^M$, and (v) $B \geq \hat{L}_{max}$, where $\hat{L}_{max}$ is the maximum length of a packet. The complete proofs can be found in [16].

4.1 Fairness Properties

Theorems 1–3 show the fairness property guaranteed by TD-FQ. Theorem 1 is for flows of the same traffic type, while Theorem 2 is for flows of different types. Theorem 3 provides some bounds on differences of services received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_N^M.

Theorem 1. *For any two active flows i and j of the same traffic type, the difference between the normalized services received by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged, error-free, and remain in the same state (leading, seriously lagging, moderately lagging, or satisfied) satisfies the inequality:*

$$\left|\frac{\Phi_i(t_1,t_2)}{r_i} - \frac{\Phi_j(t_1,t_2)}{r_j}\right| \leq \varepsilon \cdot \left(\frac{\hat{L}_{max}}{r_i} + \frac{\hat{L}_{max}}{r_j}\right),$$

where $\Phi_i(t_1,t_2)$ represents the services received by flow i during $[t_1,t_2)$, $\varepsilon = 3$ if both flows belong to the same lagging set (L_R^S, L_R^M, L_N^S, or L_N^M) or both flows are satisfied, $\varepsilon = 3 + \alpha_R$ if both flows are real-time leading flows, and $\varepsilon = 3 + \alpha_N$ if both flows are non-real-time leading flows.

Theorem 2. *For any real-time flow i and non-real-time flow j, the difference between the normalized services received by flows i and j in any time interval $[t_1, t_2)$ during which both flows are continuously backlogged, error-free, and remain leading satisfies the inequality:*

$$\left|\frac{\Phi_i(t_1,t_2)}{r_i} - \frac{\Phi_j(t_1,t_2)}{r_j}\right| \leq 3\left(\frac{\hat{L}_{max}}{r_i} + \frac{\hat{L}_{max}}{r_j}\right) + 2\alpha_N \frac{\hat{L}_{max}}{r_j}.$$

Theorem 3. *The difference between normalized ES/CS received by any two lagging sets in any time interval $[t_1, t_2)$ during which both sets remain active satisfies the inequalities:*
(1) for L_R and L_N:

$$\left|\frac{\Phi_R(t_1,t_2)}{W_R} - \frac{\Phi_N(t_1,t_2)}{W_N}\right| \leq \frac{B+\hat{L}_{max}}{W_R} + \frac{B+\hat{L}_{max}}{W_N},$$

(2) for L_R^S and L_R^M:

$$\left|\frac{\Phi_R^S(t_1,t_2)}{W_R^S} - \frac{\Phi_R^M(t_1,t_2)}{W_R^M}\right| \leq \frac{B+\hat{L}_{max}}{W_R^S} + \frac{B+\hat{L}_{max}}{W_R^M},$$

(3) for L_N^S and L_N^M:

$$\left|\frac{\Phi_N^S(t_1,t_2)}{W_N^S} - \frac{\Phi_N^M(t_1,t_2)}{W_N^M}\right| \leq \frac{B+\hat{L}_{max}}{W_N^S} + \frac{B+\hat{L}_{max}}{W_N^M},$$

where $\Phi_R(t_1,t_2)$, $\Phi_N(t_1,t_2)$, $\Phi_R^S(t_1,t_2)$, $\Phi_R^M(t_1,t_2)$, $\Phi_N^S(t_1,t_2)$, and $\Phi_N^M(t_1,t_2)$ represents ES/CS received by L_R, L_N, L_R^S, L_R^M, L_N^S, and L_M^N during $[t_1,t_2)$, respectively.

4.2 Delay Bounds

When a backlogged flow suffers from errors, it becomes lagging. Theorem 4 shows that if a lagging flow becomes error-free and has sufficient service demand, it can get back all its lagging services within bounded time.

Theorem 4. *If an active but lagging flow i becomes error-free at time t and remains backlogged continuously after time t, it is guaranteed that flow i will become non-lagging (i.e., $lag_i \leq 0$) within time Δ_t, where*

$$\Delta_t \leq \frac{\varphi(\Psi + 2\hat{L}_{max})}{r_{min}(1-\alpha_R)\widehat{C}} + (n + 1 + \frac{\varphi}{r_{min}})\frac{\hat{L}_{max}}{\widehat{C}},$$

n is the number of active flows, $\widehat{C}$ is the channel capacity, φ is the aggregate weight of all flows, φ_R is the aggregate weight of all real-time flows, φ_N is the aggregate weight of all non-real-time flows, r_{min} is the minimum weight of all flows,

$$\Psi = \frac{(W_R+W_N)(W_R^S+W_R^M)}{W_R W_R^S}\left(\frac{lag_i(t)}{r_i}\varphi_R + B + (\frac{\varphi_R}{r_i} + n - 2)\hat{L}_{max}\right) + \frac{W_R+W_N}{W_R}(\delta\varphi_R + B + (\frac{2\varphi_R}{r_i} + n - 1)\hat{L}_{max})$$

if flow i is real-time, and

$$\Psi = \frac{(W_R + W_N)(W_N^S + W_N^M)}{W_N W_N^S} \left(\frac{lag_i(t)}{r_i}\varphi_N + B + (\frac{\varphi_N}{r_i} + n - 2)\hat{L}_{max} \right) + \frac{W_R + W_N}{W_N}(\delta\varphi_N + B + (\frac{2\varphi_N}{r_i} + n - 1)\hat{L}_{max})$$

if flow i is non-real-time.

5 Simulation Results

In this section, we present some experimental results to verify the effectiveness of the proposed algorithm. We mix real-time and non-real-time traffics together. We mainly compare TD-FQ and CIF-Q and observe two performance metrics, queuing delay and throughput. Five flows are used, as shown in Table 2. Flows 1 and 2 are real-time constant-bit-rate (CBR) flows, flows 3 and 4 are non-real-time FTP flows, and flow 5 has a Poisson packet arrival. Suffering from channel errors during [5, 15) period, flows 2 and 3 will become active but lagging after the 15th second. The other flows are all leading in this experiment. For CIF-Q, we set $\alpha = 0.5$, while for TD-FQ we set $\alpha_R = 0.8$, $\alpha_N = 0.2$, $W_R = 2$, and $W_N = 1$.

Table 2: Traffic specification of the flows.

Flow no.	Traffic type	Rate	Error scenario
1	CBR	320 Kbps	Error-free
2	CBR	160 Kbps	Errors in [5,15) sec.
3	FTP	2 Mbps	Errors in [5,15) sec.
4	FTP	2 Mbps	Error-free
5	Poisson	1 Mbps	Error-free

Fig. 4 compares the queuing delays for flows 1, 2, and 5. In TD-FQ, the real-time flow 1 will experience less queuing delay compared to CIF-Q even if flow 1 remains leading all the time. This is because TD-FQ allows a real-time leading flow to keep more fraction of its normal services while remaining in the leading status. Even for the real-time lagging flow 2, TD-FQ still incurs lower queuing delay than CIF-Q due to its compensation mechanism for real-time lagging flows. The cost, as shown in Fig. 4(c), is at the slightly higher queuing delay of flow 5, which is non-real-time and leading and which contributes more compensation services in TD-FQ than that in CIF-Q.

Based on the same environment, Fig. 5 shows the throughputs of flows 2, 3, and 4. For the real-time lagging flow 2, TD-FQ gives it more services than CIF-Q due to its compensation mechanism. Even for the non-real-time lagging flow 3, TD-FQ still gives it more services than CIF-Q, because total compensation services in TD-FQ are more than that in CIF-Q. However, the cost, as shown in Fig. 5(c), is at lower throughput of flow 4, which is non-real-time and leading, and which contributes more compensation services in TD-FQ than that in CIF-Q.

6 Conclusions

We have addressed the delay-weight coupling problem that exists in many existing fair-queuing schemes. A new algorithm, TD-FQ, is proposed to solve this problem. By taking traffic types of flows into consideration when scheduling packets, TD-FQ not only alleviates queuing delay of real-time flows, but also guarantees bounded delays and fairness for all flows. Fairness properties and delay bounds guaranteed by TD-FQ are derived analytically. Simulation results have also shown that TD-FQ has smaller queuing delay for real-time flows when compared to CIF-Q.

References

[1] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 3, pp. 344–357, 1993.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Journal of Internetworking Research and Experience*, vol. 1, pp. 3–26, 1990.

[3] P. Goyal, H. M. Vin, and H. Chen, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," in *Conference on Applications, technologies, architectures, and protocols for computer communications*, 1996, pp. 157–168.

[4] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *INFOCOM*, 1994, pp. 12–16.

[5] J. Bennett and H. Zhang, "$\mathcal{W}F^2Q$: worst-case fair weighted fair queueing," in *INFOCOM*, vol. 1, 1996, pp. 120–128.

[6] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," in *Conference on Applications, technologies, architectures, and protocols for computer communications*, 1996, pp. 143–156.

[7] V. Bharghavan, S. Lu, and T. Nandagopal, "Fair queuing in wireless networks: issues and approaches," *IEEE Personal Communications*, vol. 6, pp. 44–53, 1999.

[8] Y. Cao and V. O. K. Li, "Scheduling algorithms in broadband wireless networks," in *IEEE Proc. of the IEEE*, vol. 89, no. 1, 2001, pp. 76–87.

[9] S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 7, no. 4, pp. 473–489, 1999.

[10] T. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *INFOCOM*, vol. 3, 1998, pp. 1103–1111.

[11] P. Ramanathan and P. Agrawal, "Adapting packet fair queuing algorithms to wireless networks," in *ACM/IEEE Int'l Conf. on Mobile Computing and Networking*, 1998, pp. 1–9.

[12] D. Eckhardt and P. Steenkiste, "Effort-limited fair (ELF) scheduling for wireless networks," in *INFOCOM*, vol. 3, 2000, pp. 1097–1106.

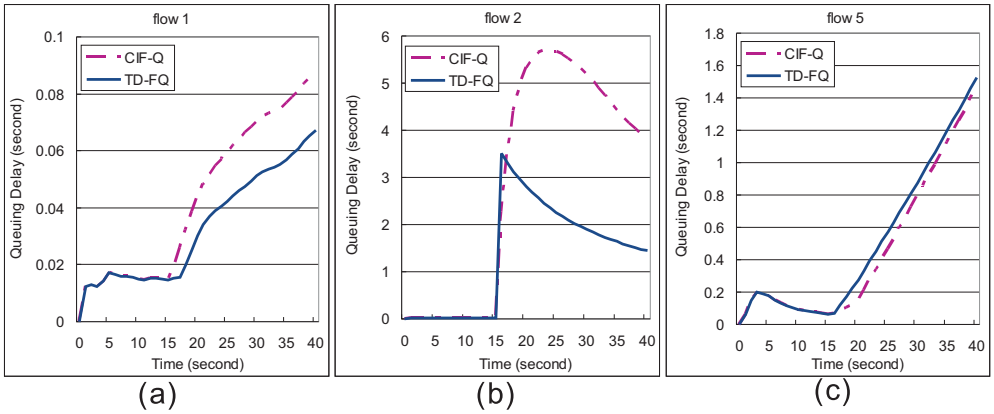

Figure 4: Comparison of queuing delays with hybrid traffics: (a) real-time leading flow 1, (b) real-time lagging flow 2, and (c) non-real-time leading flow 5.

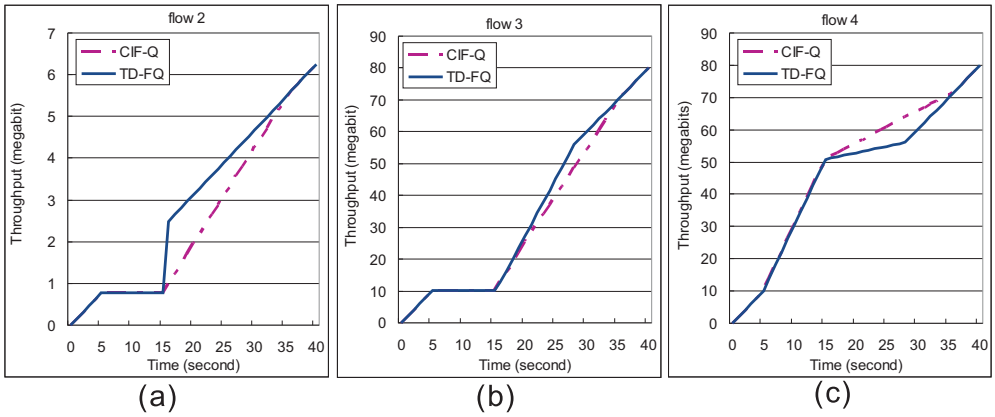

Figure 5: Comparison of throughputs with hybrid traffics: (a) real-time lagging flow 2, (b) non-real-time lagging flow 3, and (c) non-real-time leading flow 4.

[13] S. Lu, T. Nandagopal, and V. Bharghavan, "A wireless fair service algorithm for packet cellular networks," in *ACM/IEEE Int'l Conf. on Mobile Computing and Networking*, 1998, pp. 10–20.

[14] S. Lee, K. Kim, and A. Ahmad, "Channel error and handoff compensation scheme for fair queueing algorithms in wireless networks," in *IEEE ICC*, vol. 5, 2002, pp. 3128–3132.

[15] Y. Yi, Y. Seok, T. Kwon, Y. Choi, and J. Park, "$\mathcal{W}^2F^2Q$: packet fair queuing in wireless packet networks," in *Proceedings of the 3rd ACM international workshop on Wireless mobile multimedia*, 2000, pp. 2–10.

[16] Y.-C. Wang, S.-R. Ye, and Y.-C. Tseng, "A fair scheduling algorithm with traffic classification for wireless networks," National Chiao-Tung University, Taiwan, Tech. Rep., May 2004.