W^2F^2Q: Packet Fair Queuing in Wireless Packet Networks [*]

Yung Yi, Yongho Seok,
Taekyoung Kwon, Yanghee Choi
Seoul National University
Seoul, Korea
yiyung,yhseok,tkkwon,yhchoi@mmlab.snu.ac.kr

Junseok Park
Electronics and Telecommunications Research Institute
Seoul, Korea
parkjs@etri.re.kr

ABSTRACT

Recently, there is a growing interest in wireless packet communications due to the explosive growth in wireless communications and the Internet. In this stage, quality of service (QoS) provisioning in wireless/mobile packet networks is becoming more and more important. A key factor in QoS provisioning is packet-scheduling. However, conventional scheduling algorithms in wired network cannot be directly applicable to wireless communication environments because of wireless-specific characteristics : bursty and location-dependent errors.

In this paper, we propose a new wireless packet scheduling method: Wireless Worst-case Fair Weighted Fair Queuing (W^2F^2Q). The proposed W^2F^2Q is based on WF^2Q+ [1, 2], which is the most accurate packet scheduling algorithm among those emulating the ideal GPS algorithm. W^2F^2Q uses the tight globally bounded timestamp (GBT) property [9] of WF^2Q+ to detect leading and lagging status of each flow. Theoretical analysis verifies that W^2F^2Q guarantees the fairness property among flows. Also, simulation experiments show the effect of readjusting and graceful degradation in W^2F^2Q.

Keywords

Packet scheduling, fair queuing, wireless packet network

1. INTRODUCTION

The significant increase in recent activity in the area of wireless packet networks indicates that mobile hosts along with their wireless links will be an integral part of existing and future networks. With the explosive growth of wireless packet networks such as high-speed wireless Local Area Network (LAN) and next-generation wireless service, QoS provisioning has gained more and more attention in these areas, especially considering multimedia applications. Among many elements for QoS provisioning, packet scheduling is the most fine-grained element to support QoS provisioning at the base-station for downlink and uplink data flows.

[*]This work has been sponsored by the Electronics and Telecommunications Research Institute, South Korea

In wired network, various packet scheduling algorithms, e.g., Packet Fair Queuing (PFQ), have been proposed to guarantee QoS such as delay and throughput bound. The objective of every PFQ algorithm is to emulate its service as closely as possible to ideal general processor sharing (GPS) [4]. The most famous PFQ algorithms is weighted fair queuing (WFQ), equivalently packet general processor sharing (PGPS) [8]. Parekh [8] proved that WFQ can have delay and throughput bound when the data source is leaky-bucket constrained. However, WFQ has its shortcomings that it is not easy to implement because cost of maintaining a priority sorting queue and computing the virtual system time is so high. Self clocked fair queuing (SCFQ) and start time fair queuing (STFQ) reduce implementation complexity of WFQ sacrificing more GPS property than WFQ. WFQ in itself cannot fall behind GPS by one maximum packet size with respect to service given to a flow. However, WFQ can be far ahead of GPS, in which there could be a large discrepancy between WFQ and GPS. Bennett [1, 2] proposed new PFQ algorithms called worst-case fair packet fair queuing (WF^2Q, WF^2Q+) in order to solve the drawbacks of WFQ. In WF^2Q+ and WF^2Q, the amount of service given to a flow fall neither behind nor ahead of GPS by one maximum-sized packet. This property is so-called *globally bounded timestamp* (GBT) property [9]. In this sense, we will seek to extend WF^2Q+ algorithm considering wireless-specific characteristics.

The conventional PFQ algorithms developed for wired networks cannot be directly applied to wireless network, because of bursty and location-dependent errors in wireless channels. By location-dependent error, each mobile station experiences different interference and fading patterns depending on its position in a cell. In other words, the service to a mobile with bad link condition can be temporarily blocked even if it has packets to send. In PFQ, the unbacklogged flow's service share is distributed to other flows. However, in addition to unbacklogged flow's service share, blocked flow's service due to bad link conditions is also distributed to others in wireless environments. Therefore, wireless packet scheduling should keep track of lagging status of each blocked flow and make up for its lag later after recovery of its channel in order to guarantee the service fairness. This is so-called *compensation* in wireless packet scheduling

[5, 6, 7].

Recently, there have been proposed packet scheduling algorithms in wireless packet networks. Idealized wireless fair queuing (IWFQ) [5] is based on the difference between the virtual system time and the virtual finish time when a flow experiences wireless channel error. However, IWFQ cannot discriminate service discrepancy caused by wireless channel error from that by WFQ. Wireless packet scheduling (WPS) based on weighted round robin (WRR) is also proposed in [5]. WPS is simple and easy to implement. However, it is difficult to support QoS guarantee, which is originally the problem of WRR. Channel condition independent packet fair queuing (CIF-Q) [7] uses error-free reference model (STFQ) to detect lagging and leading status of each flow. However, CIF-Q has the drawbacks of STFQ (bigger service discrepancy between GPS than WFQ) and the overhead of maintaining error-free reference queuing model. Wireless fair service (WFS) [6] introduces the concept of delay-bandwidth decoupling into wireless fair queuing, which handles delay-sensitive (video and audio) and error-sensitive flow (general data) separately using look-ahead parameter. Bandwidth-delay decoupling issues are not addressed in this paper. We focus on how to extend wireline packet fair queuing algorithm for wireless networks.

The algorithm proposed in this paper, W^2F^2Q, is based on WF^2Q+. We adopt the ideas of IWFQ algorithm for wireless packet fair queuing. However, the disadvantage of IWFQ is that the scheduler cannot consider leading and lagging status caused by wireless channel errors. In contrast, W^2F^2Q takes into explicit consideration explicit lagging and leading amount of service in each flow by using the GBT property of WF^2Q+. In addition, W^2F^2Q does not use the error-free reference model such as CIF-Q, which burdens the overhead on the scheduler. Therefore, our algorithm can be implemented with minor modification of WF^2Q+. Also, we reflect various wireless scheduling issues(graceful degradation, readjusting, and unbacklogged leading/lagging flows) to our scheduling framework. Similarly to IWFQ, W^2F^2Q does not have to keep track of lagging and leading status. Therefore, it is feasible to implement W^2F^2Q with minor modification of WF^2Q+.

The rest of this paper is organized as follows. Section 2 and Section 3 briefly describes wireless network model and the motivation of wireless packet fair queuing. Section 4 describes a proposed new wireless scheduling algorithm and other issues such as readjusting, graceful degradation, unbacklogged leading/lagging flow. Section 5.1 and Section 6 show the performance evaluation results of W^2F^2Q through theoretical analysis and simulation experiments.

2. WIRELESS NETWORK MODEL

It is assumed that a wireless network is composed of one or more cells, and bandwidth in each cell is managed by a base station (BS). Examples of such networks are IEEE 802.11 LAN using Point Control Facility and CDMA cellular network supporting random access data mode. Fixed packet size at MAC-level is assumed for simplicity (denoted by L). Simplified shared-channel cellular network is considered here where each mobile terminal can experience location-dependent error patterns as in [5, 6, 7].

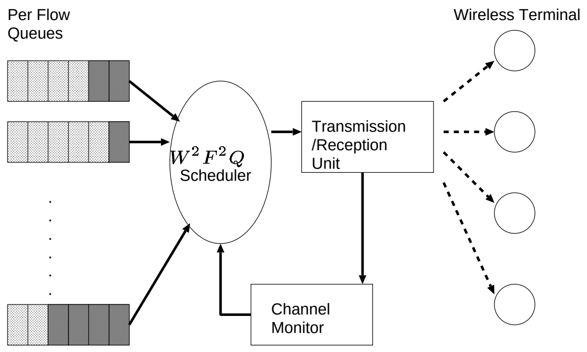


Figure 1: Wireless Scheduling Framework

Figure 1 shows a general wireless fair scheduling framework at a BS. The wireless packet scheduler proposed in this paper has three conceptual components (per-flow queues, a packet scheduler, and a channel monitor). Per-flow queues store packets for each flow. Two time-stamps (virtual start and finish time) are tagged for packets in the queue. The W^2F^2Q scheduler maintains the system virtual time for packetized emulation of GPS. It performs the eligibility test and serves packets in the increasing order of finish time. The channel monitor is used for the estimation of channel status (good or bad) between the BS and each mobile terminal. Channel monitoring using the RTS-CTS control packet of IEEE 802.11 LAN is known to be more than 95% correct [5]. Therefore, perfect channel status estimation is assumed here for brevity in this paper. As mentioned before, we focus on how to extend WF^2Q+ for wireless domain in this paper.

3. WIRELESS PACKET FAIR QUEUING

3.1 WF^2Q and WF^2Q+: Worst-case Fair Weighted Fair Queuing

In wired network, fair queuing has long been a popular paradigm for providing bounded delay and guaranteed QoS. All fair queuing algorithms are based on the fluid fair queuing model. In this model, packet flows are modeled as fluid flows. Fluid fair queuing guarantees that for an arbitrary time window $[t_1, t_2]$, any two backlogged flows i and j are serviced in proportion to their weight, which is represented by the following equation:

$$\frac{W_i(t_1,t_2)}{\phi_i} = \frac{W_j(t_1,t_2)}{\phi_j} \qquad (1)$$

where $W_i(t_1, t_2)$ and ϕ_i are the service amount received by flow i during the time window $[t_1, t_2]$ and the weight of flow i, respectively. However, in the real network, systems handle flows at the granularity of packets rather than bits. Therefore, the main objective of packetized fair queuing (PFQ) algorithm is to approximate the fluid pair queuing model as closely as possible. That is, the primary goal of PFQ is to minimize $|W_i(t_1,t_2)/\phi_i - W_j(t_1,t_2)/\phi_j|$ for any two backlogged flows i and j over time interval $[t_1, t_2]$. This objective is called as Service Fairness Index (SFI) by Golestani [3].

Among many PFQ algorithms, WFQ or PGPS [8] is widely known due to its good applicability to best-effort and guaranteed service applications. In WFQ, the selection policy of next transmission is to select the packet that would complete service in the corresponding GPS scheduler (minimum of virtual finish time). The k-th packet's virtual finish time, $f_i^k(t)$, in flow i is computed by

$$s_i^k(t) = \max(V(t), f_i^{k-1}(t)) \quad (2)$$

$$f_i^k(t) = s_i^k(t) + \frac{L}{\phi_i} \quad (3)$$

where $s_i^k(t)$ is the virtual start time of packet k in flow i at time t, L is the packet length, $V(t)$ is the virtual system time and ϕ_i is the weight of flow i at time t.

The important property of the WFQ scheduler is that WFQ in itself cannot fall behind GPS with respect to service given to a flow by one maximum packet size. However, in [1], it is shown that WFQ can be far ahead of GPS more than one maximum packet size. To solve this problem, Bennett proposed new PFQ algorithms called WF^2Q and WF^2Q+ to minimize discrepancy between GPS [1, 2], which perform the eligibility test before choosing a next packet for transmission. The eligibility test is to check whether the corresponding packet would start in GPS model or more formally whether the virtual system time is greater than the virtual start time of packet k of flow i. Through the eligibility test, the service amount of WF^2Q and WF^2Q+ is neither ahead nor behind that of the ideal GPS scheduler by one maximum packet size. Accordingly, WF^2Q and WF^2Q+ meet the GBT property defined by Definition 1 in terms of the virtual system time and the virtual start time.

Definition 1. A PFQ algorithm has the GBT property if the following condition holds:

$$s_i(t) - \frac{L}{\phi_i} \leq V(t) \leq s_i(t) + \frac{L}{\phi_i} \quad (4)$$

where L is the fixed packet size of every existing flow in the corresponding scheduler.

In WF^2Q, the virtual system time is computed using GPS emulation, which is of high implementation complexity. That's why GPS emulation makes WF^2Q of little practical value in implementing WF^2Q. Therefore, the following computation method of virtual system time is devised, which is low in implementation complexity and satisfies all important properties of WF^2Q [2].

$$V(t+\tau) = \max(V(t) + W(t, t+\tau), \min_{i \in B(t+\tau)} s_i(t+\tau)) \quad (5)$$

This scheme is called WF^2Q+ [2], which is the base scheduling algorithm of our work.

3.2 Wireless Issues

Conventional PFQ algorithms in wired networks cannot be directly applicable to wireless networks because of location-dependent channel error and temporary bursty channel error in the shared wireless channel. Location-dependent error exists because each mobile terminal suffers from different fading and interference. In wireline PFQ algorithms, service share of unbacklogged flow i is distributed to other flows. However, in wireless environments, an error-experiencing flow gives its service share to other flows for efficient link utilization and elimination of HOL (Head-Of-Line) blocking problem, even though it has packets to transmit. This type of flow is lagging behind its service share because a wireless scheduling scheme chooses one of other flows with good link conditions. Therefore, each flow can be then classified as *leading, lagging, or in-sync* by the amount of received service [5] compared to its service share in error-free environments.

The key feature of wireless fair queuing is to allow lagging flows to make up for their lags by making leading flows give up their leads according to some criterion. This is so-called *compensation scheme* or *service swapping scheme*. Most existing schemes on wireless fair queuing are based on the compensation scheme. By the compensation scheme, the scheduler guarantees long-term fairness to each flow under the assumption that an error-prone flow has sufficient time to make up for their lag after recovery of channel.

However, depending on various applications, it may be meaningless to compensate the lost lags completely. For example, in the case of audio service, let us assume that a mobile terminal experiences severe wireless channel errors for a long duration. Then, when the channel error disappears, does the corresponding scheduler have to compensate all lost lagging service (delayed audio data)? To support perfect fairness, it has to compensate all lost service of flow i. However, it seems unnecessary that flow i receives its lost lagging service generated if the packets of flow i are timely invalid. Therefore, our algorithm incorporates the maximum bound of lagging and leading service into compensation. More detailed description will be given in Section 4.

4. THE W^2F^2Q ALGORITHM

We first start with how to detect leading/lagging status of each flow, which utilizes the GBT property of WF^2Q+. Then, the compensation capability inherited from WF^2Q+ is described. To complement the weak points of the compensation capability, we incorporate three techniques: readjusting, dynamic graceful degradation, and unbacklogged leading/lagging flows. After explaining these techniques, we formally describe W^2F^2Q.

4.1 Detecting the amount of leading/lagging

We use the GBT property of WF^2Q+ to detect the amount of lagging and leading caused by wireless channel errors. Normally, in the case of wireless channel errors for flow i, the scheduler gives the service share of flow i to other flows with the good channel condition. The scheduler keeps track of the amount of flow i's leading and lagging service using the GBT property of WF^2Q+. To classify the ongoing flows by how much their leading/lagging service amount is, we introduce the following definitions.

Definition 2. A flow i is γ_i *over-lagging* if the following condition holds:

$$V(t) - s_i(t) = \frac{L}{\phi_i} + \gamma_i \quad (6)$$

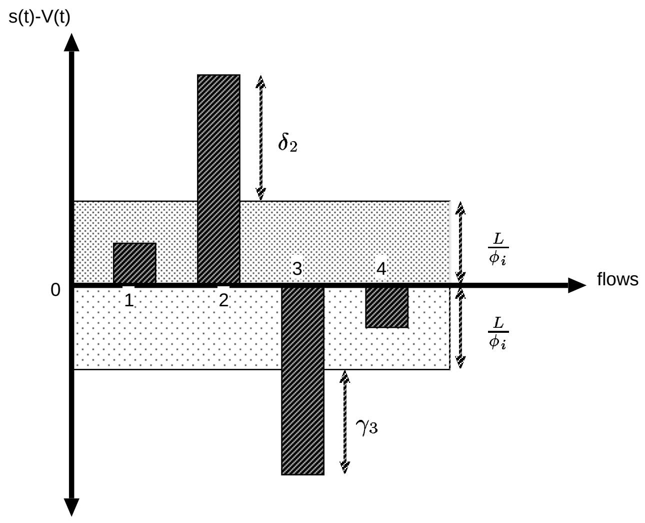


Figure 2: Detecting the amount of leading/lagging ($\phi_1 = \phi_2 = \phi_3 = \phi_4$)

Definition 3. A flow i is δ_i *over-leading* if the following condition holds:

$$s_i(t) - V(t) = \frac{L}{\phi_i} + \delta_i \tag{7}$$

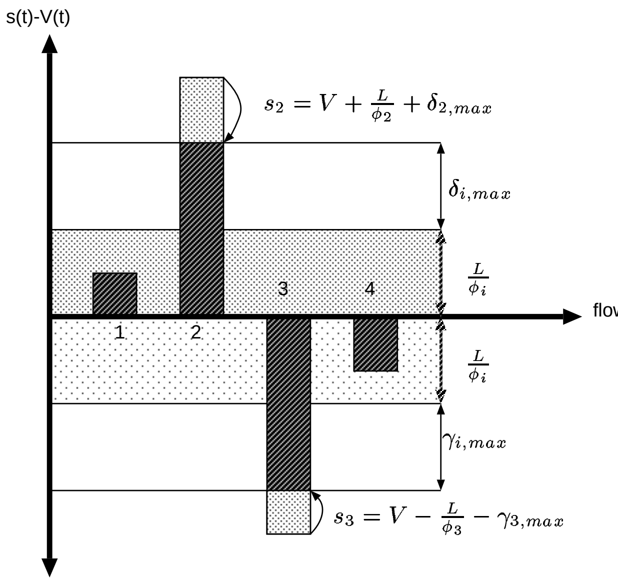
where $s_i(t)$ and $V(t)$ are the virtual start time of flow i and the virtual system time at time t, respectively. L is the fixed packet length. Without wireless errors, WF^2Q+'s GBT property makes γ_i and δ_i greater than or equal to $\frac{-2L}{\phi_i}$ and lower than or equal to 0. However, if flow i's channel is in bad state, $s_i(t)$ cannot increase, and then, breaks the GBT property temporarily. In this case, note that γ_i is positive.

Figure 2 shows the example of over-lagging (flow 3) and over-leading (flow 2) flow. The fact that the weight of all flows are same makes GBT boundary value of all flows is equal. Figure 2 shows, in case of flow 2, $s_2(t) - V(t) = \frac{L}{\phi_2} + \delta_2$. By the GBT property of WF^2Q+, it is obvious that $s_2(t)$ cannot be larger than $V(t) + \frac{L}{\phi_2}$. However, due to other flows' channel errors, it can be over-serviced. Likewise, flow 3 can be under-serviced, which makes $s_3(t) - V(t)$ go beyond the boundary of one packet size. Over-lagging and over-leading flows can be detected easily by comparing $s(t)$ and $V(t)$.

4.2 Compensation of WF^2Q+

Let us first explain that WF^2Q+ has compensation capability without wireless extensions. In WF^2Q+, the k-th packet of the unbacklogged flow i is assigned the virtual start time, $\max(V(t), f_i^{k-1}(t))$. On the other hand, the k-th packet of the flow i inherits its virtual start time by $f_i^{k-1}(t)$ when the flow i has packets to be serviced. Obviously, flow i's virtual start time will become relatively small when it experiences wireless channel errors and thereby cannot be serviced temporarily. After recovery of wireless channel, flow i will be serviced ahead of other leading and in-sync flows because of its low virtual start time. Clearly, WF^2Q+ has compensation capability and supports long-term fairness. However, this compensation property of WF^2Q+ does not take into account the possible shortcomings. Therefore, we incorporate three techniques, which will be described in the following sections.

4.3 Readjusting



s(t)-V(t)
$s_2 = V + \frac{L}{\phi_2} + \delta_{2,max}$
$\delta_{i,max}$
$\frac{L}{\phi_i}$
3
4
flows
1
2
$\frac{L}{\phi_i}$
$\gamma_{i,max}$
$s_3 = V - \frac{L}{\phi_3} - \gamma_{3,max}$

Figure 3: Example Readjusting Procedure ($\phi_1 = \phi_2 = \phi_3 = \phi_4$)

To control the amount of compensation for lost services, the readjusting technique [5, 6] is adopted in W^2F^2Q. The objective of readjusting is two-fold. For a lagging flow, readjusting restricts the amount of compensation, which can be obtained from other non-lagging flows. The scheduler does not necessarily make up for the lag fully if the lag is longer than some pre-specified limit. In the context of multimedia applications, readjusting is to discard over-delayed packet in the corresponding queue in order to prevent meaningless scheduling monopoly of the lagging flow after the recovery of the wireless channel. On the other hand, for a leading flow, readjusting bounds the quantity of concession to other flows.

Let $\delta_{i,max}$ and $\gamma_{i,max}$ denote the maximum allowable leading and lagging amount of flow i. Values are pre-specified according to the scheduling strategy. Usually, they are assigned values in proportion to the weight of flow. In addition, the scheduler has global maximum leading and lagging amount (δ and γ) to be distributed to flows by the following rules.

$$\delta_{i,max} = \delta \times \frac{\phi_i}{\sum_{j \in F} \phi_j} \tag{8}$$

$$\gamma_{i,max} = \gamma \times \frac{\phi_i}{\sum_{j \in F} \phi_j} \tag{9}$$

where F is the set of all flows. If δ and γ are large enough to absorb all lagging and leading amount caused by wireless channel errors, the readjusting is not used and the scheduler falls back to the simple compensation mode of WF^2Q+.

The following formal readjusting procedure is applied to each flow in every scheduling loop.

- For leading flow i, if $s_i - V - L/\phi_i$ is greater than $\delta_{i,max}$, then

$$s_i = V + \frac{L}{\phi_i} + \delta_{i,max} \tag{10}$$

- For leading flow i, if $V - s_i - L/\phi_i$ is greater than $\gamma_{i,max}$, then

$$s_i = V - \frac{L}{\phi_i} - \gamma_{i,max} \tag{11}$$

Figure 3 shows an example of the readjusting procedure. As in Figure 2, weights of all flows are same. In real environments, however, different reserved weight makes the boundary value of GBT property (L/ϕ_i) different for each flow i. In the case of flow 2, it is more over-leading than $\delta_{2,max}$. Therefore, the readjusting procedure reduces its over-leading amount to $\delta_{2,max}$ by reducing $s_2(t)$, which is similar to the case of over-lagging flow 3.

Generally, for delay-sensitive multimedia flows, after dropping over-delayed packets according to some delay bound Δ, newly reduced virtual start time is assigned to a new HOL packet.

Readjusting step in scheduling algorithm is performed at every scheduling loop. However, it can be performed once per several scheduling loops to reduce the computation complexity of scheduling algorithm. The dimension of γ and δ can be $bits/\phi_i$, $bits/r_i$, or $bits$ according to the method of virtual time computation, where ϕ_i and r_i is the reserved weight and bandwidth of flow i, respectively.

4.4 Dynamic Graceful Degradation

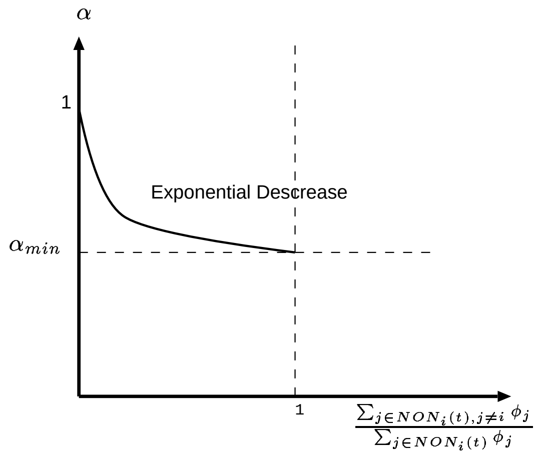


Figure 4: Determining α value

The graceful degradation technique was first addressed in CIF-Q [7]. According to CIF-Q, during any time interval while it is error-free, a leading backlogged flow should be guaranteed to receive at least a minimum fraction of its service in an error-free system. CIF-Q used the fraction parameter α, which causes a leading flow to give up at most $(1-\alpha)$ of its own leading quantity.

In WFS [6] and CIF-Q [7], there are explicit variables such as lag_i and E_i, each of which represents leading and lagging amount of each flow, respectively. Using lag_i and E_i, a leading flow that comes to get its service gives a portion of its service share to other lagging flows. In other words, the leading flow yields its service share for graceful degradation.

In W^2F^2Q, however, compensation is performed automatically as described in Section 4.2. Lagging flow i experiencing wireless channel errors monopolizes the scheduler's service after the recovery of its channel, because its s_i is smaller than those of other leading and in-sync flows and flow i has higher priority than other leading and in-sync flows. Formally, if $V - s_i(t) = L/\phi_i + \gamma_i$, flow i has lagging amount of at least γ_i. In this situation, the lagging flow gets more chance to serve its packet than other leading and in-sync flows. Leading and in-sync flow can be starved, because service is concentrated on lagging flows, which violates the important *"short-term fairness"* property of the scheduler. In W^2F^2Q, lagging flows having scheduling rights distributes its some portion to other leading and in-sync flows. In other words, lagging flows are responsible for graceful degradation in our algorithm, which is different from those of [6, 7]. In W^2F^2Q, however, if s_i of the lagging flow is within one packet size compared to V, graceful degradation is not performed. Insufficient lagging amount has to be accumulated to distribute share to non-lagging flows. The GBT property of scheduling eventually decides when to start graceful degradation. Therefore, formally, only over-lagging flows perform graceful degradation. Our strategy is that for each over-lagging flow, it distributes $(1-\alpha)$ of its own service share to non-lagging flows and gets α of its original service share [7]. Among non-lagging flows, in-sync flows and leading flows are allocated the lagging flow's yielded service share computed by the current state of non-lagging flows.

If $\alpha = 1$ [7], a lagging flow does not yield its service share to non-lagging flows. In [7], α is static, but α is dynamically adjusted in W^2F^2Q. The value of α should reflect the current number of lagging flows. Intuitively, the more lagging flows there are, the more service should be yielded to other flows. The boundary conditions in determining the value of α are : it is one when the number of non-lagging flows is zero. Also, α has a minimum value (α_{min}) irrespective of the number of non-lagging flows. We adopt an exponential decreasing function for α to make graceful degradation more sensitive to the number of non-lagging flows. When an exponential decrease function is employed, α is computed by

$$\alpha = Ke^{1-x} + M \tag{12}$$

where x represents the ratio of the number of non-lagging flows to the number of non-lagging and the corresponding over-lagging flow.

Solving K and M in Equation 12 from the boundary conditions,

$$\alpha = \frac{1-\alpha_{min}}{e-1}e^{1-x} + \frac{e\alpha_{min}-1}{e-1} \quad (13)$$

$$x = \frac{\sum_{j \notin NON_i(t), j \neq i} \phi_j}{\sum_{j \in NON_i(t)} \phi_j} \quad (14)$$

where $NON_i(t)$ is the union set of non-lagging flows and the corresponding lagging flow i at time t. For example, when there are four flows (flow 1, 2, 3, 4), let us assume that flow 1 and 3 are lagging, flow 2 is in-sync and flow 4 is leading. Then, $NON_1(t)$ is flow 1, 2, and 4. From the boundary conditions, $(0, 1)$ and $(1, \alpha_{min})$ should satisfy Equation 13. Equation 13 represents that α is 1 (the lagging flow does not perform graceful degradation) as there are no non-lagging flows. Also, α approaches α_{min} as the number of non-lagging flows becomes large (Figure 4). The x-axis of Figure 4, $\frac{\sum_{j \in NON_i(t), j \neq i} \phi_j}{\sum_{j \in NON_i(t)} \phi_j}$, represents how many non-lagging flows exists compared to the corresponding lagging flow i.

When $(1-\alpha)$ amount of lagging service is moved to leading and in-sync flows, weighted round robin (WRR) can be utilized hierarchically for service distribution to each non-leading flows. At the highest level, service share of lagging flows will be given to leading and in-sync flows with the weight of $\sum_{k \in L(t)} \phi_k / \sum_{j \in NL(t)} \phi_j$ and $\sum_{k \in I(t)} \phi_k / \sum_{j \in NL(t)} \phi_j$, where $L(t)$ is the set of all leading flows at time t, $I(t)$ is the set of all in-sync flows at time t, and $NL(t) = L(t) \cup I(t)$. Leading flow i receives service in proportion to its normalized leading amount (δ_i/ϕ_i). The service share of in-sync flows are allocated based on rate weight (ϕ_i).

4.5 Unbacklogging in Leading or Lagging State

Suppose that flow i has just one packet (p) to transmit now. In ordinary PFQ scheduling discipline, p has just one scheduling chance. If five scheduling chances comes to flow i and there is no packet to be served after p until that time, only one scheduling chance will be given to flow i for p and four chances of scheduling will be distributed to other backlogged flows. In wireless fair queuing, however, p can not be transmitted to the link if wireless channel error exists in the link, even though five scheduling opportunities is received by flow i. As a result, flow i will have lagging amount of four packets (four scheduling chances) services. Conceptually, the fact that flow i is lagging by k amount of service does not mean that flow i has k packets to send. This phenomenon can be observed when flow i is unbacklogged in a lagging state of service. We face the problem : *How does scheduler handle remaining lagging amount of service when a lagging flow is unbacklogged?*

Let us think about the solution of a leading flow's case. When a leading flow is unbacklogged, the first packet (q) of next backlogged period in flow i can inherit leading status from previous backlogged packets because the start time of flow i, s_i^q is computed by $\max(V, f_i^q)$. In our scheduling, two alternative procedures are possible to handle this situation. First, s_i^q is reduced to eliminate the leading amount of the previous packet. The other action is to proceed the scheduling without changing s_i^q. The first strategy can make the packet q's queuing delay smaller than the second one because q does not have to be responsible for the leading service of a precedented packet. In the first solution, however, flow i results in stealing other flow's service and hiding itself, which can break the long-term fairness. The second solution leads to the opposite result.

Table 1: Notations for algorithm description

Notation	Description
$V(t)$	the virtual system time at time t
$s_i(t)$	the virtual start time for the flow i at time t
$f_i(t)$	the virtual finish time for the flow i at time t
L	the length of the arrived and transmitted packet
ϕ_i	service weight for flow i
$c\_status(i)$	true if flow i's channel state is good
$B(t)$	the set of backlogged flows at time t

An unbacklogged lagging flow is different from an unbacklogged leading flow. As shown in the example above, an unbacklogged lagging flow can be generated when a lagging flow does not have as much demand as amount of lagging. In the example at the start of this section, four lagging units do not always mean that the lagging flow used to have demand of transmitting four packets. Thus, it is natural to distribute remaining amount of lagging of flow i to other flows. We setup the following strategy of distributing remaining lag to other flows as in [7].

$$s_j = s_j - \frac{\phi_i \times (V - s_i - L/\phi_i)}{\sum_{k \in B} \phi_k} \quad (15)$$

where i is the corresponding unbacklogged lagging flow and j includes all flows except for flow i in B. The equation above is different from that of CIF-Q in that CIF-Q changes the explicit lagging variable (lag_i).

4.6 Algorithm Description

Here, we present the formal W^2F^2Q algorithm. The notations for algorithm description are shown in Table 1.

Algorithm 1 *Enqueue*(i, P), i : flow id, P : newly arriving packet

```
1: if $\hat{Q}_i(t) = \emptyset$ then
2:     $\hat{Q}_i \leftarrow P$;
3:     $s_i(t) \leftarrow max(f_i(t), V(t))$;
4:     $f_i(t) \leftarrow s_i(t) + L/\phi_i$;
5:     $V \leftarrow max(\min_{j \in B(t)} s_j, V)$;
6: end if
7: insertto(P,$\hat{Q}_i$);
8: return;
```

The Enqueue operation of W^2F^2Q is described in Algorithm 1. Algorithm 1 operates when a new packet (P) arrives at its queue. The scheduler decides whether P's corresponding flow (i) is active (or equivalently, backlogged) or inactive. If flow i is inactive, P becomes head-of-line packet of its queue and the start and finish tag are assigned (lines 3-4). Also, the virtual system time is updated according to the update policy of WF^2Q+ (lines 3-5 in Algorithm 1).

The Dequeue operation of W^2F^2Q is described in Algorithm 2. At first, the scheduler readjusts every queue to

Algorithm 2 *Dequeue*()

```
1: Readjust();
2: $flow = \emptyset$;
3: $lag\_minflow = \emptyset$,$lag\_minF = \emptyset$;
4: $elig\_minflow = \emptyset$,$elig\_minF = \emptyset$;
5: $nonlag\_minflow = \emptyset$,$nonlag\_minF = \emptyset$;
6: $lag\_minflow = i$ such that $f_i(t) = \max_{j \in E} f_j(t)$ ,$c\_status(i) = GOOD$ and $s_i(t) - V < -L/\phi_i$;
7: $lag\_minF = f_i(t)$;
8: $elig\_minflow = i$ such that $f_i(t) = \max_{j \in E} f_j(t)$ ,$c\_status(i) = GOOD$ and $s_i(t) - V \leq 0$;
9: $elig\_minF = f_i(t)$;
10: $nonlag\_minflow = i$ such that $f_i(t) = \max_{j \in E} f_j(t)$ ,$c\_status(i) = GOOD$ and $s_i(t) > V$;
11: $nonlag\_minF = f_i(t)$;
12: if $elig\_minflow = \emptyset$ and $lag\_minflow = \emptyset$ and $nonlag\_minflow = \emptyset$ then
13:     return NULL;
14: end if
15: $flow = \min(elig\_minflow, lag\_minflow, nonlag\_minflow)$ ;
16: if $flow == lag\_minflow$ then
17:     Do graceful degradation;
18: end if
19: deletefrom(P,$\hat{Q}_{flow}$);
20: nextPacket $\leftarrow \hat{Q}_{flow}$;
21: if $nextPacket \neq \emptyset$ then
22:     $s_i(t) \leftarrow max(f_i(t), V)$;
23:     $f_i(t) \leftarrow s_i(t) + L/\phi_i$;
24: end if
25: $V = \max(\min_{j \in B} s_j(t), (V + L/\sum_{k \in B} \phi_k))$;
26: transmit P;
```

limit leading and lagging amount of each flow (line 1). After readjusting, the W^2F^2Q scheduler chooses flow i with minimum finish time among all lagging flows on the condition that the flow i's channel status is good (lines 6-7). After selecting the lagging flow with minimum finish time, the scheduler performs the eligibility test in order to select the eligible flow with minimum finish time (lines 8-9). And then, the non-lagging flow having minimum finish time is chosen (lines 10-11). The selection of eligible and non-lagging flows is for the case that there is no lagging flow in the scheduler. The reason why the scheduler does not transmit the corresponding packet (P) right after selecting the lagging flow with minimum finish time is that the scheduler's choice of packet is based on the virtual finish time. Accordingly, although P belongs to a lagging flow, it does not always mean that the finish time of P is minimum. It should be noted that the scheduler chooses the packet to transmit based on leading, lagging, and in-sync, because the W^2F^2Q scheduler is based on WF^2Q+. Lagging flows are selected separately for graceful degradation. If the scheduler cannot discover a flow to be serviced, it just returns NULL (lines 12-13). This happens when every flow is unbacklogged or all channel status of backlogged flows is bad. In most cases, the lagging flow receives service if flows' weight is not quite different from each other and the packet size is fixed. In case without lagging flow, the eligible flow gets the corresponding service (line 15). When the flow to be serviced is lagging, it has to yield some portion of its own service share to non-lagging flows for short-term fairness. The objective of graceful degradation is to reduce lagging flow's influence on in-sync and leading flows. After serving a packet, the start and finish tags are assigned to the next packet (lines 22-23). The virtual system time will be updated by the maximum of the minimum start time of all backlogged flows and the past virtual system time plus the service amount (line 25).

4.7 Algorithm Complexity

Scheduling algorithms requiring virtual time such as WFQ and WF^2Q have very complex time-stamp computation complexity. Regarding the well-known WFQ, it has the priority queue maintenance cost of $O(\log N)$ and the virtual system time update cost of $O(N)$, where N is the number of flows in the queue. WF^2Q is more complex to maintain the eligible packets. However, the efficient virtual system update is performed in WF^2Q+, which is not ideal GPS emulation but packetized one. All complexities WF^2Q+ has is to maintain cost of eligible packets and priority queue in order to sort the eligible packets in the order of finish time, which is $O(\log N)$.

In W^2F^2Q, the computation complexity can increase due to readjusting, graceful degradation and investigation of flow's channel status. Investigating the channel state of each flow has $O(N)$ complexity. Graceful degradation is of complexity $O(N)$ because of WRR-based scheduling. Readjusting corresponds to just one flow to be serviced at the server (i.e., $O(1)$). Therefore, W^2F^2Q scheduling complexity is $O(N)$.

$O(N)$ may be too high for schedulers in the backbone network, where N is very large. However, the scheduler in this paper will be utilized only at the access network with small N.

5. THEORETICAL ANALYSIS

In this section, theoretical analysis of W^2F^2Q is given. In the first place, it will be proved that long-term throughput property of W^2F^2Q is showed compared to that of WF^2Q+. And then, the delay that newly arriving packet experiences in unbacklogged queue is analyzed. In this analysis, it is assumed that α is fixed for brevity. We will not describe the complete proofs of theorems due to the limitation of paper. The complete proofs are shown in [10].

5.1 Throughput Analysis

LEMMA 1. *If it is assumed that n flows exist in W^2F^2Q scheduler, any packet p on an error-free flow i completes its service in W^2F^2Q before time $\delta/C + t_{wf^2q}$, where C is link capacity ,δ is global maximum lagging amount of W^2F^2Q and t_{wf^2q} is service completion time of WF^2Q+ when there is no wireless channel error.*

THEOREM 1. *For any packet p of a error-free flow i in , its maximum service completion time (SC_{i,w^2f^2q}) compared to error-free WF^2Q+ when there is no wireless channel error is bounded by*

$$SC_{i,w^2f^2q} < SC_{i,wf^2q} + \frac{\delta}{C\alpha} \qquad (16)$$

,where SC_{i,wf^2q} is the service completion time of packet p in error-free WF^2Q+ scheduler when there is no wireless channel error.

THEOREM 2. *For a error-free flow i, let $S_i^{w^2f^2q}(0,t)$ and $S_i^{wf^2q}(0,t)$ denote the aggregate service received by flow i in the interval $[0,t]$ in the W^2F^2Q and error-free WF^2Q+, then the following inequality holds:*

$$S_i^{w^2f^2q}(0, t + \frac{\delta}{C\alpha}) > S_i^{w^2f^2q}(0,t) \tag{17}$$

Theorem 2 says that long-term throughput guarantee property of W^2F^2Q. In Theorem 2, error-free flow has only to wait no $\frac{\delta}{C\alpha}$ more than in order to be equivalent to WF^2Q+ service discipline.

5.2 Delay Analysis

THEOREM 3. *For any non-lagging flow i, its maximum new queuing delay $QD^i_{w^2f^2q}$ at time t compared to maximum queuing delay in WF^2Q+ scheduler under the assumption of fixed α is bounded by the following equation.*

$$\begin{cases} \dfrac{pktSize}{C(1-\alpha)\frac{\sum_{j\in NL(t)}\phi_j}{\sum_{k\in LEAD(t)}\phi_k}\frac{\frac{\delta_i}{\phi_i}}{\sum_{j\in LEAD(t)}\frac{\delta_j}{\phi_j}}} + \lambda & : \quad condition\ a \\ \frac{\delta}{C\alpha} + QD^i_{wf^2q} + \frac{\gamma_i(\sum_{j\in F, j\neq i}\phi_j)}{C\phi_i} & : \quad Otherwise \end{cases} \tag{18}$$

where "condition a" corresponds to when $\frac{\sum_{j\in LAG(t)}\gamma_j}{C\alpha} > \frac{\sum_{j\notin LAG(t)}\delta_j}{C(1-\alpha))}$. Theorem 3 shows the new queuing delay of non-lagging flow i. There are two cases for the new queuing delay computation. Under the assumption that ideal GPS scheduling is used for graceful degradation and α is fixed, if the sum of lagging amount of lagging flows is enough to serve leading flows by distributing $(1-\alpha)$ service share to leading flows, the HOL packet of flow i can be served by borrowing service from lagging flows. Otherwise, the HOL packet of flow i waits until the lagging sum of lagging flows and its own leading amount of service are eliminated. After that, it becomes in-sync and waits for its service by WF^2Q+ scheduling. λ represents the error-term for GPS scheduling in graceful degradation when the lagging flow distributes its $(1-\alpha)$ portion to non-lagging flows.

6. SIMULATION EXPERIMENTS

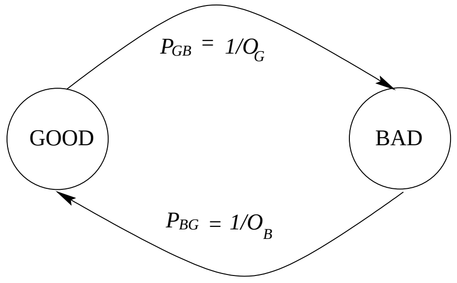


Figure 5: Two-state Markov model

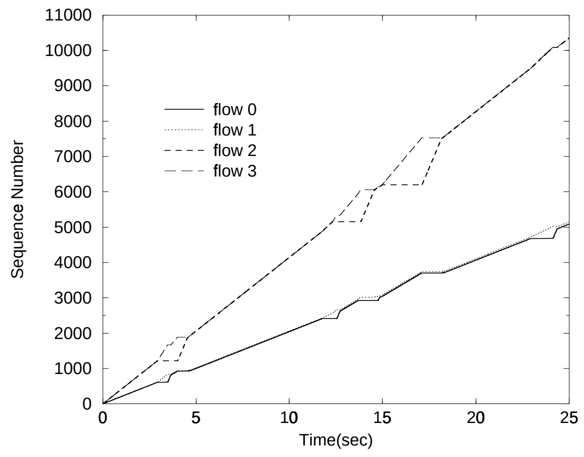


Figure 6: Automatic Compensation without readjusting and graceful degradation (WF^2Q+)

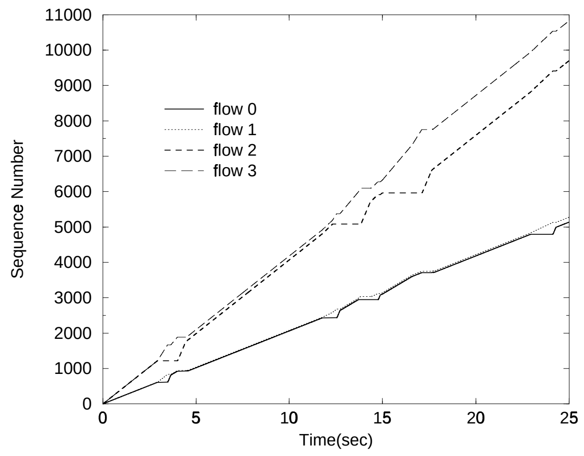


Figure 7: Readjusting

Simulation experiments are carried out to show the effectiveness of readjusting and dynamic graceful degradation. We model burst packet losses using two-state discrete Markov chain as in Figure 5. When a channel is in the bad state, packets sent in this period are lost. The Markov property of each state enables us to express the transition probability between two states by the mean state occupancy time. The transition probabilities, P_{GB} and P_{BG}, are defined by the inverse of the mean state occupancy time O_G and O_B, respectively. In this paper, channel errors are applied to only two flows (flow 0 and 2). Here, O_B and O_G of flow 0 are set to 1 sec and 5 sec to illustrate the effect of the schemes in Section 4, respectively. For the same purpose, O_B and O_G of flow 2 are set to 2 sec and 5 sec, respectively. Also, it is assumed that the scheduler knows the channel status of every flow perfectly. For the purposes of comparison, the same error pattern is applied to three experiments in Figure 6, 7, and 8. Four flows are present, and flow 2 and 3 have twice as much weight as flow 0 and 1. We assume that

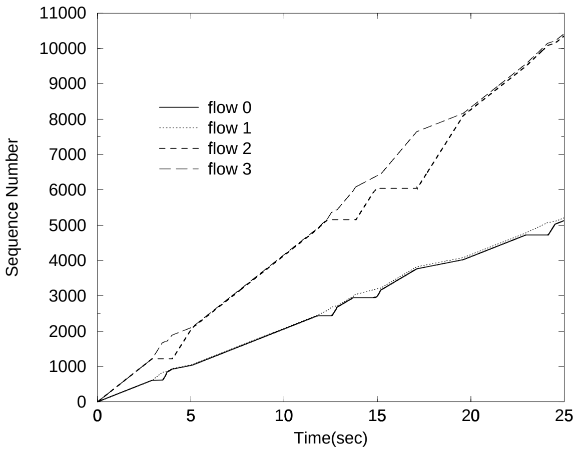


Figure 8: Dynamic Graceful Degradation

link bandwidth is 2 Mbps and that flows are continuously backlogged. A Poisson process of cross background traffic with average 1 Mbps is also generated in the experiments.

Figure 6 shows the automatic compensation ability of WF^2Q+ without wireless extensions. In Figure 6, flow 2 experiences wireless channel errors from time 3 to 5, from 13 to 16, and from 16 to 19 during which its service share is distributed to other flows. However, after the channel recovery, it monopolizes the service chance and makes up its lag. Long-term fairness is maintained within a short time. But, short-term fairness during wireless channel error is broken.

The simulation result which highlights the effectiveness of the readjusting scheme is shown in Figure 7. Due to readjusting, a few packet from the queue front are dropped. And the newly increased start time is assigned to a new HOL packet. Flow 2 proceeds below flow 1 after 15 sec, because packets with excessive lag are eliminated from the front of queue.

In Figures 6 and 7, the slope of flow 2 is sharp and that of flow 3 is zero during right after the recovery of wireless channel errors. Using graceful degradation, it is shown that the slope of flow 2 is less sharp during each error period. Also, during that time flow 3 can get some service share from flow 2, which guarantees short-term fairness while recovery. Figure 8 shows behavior of four flows when graceful degradation is applied.

7. CONCLUSION

In this paper, we propose the new wireless packet scheduling method, Wireless Worst-case Fair Weighted Fair Queuing (W^2F^2Q), based on WF^2Q which emulates the ideal GPS algorithm. W^2F^2Q uses the globally bounded timestamp (GBT) property of WF^2Q+ to detect leading and lagging status of each flow. This detection function is utilized at readjusting and graceful degradation process. The proposed W^2F^2Q seeks to achieve fairness among leading and lagging flows. The proposed algorithm is of manageable complexity $O(n)$, where n is the number of ongoing flows. Theoretical analysis verifies that the proposed algorithm guarantees long-term fairness as well as queuing delay bound. Simulation experiments are carried out to show the effectiveness of readjusting and graceful degradation.

8. REFERENCES

[1] J. Bennett and H. Zhang. wf^2q: Worst-case fair weighted fair queuing. In *IEEE INFOCOM '96*, March 1996.

[2] J. Bennett and H. Zhang. Hierarchical packet fair queuing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, October 1997.

[3] S. Golestani. A self-clocked fair queuing scheme for broadband applications. In *IEEE INFOCOM '94*, April 1996.

[4] L. Kleinrock. *Queuing Systems Vol. 2 : Computer Applications.* Wiley, New York, 1976.

[5] S. Lu, V. Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. In *ACM SIGCOMM '97*, September 1997.

[6] S. Lu, T. Nandagopal, and V. Bharghavan. A wireless fair service algorithm for packet cellular networks. In *ACM MOBICOM '98*, October 1998.

[7] T. S. E. Ng, I. Stoica, and H. Zhang. Packet fair queuing algorithms for wireless networks with location-dependent errors. In *IEEE INFOCOM '98*, June 1993.

[8] 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*, 1(3):344–357, June 1993.

[9] D. C. Stephens, J. C. R. Bennett, and H. Zhang. Implementing scheduling algorithms in high-speed networks. *IEEE JSAC*, 17(3):1145–1159, June 1999.

[10] Y. Yi, Y. Seok, and Y. Choi. w^2f^2q : Efficient packet fair queuing in wireless packet networks. In *Technical Reports.* `http://mmlab.snu.ac.kr/~yiyung/w2f2q.ps`, June 2000.