Wireless Packet Scheduling for Two-state Link Models

Yaxin Cao and Victor O. K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong

Abstract— **Packet scheduling is key to the provision of Quality of Service (QoS) differentiation and guarantees in a wireless network. Unlike its wireline counterpart, wireless communication poses special problems such as time-varying link capacity and location-dependent errors. These special problems make designing efficient and effective scheduling algorithms for wireless networks very challenging. Although many wireless scheduling algorithms have been proposed in recent years, some issues remain unresolved. This paper introduces a new wireless scheduling algorithm called *BGFS-EBA (bandwidth-guaranteed fair scheduling with effective excess bandwidth allocation)*, which addresses these issues. It is shown that BGFS-EBA distributes excess bandwidth effectively, strikes a balance between effort-fair and outcome-fair, and provides delay bound for error-free flows and transmission effort guarantees for error-prone flows. The new algorithm is compared with some recent wireless scheduling algorithms.**

I. Introduction and Motivation

As a vital component of QoS support, packet scheduling algorithms are very important for both wired and wireless data networks. However, to design a good wireless scheduling algorithm is more challenging, because wireless links are time-varying and location-dependent. Recently, new packet scheduling algorithms which account for the special characteristics of the wireless environment have emerged. [5] surveyed many representative algorithms and discussed their pros and cons. The basic principle of these algorithms is that the scheduler defers transmissions of flows whose links are in the error state and compensates those flows when the links recover.

In addition to algorithms discussed in [5], ELFS (Effort-limited Fair Scheduling) [3] and CS-WFQ (Channel-state Independent Wireless Fair Queueing) [4] point out the tradeoff between effort-fair and outcome-fair. Effort-fair means each flow should receive transmission capacity in proportion to its assigned rate. Outcome-fair means the achieved goodput of each flow should be in proportion to its assigned rate. ELFS and CS-WFQ try to achieve outcome-fair by increasing the fair queueing weight of the flows with larger error rates. To avoid reducing the overall channel efficiency excessively, an upper bound for the weight is set. The algorithms proposed are too simplistic and suffer from some fundamental shortcomings. First, they assume that a link's error-rate does not change over time. Thus, they do not exploit the benefit of improving bandwidth efficiency by swapping service opportunities. Second, increasing the weights for flows with high error-rate links directly reduces the service effort other flows receives irrespective of their rate guarantees or link status. Therefore, even a well-behaving flow with error-free links will not get its fair share of service.

Among the above algorithms, CIFQ (Channel-condition Independent Packet Fair Queueing) [1] and WFS (Wireless Fair Service) [2] are the most recent and most sophisticated. Since our proposed scheduling algorithm is designed to address the deficiencies of CIFQ and WFS, we now focus on such deficiencies.

First, as in most other wireless scheduling algorithms, they assume that each wireless link has two states, *error* and *error-free*. However, in reality the capacity of a wireless link does not jump between zero capacity and full capacity. Realistically, each state should be associated with a range of error probability. Furthermore, since 100% accurate link state prediction/estimation is not achievable in practice and the scheduling decisions are based on the scheduler's perception of the links, such estimation error probability should be accounted for.

Second, in WFS and CIFQ, a flow never transmits in a *bad* state. There are two undesirable consequences of such a policy. Firstly, flows with inferior average link qualities may be starved. Secondly, such a policy increases packet delay of flows experiencing long *bad* states. Packets with timing requirements may miss their deadlines and become useless when the link recovers from error. However, if a flow is very important, e.g. a general's commands in a battle field, we would like to guarantee certain throughput to this flow even if its link is in a *bad* state. However, when transmitting in a *bad* state, a regular (uncoded or weakly coded) packet will be split into several packets with enhanced error-correction coding protection. Compared with transmitting a regular packet in a *bad* state with high error probability, this will improve the bandwidth efficiency, because the bandwidth wastage caused by higher coding overhead is usually much smaller than that caused by higher packet error probability.

Third, since WFS and CIFQ are based on fair queueing, which in turn emulates GPS (Generalized Processor Sharing) [9], the excess bandwidth due to some flows being unbacklogged is distributed among all the flows in proportion to their fair queueing weights. However, we argue that such emulation of GPS-type of fair distribution is not necessary and effective in wireless scheduling. In situations where some flows may have achieved less goodput than its target share because of worse link quality, while some other flows may have achieved enough goodput to guarantee its QoS requirements, we believe excess bandwidth should be distributed to the former flows first, as long as the latter's QoS guarantees are not violated. In an error-free system, it has been proven [8] that as long as a flow's service received in any backlogged period can be guaranteed such that

$$W_i(t_1, t_2) \geq r_i(t_2 - t_1 - \theta_i) \quad (1)$$

where $W_i(t_1, t_2)$ is the effective service received during (t_1, t_2),

This research is supported in part by the NSFC/RGC Joint Research Scheme under Grant No. N_HKU709/00.

0-7803-7632-3/02/$17.00 ©2002 IEEE

r_i is the allocated rate of the flow, t_1 is the start time of a backlogged period, and θ_i is a non-negative constant called *latency*, for burst-constrained traffic with average rate less than or equal to r_i, the packet delay can be bounded. In addition, such a bound, independent of other flows' behavior, is the same delay bound that a GPS-based fair queueing algorithm with the same assigned rate r_i can provide. Since the availability of excess bandwidth can not be guaranteed by the network, the guaranteed worst case delay bound to a flow can not be improved by providing it unassured excess bandwidth. Therefore, in our wireless scheduling algorithm, for each flow i we try to achieve an effective service curve $r_i(t-\theta_i)$ instead of the GPS service curve in WFS and CIFQ. In this way more excess bandwidth will be available for compensating flows with bad links.

Fourth, the time-stamping schemes of WFS and CIFQ are impractical for uplink transmissions. They either require that each uplink flow monitor the service progress of all other flows, or the scheduler or uplink flows constantly send information of system virtual time or packet finish times. Neither can be realized efficiently.

In this paper we propose a new wireless scheduling algorithm which accounts for all the above issues and is capable of providing QoS guarantees.

II. System Model

We consider centralized scheduling in a wireless network, where a base station is responsible for scheduling all the packet transmissions in the network. There are two types of communication links in the system, error-free and error-prone. An error-free link is just like a wireline link where packet error probability is zero. As in other scheduling algorithms, a two-state Markov model is used for an error-prone wireless link. The scheduler only differentiates the quality of an error-prone link between two states, *good* or *bad*. The link states may be incorrectly estimated with certain probability. The links are independent of each other and their average link quality may be different. Time is divided into fixed-length slots. In each time slot, only one flow can transmit and a fixed number of bits can be transmitted. All regular data packets have the same length of one time slot. Transmissions of a regular packet in a *good* state have much higher probability of success than in a *bad* state. A regular packet can be split into m ($m = 2$ or 3 should be sufficient) *low rate packets* with more coding protection. A *low rate packet* has low transmission error probability in a *bad* state. For flows requiring reliable delivery, a transmitted packet remains at the head of transmission queue until it is successfully transmitted. We assume instant feedback on whether a transmission is successful.

III. BGFS-EBA

A. Algorithmic Details

Here we present our proposed wireless scheduling algorithm, namely, Bandwidth-guaranteed Fair Scheduling with effective excess bandwidth allocation (BGFS-EBA).

Each admitted flow i in the system has a target rate r_i, which is the average goodput rate desired. The sum of all the target rates should not exceed the total available bandwidth R. The scheduler keeps track of the goodput g_i each flow i has achieved.

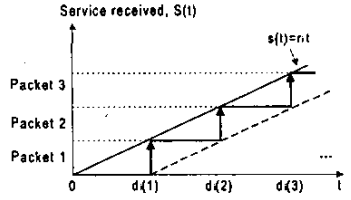

Fig. 1. Deadline calculation of a virtual flow

Instead of trying to approximate the GPS service curves as in CIFQ and WFS, we try to achieve the goodput target for each flow in any backlog period as shown in (1) with $g_i(t_1,t_2)$ replacing $W_i(t_1,t_2)$. As we have already discussed previously, if such a condition can be satisfied, the delay bound of the flow can be guaranteed. Using the same terms as in CIFQ, a flow is considered as *leading*, *lagging*, or *in sync* if its achieved goodput during the current backlog period is larger than, smaller than, or the same as its target share, i.e. $r_i t$. Note that although we use the same terms as in CIFQ, the reference system we are using is different from CIFQ. CIFQ compares the service received to the service a flow would receive in an error-free SFQ system, which is dependent on the traffic load. Our definitions are based on comparison with the load-independent function $r_i t$.

The major philosophical differences between BGFS-EBA and CIFQ or WFS are the following. First, we believe the network should be able to guarantee minimum transmission bandwidth for some flows regardless of their link quality. Second, we believe flows with better average link quality should not be over-provisioned with excess bandwidth as in WFS and CIFQ. The excess bandwidth resulting from some flows not using up its share should be allocated to the lagging flows.

Each data flow has its own queue. When a packet arrives, it is simply put at the end of the corresponding flow's queue. No time-stamping is performed.

To decide which packet to transmit next, the scheduling process is performed in two phases. In the first phase, the scheduling decision is made on an idealistic full-load error-free system. Besides the real data flows in the network, the scheduler maintains a dummy flow which does not actually have packets to send but is only used to fill up the bandwidth. Suppose there are $n-1$ real data flows, each having a target rate r_i, and the total available bandwidth is R. Then the dummy flow's rate will be $R - \sum_{i=1}^{n-1} r_i$. The flows in the idealistic system are called *virtual flows*. In the idealistic system, all the virtual flows including the dummy flow are assumed to be continuously backlogged. The virtual flows' imaginary packets, which have the same fixed size as the real packets, are called *virtual packets*. The virtual packets are assigned deadlines such that if all the virtual packets of a virtual flow i are served before their deadlines, then at each virtual packet j's deadline $d_i(j)$ the service received by flow i is no less than $r_i \cdot d_i(j)$. The deadline calculation is further illustrated in Fig. 1. The arrows in the figure represent the deadlines. The service curve $s(t) = r_i t$ represents the service received by the flow if all the virtual packets depart at their deadlines. Therefore, the deadline of a virtual packet is the latest time that a virtual packet should depart for the flow to catch up with the service curve.

In the first phase, the scheduler always schedules the virtual packet with the earliest deadline. The scheduler then decides

in the second phase whether a real packet of the corresponding data flow should be sent. Since the virtual flows are always backlogged and the packet size is fixed, for flow i, the deadline of its virtual packet $j+1$ can be derived from the deadline of its virtual packet j as $d_i(j+1) = d_i(j) + l/r_i$, where l is the packet size and r_i is the target rate of flow i. Therefore, for each virtual flow i the scheduler only needs to maintain one deadline, d_i, before which the HOL (Head of the Line) virtual packet should be served. After a virtual flow's HOL packet is scheduled, no matter which real flow's packet receives actual service in the second phase, the deadline of the virtual flow is updated as $d_i = d_i + l/r_i$. Since we have $\sum_{i=1}^{n} r_i = R$ (including the dummy flow), using such a deadline assignment and scheduling policy, it is easily shown that it is a schedulable system [6], that is, all the virtual packets can meet their deadlines.

In the second phase, to determine how much a flow i is leading and lagging its target rate, the scheduler keeps track of a parameter $G_i(t)$ called the *normalized goodput gap*,

$$G_i(t) = [g_i(t) - r_i t]/(r_i t) \qquad (2)$$

where $g_i(t)$ is the goodput achieved by flow i up to t in the current backlog period. Note that $G_i(t)$ is normalized by the target goodput $r_i t$ and it is a fraction which represents how much a flow is leading and lagging compared with its target goodput. g_i and G_i are reset to zero at the beginning of each flow's backlog period.

Fig. 2 shows the complete scheduling operation. The first three blocks are the operations performed in the first phase as described above. When a virtual flow is picked and its real flow is backlogged, normally, if the flow's link is in a *good* state its packet will be transmitted; otherwise, it will give up the current service opportunity to some other flow with a *good* link. However, there are some exceptional cases.

A threshold th_i is set for each flow, where $-1 \le th_i \le 0$. A flow with a bad link may still transmit if its gap falls below its threshold or it can not find any other flow with a good link. Setting such a threshold guarantees a minimum transmission bandwidth for the flows. Whenever a flow lags behind its target goodput substantially, the flow will start to transmit even in a *bad* state. In this way, the flows experiencing long duration of bad link state will not be totally deprived of transmission. The smaller the threshold, the smaller the guaranteed transmission bandwidth. Although this may cause lower overall bandwidth efficiency in terms of total goodput, it is necessary for the scheduler to have the ability to trade off between bandwidth efficiency and link-state-independent bandwidth guarantees.

When it is the turn for a backlogged real flow i to transmit according to the idealistic system and its link is in a *good* state, it will give up its service opportunity to some other flow if all of the following conditions are satisfied.

1. $g_i > 0$, i.e. it is leading.
2. $g_i + l_i(HOL) \ge r_i(d_i - t_i)$ where t_i is the start time of the current backlog period of flow i, d_i is the current (updated in phase one already) deadline of the corresponding virtual flow.
3. There exists at least one flow with negative gap and a good link or one flow whose gap is below its threshold.

The second condition, which corresponds to the decision diamond marked with * in Fig. 2, stipulates that giving up the current service share will not jeopardize flow i's own goodput guarantee assuming that its next transmission will be successful. Since flow i will take its turn again in the idealistic system before d_i, if the transmission is successful, flow i's goodput till d_i will be $g_i + l_i(HOL)$, where $l_i(HOL)$ is the length of the information bits in its HOL packet. The inequality ensures that by d_i flow i's goodput will still stay above its target goodput. Since the flow is leading, it must have received excess bandwidth or some other flow's bandwidth before. To compensate other lagging flows it should give up its lead. Again, the reason here is that as long as we can meet the flow's target rate, it is receiving its fair share.

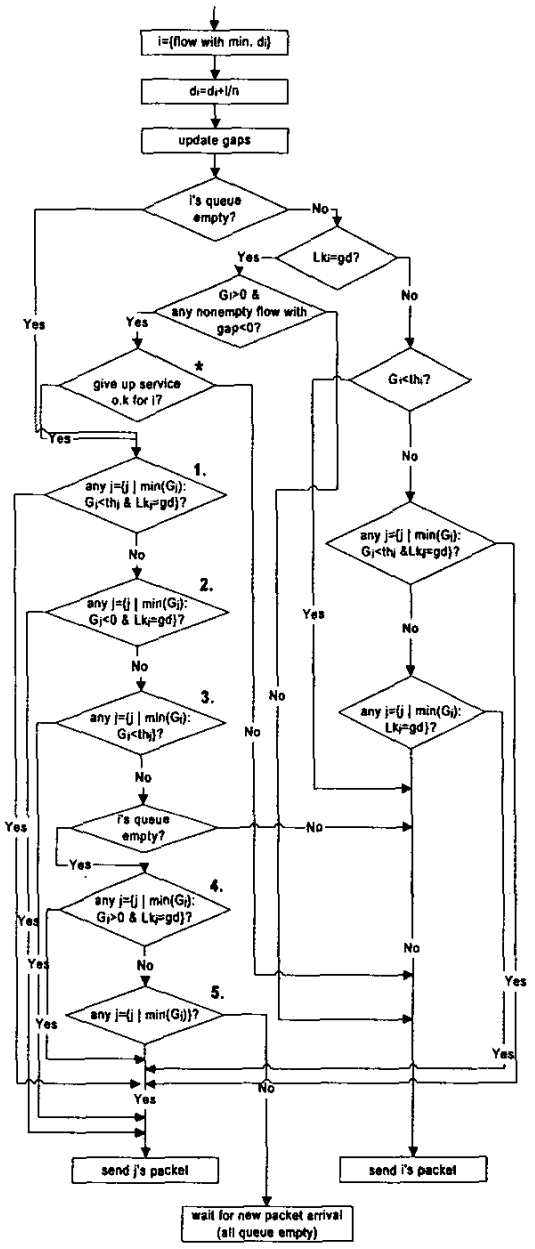

Fig. 2. Flow chart of the scheduling operations

Whenever a virtual flow is picked but its real packet queue is empty, such service opportunity represents the excess bandwidth not being fully used by the flow. The dummy flow never has any packet to transmit, therefore, when it is scheduled, the service opportunity also represents excess bandwidth. The scheduler searches for a nonempty flow with the smallest gap among flows in the following ordered sets to receive such excess bandwidth.

1. Any flow with a good link and whose gap is below its threshold.
2. Any flow with a negative gap and a good link.
3. Any flow whose gap is below its threshold.
4. Any flow with a good link.
5. All nonempty flows.

If none of the above flows exists, i.e., all the flow queues are empty; wait for a new packet arrival. Note that any flow with a negative gap, especially a flow whose gap has fallen below its threshold, has precedence in receiving excess bandwidth. The extra transmission effort allocated to the lagging flows is aimed at offsetting the adverse effect of bad link quality.

The process of searching for a flow to receive the service opportunity given up by a leading flow follows the first two steps above. If none of the flows qualifies, the leading flow will redeem the service opportunity and transmit a packet of its own.

One point not specified in the flow chart is that to improve bandwidth efficiency, only low rate packets will be transmitted in a *bad* state. Therefore, when the scheduler decides to transmit in a *bad* state, it first checks the flow's HOL packet. If it is a regular packet, it splits it into m low rate packets, and transmits one of them. The rest are inserted at the HOL of the flow queue.

Note that since no time-stamping for the real packets is used and the deadlines of virtual flows can be iteratively calculated by the scheduler, BGFS-EBA does not require time-stamping which is impractical. An uplink flow only needs to notify the scheduler when it becomes unbacklogged or backlogged, which is necessary for any scheduling algorithm.

B. Analytical Observations

Minimum Bandwidth and Goodput Guarantees for Error-prone Flows

For a continuously backlogged flow i, over a sufficiently long time, its average allocated transmission bandwidth is at least $r_i(1 + th_i)$. Consequently, if the average error probability of flow i's low rate packet, with a code rate[1] of θ, in a *bad* state is e_i, flow i will be guaranteed a long-term average goodput rate of $r_i\theta(1 + th_i)e_i$, even if the flow's link is always in a *bad* state.

Balance between Effort-fair and Outcome-fair

First, a fraction of the transmission bandwidth is used to guarantee minimum transmission effort for the flows to realize effort-fair. In addition, when the network is lightly loaded and much excess bandwidth is available, the algorithm tries to realize outcome-fair without seriously degrading the overall bandwidth efficiency.

Delay Bound for Error-free Flows

Theorem: For a flow i with an error-free link, if its traffic is constrained by a token bucket (σ_i, r_i), where σ_i is the bucket depth and r_i is the token rate, which is equal to flow i's target rate, flow i's packet delay D_i can be bounded as

$$D_i \leq \frac{\sigma_i}{r_i} + \frac{2l}{r_i} + \frac{l}{R} \tag{3}$$

Proof: Omitted due to space limit. Proof is available in [7].

[1] The ratio of the information length to the total length.

TABLE I
SIMULATION RESULTS IN EXAMPLE I

	Flow 1		Flow 2		Efficiency
	effort	goodput	effort	goodput	
CIFQ ($\alpha = 0$)	193.3	89.9	726.0	639.8	72.0%
CIFQ ($\alpha = 0.9$)	115.5	45.7	799.1	702.5	74.8%
BGFS ($th = -0.3$)	590.8	278.4	409.2	362.8	64.1%
BGFS ($th = -0.5$)	526.4	250.0	473.6	408.0	65.8%

IV. Simulation Results

We conducted simulations to compare the performance of CIFQ and BGFS-EBA. To simulate the fact that each link state corresponds to a range of BER or SNR instead of a single point, we let all the packet error probabilities vary over a range. We denote by $u(a, b)$ a uniform distribution between a and b. The parameters used in the simulations are as follows. The packet size is 200 bytes. The error probabilities of a regular (uncoded) packet are distributed as $u(0, 0.2)$ and $u(0.8, 1)$ in a *good* state and a *bad* state, respectively. In BGFS-EBA, when needed, a regular packet is split into two low rate packets, having 100 bytes of information each. The error probabilities of a low rate packet are distributed as $u(0, 10^{-3})$ and $u(0, 0.1)$ in a *good* state and a *bad* state, respectively. A *good* (or *bad*) state can be wrongly estimated as a *bad* (or *good*) state with probability 0.1. The total available bandwidth is $1Mbytes/s$. The duration of each test equals the transmission time of two million packets. All flows require reliable delivery.

A. Example I

We start with a very simple example to demonstrate the idea of guaranteeing minimum transmission bandwidth and goodput. In this example, there are only two flows in the network. Each flow has a target rate of $500Kbytes/s$. The average state durations of flow 1's link are 0.01s and 0.09s for the *good* and the *bad* states, respectively. The average durations of flow 2's link are 0.09s and 0.01s for the *good* and the *bad* states, respectively. Flows 1 and 2 both have greedy traffic sources, i.e., they are always backlogged. Table I shows the simulations results.

All the numbers except the efficiency are the average rates $inKbytes/s$. The parameter α, as defined in CIFQ, is the minimum fraction of service retained by a leading session. The *efficiency* is defined as the total goodput rate divided by the total available bandwidth, i.e. $1Mbytes/s$. In BGFS-EBA, both flows have the same threshold.

In CIFQ, there is no link-independent transmission bandwidth guarantee for a flow with a very poor link. Note that in CIFQ, flow 1 receives very little bandwidth and goodput. Even as $\alpha = 0$, where the lagging flow 1 has the most transmission opportunities in CIFQ, its effort rate is still less than 40% of its target rate. However, in BGFS-EBA, a flow is guaranteed certain amount of bandwidth regardless of its link quality (effort-fair). Beyond the guaranteed bandwidth, flow 1 is given more transmission effort to compensate for its poor link quality to help it achieve outcome-fair. Therefore, we see that flow 1 receives far more bandwidth and goodput in BGFS-EBA than in CIFQ.

TABLE II

CIFQ'S RESULTS IN EXAMPLE II

	$\alpha = 0$		$\alpha = 0.3$		$\alpha = 0.7$	
	effort	goodput	effort	goodput	effort	goodput
flow 1	249.9	156.4	160.8	111.0	151.3	105.9
flow 2	124.9	71.7	124.8	80.0	124.8	80.4
flow 3	249.9	225.2	284.0	251.9	290.3	255.9
flow 4	249.9	225.2	282.3	250.3	284.3	251.8
flow 5	125.0	115.6	147.5	130.3	148.8	130.4
efficiency	79.4%		82.3%		82.4%	

TABLE III

BGFS-EBA'S RESULTS IN EXAMPLE II

	$th = -0.2$		$th = -0.3$		$th = -0.5$	
	effort	goodput	effort	goodput	effort	goodput
flow 1	279.8	200.1	278.3	199.5	277.9	199.3
flow 2	149.8	101.8	145.5	101.0	143.9	100.2
flow 3	227.8	201.5	230.1	204.1	231.0	205.3
flow 4	227.5	201.4	230.0	204.0	231.0	205.2
flow 5	115.0	100.7	116.1	102.3	116.5	103.2
efficiency	80.6%		81.1%		81.3%	

Also, flow 1 receives more bandwidth when $th = -0.3$ than $th = -0.5$.

Note that BGFS-EBA's overall efficiency is lower than that of CIFQ. This is the tradeoff for providing minimum bandwidth guarantees. In fact, this is an extreme case, where flow 1's link quality is very bad and its target rate is quite high. In more general cases, the efficiency difference between the two algorithms will not be so big.

B. Example II

In the second example we show a more complex scenario. There are five greedy flows in the system. Flows 1 and 2 have worse links than flows 3, 4 and 5. For flows 1 flow 2, the average state durations of the good state and bad state are $0.03s$ and $0.07s$, respectively. For flows 3, 4, and 5, the average state durations of the good state and bad state are $0.09s$ and $0.01s$, respectively. Flows 1, 3 and 4 all have the same goodput target rate, $200Kbyte/s$, while flows 2 and 5 both have the same target rate, $100Kbytes/s$.

The simulation results are shown in Tables II and III. For CIFQ, the average goodput rates of the flows with better links (flow 3, 4 and 5) are more than 10% higher than their target rates while flows 1 and 2 have average goodput rates far from their target rates. For BGFS-EBA, almost each flow's goodput is around its target rate. This is because the scheduler uses the excess bandwidth to compensate flows with bad links while trying to make sure that such compensation does not jeopardize the service targets of flows with better links. Since the thresholds are not set very aggressively, the efficiency is maintained at a high level. Outcome-fairness is achieved.

In fact, when we eliminate flow 5 but with no other parameters changed, we find, in CIFQ, flow 1 and 2 still miss their target rates by at least 20%. In BGFS, all flows achieve at least their target rates. This is due to the way excess bandwidth is distributed and the discrimination of CIFQ against flows with worse link quality. Therefore, the flows with better links are over-provisioned.

C. Example III

In this last example we demonstrate that the delay bounds for error-free flows can be guaranteed. In this example there are 6 flows in the network, which is fully loaded with the sum of all the target rates equal to the bandwidth. Table IV shows the parameters of the traffic and the links. The two Poisson sources

TABLE IV

PARAMETERS IN EXAMPLE III

	Target Rate (Kbyte/s)	Traffic	Avg. Link State Duration (s)	
			good state	bad state
flow 1	200	Exponential on/off	error-free	
flow 2	100	Poisson	error-free	
flow 3	200	Poisson	0.09	0.01
flow 4	200	Greedy	0.09	0.01
flow 5	200	Greedy	0.03	0.07
flow 6	100	Greedy	0.09	0.01

TABLE V

PACKET DELAYS OF BGFS-EBA IN EXAMPLE III

	$th = -0.3$		$th = -0.5$		Analytical
	max. delay	avg. delay	max. delay	avg. delay	bound
flow 1	0.0098s	0.0009s	0.0096s	0.0007s	0.0122s
flow 2	0.019s	0.0023s	0.018s	0.0018s	0.0242s
flow 3	0.104s	0.014s	0.11s	0.015s	n/a

both have an average rate of $200Kbytes/s$. The exponential on/off source's average on time is 500ms and average off time is 100ms. When it is on, the sending rate is $400Kbytes/s$. To provide any delay guarantees the traffic bursts should be constrained. Flows 1, 2 and 3 are all token-bucket-constrained. Each flow's token accumulating rate is just its target rate. The depth of the bucket is 2000 bytes. In BGFS-EBA, if a regular packet is split into low rate packets, its transmission is considered complete only when all its low rate packets have been received.

The analytical bounds are calculated based on (3). Table V shows that for error-free flows, all the delays are within the analytical bounds. Compared with the results in Table VI, we see that the delays for error-free flows in the two algorithms are similar. However, for error-prone flow 3, the delays in our proposed algorithm are smaller. This is because in BGFS-EBA, error-prone flows send their packets more aggressively.

REFERENCES

[1] T. S. Ng, I. Stoica, H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," Proc. of INFOCOM98, pp. 1103-1111, March 1998.

[2] S. Lu, T. Nandagopal, and V. Bharghavan, "Design and analysis of an algorithm for fair service in error-prone wireless channels," Wireless Networks, vol. 6, pp. 323-343, 2000.

[3] D. A. Eckhardt and P. Steenkiste, "Effort-limited fair (ELF) scheduling for wireless networks," Proc. of INFOCOM 2000, pp. 1097-1106, 2000.

[4] P. Lin, B. Benssou, Q. L. Ding, and K.C. Chua, "CS-WFQ: a wireless fair scheduling algorithm for error-prone wireless channels," Proc. Computer Communications and Networks 2000, pp. 276 -281, 2000.

[5] Y. Cao and V. O. K. Li, "Scheduling algorithms for broadband wireless networks," Proc. of the IEEE, Vol. 89, No. 1, pp. 76 - 87, Jan. 2001.

[6] R. L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks," IEEE JSAC, vol. 13, no. 6, pp. 1048-1056, Aug. 1995.

[7] Y. Cao and V. O. K. Li, "Wireless Packet Scheduling for Two-state Link Models," Technical Report, Dept. of Electrical and Electronic Engineering, The University of Hong Kong.

[8] D. Stiliadis and A. Varma, "Latency-rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," IEEE Trans. on Networking, vol. 6, no. 5, pp. 611-624, Oct. 1998.

[9] A. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," IEEE/ACM Trans. Networking, vol. 1, pp. 334-357, June 1993.

TABLE VI

PACKET DELAYS OF CIFQ IN EXAMPLE III

	$\alpha = 0.1$		$\alpha = 0.3$	
	max. delay	avg. delay	max. delay	avg. delay
flow 1	0.0095s	0.0008s	0.0096s	0.00077s
flow 2	0.019s	0.0024s	0.0191s	0.0026s
flow 3	0.276s	0.057s	0.277s	0.0386s