Scheduling Algorithms in Broad-Band Wireless Networks

YAXIN CAO AND VICTOR O. K. LI, FELLOW, IEEE

Invited Paper

Scheduling algorithms that support quality of service (QoS) differentiation and guarantees for wireless data networks are crucial to the development of broad-band wireless networks. Wireless communication poses special problems that do not exist in wireline networks, such as time-varying channel capacity and location-dependent errors. Although many mature scheduling algorithms are available for wireline networks, they are not directly applicable in wireless networks because of these special problems. This paper provides a comprehensive and in-depth survey on recent research in wireless scheduling. The problems and difficulties in wireless scheduling are discussed. Various representative algorithms are examined. Their themes of thoughts and pros and cons are compared and analyzed. At the end of the paper, some open questions and future research directions are addressed.

Keywords*—*Broad-band wireless networks, QoS, scheduling.

I. INTRODUCTION

In the future, broad-band wireless networks will be an integral part of the global communication infrastructure. With the rapid growth in popularity of wireless data services and the increasing demand for multimedia applications, it is expected that future wireless networks will provide services for heterogeneous classes of traffic with different quality of service (QoS) requirements. Currently, there is an urgent need to develop new technologies for providing QoS differentiation and guarantees in wireless networks. Among all the technical issues that need to be resolved, packet scheduling in wireless networks is one of the most important.

Scheduling algorithms provide mechanisms for bandwidth allocation and multiplexing at the packet level. Admission control and congestion control policies are all dependent on the specific scheduling disciplines used. Many scheduling algorithms, capable of providing certain guaranteed QoS, have been developed for wireline networks. However, these existing service disciplines, such as fair queueing scheduling [1], virtual clock [4], and EDD [2], are not directly applicable in wireless networks because they do not consider the varying wireless link capacity and the location-dependent channel state.

The characteristics of wireless communication pose special problems that do not exist in wireline networks. These include: 1) high error rate and bursty errors; 2) location-dependent and time-varying wireless link capacity; 3) scarce bandwidth; 4) user mobility; and 5) power constraint of the mobile hosts. All of the above characteristics make developing efficient and effective scheduling algorithms for wireless networks very challenging.

Recently, new propositions and schemes for packet scheduling in wireless networks, which account for the special characteristics of the wireless environment, have emerged. However, to the best of our knowledge, none of this work systematically addresses the problem of packet scheduling in wireless networks. This paper presents a comprehensive survey and in-depth discussion on packet scheduling in wireless networks, and more specifically, in cell-structured wireless networks. It is expected that, as the first step of wireless and wireline network integration, wireless networks will be mainly used as access networks to the wired world for mobile users. The most commonly adopted and deployed wireless networks are based on a cell structure. Although many of the underlying ideas of the solutions to be discussed in this paper can also be used in the design of scheduling algorithms for *ad hoc* networks (also called packet radio networks), scheduling issues in *ad hoc* networks are beyond the scope of this paper.

The remainder of the paper is organized as follows. In Section II, the system model, including the wireless network model and the wireless link model, is presented. Section III

Manuscript received February 23, 2000; revised September 4, 2000. This work was supported in part by the Research Grants Council, Hong Kong, under Grant HKU 7224/99E.

Y. Cao is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: ycao@usc.edu).

V. O. K. Li is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Pokfulam, Hong Kong (e-mail: vli@eee.hku.hk).

Publisher Item Identifier S 0018-9219(01)00453-4.

discusses the major issues in the design of wireless scheduling algorithms. Section IV–VIII examine various representative wireless scheduling algorithms and contrast their advantages and disadvantages. Further discussion of the algorithms are presented in Section IX. Finally, open research questions and future research directions are addressed in Section X.

II. System Model

A. Wireless Network Model

In this paper, we are interested in cell-structured wireless networks. In such networks, the service area is divided into cells, and each cell has a base station. Within a cell mobile hosts communicate via the base station, and base stations are connected via wireline networks. The communication between a mobile host and a base station may consist of more than one traffic flow (or session). The base station is responsible for scheduling both downlink (from base station to mobile hosts) and uplink (from mobile hosts to base station) packet transmissions between the mobile hosts and itself. All the downlink packets are queued at the base station; therefore, the base station has full knowledge of the status of downlink queues. For uplink packet transmissions, mobile hosts need to send their transmission requests and queue status, if necessary, to the base station. The base station performs uplink scheduling based on these requests and related information. Although we assume the above model for the wireless network, the algorithms to be discussed are not necessarily only applicable in this model.

B. Wireless Link Model

All the scheduling algorithms discussed in this paper use the following wireless link model. The wireless links between a base station and each of the mobile hosts are independent of each other. Wireless links are subject to bursty errors. A two-state Markov channel model [11] is used for the state of a wireless link, which is in either one of the two states: *good* (or *error-free*) state or *bad* (or *error*) state. In a *good* state, the wireless link is assumed to be error-free. If a link is in a *bad* state, packets transmitted on the link will be corrupted with very high probability. Transitions between the two states occur randomly.

III. Major Issues in Wireless Scheduling

A. Wireless Link Variability

The biggest difference between a wireless network and a wireline network is the transmission link variability. Due to the high quality of the transmission media, packet transmissions on wireline networks enjoy very low error rate. However, wireless channels are more error-prone and suffer from interference, fading, and shadowing. As a result, the capacity of a wireless link has very high variability. During some severe bursty error state, a wireless link can be so bad that no successful packet transmissions can be achieved. Besides the time-dependent problem, wireless link capacity is also location-dependent. At a particular time instance, a base station can communicate with more than one mobile host simultaneously. Due to different physical locations, some mobile hosts may enjoy error-free communication with the base station, while others may not be able to communicate at all. This is the so-called *location-dependent error*. Furthermore, mobility of the hosts increases the variability of the transmission links. Such link variations require the scheduling algorithms to be equipped with certain dynamic mechanisms that can deal with these time-dependent and location-dependent changes.

B. Fairness

Scheduling fairness in wireline networks is usually guaranteed by dedicating a certain service rate to a flow, and the scheduling algorithm prevents different flows from interfering with each other. Since wireline media may be considered error-free, the service rate allocated is indeed the amount of service share that is received by a particular flow. However, the fairness issue in wireless networks is more complicated. It may happen that a packet is scheduled for transmission on a wireless link according to certain service discipline or fairness guideline, which are independent of link state, and the link is actually in an *error* state. If the packet is transmitted, it will be corrupted and the transmission will waste transmission resources. In this case, deferring transmission of this packet till the link recovers from the *error* state is clearly a reasonable choice. The affected flow, hence, temporarily loses its share of the transmission bandwidth. To ensure fairness, the flow should be compensated for this loss later when the link recovers. But determining how to compensate for it is not an easy task. Some fairness definitions for wireline scheduling algorithms are available in [12], [13]. The definition and objectives of fairness guarantees become more ambiguous in a wireless environment. The granularity of fairness, i.e., short-term fairness versus long-term fairness, is another factor that affects the scheduling policy. The appropriate interpretation of fairness for wireless scheduling should depend on the service model, traffic type, and channel characteristics.

C. QoS

Broad-band wireless networks will provide services for heterogeneous classes of traffic with different QoS requirements. Therefore, QoS differentiation and guarantees must be supported. To achieve this goal, the corresponding mechanism for QoS support should be integrated into the scheduling algorithm. QoS support in wireless scheduling is dictated by the service model. For DiffServ [6] type of services, at least prioritized scheduling service for aggregated traffic with QoS differentiation should be implemented in the scheduling algorithm; whereas for IntServ [5] type of services, support for per-flow-based guaranteed QoS performance, such as delay or jitter bound, should be provided by the scheduling algorithm. Of course, if a wireless link experiences frequent channel degradations, it is very difficult to guarantee QoS for the flows using this link. Nevertheless, QoS should be guaranteed for the flows, either deter-

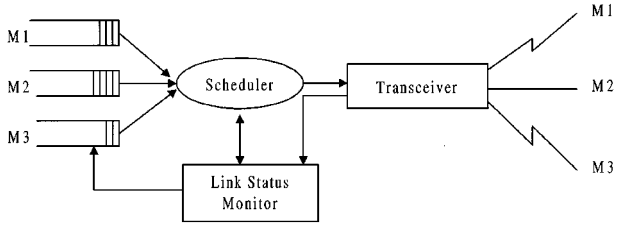

Fig. 1. CSDPS component model.

ministically or statistically, on those links where the physical channel degradation does not exceed certain thresholds.

D. Data Throughput and Channel Utilization

The most precious resource in wireless networks is the bandwidth. An efficient wireless scheduling algorithm should aim to minimize unproductive transmissions on error links, and at the same time, maximize the effective service delivered and the utilization of the wireless channels.

E. Power Constraint and Simplicity

The scheduling algorithms in cell-structured wireless networks are usually run at the base station. Therefore, the electric power required for computation of packet service order should not be a big concern because of adequate power supply at the base station. However, mobile hosts are power-constrained. A good scheduling algorithm should be designed in such a way that minimal number of scheduling-related control messages, which may contain information on mobile hosts' queue status, packet arrival times, and channel states, are required from mobile hosts. For example, a scheduling algorithm that needs to use every uplink packet's arrival time to compute scheduling order is not a good choice, because it demands a large amount of power from mobile hosts for transmitting the information of arrival times to the base station. Also the scheduling algorithm should not be too complex, so that it can be executed at high speed to schedule real-time multimedia traffic with stringent timing requirements.

IV. CSDPS AND ITS VARIANT

A. CSDPS

One of the first papers that address the problem of location-dependent and bursty errors in scheduling in wireless networks is [8]. It proposes a scheduling algorithm called channel state dependent packet scheduling (CSDPS). Fig. 1 presents the major components of a CSDPS scheduling system of the base station. A separate queue is maintained for each mobile's packets. Within one queue, packets are served in an FIFO order. Across the queues the service policy could be decided according to service requirements. In particular, the paper discusses three service policies, namely, round robin (RR), earliest timestamp first (ETF), and longest queue first (LQF).

The basic idea of CSDPS is very simple. When a wireless link experiences bursty errors, the scheduling algorithm defers transmission of packets on this link. The link status monitor (LSM) monitors the link states for all mobile hosts. When the LSM determines that a link is in a *bad* state, it marks the affected queues. The scheduler does not serve the marked queues. The queue is unmarked after a time-out period, which may, for example, be the average link "error" duration. A link is considered being in a *bad* state when the acknowledgment for a data packet from the mobile is not received. Simulations show that compared with just using a single FIFO queue for all traffic, CSDPS can achieve much higher data throughput and channel utilization. Since CSDPS alleviates the head of line (HOL) blocking problem experienced by a single FIFO queue, the average delay of the packets is reduced.

CSDPS improves scheduling performance by taking into account the location-dependent and time-dependent channel states. However, it has several drawbacks. It does not have any mechanism to guarantee bandwidth for a mobile user. A mobile user may receive much less service opportunities than its fair share because when a link is "believed" to be in a *bad* state, the affected mobile user's service opportunities in a subsequent time period are given to other mobile users although those other mobile users may already have exceeded their fair service share. No limit is imposed on the amount of service received by a mobile user. Moreover, the algorithm does not provide any guarantees on packet delay.

B. CSDPS + CBQ

In order to solve the problem of unfair bandwidth sharing in CSDPS, a scheduling scheme combining class-based queueing (CBQ) [3] and CSDPS is proposed in [10]. In this scheme, users or traffic flows are grouped into *classes*, and each *class* is committed with a certain amount of bandwidth. The CSDPS component is used to deal with wireless link variations, and the CBQ component is used to provide fairness mechanism in sharing the overall wireless channel.

CBQ is a set of hierarchical channel-sharing guidelines for ensuring that classes receive their allocated share of bandwidth over a predefined time scale. It also aims to distribute the excess bandwidth in a fair way. CBQ keeps track of the amount of service received by each class in a certain time-interval window. A class is called *unsatisfied* if it has persistent backlogs, and the service it recently received is less than its allocated fraction. The channel-sharing guideline of CBQ is that a class should be restricted from receiving service when it exceeds its allocated bandwidth share and contributes to any other class' *unsatisfied* state. Such a queue is called a *restricted* queue.

Reference [10] makes changes to the CSDPS component in the following two aspects. First, it uses the ready-to-send (RTS) and clear-to-send (CTS) pair [21], rather than the data packet and acknowledgment pair, to test the states of wireless links. RTS and CTS are exchanged between the base station and a mobile host before a data packet is transmitted. RTS and CTS messages are usually shorter than data packets. If a link is in an *error* state, using RTS-CTS wastes less transmission bandwidth than data packets. However, if a link is in a *good* state, RTS-CTS imposes extra overhead and delay on packet transmissions. Second, each link i is associated with a

variable G_i, the maximum number of tries of RTS transmissions allowed before a packet transmission attempt is given up. G_i actually represents the estimate of how good a link is and is updated whenever a packet is scheduled for transmission on link i. G_i is decreased exponentially if no CTS is received after G_i trials of RTS. If it succeeds within less than G_i trials, G_i is increased in inverse proportion to the number of RTS trials.

The original CBQ in [3] is also modified to accommodate the special needs in the wireless environment. Instead of guaranteeing a class a specific share of transmission bandwidth, it guarantees a percentage of the total *effective throughput*. *Effective throughput* is defined as the total successful data transmissions per unit time. The reasoning of the authors is that a portion of the bandwidth will be wasted by unsuccessful transmissions and RTS-CTS attempts, and the fair share of a mobile user should be based on the total effective service the base station delivered, not the theoretical bandwidth. Also, a *restricted* queue will continue to receive service if it has a *good* link and all other unsatisfied queues are experiencing *bad* links.

This algorithm enables fair sharing of the wireless channel while trying to maintain high throughput. However, it does not have an explicit mechanism for compensating those mobile users who have previously lost their service share because of link error. Thus, there is no guarantee on the rate at which a mobile user will be compensated. Also, issues on other QoS guarantees, such as delay bound and loss rate, are not discussed in [10].

V. Algorithms Using an Ideal Reference System

A. IWFQ

Derived from the fluid fair queueing (FFQ) [or generalized processor sharing, i.e., (GPS)] and weighted fair queueing (WFQ) [1], [14] models, wireless fluid fair queueing (WFFQ), and idealized wireless fair queueing (IWFQ) for packet scheduling in cell-structured wireless networks are proposed in [9]. The relationship between IWFQ and WFFQ is very similar to the relationship between WFQ and FFQ. Basically, FFQ and WFFQ are fluid scheduling models for wireline and wireless networks, respectively. WFQ and IWFQ are corresponding packet-based emulations of FFQ and WFFQ, respectively.

Since WFFQ, just like FFQ, is based on the unrealistic fluid model and is mainly presented for the purpose of theoretical analysis, we focus our discussion on its packet-based emulation, IWFQ. An IWFQ model is defined with reference to an error-free WFQ service system. Given the packet arrival sequence for each data flow, an *error-free service* is defined as the WFQ service for the flows with identical arrival sequences and completely error-free links. The service received by a flow in the IWFQ system (with link errors) is compared with the *error-free system*. A flow is said to be *leading*, *lagging*, or *in sync* at any time instant if its queue size is smaller than, larger than, or the same as the queue size in the *error-free system*. With the assumption of perfect knowledge on wireless links and a perfect multiple-access control (MAC) protocol, IWFQ works as follows.

Each flow has its own queue. When no link suffers from errors, it works just like ordinary WFQ in wireline networks. When a packet of sequence number n of flow i arrives, it is tagged with virtual service *start time* $S_{i,n}$ and *finish time* $f_{i,n}$. More specifically

$$\begin{aligned} S_{i,n} &= \max\{v(A(t)),\ f_{i,n-1}\}; \\ f_{i,n} &= S_{i,n} + L_{i,n}/r_i \end{aligned} \tag{1}$$

where

$L_{i,n}$	packet size of the arrived packet;
$v(A(t))$	*system virtual time* defined in WFQ;
r_i	service rate allocated to flow i.

Packets are stored in a nondecreasing order of the *finish times* in each queue. The scheduler always chooses to serve the packet with the smallest *finish time*. All these operations are exactly what a wireline WFQ scheduler performs. The difference occurs when link error happens. If the chosen packet cannot be transmitted because of a *bad* link state, the packet in other queues with the next smallest *finish time* will be picked. The process will continue and repeat from the beginning, if necessary, until the scheduler finds a packet with a *good* link state.

Since the time tags of a packet normally are not changed after a packet's arrival, a flow that loses its service opportunity because of link error will have packets with smaller *finish times*, compared with other flows' packets. Therefore, it will have precedence in accessing the service bandwidth when its link exits from error. So the compensation is guaranteed. However, allowing an unbounded amount of compensation can result in denial of service to other error-free flows. When a flow's link recovers from an *error* state, its packets may have the smallest virtual tags and as a result, the scheduler may only serve this flow exclusively for an extended period. To alleviate this problem, IWFQ artificially bounds the amount of lag and lead. The bounds are defined and enforced in the following ways.

1) Lagging bound: The total lag, with respect to *error-free service*, of all flows that will be compensated is bounded by B bits. A flow i with weight r_i is allowed to compensate a maximum of $b_i = B \cdot (r_i / \sum_{j \in F} r_j)$ bits, where F is the set of all flows. The scheduler checks every queue after transmitting a packet. For a *lagging* flow, if the total length of the packets with *finish times* less the current *virtual time* is greater than b_i bits, then, among these lagging packets, only the first N_i packets are retained and other lagging packets are deleted from the queue. N_i is the largest integer such that $\sum_{n=1}^{N_i} L_{i,n} \le b_i$.
2) Leading bound: If a flow leads, with respect to *error-free service*, for more than l_i bits, it will only surrender up to l_i bits of service share to other flows later on. To implement this bound, the scheduler checks each *leading* flow after transmitting one packet. If the *start*

time $S_{i,\mathrm{HOL}}$ of the HOL packet of a leading flow satisfies $S_{i,\mathrm{HOL}} > v(t) + l_i/r_i$, then set $S_{i,\mathrm{HOL}} = v(t)+l_i/r_i$ and $f_{i,\mathrm{HOL}} = S_{i,\mathrm{HOL}}+L_{i,\mathrm{HOL}}/r_i$, where $f_{i,\mathrm{HOL}}$ is the *finish time* of the HOL packet, $v(t)$ is the system *virtual time*, and $L_{i,\mathrm{HOL}}$ is the packet length of the HOL packet.

Following the above two guidelines, the amount of compensation for *lagging* flows and penalty for *leading* flows are bounded. Consequently, throughput and delay guarantees are achievable for IWFQ. Analytical results of throughput and delay bounds of IWFQ are derived in [9] for error-free links and error-prone links, respectively. Here, we need to note that delay bounds are derived only for packets not deleted from the queue by the scheduler to enforce the artificial lagging bound.

There is a problem with the described approach, namely, how can the base station know the arrival time of uplink packets? It is impractical to require all mobile hosts to convey this information to the base station every time a packet arrives. In the real implementation of the algorithm, weighted round robin (WRR) is used instead of WFQ. Hence, only information of whether an uplink queue is empty or not is needed for scheduling. Since perfect link knowledge is impossible, missing acknowledgment messages are used for detecting link errors. The simulation shows that the worst case performance of the real implementation is much worse than IWFQ, and the average performance is very close to the idealized algorithm.

Although IWFQ has some appealing properties in fairness and QoS guarantees, its limitations have been pointed out in [7]. First, since absolute priority is given to packets with the smallest *finish times*, when a flow is compensated for its previous lagged service all other error-free flows will not be served at all. For the same reason, a *lagging* flow will receive compensation at a rate independent of its allocated service rate, violating the semantics that a larger guaranteed rate implies better QoS. Second, the choice of the parameter B reflects a conflict between the delay and fairness properties. A large B means that *lagging* flows can receive more compensation after recovering from errors, but it also causes other error-free flows to be denied service for longer periods. In addition, the delay and throughput guarantees are closely coupled, which may not be desirable.

B. CIF-Q

Reference [7] proposes a wireless scheduling algorithm called channel-condition independent packet fair queueing (CIF-Q). CIF is very similar to IWFQ in the sense that it also uses an error-free fair queueing reference system and tries to approximate the real service to the ideal error-free system. Similar to the definitions in [9], a flow is said to be *leading*, *lagging*, or *satisfied* at any time instant if it receives more, less, or the same amount of service as it would have received in the corresponding *error-free system*. The main objective of CIF-Q is to address the fairness issue; therefore, link error detection, estimation, and implementation issues are not discussed.

One of the important contributions made by [7] is that it clearly specifies the properties a fair wireless scheduler should have. Although these properties may not be optimal and are subject to modifications if used in different environments and applications, they provide some useful guidelines for designing fairness mechanisms in wireless scheduling algorithms. The set of properties includes four objectives.

1) Delay and throughput guarantees: delay bound and throughput for error-free flows should be guaranteed, and not affected by other flows in *error* state.
2) Long-term fairness: after a flow exits from link error, as long as it has enough service demand, it should be compensated, over a sufficiently long period, for all of its lost service while it was in error.
3) Short-term fairness: the difference between the normalized services received by any two error-free flows that are continuously backlogged and are in the same state (i.e., *leading*, *lagging*, or *satisfied*) during a time interval should be bounded.
4) Graceful degradation for leading flows: during any interval while it is error free, a leading backlogged flow should be guaranteed to receive at least a minimum fraction of the service it would receive in an *error-free system*.

CIF-Q is designed to achieve the above goals. Start-time fair queueing (SFQ) [15] is chosen to be the core of CIF-Q, although other wireline fair queueing algorithms can also be used. Just like IWFQ, the real error-prone scheduling system S is associated with an error-free system S^r. Each flow has its own queue. When a flow is served, its HOL packet is transmitted. One of the major differences between CIF-Q and IWFQ is that *virtual time* is only kept and updated in S^r, and not in S. Arrived packets are put into queues both in S and S^r. (The queues in S^r are virtual queues used for keeping and updating each HOL packet's *virtual start time*, also called a flow's *virtual time*.) Service order is determined by employing SFQ in S^r. When there is no link error, if a packet is chosen in S^r, it is served in both S^r and S. However, when a chosen packet in S^r cannot be transmitted in S because of link error, the real packet in the queue of S is kept, but the virtual packet in the queue of S^r is still served and the corresponding flow's *virtual time* is updated according to SFQ. Therefore, a flow's *virtual time* only keeps track of the normalized service, with respect to its allocated service rate, received in the *error-free system*, not in the real system.

A parameter lag_i is associated with flow i to keep track of the difference between the service received by a flow in S and the service it receives in S^r. Since the scheduling algorithm is work-conserving, $\sum_{i\in A} lag_i = 0$, where A is the set of all active flows. To achieve graceful degradation, a parameter a is used to define the minimal fraction of service to be received by a leading flow. In particular, a *leading* flow is allowed to continue to receive service at an average rate of $a \cdot r_i$, where r_i is the allocated service rate. The value of a is configurable. The remaining part of service share of the leading flow is given up for compensating other *lagging* flows. The following summarizes the key rules of CIF-Q scheduling.

1) When scheduling the next packet to be transmitted, the HOL packet with the smallest *virtual start time*

in S^r is chosen, and the corresponding packet in S is transmitted unless one of the following situations occurs.

a) The link on which the picked packet is to be transmitted is in an *error* state.

b) The picked packet belongs to a *leading* flow and during the current leading period (time elapsed since the flow starts to lead) the flow has received more than a fraction a of the normalized service it should have received based on its allocated service rate.

2) *Lagging* flows have higher priority to receive *additional service* available due to *leading* flows giving up lead [case b)] or another flow not being able to send because of error link [case a)].

3) Instead of allocating all the *additional service* to the flow that has the largest lag, the compensation is distributed among the lagging flows in proportion to their allocated service rates.

4) If no *lagging* flow can receive *additional service* because of link errors, the *additional service* is distributed to *nonlagging* flows in proportion to their allocated service rate.

By enforcing the above rules, the four previously listed fairness properties can be satisfied. The detailed implementation of the above rules can be found in [7]. It is proven in [16] that for CIF-Q, the difference between the normalized service received by any two flows i and j during any interval $[t1, t2)$ in which both flows are continuously backlogged, error-free, and their status unchanged, is bounded by the following inequality:

$$\left|\frac{W_i(t_1, t_2)}{r_i} - \frac{W_j(t_1, t_2)}{r_j}\right| < \beta\left(\frac{L_{\max}}{r_i} + \frac{L_{\max}}{r_j}\right) \quad (2)$$

where $W_i(t_1, t_2)$ represents the service received by flow i during $[t1, t2)$, $L_{\max}$ is the maximum packet length, and $\beta = 3$ if both flows are nonleading, and $\beta = 3 + a$ otherwise.

The biggest advantage of CIF-Q is its nice properties in both long-term and short-term fairness guarantees. In addition, the other two QoS parameters, packet delay bound and flow throughput, can also be guaranteed for flows with error-free links. Compared with IWFQ, CIF-Q improves scheduling fairness by associating compensation rate and penalty rate with a flow's allocated service rate and guaranteeing flows with error-free links with a minimal service rate. In CIF-Q, all the quantities are normalized by service rate, which makes service allocation fairer. Since CIF-Q also needs to use packet arrival times to compute *virtual start times* for scheduling uplink packets, it has the same problem as IWFQ, i.e., how does the base station get the information of uplink packet arrival times? The algorithm complexity of both CIF-Q and IWFQ are relatively high because they need to simulate an *error-free* system and keep record of and update the amount of services received in the real system.

VI. Algorithms Using an Explicit Compensation Counter or Server

A. SBFA

Server-based fair approach (SBFA) is proposed in [17]. In this approach, part of the wireless bandwidth is allocated to some compensation server(s), called long-term fairness server (LTFS). An LTFS is a special data flow created for compensating flows whose packet transmissions are deferred because of link errors, and it shares the wireless channel with other regular data flows.

The algorithm works with an implicit assumption that the packet size of each flow is fixed, although different flows can have different packet sizes. The scheduler maintains two queues, packet queue (PQ) and slot queue (SQ) for each flow. When a packet of flow i arrives, it is put into PQ_i and a virtual copy of the packet called slot is put into SQ_i. The scheduling policy is applied on slot queues. Any wireline scheduling algorithm can be used for scheduling slot queues. Each slot carries an identification number specifying which flow it belongs to. When a slot is chosen from a slot queue, its corresponding flow's HOL packet in the packet queue will be transmitted if the link it uses is in a *good* state. In this case, the slot and the packet will be removed from the slot queue and the packet queue, respectively.

Fig. 2 illustrates what happens if a chosen packet can not be transmitted because of link errors. Assume there are two flows and one LTFS in the system. The RR scheduling policy is used for simplicity. At the beginning of time 0, the queue status is shown in Fig. 2(a). No packet arrives after time 0. At time 0, the scheduler picks one slot in SQ_1 and finds that link 1 is in error. Therefore, the transmission of packet p11 is deferred. Instead, the scheduler chooses to serve flow 2. Link 2 is in a good state, so the HOL packet of flow 2, namely, p21 is transmitted. Note the changes in the queues shown in Fig. 2(b) at the beginning of time 1. Packet p11 is kept in PQ_1, but one slot is removed from SQ_1. At the same time one slot with flow 1's identification number (s1) is added into the LTFS queue. Since p21 of flow 2 is transmitted, p21 and one slot are removed from the queues of flow 2. According to the original RR service order, time 1 is flow 2's turn to receive service. Link 2 is still in the *good* state at time 1; therefore, one more packet of flow 2 is transmitted. Then at time 2, LTFS takes its turn to receive service. The scheduler checks the queue of LTFS and finds a slot with identification number s1. Thus, the HOL packet of flow 1 is picked for transmission. If at this time link 1 is in a *good* state, the HOL packet will be transmitted, and p11 and s1 will be dequeued from PQ_1 and LTFS, respectively. However, if link 1 is still in a bad state, s1 will still be kept in LTFS, so that this compensation credit is kept for flow 1.

By recording the service loss of each flow in LTFS and dedicating part of the bandwidth to LTFS, flows losing their service share because of link errors will eventually be compensated. There can be more than one LTFS in the system. The exact number depends on the QoS requirements of flows or some administrative policies of the flows. It is suggested that flows with similar requirements be assigned to the same LTFS.

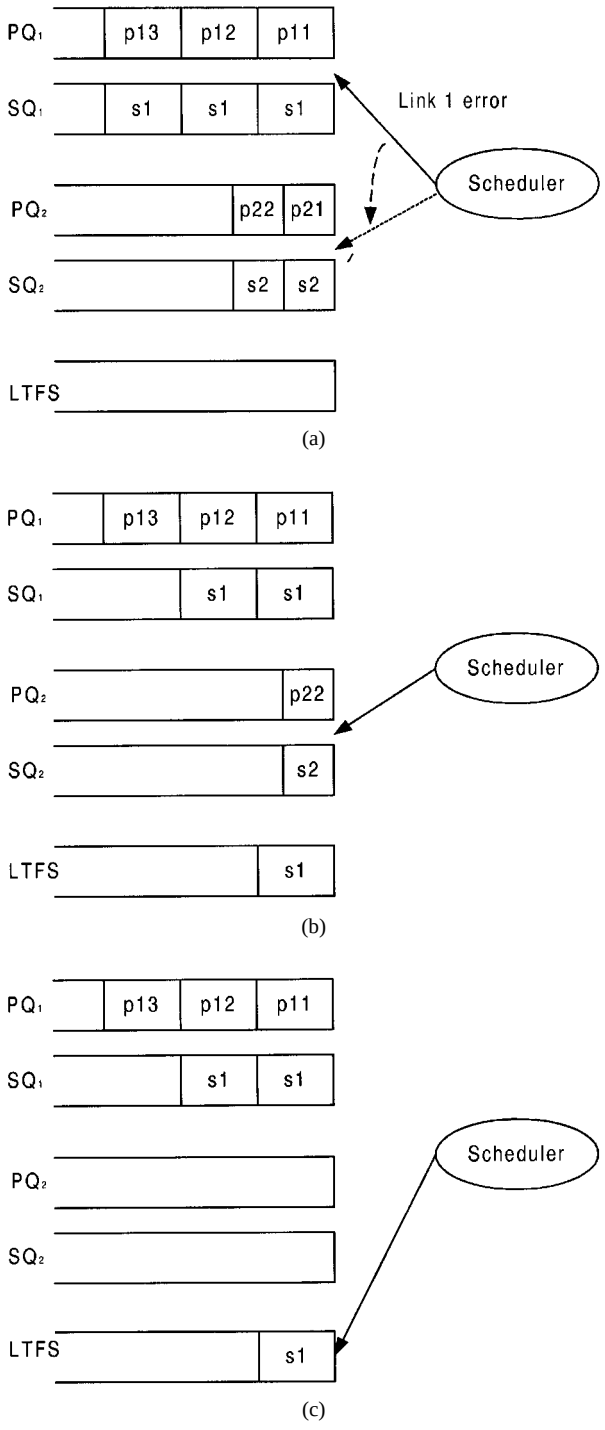

Fig. 2. An example of SBFA with RR. (a) Time 0. (b) Time 1. (c) Time 2.

SBFA can be integrated with arbitrary wireline scheduling policies, such as WFQ, WRR, or hierarchical fair service curve (H-FSC) [18]. The QoS characteristics of the algorithm depend on the wireline scheduling policy it is integrated with.

The structure of SBFA is simple and provides throughput guarantees. However, it has several drawbacks. Since SBFA is designed based on the reasoning that all flows whose wireless link is in a *good* state should always be served at its promised service rate and not a fraction of the promised rate, no restriction is imposed on flows receiving excessive service. Hence, a flow with a consistently good link may receive far more service than its promised share. Another problem is that LTFS needs preallocated network resources. For example, if LTFS is integrated with WFQ, LTFS must be guaranteed some service rate, which reduces the service capacity available to data flows and the number of flows admitted into the system. In addition, when several flows share an LTFS, the rate at which these flows receive compensation is determined only by the service rate of LTFS and is independent of the flows' allocated service rates. The information of different service rate requirements of different flows is lost in the compensation process. One possible solution to this problem would be to add more LTFS's, which certainly introduces more scheduling overhead. Finally, the algorithm does not work well if the packet size of a flow is variable. To keep in-order transmission of a flow, a slot in LFTS is always associated with the HOL packet of a flow. However, this HOL packet may not be the same packet the slot is originally associated with. If the packet size is variable, then the slot size in LFTS may not be consistent with the HOL packet size, which will cause problems in calculating and updating the *virtual time*.

B. I-CSDPS

Using a modified version of deficit round robin (DRR) scheduler [20] combined with an explicit compensation counter, a wireless scheduling algorithm called improved channel state dependent packet scheduling (I-CSDPS) is proposed in [29].

In DRR, each flow has its own queue, and the queues are served in a RR fashion. In each service round the number of packets served in each queue is determined by two parameters: deficit counter (DC) and quantum size (QS). DRR is basically a credit-based scheduling policy. The QS determines how much credit, in number of bits or bytes, is given to a flow in each round and DC keeps a record of the total credit received less the credit used. For each flow at the beginning of each round, a credit of size QS is added to DC. When the scheduler serves a queue, it transmits the first N packets in the queue, where N is the largest integer such that $\sum_{i=1}^{N} l_i \leq$ DC, where l_i is the size of the ith packet in the queue. After transmission DC is decreased by the amount of $\sum_{i=1}^{N} l_i$. If the scheduler serves a queue and finds that there are no packets in the queue, its DC is reset to zero.

To allow flows to receive compensation for their lost service opportunities due to link errors, I-CSDPS adds a compensation counter (CC) to each flow. CC keeps track of the amount of lost service for each flow. If the scheduler defers transmission of a packet because of link errors, the corresponding DC is decreased by the QS of the flow and the CC is increased by the QS. At the beginning of each round $\alpha \cdot$CC amount of credit is added to DC, and CC is decreased by the same amount, where $0 < \alpha \leq 1$. Fig. 3 gives an example of how the counters are updated. Assume there are two flows in the system with parameters $\text{QS}_1 = 100$, $\text{QS}_2 = 50$, $\alpha_1 = 1/3$, and $\alpha_2 = 1/2$. The status of the queues and the counters at the beginning of round n is shown in Fig. 3(a). The number marked on each packet represents the size of the

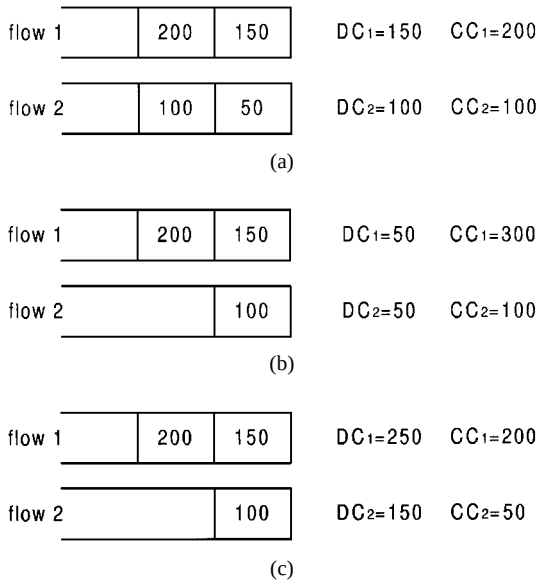

Fig. 3. An example of I-CSDPS scheduling ($QS_1 = 100$, $QS_2 = 50$, $\alpha_1 = 1/3$, and $\alpha_2 = 1/2$). (a) Beginning of round n, (b) end of round n, and (c) beginning of round $n + 1$.

packet. Assume no packet arrives thereafter, and link 2 is always in a *good* state while link 1 is in a *bad* state at round n. In round n, transmission of the HOL packet of flow 1 is deferred due to link error, and the HOL packet of flow 2 is transmitted. Note the changes of counters at the end of round n. DC_1 is decreased by the size of QS_1 and CC_1 is increased by the size of QS_1. DC_2 is decreased by the size of the transmitted packet and CC_2 is not changed. At the beginning of the next round, as shown in Fig. 3(c), each flow's DC is credited with the amount of its QS. Besides this increase, the DC of flow 1 is increased by an extra amount of $\alpha_1 \cdot CC_1 = 100$, while CC_1 is decreased by the same amount.

To avoid problems caused by unbounded compensation, upper limits are imposed on deficit counters. That is, at any time the credit accumulated in a DC can not exceed a certain value $DC_{\max}$. To increase the flexibility in compensation and resource allocation, the amount of credit transferred to a CC after a packet transmission is deferred and the parameter α, which decides the portion of compensation credit transferred to a DC, can be dynamically changed. Reference [19] describes a possible way of implementing such dynamical changes based on the load of the system, i.e., fast compensation when the system is lightly loaded and slow compensation when the system is heavily loaded.

Unlike SBFA, in I-CSDPS the rate at which a flow with lost service is compensated is associated with its allocated service rate. Also the compensation rate can be dynamically changed according to system load, thus increasing flexibility. Another advantage of I-CSDPS is its ability to handle variable-sized packets. When all the deficit counters are bounded, packet delay can be bounded. However, this bound is very loose and dependent on the number of active flows. Consider a packet arriving at the head of a queue j, which has just finished receiving service. Then the longest time it has to wait before being served is $\sum_{i=1, i\neq j}^{n} D_{\max, i}/C$, where $D_{\max, i}$ is the upper limit of the deficit counter of flow i, n is the number of active flows, and C is the transmission bandwidth. In other words, the packet can be delayed by a DC's worth by every other flow. Like the parameter B in I-WFQ, the choice of $DC_{\max}$ also represents a tradeoff between the packet delay bound and the degree of compensation a flow is allowed to receive for its lost service due to link errors. In addition, CSDPS has the same problem as SBFA in the sense that it does not impose any restriction on flows receiving excessive service.

VII. Algorithms Specifically Designed for Wireless ATM MAC

Up to now, all of the wireless scheduling algorithms discussed are in a very general form—they are not designed for any specific multiple-access control (MAC) protocol. Appropriate modifications to the algorithms are needed in order to integrate them with MAC protocols and apply them to a specific type of wireless networks, such as wireless asynchronous transfer mode (ATM) networks. In this section, we would like to discuss some algorithms specifically designed for wireless ATM networks.

Based on the cell structure, wireless ATM networks extend wireline ATM services into the wireless environment [22], [23]. Packet scheduling in wireless ATM has some special concerns related to MAC framing structures and heterogeneous traffic classes. Wireless ATM must support constant bit rate (CBR), variable bit rate (VBR), available bit rate (ABR), and unspecified bit rate (UBR) traffic classes. Different QoS requirements, such as cell loss ratio (CLR) and cell transfer delay (CTD), are specified for different classes of traffic. Most of the TDMA-based MAC protocols in wireless ATM networks use fixed-size MAC protocol data units (MPDU). Time is divided into slots and a number of slots are grouped together to form a frame, whose size could be variable. Within a frame, some slots are used for uplink transmission requests; and some are used for data transmission; and some are used by the base station for broadcasting slot allocation. The base station needs to collect all the transmission requests, and decide appropriate frame partitioning and slot allocation for packet transmissions. Many wireless ATM MAC protocols, such as D-TDMA [24], PRMA/DA [25], and MASCARA [26], fall into this generalized category. In such a system, the scheduling algorithm is coupled closely with the MAC protocol, and its main functions consist of allocating transmission slots and packing slots into frames.

In the early work of designing MAC protocols for wireless ATM networks, the issue of related scheduling algorithm design has not been emphasized. Simple FIFO service disciplines combined with different priority assignments to different classes are used to serve the transmission requests. CBR and VBR traffic are given higher priority than ABR and UBR traffic in receiving service. Slots could be reserved for CBR and UBR slots if they have enough traffic demand. Within a priority class, requests are served in a FIFO order. The default scheduling algorithm of D-TDMA and the dynamic allocation (DA) algorithm of PRMA/DA both follow the above scheduling policy.

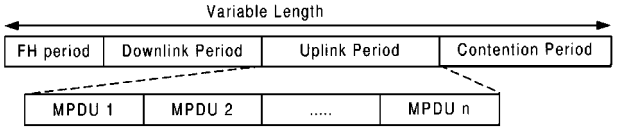

Fig. 4. Frame structure of MASCARA.

Table 1
Traffic Classes and Service Priorities in PRADOS

Priority No.	Traffic Class
5	CBR
4	Real-time VBR
3	Non-real-time VBR
2	ABR
1	UBR

A more sophisticated algorithm, named prioritized regulated allocation delay oriented scheduling (PRADOS) [27] is developed for traffic scheduling in wireless ATM. PRADOS is designed to work with a specific MAC protocol, MASCARA [26]. Fig. 4 shows the time frame structure of MASCARA. The FH period is the frame header, which is the descriptor of the current frame. The remainder of the frame consists of a downlink period for downlink traffic, an uplink period for uplink traffic and a contention period. All the periods are of variable lengths, depending on the instantaneous traffic, and all are further subdivided into a variable number of time slots. The mobile hosts use contention slots to transmit uplink transmission requests for subsequent frames and other control information.

PRADOS is based on the idea of Backward Earliest Due Date First (B-EDF) with priority [28] and combines it with a leaky bucket traffic regulator. Each connection is associated with a priority number according to which traffic class it belongs. The traffic classes and the priority assignments are shown in Table 1. The greater the priority number, the higher the priority of a connection. Additionally, a token pool is introduced for each connection. Tokens are generated at a fixed rate equal to the mean cell rate, and the size of the pool is the maximum burst size of a connection. In ATM, the information of the traffic characteristics of a connection can be obtained from the traffic contract of the connection. There are no bandwidth guarantees for UBR traffic, and so no token pool is available for UBR connections. The scheduler takes into account the priority class, contractual characteristics of the traffic, and delay constraints in slot allocation.

At the beginning of each frame, the scheduler collects a number of pending requests for slot allocation. The scheduler's operation can be separated into two parts, which are executed in parallel: 1) determining the number of requests for slot allocation from each connection that should be served in the current frame; and 2) determining the exact location in the frame of the allocated slots.

The first operation can be further divided into two steps. In the first step, the scheduler services "conforming" requests, defined as requests that belong to connections whose token pool is nonempty. UBR connections are not served in this step. Following the priority table, the scheduler serves the requests of each connection, as long as slots are available and the connection's token pool is not empty. Every time a slot is allocated to a connection, a token is removed from that connection's token pool. Within the same priority class, the scheduler gradually allocates one slot at a time to the connection that has the most tokens left in its token pool. The first step finishes when all requests have been served or all token pools are empty. If all token pools are empty and there are still requests pending, the scheduler moves to the second step to serve "nonconforming" requests. It restarts allocating slots for connections, starting from the highest priority (CBR) down to the lowest priority (UBR). The same procedure as described above is repeated until all the requests are served.

The second part of the operation is based on the idea that, in order to maximize the fraction of ATM cells that are transmitted before their deadlines, each ATM cell should be scheduled for transmission as closely to its deadline as possible. Deadlines are calculated according to ATM connections' cell delay tolerance (CDT) requirements. The initial ordering of the slots are decided following the above idea. Then the scheduler performs some necessary slot shifting and packing operations so that deadlines are not violated and downlink and uplink transmissions are within their respective periods. To achieve high utilization of the wireless channel, PRADOS is work-conserving. This part of the operation is very similar to the B-EDF algorithm. A detailed description can be found in [27].

PRADOS is a good example of how a scheduling algorithm is tied to a specific MAC protocol. The main advantage of PRADOS is that it reduces the average cell delay and the cell loss rate by taking into account the cells' timing constraints. Furthermore, since traffic is leaky bucket regulated, combined with appropriate admission control, PRADOS can provide delay bound for error-free connections. But PRADOS has no mechanism to deal with wireless link variability. Slot allocations are independent of the wireless link states. As a result, the effective wireless channel utilization will be lowered when link errors occur.

VIII. Further Discussion and Comparison

We summarize the properties of the presented algorithms in Table 2. These properties have already been discussed in the previous sections, where we introduced the algorithms. Delay bound is defined only for flows with error-free wireless links. Long-term throughput guarantee means that as long as a flow has enough service demand and the link errors it experiences are sporadic, its throughput over a sufficiently long period can be maintained above a certain value. For short-term fairness, we follow the definition in [7]. In fact, CIF-Q is the only algorithm addressing the issue of short-term fairness.

Table 2
Comparison of the Algorithm Properties

	Delay Bound	Long-term Throughput Guarantee	Mechanism for Wireles Link Variability	Short-term Fairness	Pre-allocated Resource for Compensation	Integrated with MAC protocol
CSDPS			√			
CSDPS+CBQ		√	√			
IWFQ	√	√	√			
CIF-Q	√	√	√	√		
SBFA		√	√		√	
I-CSDPS	√	√	√			
PRADOS	√					√

There are other related works that may provide insights in wireless scheduling. For the sake of completeness, they are briefly discussed below. A modified version of earliest due date (EDD), called feasible earliest due date (FEDD), is proposed in [30] for scheduling real-time traffic with deadlines in wireless networks. The FEDD policy is a scheduling policy that always schedules the packet with the earliest time to expire and whose link is currently in a *good* state. It is proven in [30] that for a large range of channel parameters and initial queue lengths and deadlines, the FEDD policy is optimal in terms of minimizing packet losses caused by deadline expiry. However, packet loss rate is the only performance criterion. No fairness issue and other QoS guarantees are discussed. Chen *et al.* propose a scheduling algorithm called priority RR with dynamic reservation update and error compensation for wireless ATM networks in [32]. One noteworthy point of this algorithm is that it tries to save mobile hosts' power by always allocating contiguous slots in a MAC frame to each mobile even if the mobile has more than one data flow. Therefore, a mobile only needs to turn on the transmitter/receiver once in the schedule broadcast phase and in the data transmission/receiving phase. Reference [31] integrates self-clocked fair queueing (SCFQ) [13] with a general form of TDMA/TDD-based MAC protocol for scheduling in wireless ATM networks. It is suggested in [31] that in case of link errors, the affected flow will get credits and free resources will be allocated to the affected flows. However, the paper does not describe the exact procedure of this compensation process. Because of the nature of SCFQ, the scheme is able to provide differentiated QoS for different traffic classes.

We believe the following is the set of objectives that a good wireless packet scheduling algorithm should try to achieve.

1) Provide long-term fairness and throughput guarantees for flows with error-free links or sporadic link errors.
2) Achieve high wireless channel utilization.
3) Minimize packet loss.
4) Provide delay (jitter, if possible) bound for flows with error-free links or sporadic link errors.
5) Support multiple classes of traffic with QoS differentiation.
6) Achieve low power consumption in mobile hosts.
7) Achieve medium algorithm complexity.

Indeed, it is impossible and not necessary to design an "optimal scheduler" to achieve all the above objectives, because of the conflicting nature of some objectives. Appropriate tradeoffs should be made depending on the system characteristics and service requirements.

IX. Future Work and Open Questions

Although the algorithms discussed provide some possible solutions to the packet scheduling problem in wireless networks, there are some issues that have not been addressed and are potential research topics.

A. Adaptive Error-Correction Coding and Deferment of Transmissions

Two approaches can be used to reduce unproductive transmissions in case of link errors. One is for the scheduler to defer transmissions, as discussed in this paper. The other approach is to use error-correction code. Adaptive error-correction coding schemes [33], [34] adjust the code rate according to the quality of the wireless channels to offset the errors. Both approaches have their advantages and disadvantages. Since perfect knowledge of the link states is not possible, for the approach of deferring transmission, the scheduler has to regularly "poll" the links by some means, such as sending RTS-CTS packets, in order to gain accurate knowledge of link states. This kind of regular "polling" introduces overhead. Also, for some packets with very stringent timing constraints, deferring transmissions may cause these packets to be dropped because of deadline expiry. To expedite the transmissions, error-correction code can be used. However, error-correction code introduces transmission overhead to a data packet. And this overhead increases with

the error-correcting capability. If, instead of using error-correction code, the scheduler defers transmissions on an error link and uses this bandwidth share to serve other packets with better links, the overhead of error-correction code will be lowered. Thus, the overall wireless channel utilization will be improved. In addition, the error-correction capability of the error-correction code in a data packet is limited. Therefore, in those situations where the links suffer from severe errors, error-correction coding may not work. Then deferment of transmission is the only choice. Although research has been conducted on the two approaches individually, little has been done to quantitatively investigate the tradeoffs of the two approaches. After the tradeoffs have been analyzed, we may try to combine the two approaches to design new wireless scheduling algorithms.

B. Scheduling in CDMA Networks—Multiple Servers and Multiple Link States

Most of the previous and current research in wireless scheduling focus on the scheduling issue of one single server, such as in the typical TDMA network. In such a system, the scheduler can serve only one packet at a time. However, in a CDMA network, multiple packets can be transmitted by the base station simultaneously. This corresponds to the multiple server case. Reference [35] addresses the issue of scheduling in a slotted CDMA network based on bit error rate (BER) requirements. The scheduler [35] tries to schedule packets based on their BER requirements while maintaining high utilization of the resources. However, fairness issues, delay bounds and throughput guarantees are not discussed. Since there is not much related work available, the question of how to perform scheduling in a wireless network with multiple servers is still open.

Almost all papers on wireless scheduling models wireless links by a two-state Markov chain. A link has full capacity when it is in a good state and has zero (or almost zero) capacity when it is in a bad state. Correspondingly, a flow's packets are either transmitted at full output rate or not served at all. However, this is not the case in a CDMA network. In particular, for a flow in a CDMA network, BER and packet error rate are dependent on its data rate [36]. When the transmission power of a data flow is fixed, BER increases as the data rate increases. Therefore, in CDMA networks, one way of dealing with link errors is to reduce the affected flows' data rates. (This approach is especially useful for best-effort traffic as in the Internet because no QoS guarantees are associated with best-effort traffic.) Since in reality the physical capacity of a wireless link does not jump only between zero capacity and full capacity, the link can have multiple states, at each of which the link has different physical capacity. As a result, the rate at which packets are transmitted can have multiple levels. Therefore, the scheduler needs to efficiently schedule packet transmission on links having more than two states. To the best of our knowledge, no work has been done in this area. In summary, to resolve the packet scheduling problem in CDMA networks, issues of multiple servers and multiple link states need to be considered.

C. Integration of Admission Control, Scheduling, and Congestion Control

Since the wireless channel is bandwidth-limited and highly variable, in order to provide guaranteed service to mobile hosts the scheduling algorithm must be supported by appropriate admission control and congestion control schemes. The performance of a scheduling algorithm is dependent upon these two components. Studying how a scheduling algorithm is affected by the admission control and congestion control policies in a wireless environment, and how to "optimally" integrate admission control, scheduling and congestion control require further study.

X. Conclusion

In this paper, we presented a comprehensive and in-depth survey on current research in wireless packet scheduling. The major issues in wireless scheduling were discussed. Various representative algorithms were analyzed and compared. We also proposed some future research directions.

References

[1] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM'89*, 1989, pp. 3–12.

[2] D. Kandlur, K. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," in *Proc. 11th Int. Conf. Distributed Computer System*, May 1991, pp. 300–307.

[3] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 365–386, Aug. 1995.

[4] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM'90*, Philadelphia, PA, Sept. 1990, pp. 19–29.

[5] J. Wroclawski, "The use of RSVP with IETF integrated services," RFC2210, Sept. 1997.

[6] S. Blake *et al.*, "An architecture for differentiated services," RFC2475, Aug. 1998.

[7] T. S. Eugene Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *Proc. INFOCOM98*, Mar. 1998, pp. 1103–1111.

[8] P. Bhagwat, A. Krishna, and S. Tripathi, "Enhancing throughput over wireless LAN's using channel state dependent packet scheduling," in *Proc. INFOCOM96*, Mar. 1996, pp. 1133–1140.

[9] S. Lu and V. Bharghavan, "Fair scheduling in wireless packet networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 473–489, 1999.

[10] C. Fragouli, V. Sivaraman, and M. Srivastava, "Controlled multimedia wireless link sharing via enhanced class-based queuing with channel-state dependent packet scheduling," in *Proc. INFOCOM'98*, vol. 2, Mar. 1998, pp. 572–580.

[11] H. Wang and N. Moayeri, "Finite state Markov channel—A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, pp. 163–171, Feb. 1995.

[12] J. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queueing," in *IEEE INFOCOM'96*, vol. 1, San Francisco, CA, Mar. 1996, pp. 120–128.

[13] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. IEEE INFOCOM'94*, Toronto, Canada, June 1994, pp. 636–646.

[14] 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.

[15] P. Goyal, H. M. Vin, and H. Chen, "Start-time fair queueing: A scheduling algorithm for integrated services," in *Proc. ACM-SIGCOMM 96*, Palo Alto, CA, Aug. 1996, pp. 157–168.

[16] T. S. E. Ng, I. Stoica, and H. Zhang. (1998, Mar.) Packet fair queueing algorithms for wireless networks with location-dependent errors. Carnegie Mellon Univ. [Online]. Available: ftp://ftp.cs.cmu.edu/user/hzhang/INFOCOM98at.ps.Z
[17] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks," in *ACM/IEEE MOBICOM'98*, Dallas, TX, pp. 1–9.
[18] I. Stoica, H. Zhang, and T. S. E. Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services," in *Proc. ACM SIGCOMM'97*, Sept. 1997, pp. 249–262.
[19] J. Gomez, A. T. Campbell, and H. Morikawa, "A systems approach to prediction, compensation and adaptation in wireless networks," in *ACM WOWMOM '98*, Dallas, TX, Oct. 1998, pp. 92–100.
[20] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *Proc. ACM SIGCOMM'95*, Berkeley, CA, 1995, pp. 231–242.
[21] *IEEE Standard for Wireless LAN Medium Access Control (MAC) and (PHY) Specifications, 802.11*, Nov. 1997.
[22] D. Raychaudhuri, "Wireless ATM networks: Architecture, system design and prototyping," *IEEE Pers. Commun.*, pp. 42–49, Aug. 1996.
[23] E. Ayanoglu, K. Y. Eng, and M. J. Karol, "Wireless ATM: Limits, challenges, and proposals," *IEEE Pers. Commun.*, pp. 18–54, Aug. 1996.
[24] D. Raychaudhuri and N. D. Wilson, "ATM-based transport architecture for multiservices wireless personal communication networks," *IEEE J. Select. Areas Commun.*, vol. 12, no. 8, pp. 1401–14, 1995.
[25] J. G. Kim and I. Widjaja, "PRMA/DA: A new media access control protocol for wireless ATM," in *Proc. IEEE ICC'96*, Dallas, TX, June 1996, pp. 240–244.
[26] N. Passas, S. Paskalis, D. Vali, and L. Merakos, "Quality-of-service-oriented medium access control for wireless ATM networks," *IEEE Commun. Mag.*, pp. 42–50, Nov. 1997.
[27] N. Passas, L. Merakos, and D. Skyrianoglou, "Traffic scheduling in wireless ATM networks," in *Proc. IEEE ATM Workshop*, 1997, pp. 391–400.
[28] C. Han and K. G. Shin, "Scheduling MPEG-compressed video streams with firm deadline constraints," in *Proc. ACM Multimedia'95*, San Francisco, CA, Nov. 1995, pp. 411–422.
[29] J. Gomez, A. T. Campbell, and H. Morikawa, "The Havana framework for supporting application and channel dependent QoS in wireless networks," in *Proc. ICNP'99*, Nov. 1999, pp. 235–244.
[30] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," in *Proc. ACM WOWMOM '99*, Seattle, WA, 1999, pp. 35–42.
[31] R. Sigle and T. Renger, "Fair queueing wireless ATM MAC protocols," in *Proc. IEEE PIMRC'98*, vol. 1, 1998, pp. 55–59.
[32] J. Chen, K. Sivalingam, P. Agrawal, and R. Acharya, "Scheduling multimedia services in a low-power MAC for wireless and mobile ATM networks," *IEEE Trans. Multimedia*, vol. 1, no. 2, pp. 187–201, June 1999.
[33] S. Yajnik, J. Sienicki, and P. Agrawal, "Adaptive coding for packetized data in wireless networks," in *Proc. PIMRC'95*, vol. 1, 1995, pp. 338–342.
[34] M. Elauod and P. Ramanathan, "Adaptive use of error-correction codes for real-time communication in wireless networks," in *Proc. INFOCOM'98*, Mar. 1998, pp. 548–555.
[35] I. F. Akyildiz, D. A. Levine, and I. Joe, "A slotted CDMA protocol with BER scheduling for wireless multimedia networks," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 146–158, 1999.
[36] R. L. Pickholtz, L. B. Milstein, and D. L. Schilling, "Spread spectrum for mobile communications," *IEEE Trans. Veh. Technol.*, vol. 40, pp. 313–322, May 1991.

Yaxin Cao received the B.S.E.E. degree (*honors*) from Tsinghua University, Beijing, China, in 1997.

He is currently a Ph.D. student at the Department of Electrical Engineering, University of Southern California, Los Angeles. His research interests include wireless networks, broad-band networks, and multimedia communications. He worked at the Nokia Research Center, Boston, MA, in 1999 as a Summer Intern and was awarded the Nokia Professional Work Experience Scholarship.

Mr. Cao is a member of Phi Kappa Phi.

Victor O. K. Li (Fellow, IEEE) was born in Hong Kong in 1954. He received the S.B., S.M., E.E., and Sc.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, in 1977, 1979, 1980, and 1981, respectively.

He joined the University of Southern California (USC), Los Angeles, in 1981, and became Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since 1997, he has been with the University of Hong Kong, Hong Kong, where he is Chair Professor of Information Engineering at the Department of Electrical and Electronic Engineering, and Managing Director of Versitech Ltd., the technology transfer and commercial arm of the University. He also serves on various corporate boards. His research is in information technology, including high-speed communication networks, personal communication systems, and distributed multimedia systems. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. He chaired the Computer Communications Technical Committee of the IEEE Communications Society 1987–1989, and co-founded the International Conference on Computer Communications and Networks.

Prof. Li is an editor of various ACM and IEEE journals and has given keynote addresses and served on the boards of numerous international conferences.