"Adaptive Packet Scheduling and Resource Allocation in Wireless Networks"

by

Yaxin Cao

CSI-02-12-07

USC VITERBI SCHOOL OF ENGINEERING
UNIVERSITY OF SOUTHERN CALIFORNIA
ELECTRICAL ENGINEERING — SYSTEMS
LOS ANGELES, CA 90089-2565

ADAPTIVE PACKET SCHEDULING AND RESOURCE ALLOCATION IN WIRELESS NETWORKS

by

Yaxin Cao

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

December 2002

Dedication

To my parents and Jie

Acknowledgements

I would like to express my sincere gratitude towards my advisor, Prof. Victor O. K. Li, for his continuous support, encouragement and guidance throughout the duration of my Ph.D. studies. The valuable technical knowledge, research methodology and paper-writing skills I learned from Prof. Li are essential for me to overcome all difficulties in both study and research.

I would also like to thank Prof. Daniel C. Lee, Prof. P. Vijay Kumar, Prof. Ashish Goel, and Prof. Zhen Zhang for serving in my qualifying and dissertation committees, and providing valuable comments on my research work and dissertation.

I am especially grateful to Dr. Lei Zhuge for all his help in both my work and my daily life. His kind assistance to me after I just arrived in Los Angeles and Hong Kong made those transitions in my life much easier. Thanks to his serious attitude towards research and sound knowledge, working and collaborating with him was a pleasant experience. My appreciation is also extended to Dr. Guangliang Li and Prof. Lawrence K. Yeung for the deep discussions on some technical issues in my research.

Special thanks are given to Diane Demetras, Milly Montenegro, Mayumi Thrasher, Danita Lee and Dr. Dixon Johnson for helping me with all the administrative issues at USC and HKU.

Finally, many thanks to my USC and HKU colleagues and friends, Dr. Tat-Keung Chan, Dr. Spiridon Bakiras, Dr. Ka-Cheong Leung, Dr. Guangcai Zhou, Dr. Zhourong Miao, Hezhi Ai, Hongtao Zhu, Mingrui Zhu, Peng Ye, Zaichen Zhang, Fung Lam and Yurong Hu for their support and friendship from which I benefited greatly.

Contents

List of Tables

List of Figures

Abstract

Providing Quality of Service (QoS) differentiation and guarantees in wireless networks is a challenging task due to the special problems of wireless communications, such as link variability and limited bandwidth. New technology needs to be developed to tackle these problems.

Wireless packet scheduling, a key component of QoS provisioning in wireless data networks, is the major focus of this dissertation. Although many wireless packet scheduling algorithms have been proposed in recent years, some issues remain unresolved. We study packet scheduling issues in both TDMA-based and CDMA-based wireless networks and propose three novel wireless scheduling algorithms applicable to different link and network models. First, we design a scheduling algorithm for TDMA networks with wireless links being modeled as having two states. The algorithm is shown to distribute excess bandwidth effectively, strike a balance between effort-fair and outcome-fair, and provide delay bound for error-free flows and transmission effort guarantees for error-prone flows. Second, extending the previous work, we propose a novel wireless scheduling algorithm applicable to wireless links with multiple states. For delay-sensitive flows, the algorithm is capable of providing statistical delay violation bounds. For best-effort flows, we propose a new notion of fairness, called long-term link-quality-weighted outcome-fair. The algorithm balances between bandwidth efficiency requirement and fairness requirement, and guarantees minimal goodput levels for best-effort flows. Third, an adaptive packet scheduling algorithm for cellular CDMA networks is proposed. The algorithm guarantees packet deadline and average data rate under the assumption of perfect power control. It is also power-efficient because it takes into account the varying channel conditions.

Due to the special characteristics of wireless mobile environment, fixed level of service guarantees and fixed level of resource allocation, commonly used in wireline networks, are not suitable and viable in wireless networks. In the last part of our dissertation, we propose a general utility-oriented adaptive QoS model for wireless networks and establish a framework for formulating the bandwidth allocation problem for users with time-varying links. Based on the framework, a high-level adaptive bandwidth allocation scheme, which guarantees utility-based QoS, ensures long-term fairness, and achieves high bandwidth efficiency, is designed.

Chapter 1

Introduction

With the developement of 3G (3rd Generation) wirless communication technologies and the rollout of wireless data services such as CDPD (Cellular Digital Packet Data) [1], and GPRS (General Packet Radio Service) [5], the future wireless networks are evolving toward broadband data networks. It is expected that future wireless networks should provide packet data services for heterogeneous classes of traffic with different QoS (Quality of Service) requirements [11], [43]. The characteristics of wireless communication pose special problems that do not exist in wireline networks. These include: (i) high error rate and bursty errors, (ii) location-dependent and time varying wireless link capacity, (iii) scarce bandwidth, (iv) user mobility, and (v) power constraint of the mobile hosts. Due to the above problems, the service models, frameworks and solutions designed for providing QoS guarantees and differentiations in wireline networks are not directly applicable in wireless networks. Currently there is an urgent need to develop new technologies for providing QoS differentiation and guarantees in wireless networks.

1.1 Packet Scheduling in Wireless Networks

Among all the technical issues that need to be resolved in wireless networks, packet scheduling 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 [61], such as Fair Queueing [6], [19], [44], CBQ (Class-based Queueing) [22], and EDD (Earliest Due Date) [34], can not be directly applied in wireless networks, because they do not consider the varying wireless link capacity and the location-dependent channel state.

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. Besides the time-dependent problem, wireless link capacity is also location-dependent. A base station can typically 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 experience high error-rate. 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 dynamic mechanisms to deal with these time-dependent and location-dependent changes.

When designing wireless scheduling algorithms, there are four major issues to be considered, namely, fairness, QoS guarantees, bandwidth efficiency, and simplicity.

Scheduling fairness in wireline networks is usually guaranteed by dedicating a certain service rate to a flow or a user, and the scheduling algorithm provides isolation mechanisms among different flows. 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. Since losses and errors are unavoidable in wireless links, the bandwidth allocated to a flow may not be equal to the actual goodput[1] achieved. Due to location-dependent errors, different users may have communication links with different statistics depending on their moving speeds and locations. Therefore, there exists a discrepancy between the transmission capacity (bandwidth) allocated to a flow and the actual goodput achieved. As suggested by many wireless scheduling algorithms [12], swapping service opportunities are allowed to improve channel utilization. That is, when a flow is scheduled to transmit but its link is relatively bad, it may give up its current service opportunity to some other flow with better links. To ensure fairness, the flow should be compensated for this temporary loss of service

[1]the total amount of successful transmissions

later when its link recovers. But determining how to compensate for it is not an easy task. Some fairness definitions for wireline scheduling algorithms are available in [6] and [26]. The definition and objectives of fairness guarantees become more ambiguous in a wireless environment. The granularity of fairness, i. e., short-term fairness vs. long-term fairness, is another factor that affects the scheduling policy. To define fairness, the following questions need to be answered. Should we guarantee bandwidth or goodput to the flows? Should we try to achieve both short-term fairness and long-term fairness or only long-fairness is necessary? The appropriate interpretation of fairness should depend on the service model and wireless link characteristics.

Broadband 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. Depending on the service requirements, either simple prioritization or more advanced per-flow guarantees, such as delay and throughput, should be provided by the scheduling algorithm. If a wireless link experiences very frequent channel degradations, it is very difficult to guarantee QoS for the flows using this link. Nevertheless, deterministic or statistical QoS guarantees should be provided for the error-free flows or flows on those links where only very few errors exist.

The most precious resource in wireless networks is the bandwidth. Under the constraints of fairness and QoS guarantees, an efficient wireless scheduling algorithm should aim to minimize unproductive transmissions on erroneous links, and at the same time, maximize the effective service delivered and the efficiency of the overall wireless channel.

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. Although mobile hosts usually are not responsible for computation of scheduling sequences, sending the base station signaling messages, which may contain information on mobile hosts' queue status, packet arrival times or and link states, demands transmission powers at the mobile host. 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 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.

1.1.1 Packet Scheduling in TDMA Networks

1.1.1.1 Scheduling for Links with Two States

Recently, new packet scheduling algorithms which account for the special characteristics of the wireless environment have emerged. One of the first scheduling algorithms that address the problem of location-dependent and bursty errors in wireless networks is CSDPS (Channel State Dependent Packet Scheduling) [8]. CSDPS introduces the idea of deferring transmission when a flow is experiencing link errors and swaps the transmission opportunity with some other flow with a better link. CSDPS+CBQ (Classed-based Queueing) [23] adds CBQ component to the original CSDPS and outperforms CSDPS in terms of fair sharing of the wireless channel. However, it does not have an explicit mechanism for compensating those mobile users who have previously lost their service share because of link errors. Thus, there is no guarantee on the rate at which a mobile user will be compensated. In SBFA (Server Based Fair Approach) [48], 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. A flow which has lost its service opportunity because of link errors will accumulate credits in an LTFS. By recording the service loss of each flow in LTFS, and dedicating part of the bandwidth to LTFS, flows which have lost their service share because of link errors will eventually be compensated. However, in SBFA 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. In addition, LTFS requires pre-allocated network resources, which reduces the service capacity available to data flows and the number of flows admitted into the system. Using a modified version of DRR (Deficit Round Robin) scheduler combined with an explicit compensation counter, a wireless scheduling algorithm called I-CSDPS (Improved Channel State Dependent Packet Scheduling) is proposed in [27]. Besides the deficit counter maintained in the standard DRR, each flow has a compensation counter which has a function similar to LTFS in SBFA. When a flow is served in each round the amount of service it receives is determined by the sum of its deficit counter and a portion of the compensation counter. I-CSDPS has the same problem as SBFA in the sense that it does not impose any restriction on flows receiving excessive service.

IWFQ (Idealized Wireless Fair Queueing) [39], CIFQ (Channel-condition Independent Packet Fair Queueing) [42] and WFS (Wireless Fair Service) [40] are all based on wireline fair queueing. They all use an error-free fair queueing scheduling system as a reference. In the real error-prone system, a flow is defined as *leading*, *lagging* or *in sync*, at any time instant, if it has received more than, less than or the same as the service it should have received in the corresponding error-free system. In IWFQ, arriving packets are tagged with timestamps just as in WFQ. When the scheduler is ready to send, the packet with the minimum finish tag and with a good link is transmitted. 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. Although IWFQ has some appealing properties in fairness and QoS guarantees, as pointed out in [42] it is deficient in terms of short-term fairness and CIFQ is designed to address these problems. WFS tries to modify IWFQ to improve short-term fairness. Adopting some ideas from CIFQ, WFS allows graceful degradation for leading flows and distributes the compensation bandwidth among lagging flows according to their lags. WFS also tries to decouple the rate and delay requirements. Instead of only one weight in fair queuing, each flow is assigned two weights, rate weight and delay weight. A flow is drained into the scheduler according to the rate weight , but served according to the delay weight. WFS only compensates a backlogged flow denied service during its scheduled slot if some other flow transmitted a packet during this slot.

All the above algorithms assume that a wireless link is either in a *good* state or a *bad* state and errors are bursty. The state durations and transitions between states are random. Transmissions in a *good* state are always error-free and transmissions in a *bad* state always fail. To compare with the existing wireless scheduling algorithms, we first design a wireless packet scheduling algorithm for the two-state link model. We compare the algorithm we proposed with WFS and CIFQ, the most recent and most sophisticated scheduling algorithms using the two-state link model. Compared with other algorithms, WFS and CIFW have well-defined fairness objectives and tighter delay bounds. However, both algorithms have deficiencies, such as unrealistic link model, impractical timestamping method, no transmission effort guarantee for flows with high error rate, excess provision for error-free flows, etc. We will discuss more about such deficiencies in the next chapter.

1.1.1.2 Scheduling for Links with Multiple States

In reality the capacity of a wireless link does not jump between zero capacity and full capacity. To model the links more accurately, more states should be used instead of such a coarse two-state model. In particular, when adaptive modulation or adaptive coding is used, a flow clearly has different effective throughput levels for different modulation constellations or coding rates. In addition, since bandwidth wastage due to losses, errors, and coding overheads is unavoidable in wireless links, the bandwidth allocated to a flow may not be equal to the actual goodput achieved. The above algorithms have not addressed this issue.

ELFS [20] and CS-WFQ [38] brought up another perspective of the wireless scheduling issue. They address the issue of *outcome-fair* vs. *effort-fair*. Since in wireless networks part of the transmission effort/capacity is unavoidably wasted due to link errors, for the same amount of transmission effort, if the links have different error rates, the outcomes will be different. Effort-fair means each flow should receive transmission capacity in proportion to its assigned weight/rate. Outcome-fair means the achieved goodput of each flow should be in proportion to its assigned weight/rate. In the algorithms discussed previously, the flows on links with larger error rate always receives less transmission effort than the ones with better links. In ELFS and CS-WFQ, the authors argue that some flows may be more important than others, and thus deserves more transmission effort even if its link's error rate is high. ELFS and CS-WFQ tries to achieve outcome fair by increasing the fair queueing weight of the flows with larger error rates. To overly reducing the overall bandwidth efficiency, an upper bound for the weight is set. Although [20] and [38] point out the tradeoff between effort-fair and outcome fair, the algorithms proposed are too simplistic and have 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 receive irrespective of their rate guarantees or link status. Therefore, even a well-behaving flow with error-free link may not get its fair share of service.

Similarly, [32] also associates the weight assignment of classical fair queueing algorithm to the link efficiency. Specifically, each flow's weight is determined by the link's efficiency raised by an exponent. By choosing different values of the exponent, the algorithm can adjust the bandwidth allocation among flows taking into account the flows' efficiencies. However, it assumes that a link's efficiency changes slowly, and between changes the efficiency is a constant. When a link's status changes rapidly, the schemes adapting to link variations by adjusting weights are not effective. Furthermore, it does not exploit the benefit of swapping service opportunities, namely, allowing a flow with a good link to transmit in place of a flow which is experiencing a bad link.

Besides the aforementioned problems, none of the existing scheduling algorithms addresses the issue of providing delay guarantees for flows with error-prone links. Nevertheless, for real-time and delay-sensitive applications delay guarantees are essential. Thus the scheduling algorithm should be able to provide certain delay guarantees even for error-prone flows if requested. For best-effort applications, the major concerns of the scheduling algorithm should be bandwidth efficiency and fairness. Since links may be different not only in instantaneous qualities but also in average qualities, to achieve perfect fairness and to maximize bandwidth efficiency are conflicting objectives in most cases. When links are fast-changing and possess multiple states, maintaining short-term fairness becomes impractical and unnecessary. In addition, when links differ in average quality, how to guarantee bandwidth for flows with inferior link qualities and balance between efficiency and fairness objectives have not been well studied.

Extending the research on scheduling for two-state wireless link models, we propose a novel scheduling algorithm, which applies to wireless links with multiple states and addresses the above issues. For flows requiring timely delivery, the algorithm provides statistical delay violation bounds even when the links are not perfect. For best-effort flows, we maintain that neither effort-fair nor outcome-fair is suitable in wireless networks. Instead, we propose the notion of *long-term link-quality-weighted* *outcome-fairness (LT-LQW outcome-fairness)*, which we believe is more meaningful considering the heterogeneous link qualities. In addition, we show that by using a very simple scheduling scheme we can guarantee certain effort and goodput levels for each best-effort flow, improve bandwidth efficiency and maintain LT-LQW outcome-fair.

1.1.2 Packet Scheduling in CDMA Networks

Most of the existing wireless packet scheduling algorithms focus on the scheduling in TDMA (Time Division Multiple Access) networks. In such a system, the scheduler can serve only one packet at a time. However, in a CDMA (Code Division Multiple Access) network, multiple packets can be transmitted by the base station simultaneously, those packet scheduling algorithms designed for TDMA-based single-server systems are generally not applicable. On the other hand, variable service rates provided by multi-code CDMA [31] can be used to increase flexibility in packet scheduling. Furthermore, the fast closed-loop power control enables the base station to gather fairly good estimates of the channel condition, which facilitates the design of packet scheduling algorithms with channel status awareness.

Unlike the large amount of work on scheduling in TDMA networks, there are much less existing packet scheduling algorithms for CDMA networks. [2] and [7] both designed slot-based algorithms, where a CDMA frame is divided into multiple slots for transmitting data and control packets to/from different users. In particular, [2] addresses the issue of scheduling in a slotted CDMA network based on bit error rate (BER) requirements. The scheduler [3] 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. [7] considers the problem of delivering non-real time data traffic, whose service requirement is specified in terms of a required average data rate over a specific interval, by one-by-one scheduling in a DS-CDMA system. The scheme, which allows only one user in each cell to transmit at a time, is shown to consume less energy compared with continuous transmission. While slotted-CDMA may achieve higher capacity from reducing intra-cell interference, it requires smart slot assignment algorithm (which is usually very difficult to design) and fundamental modifications of frame structure and media access control (MAC) protocol from the existing CDMA proposals.

In [50] packets are scheduled according to their calculated priorities and the system resource constraints. The priority of a packet is defined as inversely proportional to the remaining time before the packet expires. The adaptive CDMA scheduling algorithm we propose has the similar strategy to [50] in the sense that the standard CDMA frame structure and MAC protocol are maintained, and priority-based scheduling is performed. However, our packet scheduling algorithm is adaptive to not only the packet delay deadline required by the application, but also the deviation of data rate from its target value and the channel status variations. In the CDMA scenario the channel status is more coupled with power consumption instead of packet loss as in TDMA, and it is not yet considered by any of the existing CDMA packet scheduling methods. Our algorithm is the first one to take into account the issues of data delay tolerance, average rate fairness, and channel variations altogether. In addition, our algorithm works with realistic channel model instead of the simple two-state model used in the majority of TDMA packet scheduling methods (if at all they consider the issue). We also study and discuss the relation between packet scheduling and flow admission control as well.

1.2 Adaptive Service and Resource Allocation

Since the wireless channel is highly variable and wireless network users are mobile, fixed level of service guarantees and fixed level of resource allocation, commonly used in wireline networks, are not suitable and viable in wireless networks. We believe that a more flexible adaptive service model which allows variable QoS is needed. In an adaptive QoS service model, the user applications are required to be adaptable, and the resource and the service capacity allocated to the applications are not fixed, but adjusted according to the condition of the network. In such a service model, applications allow graceful service degradation while maintaining certain minimal service requirement. We maintain that an adaptive QoS service model is necessary and valid in wireless mobile networks due to the following reasons: (i) Highly volatile wireless links, unpredictable user mobility, and uncoordinated multiple access of the wireless channel make it impossible to impose very rigid QoS requirements; (ii) most of the new multimedia applications are designed to have certain adaptability by, for example, dynamically adjusting playback time or layered coding [41]; (iii) from the user's perspective, lower service quality within the tolerable region is certainly better than denial or disruption of service during adverse network conditions; (iv) the network resources can achieve high utilization, thus accommodating more users.

Wireless network service models that incorporate the concept of adaptive QoS can be found in [9], [10], [49], [54], [55]. However, none of the papers discusses the problems of location-dependent link variations and links with multiple physical states. The only channel variation some of these papers consider is the change of total available bandwidth/capacity. However, as we have pointed out, different users may experience different link capacity due to different locations. To combat wireless link variability and improve bandwidth utilization, we believe bandwidth should be allocated in an adaptive and link-state-dependent way. Irrespective of the amount of bandwidth received, the ultimate measure of the effectiveness of network services is the level of users' satisfaction, which is dependent on the specific application type. It is well known that the performance of an application does not always increase linearly as the bandwidth it receives increases. To capture the heterogeneity of different applications and to have a consistent performance measure, we propose a utility-oriented wireless adaptive QoS service model based on utility functions.

Furthermore, there exist two dimensions of adverse dynamics, namely, the dynamics of physical channels and the dynamics of users' requests. The dynamics of physical channels refer to the inherent variability of wireless channels, which is both time-dependent and location-dependent. In addition to the channel dynamics, the uncoordinated users' transmission requests make up the other dimension of the challenging dynamics. The broadcast nature of wireless communication causes conflicts or interference among users' transmissions. Since users generate their traffic independently, without any regulatory policy, they would contend for wireless channel access in an uncoordinated way. Also, mobility of the users adds to the dynamics of users' transmission requests. These requests can be call-based, burst-based or packet-based, each of which applies to a different time scale. In order to utilize the bandwidth efficiently and fairly, and to satisfy the users' service requirements, a resource allocation scheme should account for both dimensions of the dynamics of the wireless and mobile environment, and actively adapts to them.

There has been a great amount of work on wireless resource management, focusing on multiple access [3] and channel allocation [35]. Most of the previous work tackled one aspect of the bandwidth allocation problem, i.e. the dynamics of user requests. That is, resolving conflicts due to users' uncoordinated requests and allocating resources, such as transmission slots and call channels, appropriately to satisfy those requests. However, there is less research effort on adding explicit adaptive mechanisms to bandwidth allocation schemes to deal with the variability of wireless links.

In most wireless scheduling schemes [12], where a two-state Markov channel model is used, a user will not receive any bandwidth when its link experiences a long-lasting shadowing degradation. However, in reality, the capacity of a wireless link will have more than two states. When slow variations are dominant, a more desirable approach is to change both the code length and the amount of bandwidth allocated to a user as its link changes state. Thus a user will still be able to receive some service when its link quality degrades and the overall bandwidth will be utilized more effectively by allocating more bandwidth to users who can better utilize it. When fading and shadowing occur simultaneously, the fast variations are superimposed on slow variations, with the latter actually determining the short-term (in the order of seconds) average link quality. To improve bandwidth utilization, in addition to the swapping mechanism at the packet level, a high-level bandwidth allocation scheme should adjust the average bandwidth share (e.g. the scheduling weight) of each user as the average link quality changes.

Wireless links are usually subject to two types of variations, i.e. slow variations (shadowing) and fast variations (fading). For typical cellular communications, the duration of shadowing is in the order of seconds or tens of seconds, while fading usually lasts for milliseconds or shorter. Traditionally, low level mechanisms, such as error correction coding [21], [60] and swapping transmission opportunities in adaptive packet scheduling [12], [39], [42] are used to handle physical link variability. However, such mechanisms work for relatively small time scales, e.g. the duration of a bit or a packet, which are comparable to the duration of fast fading. For slow link variations, such mechanisms alone are inadequate. In adaptive coding, the decision of using which codeword for a packet or a symbol is usually only based on the instant wireless link condition of the particular user. A user with very bad link quality may waste a great amount of bandwidth on coding overhead. The overall effective bandwidth utilization and fairness among different users are not concerned. In most wireless scheduling schemes [12], where a two-state Markov channel model is used, a user will not receive any bandwidth when its link experiences a long-lasting shadowing degradation. However, in reality, the capacity of a wireless link will have more than two states. When slow variations are dominant, a more desirable approach is to change both the code length and the amount of bandwidth allocated to a user as its link changes state. Thus a user will still be able to receive some service when its link quality degrades and the overall bandwidth will be utilized more effectively by allocating more bandwidth to users who can better utilize it. When fading and shadowing occur simultaneously, the fast variations are superimposed on slow variations, with the latter actually determining the short-term (in the order of seconds) average link quality. To improve bandwidth utilization, in addition to the swapping mechanism at the packet level, a high-level bandwidth allocation scheme should adjust the average bandwidth share (e.g. the scheduling weight) of each user as the average link quality changes.

Since the dynamics of physical wireless links exist over a wide range of time scales (from microseconds to seconds), an ideal system should incorporate adaptive mechanisms working at all levels of time scales. Even in wireline broadband networks, using multi-level resource allocation mechanisms at different time scale is considered as necessary because of traffic dynamics [33]. In addition to the packet-level adaptive scheduling mechanisms we designed to combat fast link variations, we establish a very general modeling framework of the high-level bandwidth allocation problem based on the utility-oriened adaptive QoS service model and design an adaptive bandwidth allocation scheme to deal with slow link variations.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows. In Chapter 2, we first discuss the problems of existing packet scheduling algorithms for TDMA-based wireless networks with two-state link models. Then we propose a new wireless packet scheduling algorithm to solve these problems. In Chapter 3, we present a wireless packet scheduling algorithm for links with multiple states. Statistical delay violation bounds are provided for delay-sensitive traffics. A new notion of fairness suitable for wireless networks is propsed. Chapter 4 discusses the issues of adaptive scheduling in CDMA-based cellular networks. A novel adaptive scheduling algorithm, which takes into account data delay tolerance, average rate fairness, and channel variations altogether, is proposed. In Chapter 5, we propose a general utility-oriented adaptive QoS model for wireless networks and establish a framework for formulating the bandwidth allocation problem for users with time-varying links. We design an adaptive bandwidth allocation scheme based on the new service model. Finally, Chapter 6 concludes the dissertation.

Chapter 2

Wireless Packet Scheduling for Links with Two-states

2.1 Deficiencies of CIFQ and WFS

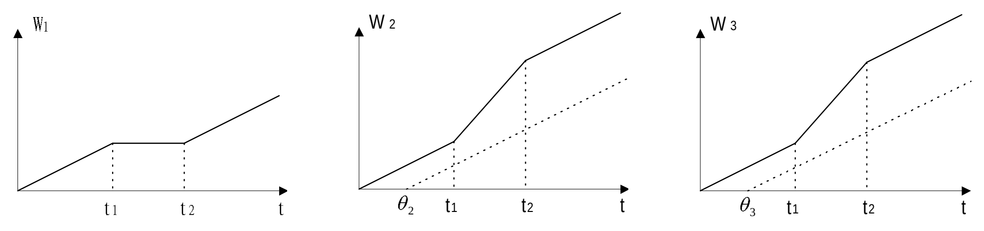
As you have discussed, CIFQ and WFS are the most recent and most sophisticated packet scheduling algorithms for two-state wireless link models. However, they still have some deficiencies. Since our proposed scheduling algorithm is designed to address these deficiencies, we now discuss them in detail.

First, as in most other wireless scheduling algorithms, they assume that each wireless link has two states, *good* and *bad*. The packet error probability in a *good* state is zero, while the packet error probability in a *bad* state is 100%. However, in reality the capacity of a wireless link does not jump between zero and one. A two-state link model is, in practice, established by setting a threshold of a link parameter [58], [63], such as SNR (Signal-to-Noise Ratio) or BER (Bit Error Rate)[1], which reflects the link quality. When the instant SNR or BER value is above the threshold, the link is considered *good*; otherwise, it is considered *bad*. Therefore, if each state is associated with one error probability, this value is in fact the average error probability corresponding to a range, not a single point, of SNR or BER. For example, even for a slowly moving user with moderate transmission rate, its BER ranges from 10^{-1} to 10^{-10} [63]. Since SNR or BER is a continuous function, no matter how the threshold is set, assuming zero loss for a *good* state and 100% loss for a *bad* state is unrealistic. Furthermore, the decision of whether a link is good during a packet's transmission is made before the packet transmission takes place, and 100% accurate prediction/estimation of a wireless link is never achievable in practice. Therefore, when a scheduler *believes* a link is in a *good* state, it may actually be in a *bad* state, or vice versa. Since the scheduling decisions are based on the scheduler's perception of the links, such estimation errors should also be accounted for.

Second, in WFS and CIFQ, a flow never transmits in a *bad* state, because they assume that nothing can be successfully transmitted in a *bad* state. However, as we have said, the capacity of a *bad* state is not necessarily zero and a *good* state can be incorrectly designated as a *bad* state. There are two undesirable consequences of such a policy. Firstly, flows experiencing, on the average, long *bad* states, and short

[1]given a fixed modulation scheme

good states, may be starved. In wireless networks, because of different locations and moving speeds, different users/flows may have different average durations for each link state. In WFS and CIFQ, flows with very bad links will receive much less service effort (bandwidth) than its allocated share while flows with relatively good links will receive excess bandwidth. Secondly, such a policy increases packet delay of flows experiencing long *bad* states. When a flow's link is in a *bad* state, it will not transmit at all. Then all the packets in the queue will be delayed for at least the duration of the *bad* state. 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 command 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. For example, when BER is 0.01 and the packet length is 255 bytes, the error probability of an uncoded packet is almost 1, while the error probability of a coded packet using (255,128) (50% information bits in a packet) Reed-Solomon code is less than 10^{-16}.



(a) service received by flow 1 (b) service received by flow 2 (c) service received by flow 3

Figure 2.1: Distribution of excess bandwidth in GPS

Third, since WFS and CIFQ are based on fair queueing, which in turn emulates GPS (Generalized Processor Sharing) [44], the excess bandwidth due to some flows being unbacklogged (or idle) is distributed among all the flows in proportion to their fair queueing weights. This is illustrated in Figure 2.1. In this figure, all flows have the same weight. Flow 1 becomes unbacklogged during (t_1, t_2), then its share of bandwidth is distributed to flows 2 and 3 evenly (the solid lines). However, we argue that such emulation of GPS-type of fairness is not necessary and effective in wireless scheduling. In wireless networks, the same amount of transmission bandwidth allocated to two flows may not result in the same amount of goodput because of the difference between the two flows' link qualities. 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. In such situations, we believe excess bandwidth should be distributed to the former flows first, as long as the latter's QoS guarantees are not violated. In this example, suppose the overall bandwidth is 1. At time t_1 flow 3 may have already achieved $t_1/3$ of goodput (its target share), whereas flow 2 has only $t_1/5$ of goodput because of a worse link. Then excess bandwidth should be allocated to flow 2 to help it reach its target rate of 1/3, rather than being distributed evenly between flows 2 and 3.

In an error-free system, it has been proven [56] that as long as a flow's service received in any backlogged period can be guaranteed such that

$$W_i(t_1, t_2) \geq r_i(t_2 - t_1 - \theta_i) \tag{2.1}$$

where $W_i(t_1, t_2)$ is the effective service received during (t_1, t_2), 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. Both of them are centralized algorithms, which should be run at the base station. The base station is responsible for scheduling both downlink and uplink packet transmissions. In WFS, the system virtual time, which is dependent on the service progress of all the flows and maintained by the scheduler, is needed in time-stamping the packets. For downlink transmissions such a process does not impose any problem, because all the downlink flows are queued at the base station where the scheduler is located and the global information maintained by the scheduler can be easily retrieved by each flow. However, for uplink transmissions the packets are queued at mobile hosts, to have the system virtual time, either each uplink flow needs to monitor the service progress of all other flows, or the scheduler needs to constantly broadcast the system virtual time to all the flows. Neither way can be realized efficiently. Furthermore, there is another problem with WFS. How does the scheduler know the finish times of the uplink packets? To do so, all the uplink flows need to constantly send information of the finish times to the scheduler, which will require huge amount of signaling bandwidth and cause significant transmission power consumption at the mobile hosts. In CIFQ, when a flow becomes unbacklogged its own virtual time is updated as the maximum of the system virtual time and its own virtual time. Thus, it has the same problem as WFS in maintaining the system virtual time for uplink flows. For the scheduling algorithms to be realizable, they should have the following two properties: (i) if any time-stamping of the packets are used, the timestamps should not depend on the information of other flows; (ii) the uplink flows do not need to send time-stamp information of each packet to the scheduler, and the scheduler only needs to know whether an uplink flow is backlogged.

2.2 System Model

We consider centralized scheduling in a wireless network, where a scheduler in a base station is responsible for scheduling all the packet transmissions in the network. The data flows being served may not be located at the base station. 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 duration of each state is exponentially distributed. A link states may be incorrectly estimated with a certain probability. The links are independent of each other and the average durations of the *good* and *bad* states may not be the same for different links. 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. Having less information bits but the same total length as a regular packet, a *low rate packet* has low transmission error probability in a *bad* state. Packets of each flow requires in sequence delivery. 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.

2.3 BGFS-EBA

2.3.1 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 rate of goodput it would like to achieve. 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. 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 (2.1) 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 flows which lag behind their goodput targets.

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

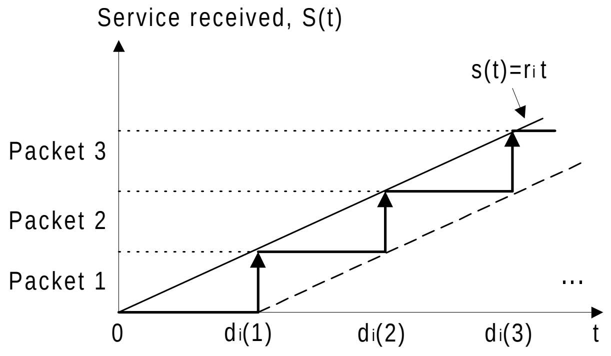


Figure 2.2: Deadline calculation of a virtual flow

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 Figure 2.2. 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 [18], 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 called the *normalized goodput gap*[2], which is defined in (2.2).

$$G_i(t) = \frac{g_i(t) - r_i t}{r_i t} \tag{2.2}$$

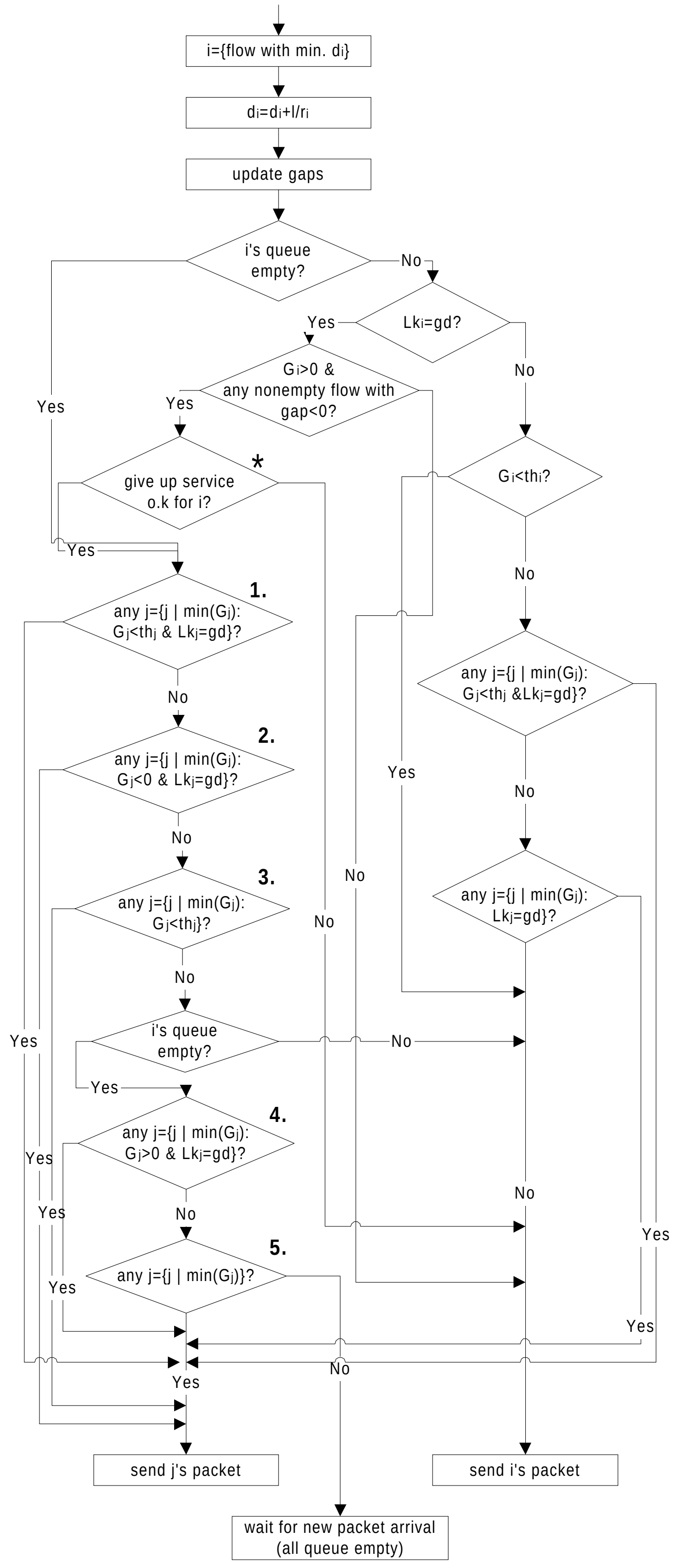
where $g_i(t)$ is the goodput achieved by flow i up to t in the current backlog period. Note that $G_i(t)$ is normalized by the target goodput $r_i t$ and it is a fraction which represents how much a flow is leading and lagging compared with its target goodput. g_i and G_i are reset to zero at the beginning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

[2]Note again the difference between the definitions of *gap* in BGFS-EBA and *lag* in CIFQ.

but the same absolute lag will be treated the same way. We do not believe this is reasonable since two flows, say, one with a rate of 1Mbps and the other with a rate of 1bps, both having a lag of 100 bytes, will be treated the same way in CIFQ and WFS, but obviously they are lagging behind their target rates in very different degrees of severity.

Figure 2.3 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 $-1 \leq th_i \leq 0$. 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 efficiency and link-state-independent bandwidth guarantees.



i={flow with min. d_i}
d_i=d_i+l/r_i
update gaps
i's queue empty?
No
Yes
Lk_i=gd?
Yes
No
G_i>0 & any nonempty flow with gap<0?
Yes
give up service o.k for i?
*
Yes
G_i<th_i?
No
1.
any j={j | min(G_j): G_j<th_j & Lk_j=gd}?
No
2.
any j={j | min(G_j): G_j<0 & Lk_j=gd}?
No
3.
any j={j | min(G_j): G_j<th_j}?
No
i's queue empty?
Yes
4.
any j={j | min(G_j): G_j>0 & Lk_j=gd}?
No
5.
any j={j | min(G_j)}?
any j={j | min(G_j): G_j<th_j &Lk_j=gd}?
any j={j | min(G_j): Lk_j=gd}?
Yes
send j's packet
send i's packet
wait for new packet arrival (all queue empty)

Figure 2.3: Flow chart of the scheduling operations

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. $g_i + l_i(HOL) \geq r_i(d_i - t_i)$ 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 Figure 2.3, 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.

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. Therefore, when a flow can not meet its gap threshold, it will not only transmit aggressively in its own turn, but also have more chance of 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.

There are two major reasons why we separate the scheduling operations into two phases and use a dummy flow. One is that we need to explicitly identify which part of the bandwidth is excess bandwidth, so that we can allocate it to flows which need it most. In CIFQ and WFS the excess bandwidth is implicitly distributed among all the flows. The other reason is that when all the flows are error-free we would like to avoid the virtual clock penalty effect [62] on flows receiving excess bandwidth while some other flows are not backlogged.

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. In fact the information of whether a flow is backlogged or not is necessary for any scheduling algorithm.

Furthermore, the scheduler only maintains one parameter for each flow, namely, the normalized goodput gap, instead of four parameters in CIFQ.

2.3.2 Analytical Observations

2.3.2.1 Minimum Bandwidth and Goodput Guarantees for Error-prone Flows

A virtual flow i is guaranteed a service rate r_i in the idealistic system. Whenever the corresponding real flow's average goodput rate falls below $r_i(1+th_i)$, the real flow will send at least at rate r_i, and $r_i \geq r_i(1+th_i)$. Real flow i may send at a rate less than r_i only when its average goodput rate is larger than $r_i(1+th_i)$. To achieve such a goodput rate, at least the same amount of average transmission bandwidth is required. Therefore, 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[3] 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.

2.3.2.2 Balance between Effort-fair and Outcome-fair

In the long run, flows with worse link quality that are not able to meet target goodput rates are guaranteed minimum transmission bandwidth. Flows with better

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

links that are able to meet target goodput rates are guaranteed no more than their target rate if there are lagging flows. Excess bandwidth is allocated to compensate flows with worse link quality. Thanks to the above mechanisms, the algorithm manages to strike a 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.

2.3.2.3 Delay Bound for Error-free Flows

Theorem 1 *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*

$$D_i \leq \frac{\sigma_i}{r_i} + \frac{2l}{r_i} + \frac{l}{R} \tag{2.3}$$

Proof: It is proven in [56] that when a flow's received service can be guaranteed in any backlog period as shown in (2.4),

$$W_i(t_1, t_2) \geq r_i(t_2 - t_1 - \theta_i) \tag{2.4}$$

where $W_i(t_1, t_2)$ is the amount of effective service received during (t_1, t_2), 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*, and its traffic is token bucket constrained with parameters (σ_i, r_i), then the packet delay can be bounded as shown in (2.5).

$$D_i \leq \frac{\sigma_i}{r_i} + \theta_i \tag{2.5}$$

In BGFS-EBA, it is easily shown that a virtual flow i is guaranteed service as in (2.4) with $\theta_i = l/r_i$. (This is actually the dotted line with a slope of r_i in Figure 2.2.) For real error-free flow i, after the first packet of a backlog period is transmitted, the subsequent packets are guaranteed to be transmitted at rate r_i; therefore, the service received by the real flow i can also be lower bounded in the form of (2.4). We only need to determine the latency θ_i for the real flow i. The latency is in fact the virtual flow i's latency plus the worst-case delay from the arrival of the first real packet of a backlog period to the time its corresponding virtual flow is scheduled again in the idealistic system. It is not difficult to show that such worst-case delay is $l/R + l/r_i$. Therefore, the latency for real flow i is $l/R + 2l/r_i$. Hence the delay bound in (2.3) is established.

2.4 Numerical Results

To assess the performance of our proposed wireless scheduling algorithm, we implemented both CIFQ and BGFS-EBA in Network Simulator-2 [30] and conducted simulations to compare the two algorithms.[4]

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 $u(0, 0.2)$ 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. The duration of each state is exponentially distributed. A *good* (or *bad*) state can be wrongly estimated as a *bad* (or *good*) state with probability 0.1. The total available bandwidth is $1 Mbytes/s$. The duration of each test equals the transmission time of two million packets. All flows require reliable delivery.

[4]Since some algorithmic details of WFS are not clearly specified in the paper, we did not implement WFS.

Table 2.1: 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%

2.4.1 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. The results of the simulations are listed in Table 2.1.

All the numbers except the efficiency are the average rates in $Kbytes/s$. 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, because a flow always gives up its bandwidth to others when its link is in a *bad* state, 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. Also, flow 1 receives more bandwidth when $th = -0.3$ than $th = -0.5$.

We notice that the total effort rate of CIFQ is smaller than the total available bandwidth. The reason is that in CIFQ the scheduler will stay idle if it can not find any flow with a *good* link. Therefore, a small amount of bandwidth is wasted on idling. Also note that BGFS-EBA's overall bandwidth efficiency in terms of total goodput achieved 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.

2.4.2 Example II

In the second example we show a more complex scenario. There are five greedy flows in the system. Table 2.2 shows the target rates and the average link state durations of each flow.

Table 2.2: Flow parameters in example II

	Target Rate (Kbyte/s)	Avg. Link State Duration (s)	
		good state	bad state
flow 1	200	0.03	0.07
flow 2	100	0.03	0.07
flow 3	200	0.09	0.01
flow 4	200	0.09	0.01
flow 5	100	0.09	0.01

Table 2.3: 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 2.4: 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%	

Table 2.5: CIFQ's results in example III

	$\alpha = 0$		$\alpha = 0.3$		$\alpha = 0.7$	
	effort	goodput	effort	goodput	effort	goodput
flow 1	283.9	177.4	173.6	128.2	164.2	118.4
flow 2	142.4	83.1	140.3	84.2	134.1	85.7
flow 3	285.0	265.3	341.0	304.7	347.9	309.0
flow 4	285.5	266.1	341.6	305.3	350.1	310.8
efficiency	79.2%		82.2 %		82.4%	

The simulation results are shown in Tables 2.3 and 2.4. 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. Just as we have discussed previously, this is due to the way excess bandwidth is distributed and the discrimination of CIFQ against flows with worse average link quality. Therefore, the flows with better links are over-provisioned with transmission effort.

2.4.3 Example III

In the third example we eliminate flow 5 from example II to see how the bandwidth will be distributed when there is more than enough bandwidth for flows to reach their target rates. No other parameter is changed. The results are show in Table 2.5 and Table 2.6.

For CIFQ, even when a sufficient amount of bandwidth is available, flow 1 still misses its goodput target, while flows 3 and 4 receive far more than their fair shares. In BGFS-EBA, after the scheduler manages to satisfy everyone's goodput target,

Table 2.6: BGFS-EBA's results in example III

	$th = -0.2$		$th = -0.3$		$th = -0.5$	
	effort	goodput	effort	goodput	effort	goodput
flow 1	276.4	200.0	276.1	199.6	276.1	199.6
flow 2	159.1	116.6	159.1	116.7	159.1	116.7
flow 3	282.4	249.0	282.6	249.4	282.6	249.4
flow 4	282.1	249.1	282.2	249.3	282.2	249.3
efficiency	81.5 %		81.5 %		81.5 %	

Table 2.7: Parameters in example IV

	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

the scheduler favors flows 3 and 4 in allocating the rest of the excess bandwidth. It also matches BGFS-EBA's policy that when there is no lagging flow, flows with positive lags can receive excess bandwidth. Since every flow's target has been met, the scheduler now tries to maintain high bandwidth efficiency.

2.4.4 Example IV

We now demonstrate that the delay bounds for error-free flows can be guaranteed. In this example there are 6 flows, which is fully loaded with the sum of all the target rates equal to the bandwidth. Table 2.7 shows the parameters of the traffic and the links. The two Poisson sources both have an average rate of $200 Kbytes/s$. The

Table 2.8: Packet delays of BGFS-EBA in example IV

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

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. The rate of each flow's token bucket is just its target rate. The depth of the bucket is 2000 bytes, which is equal to the length of ten packets. 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 time unit used in the tables is seconds. The analytical bounds are calculated based on (2.3). The simulation results in Table 2.8 show that for error-free flows, all the delays are within the analytical bounds. Compared with the results in Table 2.9, 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.

Remarks: Although in the experiments shown in this paper the thresholds of all the flows are set to be the same, it is not necessary for all the flows to have the same threshold. In fact, to improve bandwidth efficiency, higher thresholds can be set for flows with better links and lower thresholds can be set for flows with worse

Table 2.9: Packet delays of CIFQ in example IV

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

links. In general, to maintain high efficiency, the thresholds of flows with poor links should not be set too aggressively. Also, for a flow with high priority, the threshold can be set to zero or close to zero, so that most of the time the flow will not give up its service opportunity even when its link is in a *bad* state. Whereas for a flow with no bandwidth guarantee, a threshold of -1 can be used, so that the flow will always give up its service opportunity when its link is in a *bad* state. The threshold may also be changed dynamically according to the link quality and traffic load. How to design a proper admission control scheme and combine it with thresholds setting to maintain certain efficiency target, and how to dynamically adjust the thresholds will be subjects of future research.

Chapter 3

Wireless Packet Scheduling for Links with Multiple States

3.1 System Model

We consider a typical cell-structured wireless network, where multiple users/flows communicate with a base station responsible for scheduling transmissions. Transmission bandwidth is shared by the users (or flows) in a time-division-multiplexing manner, where time is slotted and only one flow is served in each time slot. Adaptive modulation and coding may be used. Each wireless link between a user and the base station is time-varying with multiple states determined by the link's physical capacity and the particular error-correction code and modulation level used. The parameters used to characterize a state of a link include *effective output* and *state probability*. The *effective output* is the number of useful information bits successfully delivered in the particular state in one slot. We assume that this parameter has taken into account the coding overhead, the modulation level, and the corresponding error rate associated with the link state. The *state probability* is the probability of being in the state in each slot. We focus on fast-fading scenarios where the link states in two different slots can be considered independent of each other. Without loss of generality, we assume that the marginal distribution of the link states is stationary and ergodic. Assuming no interference and each user's location and moving speed to be independent of each other, all the links are independent of each other and may have different link statistics.

There are two types of data flows, delay-sensitive flows and best-effort flows. Each delay-sensitive flow has two target parameters, *delay bound* and *delay violation probability bound*. The objective of scheduling a delay-sensitive flow is to guarantee that the probability of the flow's traffic delayed by more than the *delay bound* is no larger than the *delay violation probability bound*. To provide any delay guarantee, the incoming traffic must be constrained in bursttiness and rate. We consider token-bucket constrained traffic for delay- sensitive flows. For best-effort flows, the scheduler tries to achieve high bandwidth efficiency while maintaining LT-LQW outcome-fair among the flows. Since the upper layer packets are usually large and need to be segmented at the wireless link layer, from the link layer's view point, the incoming traffic is considered as bit-streams. Each flow has its own queue and infinite buffer is assumed.

3.2 Providing Delay Violation Bound for a Single Delay-Sensitve Flow

First we discuss how to provide statistical delay violation bounds to a single flow whose link has multiple states. As time is slotted, we model the system as a discrete-time system with fluid-modeled traffic arriving and departing the queue only at the boundaries of time slots. The model is validated with reasonable requirements that the bits arriving in one slot can not be served in the same slot, and the bits served in a slot is said to complete the service only at the end of the slot. In this section all the variables that represent time are in the unit of a time slot. For a link I with L states, denote the *effective output* and *state probability* of state l by u_{Il} and p_{Il}, respectively. Denote the effective service received by the corresponding flow I during an interval $[\mu, \mu + t]$ by $W_I(t)$, and denote the total incoming traffic during the same interval by $A_I(t)$. If $A_I(t)$ and $W_I(t)$ satisfy

$$A_I(t) \leq_{st} B_I(t) \tag{3.1}$$

$$W_I(t) \geq_{st} S_I(t) \tag{3.2}$$

where $B_I(t)$ and $S_I(t)$, called *statistical traffic envelope* and *statistical service envelope*, respectively, are non-negative non-decreasing random processes, and $\leq_{st}$ represents stochastic inequality. Then given a d_0, the probability of a bit being delayed for more than d_0, $P\{D_I > d_0\}$ is bounded as follows

$$P\{D_I > d_0\} \leq P\{\max_{t \geq 0}\{B_I(t) - S_I(t + d_0)\} > 0\} \tag{3.3}$$

where D_I is the delay experienced by a bit. We utilize the above theorem, which is proven in [47], to find a solution for bounding the delay violation probability in the practical scenario that we are interested in.

In practice, statistical traffic envelopes are difficult to enforce. In this paper we consider deterministic traffic envelopes, in particular, token bucket constrained traffic. For a flow regulated by a token bucket with parameter set (α, ρ), where α is the bucket depth and ρ is the token filling rate, its $A_I(t)$ satisfies

$$A_I(t) \leq \alpha + \rho t \tag{3.4}$$

Note that the minimal service time unit is one slot and every bit only completes service at the end of a slot. Suppose we allocate one slot to flow I every M slots. Now the problem we need to solve is: Given (α, ρ), d_0 and delay violation probability bound P_v, what's the minimal bandwidth , i.e. the largest M, which needs to be allocated to the flow in order to guarantee that $P\{D_I > d_0\} \leq P_v$?

Without loss of generality, suppose at time 0, flow I's queue is empty. Then for any $t > 0$,

$$A_I(t) \leq \alpha + \rho t \tag{3.5}$$

$$W_I(t) = \sum_{i=1}^{\lfloor \frac{t}{M} \rfloor} U_i \tag{3.6}$$

where U_i is the information bits transmitted by the flow in the ith transmission/service, and $\lfloor \phi \rfloor$ is the largest integer that is no larger than ϕ. Note in our system model, U_i, $i = 1, 2, ...$ are i.i.d. Let traffic envelope $B_I(t)$ and service envelope $S_I(t)$ be

$$B_I(t) = \alpha + \rho t \tag{3.7}$$

$$S_I(t) = \sum_{i=1}^{\lfloor \frac{t}{M} \rfloor} U_i \tag{3.8}$$

Then,

$$P\{B_I(t) - S_I(t + d_0) > 0\} = P\{\alpha + \rho t - \sum_{i=1}^{\lfloor \frac{t+d_0}{M} \rfloor} U_i > 0\} \tag{3.9}$$

Let $\lfloor \frac{t+d_0}{M} \rfloor = n$ and $\frac{t+d_0}{M} - \lfloor \frac{t+d_0}{M} \rfloor = \delta$. Then

$$t = Mn + M\delta - d_0 \tag{3.10}$$

Following (3.9) and (3.10),

$$P\{B_I(t) - S_I(t + d_0) > 0\} \tag{3.11}$$

$$\leq P\{\alpha + \rho M n + \rho M \delta - \rho d_0 - \sum_{i=1}^{n} U_i > 0\} \quad (3.12)$$

$$= P\{\sum_{i=1}^{n} (\rho M - U_i) > \rho d_0 - \rho M \delta - \alpha\} \quad (3.13)$$

$$\leq P\{\sum_{i=1}^{n} (\rho M - U_i) > \rho d_0 - \rho M - \alpha\} \quad (3.14)$$

The last step above follows because $0 \leq \delta < 1$. Since the average arrival rate of the information data is ρ, and the average departure rate is $\frac{E\{U_i\}}{M}$, to have a stable queue we should have

$$\rho M < E\{U_i\} \quad (3.15)$$

Let $V_i = \rho M - U_i$, and $\rho d_0 - \rho M - \alpha = a$. Then (3.14) becomes

$$P\{B_I(t) - S_I(t + d_0) > 0\} \leq P\{\sum_{i=1}^{n} V_i > a\} \quad (3.16)$$

where $E\{V_i\} < 0$. Since the flow's service period is M slots, for the scheduler to guarantee the delay violation probability bound, d_0 should be no less than M. Therefore, for any $t > 0$, there exists an integer $n \geq 1$ according to mapping (3.10). Following (3.16) we have

$$P\{\max_{t \geq 0}\{B_I(t) - S_I(t + d_0)\} > 0\} \leq P\{\max_{n \geq 1}\{\sum_{i=1}^{n} V_i\} > a\} \quad (3.17)$$

Now we try to find an upper bound for (3.17) by applying a corollary of Wald's identity [24]. The corollary is as follows. Let X_i, $i = 1, 2, ...$ be i.i.d, and $E\{X_i\} < 0$.

Let $\theta(r) = ln(E\{exp(rX_i)\})$ be the semi-invariant moment generating function of each X_i. Let $S_n = X_1 + ... + X_n$. If there exists an $r^* > 0$ such that $\theta(r^*) = 0$, then given a constant $b \geq 0$,

$$P\{\max_{n\geq 1}\{S_n\} \geq b\} \leq exp(-r^*b) \tag{3.18}$$

The proof is given in [24].

Since V_i, $i = 1, 2, ...$ are i.i.d. and $E\{V\} < 0$, if we have

$$a = \rho d_0 - \rho M - \alpha \geq 0\ , \tag{3.19}$$

then following (3.3), (3.17) and (3.18), we have

$$\begin{aligned} P\{D_I > d_0\} &\leq P\{\max_{t\geq 0}\{B_I(t) - S_I(t+d_0)\} > 0\} \\ &\leq P\{\max_{n\geq 1}\sum_{i=1}^{n} V_i > a\} \\ &\leq exp(-r^*a) \end{aligned} \tag{3.20}$$

where r^* is the positive root of the equation $ln(E\{exp(rV)\}) = 0$. Therefore, given M and d_0, (3.20) is an upper bound (and a reasonable estimate) of the delay violation probability. In particular, for a link I with L states,

$$ln(E\{exp(rV)\}) = ln\{\sum_{l=1}^{L} p_{Il} \cdot exp[r(\rho M - u_{Il})]\} \tag{3.21}$$

where u_{Il} and p_{Il} are the *effective output* and *state probability* of state l, respectively.

From the above analyses we make the following conclusion. Given delay violation probability bound P_v, and all the other necessary parameters, if we find an $M \geq 1$ such that (3.15), (3.19) and the following conditions are all satisfied,

$$exp\{-r^*(\rho d_0 - \rho M - \alpha)\} \leq P_v \tag{3.22}$$

the delay violation probability bound can be guaranteed. Note r^* is the positive root of the semi-invariant moment generating function of each $V_i = \rho M - U_i$.

Note $r^* > 0$ and $P_v \leq 1$; therefore, condition (3.22) implicitly requires that $\rho d_0 - \rho M - \alpha \geq 0$. That is, condition (3.19) is incorporated in (3.22) already.

Theorem 2 *Let a flow's traffic be token-bucket constrained with bucket parameters* (α, ρ)*, and the number of information bits it can send in each service slot* i *be* U_i*,* $i = 1, 2...$*, which are i.i.d. Given a delay bound* d_0 *and a delay violation probability bound* P_v*, we can guarantee that* $P\{D_I > d_0\} \leq P_v$ *by guaranteeing that the flow receives at least one service slot in every* M *slot time, if both of the following conditions are satisfied.*

$$\begin{cases} M < \frac{E\{U_i\}}{\rho} \\ exp\{-r^*(\rho d_0 - \rho M - \alpha)\} \leq P_v \end{cases} \tag{3.23}$$

where r^* *is the positive root of* $ln(E\{exp[r(\rho M - U_i)]\})$*.*

Following the above theorem, to guarantee a delay violation probability bound with the least required bandwidth, we just need to pick the largest M that satisfy the conditions in (3.23).

Now for a single flow, the delay violation bound requirement becomes equivalent to the requirement of guaranteeing a service effort period M. (Note that it is not the actual effective output.) The next question is: when there are N_d flows, how do we guarantee a service effort period M_I for each flow I, $I = 1, 2, ..., N_d$? In fact, it can be easily shown [18] that as long as M_I's satisfy

$$(\sum_{I=1}^{N_d} \frac{1}{M_I}) \cdot \frac{s}{\tau} \leq B_d \tag{3.24}$$

where s is the slot size in bits, τ is the slot duration, and B_d is the bandwidth available to the N_d delay-sensitive flows, such service effort guarantees can be provided by using an EDF (earliest-deadline-first) based scheduling. We will elaborate more on this when we discuss scheduling multiple delay-sensitive and best-effort flows simultaneously in section 3.4.

3.3 Issues of Scheduling Best-Effort Flows

For best-effort traffic, no strict guarantees are provided. Nevertheless, the network still needs to facilitate fair sharing of the resource and avoid large differences in different flows' goodputs.

In most of the existing wireless scheduling, since the links are modeled as either error-free or 100% in error, only transmission in error-free state is possible. Therefore, improving efficiency is not a concern in such a scenario. Maintaining both short-term and long-term fairness has been the main focus of the previous research. However, things are more complicated when links have multiple effective throughput levels. For example, suppose there are two flows with the same average link statistics and the same share of bandwidth. At the beginning of a slot, flow one has less goodput than flow two, and flow one can send more information bits, if served in this slot, because of better instantaneous link quality. Which flow should be chosen to receive service? Choosing flow two improves bandwidth efficiency, while choosing flow one improves fairness in terms of effective goodput level. Furthermore, links may be different not only in instantaneous quality but also average quality. More service opportunities need to be assigned to flows with inferior links to ensure that they have similar goodput levels as the others. However, this will decrease bandwidth efficiency. On the other hand, if the scheduler always chooses to serve the flow with the best instantaneous quality, some flows may be starved.

In wireline networks transmission outcome is always consistent with transmission effort. However, in wireless networks, the same amount of transmission effort may result in different outcome, due to users' different link qualities. Consequently, as we have discussed, there are two types of fairness notions, effort-fair and outcome-fair. As for end users, the effective service received is directly related to the useful data (outcome) sent/received, not the bandwidth (effort) it received. Therefore, guaranteeing only effort-fair without considering the actual outcome is not meaningful. On the other hand, guaranteeing only outcome-fair irrespective of the link quality differences may result in very low bandwidth efficiency. We believe that the fairness notion should be based on outcome and also related to link qualities. Since the exact amount of any short-term outcome depends on the instantaneous link status which is changing randomly, for fast-changing links with multiple instantaneous throughput levels, it is very difficult, if not impossible, to maintain short-term fairness based on outcome. Guaranteeing short-term effort-fairness is achievable by scheduling flows just according to classical wireline scheduling policies that provide isolation mechanisms among flows. However, such isolation prevents flows from giving up transmission opportunities to other flows with better instantaneous link qualities, which may cause low bandwidth efficiency. In all, maintaining short-term fairness for best-effort flows in wireless networks is impractical and unnecessary. Following the above reasoning, we propose a new fairness notion, called *long-term link-quality-weighted outcome-fairness*, which is defined as follows. Considering two continuously backlogged flows over a sufficiently long time interval T, if the average effective output levels of the two flows' links are $\bar{O}_1$ and $\bar{O}_2$, respectively; and the total effective output achieved, $W_1(T)$ and $W_2(T)$ satisfy

$$\left| \frac{W_1(T)}{w_1 \bar{O}_1} - \frac{W_2(T)}{w_2 \bar{O}_2} \right| < \epsilon \tag{3.25}$$

where ϵ is a small constant, and w_1 and w_2 are generic bandwidth weights, we say LT-LQW outcome-fair is achieved.

Denote the total available bandwidth by B_w, of which B_d is allocated to delay-sensitive flows. The bandwidth available to best-effort flows is thus $B_b = B_w - B_d$. Each best-effort flow J is assigned a generic bandwidth weight w_J. We assume that the weights are determined by some call-level[1] bandwidth allocation module that may have already taken into consideration each flow's traffic demand and importance level. Denote by $\bar{O}_J$ the average effective output that can be achieved on a link J in one time slot, and s the slot size in bits. According to the call-level allocation with perfect isolation among flows, each flow J should receive $B_J = B_b \cdot w_J / (\sum_{I=1}^{N_b} w_I)$ of bandwidth, and, consequently, achieves effective goodput at

$$r_J = B_J \frac{\bar{O}_J}{s} \tag{3.26}$$

where N_b is the total number of best-effort flows. We call r_J, which represents estimated goodput level the network operator would like flow J to achieve without considering packet-level adaptation, the *goodput target* of flow J.

It is easy to see that (3.25) is equivalent to the following.

$$\left| \frac{W_1(T)}{r_1} - \frac{W_2(T)}{r_2} \right| < \epsilon \tag{3.27}$$

To achieve LT-LQW outcome-fair, flows' goodput levels should be in proportion to their average link qualities. As flows with worse links have smaller goodput targets

[1]compared to the actual scheduling at the packet-level

than those with the same bandwidth weights but better links, LT-LQW outcome-fair facilitates more efficient use of bandwidth than pure outcome-fair. Furthermore, this fairness notion focuses only on long-term performance. As no strict short-term fairness is required, it gives the packet scheduler more freedom in improving bandwidth efficiency by selectively scheduling transmissions on links with better instantaneous qualities.

If maximizing the bandwidth efficiency is the sole objective of packet scheduling, the scheduler just needs to always schedule the flow with the best instantaneous link quality to transmit in each time slot. However, as fairness must also be considered, the objectives of scheduling for best-effort flows should be multi-dimensional. First, the scheduling policy should provide certain minimal goodput level (its target goodput) for each flow. Second, the policy should try to maintain LT-LQW outcome-fair. Third, the policy should try to achieve high bandwidth efficiency while maintaining fairness properties.

We propose a simple scheduling scheme for best-effort flows and show its effectiveness in achieving the above objectives by intuitive reasoning and simulations. The algorithmic details are presented in the next section as a part of the complete scheduling algorithm for both delay-sensitive and best-effort flows.

3.4 Scheduling Algorithm for Delay-Sensitive and Best-Effort Flows

The complete algorithm for scheduling both delay-sensitive and best-effort flows is presented in Figure 3.1, where the flow sets $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ are defined as follows.

$$\mathcal{A} = \{\text{all flows}\} \tag{3.28}$$

$$\mathcal{B} = \{\text{backlogged best-effort flows}\} \tag{3.29}$$

$$\mathcal{D} = \{\text{backlogged delay-sensitive flows}\} \tag{3.30}$$

$$\mathcal{C} = \mathcal{B} \cup \mathcal{D} \tag{3.31}$$

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. Each real flow, best-effort or delay-sensitive, has a corresponding *virtual flow* in the error-free system. The virtual flows are assumed to be always backlogged. Each virtual flow J is assigned bandwidth (service rate) B_J. We have $\sum_{J=1}^{N} B_J \leq B_w$, where B_w is the total available bandwidth, and N is the total number of users. Each virtual flow J maintains a deadline Δ_J. When the scheduler starts to work, each flow's deadline is initialized as $\Delta_J = s/B_J$, where s is the total number of bits that can be transmitted in one time slot when it is error-free. In the first phase, the scheduler always

```
Scheduling Procedure: (when $\mathcal{C} \neq \emptyset$)
 1.  $id = \{i \mid \min \Delta_i : i \in \mathcal{A}\}$;
 2.  $\Delta_{id} = \Delta_{id} + \frac{s}{B_{id}}$;
 3.  if ($id \in \mathcal{D}$) then
 4.     $sid = id$;
 5.  else if ($id \in \mathcal{B}$) then
 6.     if ($g_{id} < t$) then
 7.        $sid = id$;
 8.     else
 9.        $fid = \{i \mid \max O_i : g_i \leq g_{id}, O_i \geq O_{id}, i \in \mathcal{B}\}$;
10.        if ($fid$ exists) then
11.           $sid = fid$;
12.        else
13.           $sid = id$;
14.        end if
15.     end if
16.  else    (// $id$ is not backlogged)
17.     $fid = \{i \mid \min g_i : i \in \mathcal{B}\}$;
18.     if ($fid$ exists) then
19.        $sid = fid$;
20.     else
21.        $sid = \{i \mid \max O_i : i \in \mathcal{D}\}$;
22.     end if
23.  end if
24.  $sid$ transmits;
25.  if ($sid$ is a best-effort flow) then
26.     $g_{sid} = g_{sid} + \frac{data\ sent}{r_{sid}}$;
27.  end if
28.  $t = t + \frac{s}{B_w}$;
end.
```

Figure 3.1: Pseudo code of the scheduling procedure

chooses to serve the virtual flow with the smallest deadline. To avoid unwanted synchronization, ties of deadlines are broken randomly. Regardless of which real flow is served in the second-phase scheduling decision, after a virtual flow is picked in the first phase, its deadline Δ_J is updated as $\Delta_J = \Delta_J + \theta_J$, where $\theta_J = s/B_J$. Without loss of generality, suppose the virtual flows become backlogged at time $t = 0$. Since $\sum_{J=1}^{N} B_J \leq B_w$ and the service unit is fixed, it is proven [18] that in the above system each flow J will be served at least once for every θ_J, and the total amount of service received by the flow is at least $m\theta_J B_J$ at $t = m\theta_J$, where $m = 1, 2, \ldots$. Therefore, each virtual flow is guaranteed a service rate B_J.

Now we explain how the bandwidth is assigned to each virtual flow. For a delay-sensitive flow I, as shown in section 3.2, the requirement of guaranteeing a statistical delay violation bound can be mapped to the requirement of guaranteeing the flow at least one service opportunity every M_I slots, and M_I can be calculated as the largest integer that satisfies (3.23). In the above idealistic system, if we set $\theta_I = M_I\tau$, where τ is the duration of a slot, virtual flow I will be scheduled at least once for very M_I slots, which is exactly what is required by a delay-sensitive flow. Accordingly, the bandwidth B_I assigned to the virtual flow corresponding to the delay-sensitive flow I is

$$B_I = \frac{s}{\tau M_I} \tag{3.32}$$

For each virtual flow corresponding to a best-effort flow K, the bandwidth received is $B_K = B_b \cdot w_K/(\sum_{I=1}^{N_b} w_I)$, where w_K, N_b and B_b are flow K's bandwidth weight, the total number of best-effort flows, and the bandwidth available to best-effort flows, respectively.

Figure 3.1 describes the scheduling procedure performed for each time slot when there is at least one backlogged flow. Lines 1 and 2 are the first-phase operations, where the virtual flow (id) with the smallest deadline is picked.

In the second phase, if id is a backlogged delay-sensitive flow, its corresponding real flow is always scheduled to transmit (lines 3-4). Thus, each delay-sensitive flow is guaranteed to have the same service effort as its corresponding virtual flow. Therefore, following the above bandwidth assignments and the scheduling policy, the delay violation probability bound of each delay-sensitive flow can be guaranteed, and such a guarantee is independent of other flows' behavior or link status.

Each best-effort flow K has a *goodput target* r_K, which is defined in (3.26). The scheduler keeps track of the service progress of each best-effort flow by using a parameter g_K called *normalized service time.* g_K is defined as

$$g_K = g'_K + \frac{W_K}{r_K} \tag{3.33}$$

where W_K is the total effective output the flow has achieved in the current backlogged period, and g'_K is the *normalized service time* at the beginning of the flow's current backlogged period. When a flow is continuously backlogged and $g'_K = 0$, g_K represents the equivalent service time the flow would have received if its goodput rate were maintained at r_K. To maintain LT-LQW outcome-fair is to ensure that g_K's do not differ greatly.

Lines 5-15 describe the second-phase operations when the flow (id) picked in the first-phase is a backlogged best-effort flow. First, g_{id} is compared with t, which records the real time passed after the scheduler starts to work. When $g_{id} < t$, the average goodput the flow has achieved is smaller than r_{id}, its goodput target. In this case, flow id will be scheduled to transmit regardless of the link status (lines 6-7), because we would like to guarantee a minimal goodput level (goodput target) for each best-effort flow. Otherwise, if flow id's average goodput is already above its target, the scheduler will try to find a backlogged flow with the best instantaneous quality among all the best-effort flows that has a smaller *normalized service time*, and an instantaneous link quality no worse than flow id's instantaneous link quality in the current slot (line 9). Note that O_i is the effective output that can be achieved if flow i is scheduled to transmit in the current slot. If such a flow exists, flow id will give up the current slot to this flow[2]. Otherwise, flow id will transmit in the current slot. Such operations incorporate considerations for both fairness and bandwidth efficiency. Flows with smaller *normalized service times* are flows receiving less service than what LT-LQW outcome-fairness requires; therefore, having flow id give up its service opportunity to such flows helps maintain fairness. On the other

[2]If more than one flow has the best instantaneous link quality, the one with smaller g_i is chosen.

hand, to improve bandwidth efficiency, flow *id* only gives up its service to the lagging flow with the best instantaneous link quality.

As each virtual flow K is guaranteed to be served at rate B_K, and its corresponding best-effort flow does not give up its service whenever its average goodput is below its target, a backlogged flow can be guaranteed a minimal long-term goodput as in (3.26). Note that if no swapping service is allowed, LT-LQW outcome-fair can be assured. However, it will result in low bandwidth efficiency. The simple service swapping mechanism described above actually balances between maintaining fairness and improving bandwidth efficiency.

If the virtual flow (*id*) picked in the first-phase corresponds to an unbacklogged flow, the flow does not have enough traffic to make full use of its bandwidth share. Such excess bandwidth is used to first compensate the best-effort flow with the least *normalized service time* (lines 17-19). If no best-effort flow is backlogged, the delay-sensitive flow with the best instantaneous link quality gets the service (lines 20-22).

At the beginning of each backlogged period of a best-effort flow K, its g'_K is set as

$$g'_K = \max(g_K, \bar{g}_K) \tag{3.34}$$

where $\bar{g_K}$ is the average value of the currently backlogged best-effort flows' g_K's. The reason for such an operation is to let an idle flow forgo its "false goodput credit" accumulated while it is idle. If a flow has no data to send for a long period, its g_K

will not increase. When it becomes backlogged again, without the above operation, it would appear that the flow is lagging behind other flows greatly, as it has a very small *normalized service time* compared with others. Then the flow would transmit aggressively at its own turn of service and have precedence in receiving service given up by others. However, since such "lagging" results from the flow having nothing to send, not from unfavorable link quality or bias of the scheduling policy, it is unreasonable to let it have precedence over other backlogged flows. In this case, g'_K is set to $\bar{g}_K$. Since the algorithm has a mechanism to maintain fairness, the g_K's of the backlogged flows are not expected to differ greatly, $\bar{g}_K$ is a reasonable estimate of the service progress of whole system. If after an idle period, the flow's g_K is still larger than $\bar{g}_K$, it means the flow had achieved more goodput than its fair share before it became idle. In this case, it should keep its old g_K to let other flows have precedence.

3.5 Numerical Results

Simulations have been performed to evaluate the algorithm. In the simulations, the system bandwidth is $1Mbps$. Each time slot lasts $1ms$. Hence, 250 bytes can be sent in one slot. Of the 250 bytes, at least 50 bytes are used for control, header, and minimum error-correction coding. Each link has four possible states with effective outputs of 200 bytes, 150 bytes, 100 bytes, 50 bytes, respectively. Links may have

Table 3.1: State probabilities of three types of links used in the simulations

	state 1	state 2	state 3	state 4
type I	0.05	0.55	0.35	0.05
type II	0	0	0.55	0.45
type III	0.3	0.6	0.05	0.05

different probabilities of being in each state. The three types of links used in the simulations are listed in Table 3.1. Listed in the table are the state probabilities.

3.5.1 Delay Violation Bound for a Single Delay-Sensitive Flow

First we test the analytical results in section 3.2. In this test, there is only one delay-sensitive flow with type I link in the system. The token filling rate and the depth of the token bucket are $40Kbps$ and 50 bytes, respectively. The flow is continuously backlogged with fluid traffic to maximize the traffic load. However, the flow does not use the whole available bandwidth. The flow is provided service of one slot every M slots. For *delay bounds* of $36ms$, $38ms$ and $40ms$, we change the service period M and calculate the analytical delay violation probability bounds according to (3.20). The analytical results are compared with simulation results in Figure 3.2. The figure verifies that the delay violation percentages are smaller than the analytical bounds.

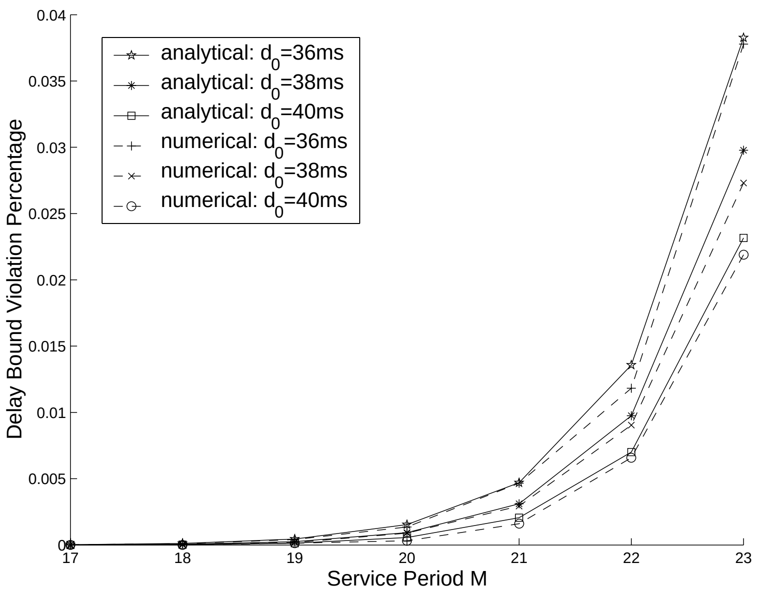


Figure 3.2: Delay violation bounds for a single flow

3.5.2 Scheduling Both Types of Flows Together

In this example, there is a total of eight flows in the network. Flows 1 and 2 are best-effort flows with type II links. Flows 3, 4, 5 and 6 are best-effort flows with type III links. Flows 7 and 8 are delay-sensitive flows with type I links. Flows 7 and 8 have the same token bucket parameters, which are $40Kbps$ and 50 bytes for the token filling rate and the depth of the token bucket, respectively. The required *delay bounds* for flows 7 and 8 are both $40ms$. All the best-effort flows have the same bandwidth weight.

Figure 3.3 and Table 3.2 show the results when all flows have greedy traffic, which is the worst case for delay-sensitive flows. From Table 3.2 we see that when

Table 3.2: Results of delay-sensitive flows

	service period M	analytical delay violation bound	simulation delay violation percentage
flow 7	21	2.06×10^{-3}	1.86×10^{-3}
flow 8	20	5.67×10^{-4}	5.03×10^{-4}

multiple flows with both types exist, the analytical delay violation bounds can still be guaranteed.

Figure 3.3 compares the *target normalized goodputs* of three scheduling schemes. The *target normalized goodput* is defined as a flow's goodput divided by its goodput target. Achieving LT-LQW outcome-fair means that each flow has the same *target normalized goodput.* The three schemes being compared are the algorithm we propose, classical fair-queueing with no service swapping, and one which always schedules the flow with the best instantaneous link quality (If more than one flow has the best instantaneous link quality, the one with the smallest *normalized service time* is scheduled). We note that both our algorithm and classical fair-queueing achieve LT-LQW. In fact, classical fair-queueing also achieves effort-fair. However, as it does not exploit the benefit of service swapping, bandwidth efficiency is lower. Bandwidth efficiency is calculated as the total goodput divided by the total goodput achievable if all the links stay in the best link state. The bandwidth efficiencies of the three schemes are 75.21%, 65.4%, and 89.9%, respectively. On the other hand, the scheme that maximizes the bandwidth efficiency result in extremely unfair share of bandwidth. We find in the figure that flows 1 and 2 almost achieve zero goodput

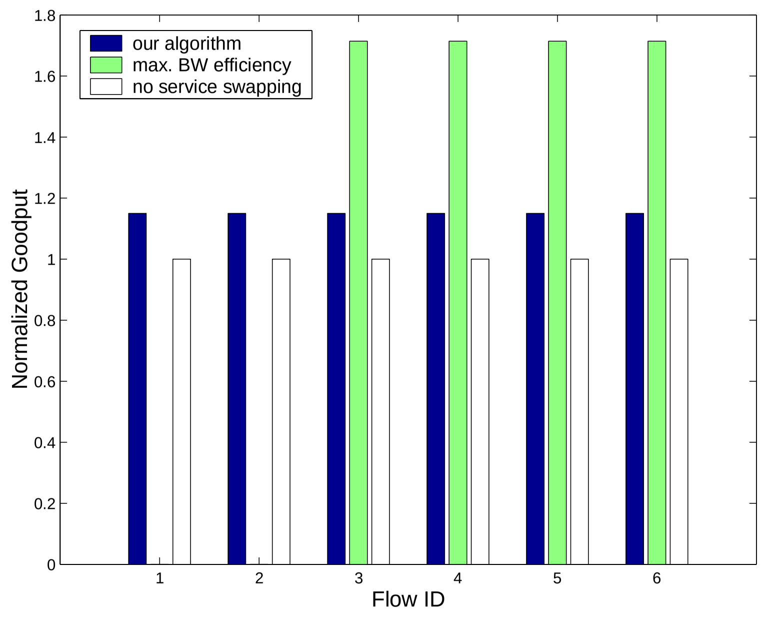


Figure 3.3: Comparison of normalized goodput under different schemes

in the third scheme, while flows 3, 4, 5, and 6 achieve far greater goodput than their targets, because they have better links than flows 1 and 2. Therefore, the results show that the scheme we propose balances the requirements of bandwidth efficiency and fairness.

We also simulated the scenarios when the best-effort flows are not greedy (Poisson and on-off traffics). To avoid distortion of the idle periods on the average goodput, we calculate the long-term average goodput of a flow by dividing its total effective output by the total backlogged time. Using such a definition, we find similar goodput distribution as that in Figure 3.3.

Chapter 4

Adaptive Scheduling in Cellular CDMA Networks

4.1 CDMA System Related Issues

4.1.1 Capacity Constraints

In CDMA all streaming service users (e. g. voice) and packet data service users share the same spectrum. In this paper we assume streaming services always have the highest priority and are one hundred percent guaranteed (i. e. no loss), and thus we have variable amount of remaining resource to be allocated by the packet scheduling algorithm for packet data users. (Note that our algorithm can be applied to all-packet-data systems as well.) The uplink (from mobile user to base station) capacity of a CDMA system is limited by the total interference power in the system, while in the downlink (from base station to mobile user) it is the maximum base station power that limits the system capacity. For convenience in this paper we consider a single-cell system (i. e. other-cell interference is not included).

For either uplink or downlink, we have found a capacity constraint relating the data rate R_i and the received bit energy to interference density ratio (BIR) $\gamma_i = \frac{S_i}{I_i}\frac{W}{R_i}$ of all user i to the system channel bandwidth W, where S_i, I_i, and R_i are received signal power, interference power, and data rate of user i, respectively. The BIR γ_i is a quality of service index and determined by the bit error rate requirement.

Specifically [65], [66], for the uplink of a single-cell system, the capacity constraint is

$$\sum_{i=1}^{N} C_i \leq W(1-\eta) \tag{4.1}$$

where N is the number of users in the cell, η is the noise to interference density ratio (a quantity for the interference limit), and

$$C_i = R_i\gamma_i \quad \text{or} \quad C_i = \left[\frac{1}{R_i\gamma_i} + \frac{1}{W(1-\eta)}\right]^{-1} \tag{4.2}$$

is called the *virtual bandwidth* consumed by user i. The difference between the two expressions of C_i is: The simpler expression is an approximation where the total received signal power from all users is counted in the interference power, while the more complicated expression has excluded the desired user signal from the interference in the calculation.

The capacity constraint for the downlink of a single-cell CDMA system is

$$\sum_{i=1}^{N} C_i \leq \frac{W(1-\beta)}{1-f_o} \tag{4.3}$$

where β is the fraction of the maximum base station power reserved for control purposes (synchronization, paging, pilot, etc.), f_o or the *orthogonality factor* is the portion of the interference cancellation (at the user side) by the downlink orthogonal codes, and

$$C_i = R_i\gamma_i \quad \text{or} \quad C_i = \left[\frac{1}{R_i\gamma_i} + \frac{1}{W}\right]^{-1} \tag{4.4}$$

The reason for our having two expressions for C_i is exactly the same as in the uplink.

Since the different C_i expressions only slightly change the capacity calculation, but do not affect the packet scheduling algorithm, for simplicity we just use the simpler version $C_i = R_i\gamma_i$ in this paper for both uplink and downlink.

4.1.2 Packet Transmission Scenarios

In the existing CDMA proposals data packets are transmitted in fixed-length frames (e. g. 10 ms). In general cases user transmission rates are variable, and thus the packet scheduling problem becomes transmission rate assignment problem (in the fixed-length frames) for the users requiring data services. Variable transmission rates can be realized by the multi-code CDMA scheme [31], where multiple packets from a user are transmitted at the same time on associated code channels.

In order for the base station to perform the packet scheduling or transmission rate assignment, the users need to send requests before data transmission. Sending the request for each data packet is inefficient. However (as in [3]), it is reasonable to assume that packets arrive in batches (bursts), and all the packets in a batch have the same delay deadline. Note that the packet delay deadline in this paper is not directly associated with the data rate, but is determined by the application requirement. A user sends to the base station a burst transmission request for all the packets in the burst. This request should include the amount of data (e. g. number of bits) in the burst and the delay deadline of the burst.

The base station needs to maintain a request queue for each user who has sent a burst transmission request. Prior to each frame time, the base station performs packet scheduling algorithm for the head-of-line burst transmission requests of all request queues, and assigns the *data rate* of transmission from each user in the next frame. From section 4.1.1 we know that when the required receiving BIR γ_i is given for each user, the cell capacity is actually a constraint on the transmission data rate R_i from each user, and thus it is a rate assignment problem.

Besides the issues of packet delay deadline and cell capacity, the packet scheduling algorithm may also need to control the average data rate from the user, and consider the channel status of each user (to avoid too much transmission power from the user with a bad channel). In the next section we propose a packet scheduling algorithm taking all these issues into account.

The downlink scenario is a little simpler since the base station has all the knowledge of the data to be transmitted. However, the base station may ask a user to report the pilot strength measurement before transmitting data to it in order for the base station to estimate the channel status and determine the transmission power. Assume the base station knows the delay allowed for transmitting each packet on this last hop to the receiver (from the delay information tagged on the packet, for example). The downlink packet scheduling algorithm should be very similar to the uplink one.

4.2 Adaptive Packet Scheduling Algorithm

Since the uplink and downlink have similar capacity structure (eq. (4.1) and (4.3)) and packet scheduling procedure, we use uplink in presenting our packet scheduling algorithm.

Here is the overview of the algorithm: First a priority is calculated for the packets in the head-of-line bursts of each request queue according to their delay deadline and the data rate lag (will be defined) of each user. Then based on its channel status, the cell capacity, and also the data rate lag, a transmission rate for the next frame is assigned to each user following the order of their packet priorities.

4.2.1 Priority Calculation

The priority of each packet consists of two parts: *delay guarantee priority* and *lag adjustment priority.*

A packet with closer deadline should have a higher priority of transmission to avoid its loss due to expiration. Let t be the current time, a be the packet arrival time, and d be the maximum delay of the packet before expiration, the *time-to-live* of the packet is defined to be

$$t_l = \frac{a + d - t}{T_f} \tag{4.5}$$

where T_f is the frame time. In the request queue at the base station, the packet arrival time is approximated by the arrival time of the request so the time of generating and transmitting the request is ignored. (In the downlink we can have accurate packet arrival time.)

By its definition t_l is actually the remaining time of the packet in the unit of frame time. The *delay guarantee priority* is thus chosen to be inversely proportional to t_l.

On the other hand, a user with larger lag on the average data rate should have higher transmission priority, so that it can eventually reach the target data rate. Assume that the target data rate and BIR have been agreed on between the user and the base station during the flow admission phase at the beginning of their communication (see section 4.2.3 for details). The *data rate lag* is defined as the difference between the target data rate R^* and the average data rate $\bar{R}$ from the beginning of the flow until the current time, or

$$\Delta R = R^* - \bar{R}$$

If $\Delta R < 0$ which means the user has received more service than it needs up to now, its priority in the lag adjustment part should have negative value so that it gives the transmission opportunity to users with positive lags. Also the lag adjustment priority should be related to the *normalized lag* with respect to the target data rate.

We design the *lag adjustment priority* as (for each user)

$$p_{r\_lag} = \frac{\Delta R}{\mu R^* - |\Delta R|}, \qquad |\Delta R| \le \mu R^* \tag{4.6}$$

where μ is the maximum allowed deviation of the average data rate from the target value. Note that p_{r_lag} has the same sign as ΔR, and its absolute value increases faster when the data rate deviation $|\Delta R|$ gets closer to its limit μR^* (can be seen from the first derivative of p_{r_lag}). In case $|\Delta R| > \mu R^*$, just set $|\Delta R| = \mu R^*$ in the calculation so that p_{r_lag} goes to ∞ or $-\infty$ depending on the sign of ΔR. (In implementation we need to specify a big but finite number for ∞.)

Lag adjustment is also the way to achieving fairness among the users. We regard a user as fairly serviced if its target data rate and quality of service (BIR) are guaranteed on the average. If assuming perfect power control or the average received

BIR is kept at its target value, fairness is achieved when the average data rate of each user reaches its target data rate.

Combining the delay guarantee priority and lag adjustment priority, the overall packet transmission priority is calculated as

$$\begin{gathered} p_r = \alpha p_{r\_lag} + t_l^{-1} = \frac{\alpha \Delta R}{\mu R^* - |\Delta R|} + \frac{T_f}{a + d - t}, \\ |\Delta R| \leq \mu R^* \end{gathered} \tag{4.7}$$

where the coefficient α is configured to achieve a compromise between the lag adjustment and delay guarantee, or user fairness and data loss. Note that at a certain time all the packets in a head-of-line burst must have the same priority, since the delay deadline and data rate deviation are the same for all of them, so they can be all transmitted at a time if enough data rate is assigned to the corresponding user.

4.2.2 Transmission Rate Assignment

Due to power control, the transmission power from a user is proportional to the channel degradation, or inversely proportional to the fading-times-shadowing envelope given the path loss. When channel status is unfavorable, the user needs large transmission power to combat fading and shadowing. If even the maximum power in the user device can not satisfy the transmission requirement, the transmitted packet will be received at a low BIR and thus a high probability of error. The packet scheduling algorithm, therefore, should take the channel status into consideration, and lower the user transmission power as much as possible.

For simplicity, in this paper we assume perfect power control which means the received BIR γ_i averaged over every frame is kept at the required value for each user. In our scheduling algorithm, when the channel degradation of a user is too serious (exceeding a predefined threshold), the user will not be allowed to transmit in the next frame, and the opportunity is given to other users with better channels. In this way the average transmission power from a user and the total transmission power from all users can both be reduced. The channel degradation threshold should be set such that it takes effect before the maximum power in the user device is reached. After all these considerations we can assume that packets are always correctly-received once they are transmitted.

The user transmission power, which is proportional to the channel degradation, is equal to the transmission bit energy times the data rate. Therefore, to avoid large transmission power when channel degradation is serious, we should assign data rate as inversely proportional to channel degradation. In practice the base station can estimate the channel status from the user's *unit-rate transmission power* defined as the transmission power divided by the data rate (so that the data rate factor in the transmission power is eliminated and only the channel status effect is left). Since the base station issues all the power control commands instructing the user to increase or decrease transmission power, it should be able to track the change of the transmission power from the user. Suppose the initial unit-rate transmission power from a user is s_0 when the flow starts (s_0 may be unknown to the base station in the uplink[1]). Let $s_t = g_t s_0$ be the current unit-rate transmission power from the user, and $\bar{s} = \bar{g} s_0$ be its average unit-rate transmission power since the flow starts. The base station knows g_t and $\bar{g}$ because the power control commands actually specify relative power changes (e. g. increase 1 dB from the previous value). The ratio

$$\frac{s_t}{\bar{s}} = \frac{g_t}{\bar{g}}$$

is used in our algorithm as an index of the channel status (smaller $s_t/\bar{s}$ or $g_t/\bar{g}$ corresponds to better channel). We use this ratio over the average value in order to decouple the channel path loss from the transmission rate assignment as both s_t and $\bar{s}$ contain the path loss factor. Otherwise users closer to the base station tend to receive more services since their path losses are smaller, which is not a reasonable assignment scheme. *Transmission rate is thus assigned to be inversely proportional to $s_t/\bar{s}$ or $g_t/\bar{g}$.*

However, the current value s_t or g_t may not be a good estimate of the channel status for data transmission in the next frame, if we consider the fast channel variations (may be much faster than the frame rate). A better estimate, as suggested in [37], is the "local average" of the channel envelope which is the average value over a few frames. Therefore, the s_t or g_t used in our algorithm is actually a value averaged

[1]In the downlink the base station obviously has full knowledge of the transmission power.

over a time window w_{av} which covers a fixed number of frames up to the current time. Note that if a user has no transmission in a frame, this frame time should not be counted in the calculation of the local average s_t or g_t.

On the other hand, the assigned transmission rate should be proportional to the target value. In order to let the average data rate converge to the target value more quickly, the lag adjustment is also included in our transmission rate assignment. The final assigned transmission rate is (for each user)

$$\begin{gathered} R_t = (R^* + \Delta R)\left(\frac{s_t}{\bar{s}}\right)^{-1} = (R^* + \Delta R)\left(\frac{\bar{g}}{g_t}\right), \\ \text{when} \quad \frac{\bar{g}}{g_t} \text{ ¿ } \xi \end{gathered} \tag{8a}$$

where ξ is the channel status threshold such that

$$R_t = 0 \quad \text{when} \quad \frac{\bar{g}}{g_t} < \xi \tag{8b}$$

In addition, we put a lower limit R_{min} on R_t

$$R_t \text{ ¿ } R_{min} \quad \text{when} \quad \frac{\bar{g}}{g_t} \text{ ¿ } \xi \tag{8c}$$

because if a user has enough priority and the channel degradation is below the threshold it should be allowed to transmit a certain amount of backlogged data. R_{min} also corresponds to the rate of the "fundamental channel" in multi-code CDMA. A user is allowed to send all the packets of a head-of-line burst in the next frame if a large enough R_t is assigned by (8a).

The rate assignment is performed to all the packets in the head-of-line bursts of the request queues, in the order of user packet priorities calculated in section 4.2.1. Users with negative priority, meaning their average data rates are so much ahead of the target values that can not even be compensated by the delay priority, are not allowed to transmit in the next frame. Expired packets are discarded and counted in the data loss rate. If after a round of rate assignment with all request queues there is still extra cell capacity (in terms of virtual bandwidth (4.1)), the second round of rate assignment begins following the same procedure. Packet priorities may need to be re-calculated for the second round rate assignment since the head-of-line bursts may have changed. This procedure continues until the cell capacity limit is reached, or no head-of-line bursts with positive priorities are left.

4.2.3 Relation to Flow Admission

Like streaming services, packet data flows need to be admitted as well before requesting burst transmissions. In the flow admission phase users tell the base station about their target data rates and quality of services requirements (BIR). The base station admits flows based on its estimation on the resource required by each flow. The estimation may be characterized by the mean and variance of the resource consumption by each flow. Under the assumption of perfect power control and no other-cell interference, the only variation of the resource consumption (the virtual bandwidth $R_i\gamma_i$) comes from the data rate. The total virtual bandwidth $C = \sum_{i=1}^{N} C_i$ in this case can usually be approximated by a Gaussian random variable. Given the *outage probability* δ in the uplink, which is the probability of the total interference power exceeding the acceptable level, or equivalently the probability of $\{C > W(1-\eta)\}$ (eq. (4.1)), the flow admission criterion for the uplink is [65]

$$m_c + \tau\sqrt{v_c} \leq W(1-\eta) \tag{9}$$

where $\tau = Q^{-1}(\delta)$ with $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{\frac{t^2}{2}}dt$ being the tail probability function of a (0, 1) Gaussian random variable, and

$$m_c = \sum_{i=1}^{N} m_{R_i}\gamma_i, \qquad v_c = \sum_{i=1}^{N} v_{R_i}\gamma_i^2 \tag{10}$$

are the mean and variance of the total virtual bandwidth C. The mean and variance of R_i — m_{R_i} and v_{R_i} — can be either estimated from a known traffic model of the flow from user i, or reported by the user. We can also write a similar flow admission constraint for the downlink.

If no packet scheduling is performed, the portion of data causing the total virtual bandwidth C exceeding $W' = W(1-\eta)$ is regarded as being lost. Applying the Gaussian approximation, the average loss rate is, approximately,

$$\begin{aligned} p_l &= \mathrm{E}\left[\frac{C - W'}{C} \middle| C > W' \right] \Pr\{C > W'\} \\ &= \delta \int_{W'}^{\infty} \frac{1}{\sqrt{2\pi v_c}} \left(1 - \frac{W'}{c}\right) e^{-\frac{(c-m_c)^2}{2v_c}} \, dc \end{aligned} \tag{11}$$

where δ is the outage probability used in flow admission. The integral in this formula can be evaluated numerically when all the parameters are given.

The packet scheduling algorithm can take advantage of the delay allowed for each packet and thus avoid overload of the system ($C > W'$) as much as possible. As a result, data loss rate can be reduced with packet scheduling. However, the channel status threshold ξ set in our scheduling algorithm affects the loss rate as it suppresses transmission in a channel below the threshold. On the other hand, since the flow admission is based on Gaussian approximation and the knowledge of traffic models, when we have rather bursty or irregular traffic the chance of overloading the system may be higher than expected. At this time the packet scheduling algorithm can help absorb the excess variation of the traffic.

4.3 Numerical Results

We consider the uplink of a single-cell system with three types of traffic: voice, low speed (type I) data, and high speed (type II) data. The simulation parameters are listed in Table 4.1. The voice stream is modelled as an on-off source with exponentially distributed talk and silent spurts. The data traffic (both types) have Poisson-arrival packets. For simplicity of the simulation we define the same delay deadline for all packets in a data type. The channel model is the product of a Rayleigh fading process and a lognormal shadowing process.

In our simulation, we assume that at each frame time the voice traffic has the highest priority and is always transmitted. The remaining resource (virtual bandwidth) after considering the voice is allocated among the two types of data users using the packet scheduling algorithm in section 4.2. The assigned transmission rate (for flows with positive priority) has a lower limit of $R_{min} = 14.4$ Kbps. For simplicity we assign each data rate as multiples of the "basic" rate 14.4 Kbps.

Setting the outage probability $\delta = 0.1$, from (9) we know that $N_v = 43$ voice users, $N_{d,1} = 3$ type I data users, and $N_{d,2} = 2$ type II data users can be accommodated in the cell. Operating on these parameters, Table 4.2 shows the simulation results of our packet scheduling algorithm (the "with channel status" part), where we can see that the average rates of data users are very close to their target values (14.4 Kbps with type I data and 144 Kbps with type II data), and the data loss rates are low. The average power listed in this table is actually the normalized unit-rate

Table 4.1: Simulation parameters (uplink)

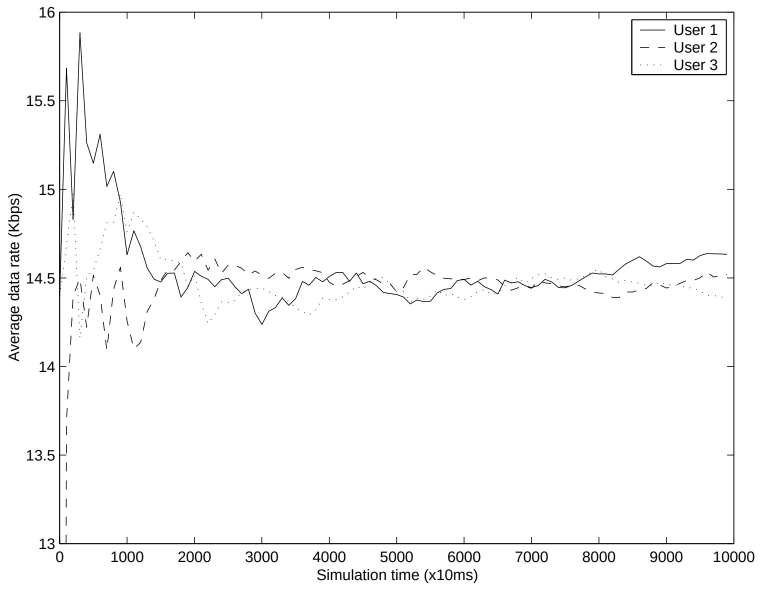
Item	Symbol	Value
System bandwidth	W	2.5 MHz
Noise to interference ratio	η	0.1
Rayleigh fading parameter	σ_r	0.5
Shadowing deviation	σ_ϵ	6 dB
Lag adjustment priority coefficient	α	1
Maximum lag deviation factor	μ	0.1
Channel status threshold	ξ	0.001
Outage probability	δ	0.1
Frame length	T_f	10 ms
Local average window	w_{av}	50 ms
Simulation length	T_t	50000 frames
Voice rate	R_v	14.4 Kbps
Voice activity factor	ρ_v	0.35
Voice BIR	γ_v	4 dB
Average talk spurt	T_v	0.7 s
Target data rate (I)	$R_{d,1}$	14.4 Kbps
Data BIR (I)	$\gamma_{d,1}$	6 dB
Data delay deadline (I)	d_1	100 ms
Data arrival rate per frame (I)	$\lambda_{d,1}$	144 bits
Target data rate (II)	$R_{d,2}$	144 Kbps
Data BIR (II)	$\gamma_{d,2}$	6 dB
Data delay deadline (II)	d_2	50 ms
Data arrival rate per frame (II)	$\lambda_{d,2}$	1.44 Kbits

Table 4.2: Simulation results for data users ($\xi = 0.001, \delta = 0.1$)

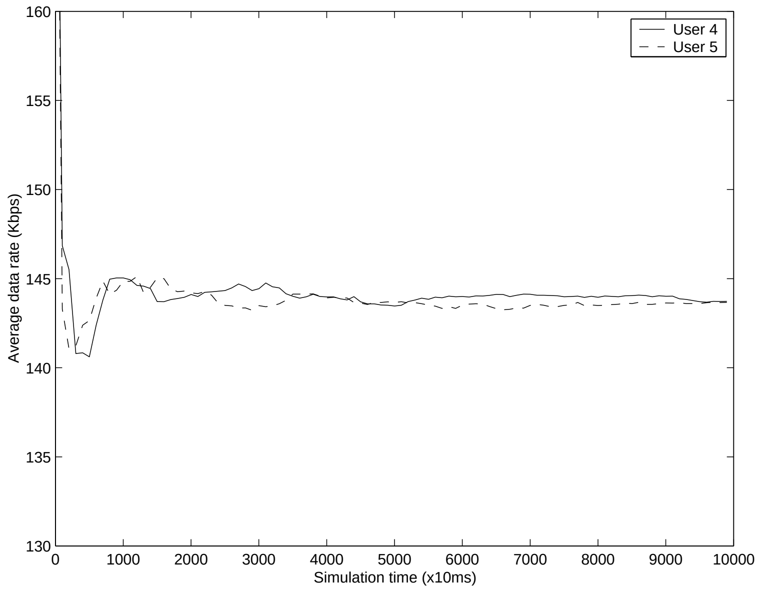
User id	Type	With channel status			No channel status		
		Avg data rate (Kbps)	Data loss rate	Avg power	Avg data rate (Kbps)	Data loss rate	Avg power
1	I	14.35	0.0005	99.2	14.36	0.0029	184.8
2	I	14.49	0.0008	85.4	14.36	0.0097	171.9
3	I	14.42	0.0007	89.9	14.34	0.0072	177.1
4	II	143.1	0.0010	92.4	143.9	0	179.3
5	II	143.8	0.0010	103.7	144.6	0	203.3

transmission power $s_t/\bar{s}$ defined in section 4.2.2. To avoid the ambiguity of packet size in the measurement, the delay and loss rate are calculated in terms of "bits". Figure 4.1 illustrates the average rate changes of data users at the beginning phase of the simulation.

None of the existing CDMA packet scheduling algorithms [2], [7], [50] have considered user channel variations. To see the effect of considering channel status in the packet scheduling, we simulate the algorithm with the simple rate assignment $R_t = R^* + \Delta R$ where no channel status is monitored. The simulation results are also shown in Table 4.2. It can be seen that under the channel status threshold $\xi = 0.001$ incorporating channel consideration can save almost half of the transmission power. Recall that in our algorithm a user is not allowed to transmit if the channel degradation is above the threshold. Hence, the performance of our packet scheduling algorithm is dependent on the threshold ξ. As seen from Table 4.3, where the loss rate and normalized unit-rate transmission power are averaged among all



(a) low speed users



(b) high speed users

Figure 4.1: Convergence of user data rate ($\delta = 0.1$)

Table 4.3: Performance of the algorithm with different ξ ($\delta = 0.1$)

Threshold ξ	Avg loss rate	Avg power
0.001	0.0009	93.29
0.005	0.0086	77.67
0.01	0.0115	75.73
0.05	0.0363	64.15

data users, that the two objectives of loss rate reduction and transmission power saving actually compromise each other with the change of ξ.

In another experiment we show the effectiveness of guaranteeing user fairness by our packet scheduling algorithm, which can not be achieved by existing CDMA packet scheduling algorithms. We compare the performance of our algorithm with the packet scheduling without considering data rate deviation (lag) adjustment as in the existing algorithms (this is equivalent to setting $\alpha = 0$ in (4.7) and $\Delta R = 0$ in (8a)). To facilitate the comparison, we let two "malignant" users, one high speed and the other low speed, generate 20% more traffic than what is specified in their target data rates (i. e. their data arrival rates are 20% higher than other users of the same type). From the results in Table 4.4, we see that in packet scheduling without lag adjustment the malignant users (user 3 and user 5) can actually transmit at rates 20% higher than the target rate, while in our algorithm with the lag adjustment the malignant users are forced to drop packets so that their specified target rates are maintained. In this experiment the maximum rate deviation factor μ is set to be 0.01, so the rate deviation is confined within 1% of the target rate of each user.

Table 4.4: Simulation results with malignant users* ($\xi = 0.001, \delta = 0.1, \mu = 0.01$)

User id	Type	With lag adjustment		No lag adjustment	
		Avg data rate (Kbps)	Data loss rate	Avg data rate (Kbps)	Data loss rate
1	I	14.36	0.0027	14.27	0.0001
2	I	14.38	0.0077	14.36	0.0002
3*	I	14.54	0.1549	17.19	0.0001
4	II	143.3	0.0040	142.9	0.0066
5*	II	145.4	0.1587	171.6	0.0051

Table 4.5: Change of measurements with outage probability

Outage probability	Calculated loss rate	Simulated loss rate	Avg virtual bandwidth (MHz)
0.1	0.0055	0.0009	1.863
0.2	0.0130	0.0031	1.973
0.3	0.0212	0.0050	2.032
0.4	0.0360	0.0086	2.126
0.5	0.0505	0.0182	2.177

Finally, let us look at the effect of outage probability change on the packet scheduling. In section 4.2.3 we have provided a method of estimating data loss rate from outage probability (eq. (11)). However, Table 4.5 shows that our packet scheduling algorithm has much lower loss rates than those calculated from (11). As mentioned in section 4.2.3, this is because the packet scheduling algorithm takes advantage of acceptable delay and queueing of the packets, and thus is able to send the "jammed" packets at later times when there is enough system resource. Therefore, from the perspective of data loss rate, with the packet scheduling algorithm the system is usually able to accommodate more packet data users than what flow admission control allows. It is also shown in Table 4.5 that higher bandwidth utilization (characterized by the virtual bandwidth) can be achieved at the cost of higher data loss rate.

Chapter 5

Utility-oriented Adaptive Service and Bandwidth Allocation

5.1 System Modeling

5.1.1 Modeling the Time-varying Links

It has been proven that the finite-state Markov channel (FSMC) can accurately model both fading and shadowing channels [4], [58], [36]. Each channel state corresponds to some channel quality and/or response at the receiver. The quantitative measures of

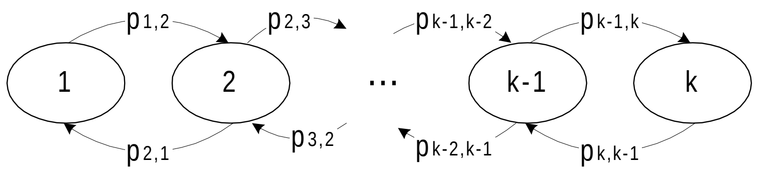


Figure 5.1: State transition diagram of a k-state Markov Channel

mutual information, or other parameters. Figure 5.1 shows the state transition diagram of a k-state Markov channel model. To completely describe such a Markov channel, we need the state transition probabilities, average state-holding times, and some parameter that reflects the physical characteristics of each state. The transition probabilities can be specified by a transition probability matrix $\mathcal{P}$ as shown in (1), where $p_{i,j}$ is the transition probability from state i to state j.

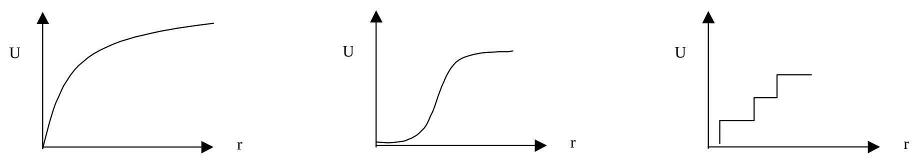
$$\mathcal{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,k} \\ \cdots & \cdots & \cdots & \cdots \\ p_{k,1} & p_{k,2} & \cdots & p_{k,k} \end{bmatrix} \tag{1}$$

To avoid confusion, from now on we use *channel* to represent the overall wireless channel shared by all users, and *link* to represent a wireless link between two specific communicating parties (e.g. a base station and a mobile user). Each wireless link is modeled by an FSMC. Assuming that all the users move freely in the same region, all the links are independent and identical. To capture the link characteristics of each state of the Markov chain, we associate each state m with a parameter called *bandwidth degradation ratio* D_m, where $0 \leq D_m < 1$. The *bandwidth degradation ratio* represents the overall degree of bandwidth wastage incurred by unsuccessful transmissions, coding overhead, and other factors. More specifically, if the bandwidth allocated to the user is r, and its link is currently in state m, $D_m \cdot r$ of bandwidth will be wasted. We call $(1 - D_m) \cdot r$ the *effective bandwidth* received by the user.

5.1.2 Utility-oriented Adaptive QoS Service Model

Utility, a concept originally used in economics, has been brought into networking research [10], [49], [52], in recent years. The utility represents the "level of satisfaction" of a user or the performance of an application. A utility function, which is monotonically non-decreasing, describes how the utility received by a user changes with the amount of resources allocated or service delivered. The key advantage of the utility function is that it inherently reflects a user's QoS requirements and can quantify the adaptability of a user or an application. Also, knowing applications' utility functions can facilitate more effective use of network resources. Some example of utility functions [1] of different types of applications are shown in Figure 5.2. Figure 5.2 (a) shows a possible utility curve of elastic applications, such as FTP or email. There is a diminishing marginal rate of performance enhancement as resource is increased, so the curve is strictly concave everywhere. Figure 5.2 (b) shows an example of real-time applications, such as streaming video. For this type of application, the utility increases rapidly after the resource received exceeds certain threshold and then tends to saturate as resource continues to increase. The last example shows a utility function of a multimedia application using layered coding. Delivering each additional layer has certain resource demand and a layer is either delivered completely or not delivered, therefore, the function is a staircase function.

[1]A utility function can be a multiple-variable function, with each variable representing one type of resource. For simplicity and without loss of generality, we only use one-variable utility function in this paper. Our work is readily extendable to multiple-variable utility functions.



(a) email or FTP (b) real-time audio/video (c) layered coding

Figure 5.2: Examples of utility functions

In an adaptive QoS model, the user applications are required to be adaptable to service degradations, and the bandwidth allocated to the user is not fixed, but adjusted according to the condition of the network. We propose a utility-oriented adaptive QoS service model for wireless networks. In this service model, each user [2] i signals its utility function $U_i(r)$, *minimum utility level* $u_{i,min}$ and *maximum utility level* $u_{i,max}$ to the network, where r is the amount of *effective bandwidth* received by the user. At any time instance, the *instant utility value* of the user is either zero or in the range of $[u_{i,min}, u_{i,max}]$. The network tries to dynamically allocate bandwidth such that each user's *instant utility* is maintained above $u_{*,min}$ and in the long run the bandwidth is allocated fairly and utilized efficiently.

5.1.3 Formulation of the Bandwidth Allocation Problem

Based on the service model and the wireless link model we have discussed, the following is the complete description of the bandwidth allocation problem. There

[2]The word "user" represents either a user or an application.

are n active users who share transmission bandwidth R. The communication link of each user follows a k-state Markov channel model. The statistical characteristics of the Markov model for all links are *i.i.d.* The average state-holding time of link state m is t_m, and the *bandwidth degradation ratio* of the state is D_m, where

$$0 \leq D_m < 1, \ \forall \ 1 \leq m \leq k \tag{2}$$

Without loss of generality, we assume that

$$D_m > D_n, \ if \ m < n \tag{3}$$

To associate the *degradation ratio* to each user, if user i's link is currently in the mth state, $D_{i,m}$ will be used. The utility function of user i is $U_i(r)$, where

$$\begin{cases} U_i(r) = 0 & : \quad r < r_{i,min} \\ u_{i,min} \leq U_i(r) \leq u_{i,max} & : \quad r_{i,min} \leq r \leq r_{i,max} \\ U_i(r) = u_{i,max} & : \quad r > r_{i,max} \end{cases} \tag{4}$$

We call $r_{i,min}$ the *minimum effective bandwidth level* and $r_{i,max}$ the *maximum effective bandwidth level* of user i. If at a particular time instance, user i is allocated r_i amount of bandwidth and its link is in state m, then the *instant utility* it receives is

$$u_i = U_i((1 - D_{i,m}) \cdot r_i) \tag{5}$$

To fully utilize the bandwidth, no bandwidth is reserved at any time, i.e.

$$\sum_{i=1}^{n} r_i = R \tag{6}$$

is always satisfied. It is assumed that each user i, $i = 1, 2, \cdots n$, always generates enough traffic to fully consume the allocated bandwidth as long as the *effective bandwidth* it receives does not exceed $u_{i,max}$. How do we dynamically allocate the bandwidth?

To answer the above question, we first need to clarify our objectives. The key issues of all bandwidth allocation problems are: QoS requirements, fairness, and bandwidth utilization.

1. *QoS requirements*: In the utility-oriented adaptive service model, the users' QoS requirements are reflected in the utility functions. One of the objectives of the bandwidth allocation scheme is to guarantee the *minimum utility level* of each user. Considering the unpredictable nature of wireless links, such guarantees should be probabilistic rather than deterministic, in order to achieve high utilization of the bandwidth. Define *utility outage* as the event that user i's *instant utility* level falls below $u_{i,min}$. Therefore, the bandwidth allocation scheme should guarantee for each user that the probability of *utility outage* is smaller than a certain threshold p_{outage}.

2. *Fairness*: Since the ultimate service criterion is the amount of received utility, the fairness criterion should also be based on utility. The fairness objective is that in the long term every user should experience the same amount of normalized utility increase/decrease. Consider users i and j with *average utility*,[3] $u_{i,avg}$ and $u_{j,avg}$, respectively,

$$G_i = \frac{u_{i,avg} - u_{i,min}}{u_{i,min}} \tag{7}$$

represents the *normalized gap* of the *average utility* received by user i and its *minimum utility level.* Since the average statistics of all user's link models are the same, we want all users to have the same *normalized gap* in the long run, i.e.

$$G_i \simeq G_j, \forall i, j \tag{8}$$

3. *Bandwidth utilization*: The total effective utility delivered is the criterion for measuring the bandwidth utilization. The more efficiently the bandwidth is utilized, the more total utility is delivered. Therefore, for a fixed amount of bandwidth we try to acheive a large value of $\sum_{i=1}^{n} u_{i,avg}$, where n is the total number of users.

[3]In this general model, the averaging method and the measuring window are flexible. It can be simple time average or weighted average, and the measuring window may start from the beginning of the data session or may just be a fixed time interval, which should be much larger than the state-holding times. The choice can be up to the service provider.

5.2 Bandwidth Allocation Scheme

5.2.1 Adaptive Bandwidth Allocation

There are three factors to consider when allocating bandwidth. These are link states, utility functions, and fairness. To improve the bandwidth utilization, an intuitive idea would be to dynamically adjust a user's bandwidth share when its link state changes, and allocate more bandwidth to users with better links and/or with more efficient mapping from the allocated bandwidth to utility. However, in the long run such allocation should not result in discrimination against any user.

When a wireless link changes to a state with a larger $D_{i,m}$, we say the link degrades. When a link changes to a state with a smaller $D_{i,m}$, we say the link upgrades. The basic idea of the bandwidth allocation scheme is that when a user's link degrades, it may surrender some bandwidth to another user with a smaller *normalized gap*, such that there is a net gain in the combined *instant utility*. When a link upgrades, the user may receive some bandwidth from another user with a larger *normalized gap* to achieve a gain in the combined *instant utility*.

In real systems, tracking a link's state changes is achievable (e.g. by measuring average SNR level) with small tracking delays for *slow link variations*. We assume that accurate knowledge of link states is available. Thus the tracking delay is zero.

The adaptive bandwidth allocation mechanism is invoked whenever there is a link-state change. Assume at a particular time user i's link state changes to state p. The following four steps are the operations performed.

1. All the users' *average utility* level and *normalized gap* are updated.

2. Users are sorted in increasing order of *normalized gap*.

3. If the *instant utility level* of user i is below $u_{i,min}$, some other users' bandwidth will be reduced and reallocated to user i to meet its $u_{i,min}$.

4. If there is no step 3, user i may give up part of its bandwidth to another user if the link degrades, whereas it may receive some extra bandwidth if the link upgrades.

We call the user who gives up part of its bandwidth to others the *benefactor*, and the user who receives bandwidth from others the *beneficiary*. In step 3, to satisfy user i's $u_{i,min}$, the bandwidth allocation scheme searches for *benefactor(s)* starting from the user with the largest *normalized gap*. A larger *normalized gap* means the user has gone ahead of other users in terms of normalized utility received. For the sake of fairness, such users should be among the first to give up some share of their bandwidth when other users need more bandwidth to sustain their *minimum utility* *levels*. Suppose the user with the largest *normalized gap* is user j, whose link is currently in state q and it is above $u_{j,min}$. User j will yield

$$min(\frac{r_{i,min}}{1-D_{i,p}} - r_i,\ r_j - \frac{r_{j,min}}{1-D_{j,q}}) \tag{9}$$

amount of bandwidth to user i, where r_i and r_j are the bandwidth allocated to users i and j, respectively, before the link state transition. If $min(\ )$ takes the value of the second term in the parenthesis, it means user j can provide enough bandwidth to user i to satisfy $u_{i,min}$ while maintaining $u_{j,min}$. If $min(\ )$ takes the value of the first term in the parenthesis, it means user j's surplus bandwidth alone is insufficient for user i to reach $u_{i,min}$. In this case, the *instant utility* of user j is kept at the minimum level, and all surplus bandwidth is allocated to user i. Then the user with the second largest *normalized gap* will be the next candidate for *benefactor*. This procedure will be repeated until $u_{i,min}$ can be reached or all the users have been checked.

If after the link state transition, user i is still above $u_{i,min}$, user i's bandwidth share may be adjusted to improve the bandwidth utilization. When user i's link degrades, the bandwidth allocation scheme will search for an appropriate *beneficiary* to receive some bandwidth from user i and decide the amount to be transferred. The users with smaller *normalized gaps*, which means they are lagging behind other users in receiving effective service, are given higher priority to be a *beneficiary*. Therefore, the users are checked in increasing order of *normalized gap*. Suppose user j, whose link is in state q, is the *beneficiary* candidate being checked. We have

$$\begin{cases} u_i = U_i((1 - D_{i,p}) \cdot r_i) \\ u_j = U_j((1 - D_{j,q}) \cdot r_j) \end{cases} \tag{10}$$

The bandwidth allocation scheme tries to maximize the combined *instant utility* of the two users with some constraints. The optimization problem is:

maximize:

$$u'_i + u'_j \tag{11}$$

where

$$\begin{cases} u'_i = U_i((1 - D_{i,p}) \cdot (r_i - x)) \\ u'_j = U_j((1 - D_{j,q}) \cdot (r_j + x)) \end{cases} \tag{12}$$

subject to:

$$\begin{cases} x \geq 0 \\ u'_i \geq u_{i,min} \\ u'_j \geq u_{j,min} \end{cases} \tag{13}$$

(13) can be simplified as

$$max(0,\ \frac{r_{j,min}}{1 - D_{j,q}} - r_j) \leq x \leq r_i - \frac{r_{i,min}}{1 - D_{i,p}} \tag{14}$$

Since the utility functions are bounded and monotonically non-decreasing and the constraints are linear, the above optimization problem is guaranteed to have solution(s), which can be easily solved by numerical methods[4] [29], [59].

If the solution of $x > 0$, then reallocate the bandwidth of users i and j as

$$\begin{cases} r_i = r_i - x \\ r_j = r_j + x \end{cases} \tag{15}$$

If $x = 0$, then the same procedure is repeated for the user with the next smallest *normalized gap*. This procedure is repeated until one *beneficiary* is found or all the users with smaller *normalized gap* than user i's have been searched.

The main difficulty in allocating bandwidth is how to combine utilization and fairness considerations and strike a balance between achieving high bandwidth utilization and fairness among users. If achieving high bandwidth utilization is the sole objective, some users may suffer starvations. If absolute fairness, such as keeping all the users at the same instant utility level in [10], is maintained at all times, bandwidth utilization is sacrificed. The operations described actually combine the considerations of both long-term fairness and short-term maximization of bandwidth utilization. First, only users who are lagging behind user i in *normalized average utility* are in the *beneficiary* candidate list. Considering long-term fairness objective, when a user gives up its bandwidth, such bandwidth is transferred to the users

[4]Although in limited cases a local optimal instead of the global optimal may be found as a solution using numerical methods when the objective function (11) is not strictly concave, it is still a valid solution, because as long as $x > 0$ we still get improvement in the overall *instant utility*.

who have received less utility than its fair share, so that they can catch up. The smaller the user's *normalized gap*, the higher its priority in the candidate list. Second, reallocating bandwidth between the *benefactor* and the *beneficiary* is aimed at maximizing the combined *instant utility*, and hence the bandwidth utilization.

Similarly, when user i's link upgrades, user i becomes the *beneficiary* and users with larger *normalized gap* are the candidates for *benefactor*. The scheme checks the candidates starting from the one with the largest *normalized gap*. Suppose user j, whose link is in state q, is the *benefactor* candidate being checked. The optimization problem becomes:

maximize:

$$u'_i + u'_j \tag{16}$$

where

$$\begin{cases} u'_i = U_i((1 - D_{i,p}) \cdot (r_i + x)) \\ u'_j = U_j((1 - D_{j,q}) \cdot (r_j - x)) \end{cases} \tag{17}$$

subject to:

$$max(0, \frac{r_{i,min}}{1 - D_{i,p}} - r_i) \leq x \leq r_j - \frac{r_{j,min}}{1 - D_{j,q}} \tag{18}$$

If $x > 0$, then reallocate the bandwidth of user i and j as

$$\begin{cases} r_i = r_i + x \\ r_j = r_j - x \end{cases} \tag{19}$$

If $x = 0$, then the same procedure is repeated for the user with the next largest *normalized gap*. This is repeated until one *benefactor* is found or all the users with larger *normalized gap* than user i's have been searched.

To summarize, in the long-term the scheme tries to achieve fair allocation by compensating lagging users and penalizing leading users, while in the short-term the scheme tries to make the best of the bandwidth available by maximizing *instant utility*. At any time, maintaining the users' *minimum utility level* is of the highest priority.

Note that the algorithm is a centralized algorithm performed by the base station or a central control unit which has sufficient computational power. Since wireless networks are usually edge networks, the number of users in the network is not expected to be very large. Therefore, the computational load of the operations, such as updating normalized gap, is not a major issue.

Besides the link state changes, adjustments in bandwidth allocation is also needed when the overall available bandwidth changes or users arrive/depart. In fact, the two situations are quite similar to the bandwidth allocation scheme. When a new user arrives, it is allocated enough bandwidth to let it reach its *minimum utility level*. Since no bandwidth is reserved for new users, some current users' bandwidth must be reduced to accommodate the new user. For reduction in the overall bandwidth, the amount reduced is also obtained from the current users. Therefore, for users currently in the systems both cases result in reduction in their bandwidth share.

Similarly, increases in the overall bandwidth and user departures mean bandwidth increment for the current users.

If r is the amount of deficient bandwidth which needs to be collected from the current users, either because of a decrease in overall bandwidth or a user's arrival, user j having the largest *normalized gap* G_j is to give up $min(max(0,\ r_j - \frac{r_{j,min}}{1-D_{j,q}}),\ r)$ amount of bandwidth, where q is the current link state of user j. If it is still not enough, the user with the second largest *normalized gap* is chosen to give up bandwidth to make up the deficit. This procedure will be repeated until enough bandwidth has been collected or all the current users have been searched. If after searching all the current users, the collected bandwidth is still not enough, the bandwidth allocation scheme will start the second round of collection, again starting from the user with the largest *normalized gap*. This time each chosen user will give up all of its bandwidth or the deficient amount until the deficit becomes zero. Similarly, if there is surplus bandwidth, the users with the first k smallest *normalized gap* are chosen to receive the surplus bandwidth. Each user can increase its *effective bandwidth* up to the *maximum effective bandwidth level.*

5.2.2 Admission Control and Utility Outage Probability

It is clear that to guarantee users' *minimum utility level,* certain admission control should be enforced to limit the number of users in the system. Given an FSMC's transition probability matrix $\mathcal{P}$, the steady state probability $\Pi = [\pi_1, \pi_2, ...\pi_k]$ can be calculated by solving the following vector equation.

$$\Pi = \Pi\mathcal{P} \tag{20}$$

If state i's average holding time is t_i, then at a particular time the probability of the link being in state i is

$$p_i = \frac{\pi_i \cdot t_i}{\sum_{i=1}^{k} \pi_i \cdot t_i} \tag{21}$$

Recall that when a user's *instant utility* falls below its *minimum utility level*, then there is a *utility outage* for the user. If the total bandwidth needed to keep all the users above their *minimum utility levels* is larger than the available bandwidth, then there will be at least one user experiencing a *utility outage*. The probability p_o of such event at any time is

$$\begin{aligned} p_o &= P_r\{\sum_{i=1}^{n} \frac{r_{i,min}}{1 - D_{i,m_i}} > R\} && (22) \\ &= \sum_{A} \prod_{1 \le i \le n} p_{m_i} && (23) \end{aligned}$$

where $A = \{m_1, m_2, \ldots m_n \mid 1 \le m_1, m_2 \ldots m_n \le k, \sum_{l=1}^{n} \frac{r_{l,min}}{1-D_{l,m_l}} > R\}$, R is the total available bandwidth, m_i is user i's link state at the time instance. It is obvious that $p_{i,o} \le p_o$, where $p_{i,o}$ is the *utility outage probability* of user i. Therefore, if we can guarantee that $p_o \le p_{outage}$, where p_{outage} is the desired threshold, we will have $p_{i,o} \le p_{outage}$.

Consequently, the admission control policy is as follows. When a new user arrives, calculate p_o according to (23), where n is the total number of users including the new user. If $p_o \leq p_{outage}$, the new user is admitted; otherwise, the user is rejected. If a user j is admitted, it is initially allocated $r_{j,min}/(1 - D_{j,q})$, where q is user j's current link state. The assigned amount of bandwidth to j is contributed by the users currently in the network following the algorithm we described previously. Given p_{outage}, R, the channel model and all the possible classes of utility functions, (23) can also be used to find the feasible region of providing statistical minimum utility guarantees.

5.3 Numerical Results

In our simulations, we choose three classes of users corresponding to the three types of utility functions in Figure 5.2. However, we use piecewise linear functions instead of high order non-linear functions to characterize the utility curves. The reasons of using such piecewise linear functions are as follows. First, a piecewise linear function can approximate a high order non-linear function fairly well. We just need to increase the number of line pieces to increase accuracy. Second, the utility, which represents a user's perception of service quality at the application level, is often measured by subjective tests using mean-opinion score (MOS) [49]. In such tests, the utility curves are modeled as piecewise linear functions. Experimental results on measuring MPEG-2 video quality show that the utility function of some video on demand

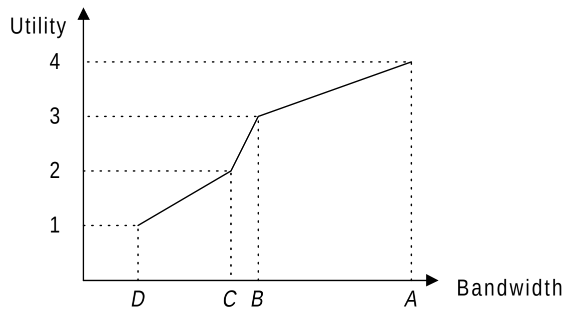


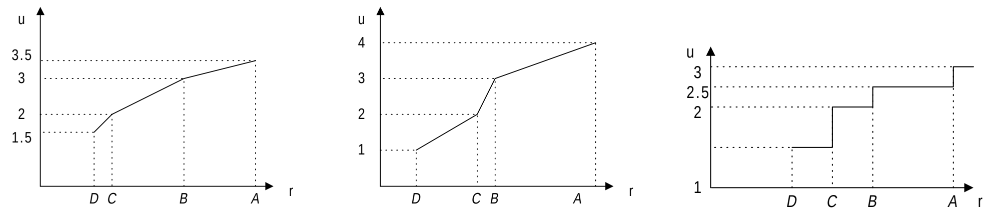
Figure 5.3: An example of utility function for MPEG-2 video

applications can be modeled as the one shown in Figure 5.3 [49], where $C = 1.51D$, $B = 1.6D$, and $A = 2.41D$. Third, since the utility curves are heterogeneous, a user signals its utility curve to the network by specifying a list of key points on the curve and the network connects those points using lines to get the utility curve, which is indeed a piecewise linear function. Last, piecewise linear functions simplify the computation in optimization.

Without loss of generality, we use the utility functions in Figure 5.4 to represent three classes of users. The horizontal axis is the *effective bandwidth* allocated to a user, and the vertical axis is the utility received. In each figure, D is the *minimum effective bandwidth level* and A is the *maximum effective bandwidth level.* The total available bandwidth $R = 100$. Since all the values are of relative importance only, we do not specify any units for the parameters in the simulation.

A three-state Markov channel model is used to model wireless links. The transition probability matrix is shown in (24). The average state-holding times are $t_1 = 3$,

Figure 5.4: Three classes of users in the simulations



(a) Class I: $D = 7.5$, $C = 10$, $B = 20$, $A = 30$

(b) Class II: $D = 15$, $C = 23$, $B = 24$, $A = 36$

(c) Class III: $D = 10$, $C = 15$, $B = 20$, $A = 30$

$t_2 = 5$, and $t_3 = 4$, and the degradation ratios are $D_1 = 0.4$, $D_2 = 0.2$, and $D_3 = 0$, where the subscript denotes the corresponding state.

$$\mathcal{P} = \begin{bmatrix} 0 & 0.8 & 0.2 \\ 0.5 & 0 & 0.5 \\ 0.2 & 0.8 & 0 \end{bmatrix} \tag{24}$$

Since there is no existing bandwidth allocation scheme for networks with multiple-state (more than two) links, the adaptive bandwidth allocation scheme is compared with a static bandwidth allocation scheme in terms of utility outage times, fairness property and bandwidth utilization, which correspond to the three objectives defined at the beginning of the paper. In the static bandwidth allocation scheme, each user is allocated a fixed amount of bandwidth r_i, given by

$$r_i = R\frac{r_{i,min}}{\sum_{i=1}^{n} r_{i,min}} \tag{25}$$

Table 5.1: Comparisons when $K_1 = 2$, $K_2 = 2$, and $K_3 = 2$

	Normalized Gap		Utility Outage Time	
	Adaptive	Static	Adaptive	Static
user 1	0.35304	0.04177	23.2607	2012.6972
user 2	0.35325	0.03624	19.3312	2038.2643
user 3	0.35320	0.40172	41.5280	1985.2418
user 4	0.35398	0.38246	41.5824	2148.8781
user 5	0.35330	0.07510	10.0698	1913.9185
user 6	0.35298	0.11179	11.9377	1878.1480
Utilization Improvement			17.12%	

where $r_{i,min}$ is each user's *minimum effective bandwidth level* and n is the number of users. The initial allocation of the bandwidth for the adaptive scheme at the beginning of the simulation also follows (25). For fair comparison, the overall available bandwidth R and the user set are fixed during each simulation. The duration of each simulation is 10000.

Table 5.1 shows the numerical results of the case where there are two users in each class. The table shows that for the adaptive scheme the maximum difference in *normalized gap* between any two users is 0.001 (between user 4 and user 6), whereas for the static scheme the corresponding value is 0.36548 (between user 3 and user 1). Since the difference in *normalized gaps* is a measure of the fairness, it is clear the adaptive scheme is much fairer. The result also shows that users experience much less utility outage time in the adaptive scheme than in the static scheme. Using equation (23) it can be calculated that the upper bound p_o is 0.00416, which corresponds to the maximum *utility outage time* 41.6. It is shown that none of the user's *utility*

Table 5.2: Comparisons when $K_1 = 2$, $K_2 = 1$, and $K_3 = 3$

	Normalized Gap		Utility Outage Time	
	Adaptive	Static	Adaptive	Static
user 1	0.45199	0.11268	2.1284	2012.6972
user 2	0.45218	0.10734	0	2038.2643
user 3	0.45186	0.72108	0	1985.2418
user 4	0.45218	0.06785	0	2148.8781
user 5	0.45228	0.07510	0	1913.9185
user 6	0.45175	0.11179	0	1878.1480
Utilization Improvement			22.38%	

Table 5.3: Comparisons when $K_2 = 2$ and $K_3 = 2$

	$max \mid G_i - G_j \mid, \forall i, j$		Max. Utility Outage Time		Utilization Improvement
	Adaptive	Static	Adaptive	Static	
$K_1 = 1$	0.00068	0.75880	0	0	6.35%
$K_1 = 2$	0.00100	0.36548	41.5824	2038.2643	17.12%
$K_1 = 3$	0.00205	0.30330	674.4124	2192.5303	21.58%
$K_1 = 4$	0.00147	0.16114	2046.0076	7407.9264	219.84%

outage time is larger than the upper bound. Finally, there is a 17.12% utilization improvement in terms of the total utility delivered for the adaptive scheme.

Various user sets are tested in our simulations. For illustration another example is presented in table 5.2. From Tables 5.1 and 5.2 it is clear that the adaptive scheme outperforms the static scheme in all three aspects: minimum QoS guarantee, fairness, and utilization.

Since in the static scheme there is no dynamic adjustment of bandwidth, when the network load increases as the number of user increases the users will spend a great amount of time experiencing *utility outage*, which lowers bandwidth utilization significantly. However, in the adaptive scheme such a problem is less serious because of dynamic allocation. Therefore, it is expected that the utilization improvement of the adaptive scheme is more significant with higher network load. This is demonstrated by the data in Table 5.3. In Table 5.3 the number of class II and III users are fixed at $K_2 = K_3 = 2$, while the number of class I users is increased from 1 to 4. Columns 2 and 3 show the maximum difference of *normalized gaps* between any two users. Columns 4 and 5 show the maximum total *utility outage* time experienced by any user. Column 6 shows the utilization improvement of the adaptive scheme over the static scheme. We see that as K_1 increases, the advantages of the adaptive scheme strengthens in terms of utilization improvement and *utility outage* time. In particular, when $K_1 = 4$ and the network is heavily loaded, the utilization improvement is dramatic. At the same time, the fairness property of the adaptive scheme is maintained.

The simulation results show that the adaptive bandwidth allocation scheme achieves all three objectives listed in the previous section.

Chapter 6

Conclusions and Future Work

In this dissertation, we studied packet scheduling and bandwidth allocation for adaptive service in wireless networks.

We studied packet scheduling issues in both TDMA-based and CDMA-based wireless networks and proposed three novel wireless scheduling algorithms applicable to different link and network models. First, we designed a scheduling algorithm for TDMA networks with wireless links being modeled as having two states. Compared with some recent wireless scheduling scheme, the algorithm is shown to distribute excess bandwidth effectively, strike a balance between effort-fair and outcome-fair, and provide delay bound for error-free flows and transmission effort guarantees for error-prone flows. Second, extending the work for two-state wireless links, we proposed a novel wireless scheduling algorithm applicable to wireless links with multiple states. For delay-sensitive flows, the algorithm is capable of providing statistical delay violation bounds. For best-effort flows, we proposed a new notion of fairness, called long-term link-quality-weighted outcome-fair, which we believe is more suitable in wireless networks than pure outcome-fair or effort-fair. The algorithm balances between bandwidth efficiency requirement and fairness requirement, and guarantees minimal goodput levels for best-effort flows. Third, an adaptive packet scheduling algorithm for cellular CDMA network was proposed. Unlike other existing CDMA scheduling schemes that only considers data delay tolerance, our algorithm takes into account delay tolerance, average rate fairness, and channel variations altogether. It guarantees packet deadline and average data rate under the assumption of perfect power control.

Since fixed level of service guarantees and fixed level of resource allocation are not suitable in wireless networks, we proposed a general utility-oriented adaptive QoS model for wireless networks and established a framework for formulating the bandwidth allocation problem for users with time-varying links. To deal with slow link variations, we designed a high-level adaptive bandwidth allocation scheme. The scheme is capable of providing QoS guarantees, ensuring long-term fairness, and achieving high bandwidth efficiency.

In our work in Chapter 3, we assume that the link states in consecutive time slots are independent of each other to simplify the analyses. One of the our future work will be to study how to provide statistical delay guarantees when the independence assumption is relaxed. We expect that when heavy-tail correlation exists, it will be very difficult to provide delay guarantees while maintaining high bandwidth efficiency. However, if the correlation decays very fast in time, we conjecture that providing statistical delay guarantees with efficient use of bandwidth is achievable. To solve this problem, some asymptotic methods, such as large deviation technique, may be used.

Another interesting problem in wireless scheduling is how to minimize packet loss due to missing deadlines when data packets having predefined deadlines are transmitted on error-prone links. Some real-time applications generate data that become useless if not delivered within certain time limit. In wireline networks, it is proven that EDF is the optimal scheduling algorithm in delivering such traffic. However, EDF can not be optimal in wireless networks, because the variability of wireless links. To minimize packet loss, the scheduler needs to consider not only the packet deadlines but also instantaneous link quality and link quality statistics when scheduling the packets. In such a scenario, we first need to find whether there exists an optimal scheduling algorithm. If the answer is positive, we will try to find the optimal algorithm(s).

In addition, we plan to extend our utility-oriented QoS service model to address the multi-dimensional resource allocation problem. In this case, the network resources will include not only bandwidth but also transmission power and other necessary resources. The corresponding adaptive bandwidth allocation scheme will also be extended to a general adaptive resource allocation scheme responsible for allocating more than one types of resources. Since the dynamics of physical wireless links exist over a wide range of time scales (from microseconds to seconds), an ideal system should incorporate adaptive mechanisms working at all levels of time scales. Therefore, we will try to integrate the high-level adaptive resource allocation scheme with the low-level adaptive packet scheduling algorithms. We believe the integration will improve further the resource efficiency and achieve better QoS performance.

Reference List

[1] J. Agosta and T. Russell, CDPD - Cellular Digital Packet Data Standards and Technology, McGraw-Hill, 1997.

[2] 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, Apr. 1999.

[3] I. F. Akyildiz, J. McNair, L. C. Martorell, R. Puigjaner and Y. Yesha, “Medium access control protocols for multimedia traffic in wireless networks,” IEEE Networks, pp. 39-47, Jul.-Aug. 1999.

[4] F. Babich and G. Lombardi, “A Markov model for the mobile propagation channel,” IEEE Trans. Vehicular Tech., pp. 63-73, Jan. 2000.

[5] R. J. Bates, GPRS, McGraw-Hill, 2002.

[6] J. R. Bennett and H. Zhang, “WF2Q: worst-case fair weighted fair queueing,” IEEE INFOCOM’96, vol. 1, pp. 120-128, San Francisco, March, 1996.

[7] F. Berggren, S.-L. Kim, R. Jantti, and J. Zander, “Joint power control and intracell scheduling for wireless multimedia networks,” IEEE J. Select. Areas Commun., vol. 19, no. 10, pp.1860-1870, Oct. 2001.

[8] P. Bhagwat, A. Krishna, and S. Tripathi, “Enhancing throughput over wireless LANs using channel state dependent packet scheduling,” Proc. of INFOCOM’96, pp. 1133-1140, March 1996.

[9] G. Bianchi, A. Campbell and R. Liao, “On utility-fair adaptive services in wireless networks,” 1998 Sixth International Workshop on Quality of Service, pp. 256-267, 1998.

[10] G. Bianchi, and A.T. Campbell, “A Programmable MAC Framework for Utility-based Adaptive Quality of Service Support,” IEEE J. Select. Areas Commun., vol. 18, no. 2, pp. 244-256, Feb. 2000.

[11] L. Bos and S. Leroy, “Toward an all-IP-based UMTS system architecture,” IEEE Network, vol. 89, no, 1, pp. 76-87, Jan. 2001.

[12] 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.

[13] Y. Cao and V. O. K. Li, "Utility-oriented adaptive QoS and bandwidth allocation in wireless networks," Proc. of IEEE ICC'02, May. 2002.

[14] Y. Cao and V. O. K. Li, "Wireless packet scheduling for two-state link models," Proc. of IEEE GLOBECOM'02, Nov. 2002.

[15] Y. Cao and V. O. K. Li, "Scheduling algorithms for wireless networks," Proc. of International Computer Syposium'00, Dec. 2000.

[16] Y. Cao and V. O. K. Li, "Bandwidth-guranteed fair scheduling with effective excess bandwidth allocation for wirelss networks," under review, IEEE Trans. Vehicular Tech., 2002.

[17] Y. Cao and V. O. K. Li, "Scheduling delay-sensitive and best effort traffics in wireless networks," under review, IEEE ICC'03, 2002.

[18] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," IEEE J. Select. Areas Commun., vol. 13, no. 6, pp. 1048-1056, Aug. 1995.

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

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

[21] M. Elauod and P. Ramanathan, "Adaptive use of error-correction codes for real-time communication in wireless networks," Proc. INFOCOM'98, pp. 548-555, Mar. 1998.

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

[23] C. Fragouli, V. Sivaraman and M. Srivastava, "Controlled multimedia wireless link sharing via enhaced class-based queuing with channel-state dependent packet scheduling," Proc. of INFOCOM'98, vol. 2, pp. 572-580, March 1998.

[24] R. G. Gallager, Discrete Stochastic Processes, Kluwer Academic Publisher, 1996.

[25] V. K. Garg, IS-95 CDMA and CDMA 2000 - Cellular/PCS Systems Implementation, Prentice Hall, 2000.

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

[27] J. Gomez, A. T. Campbell and H. Morikawa, "A systems approach to prediction, compensation and adaptation in wireless networks," ACM WOWMOM'98, pp. 92-100, Dallas, Texas, October 1998.

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

[29] D. M. Himmelblau, Appied Nonlinear Programming, McGraw-Hill, 1972.

[30] http://www.isi.edu/nsnam.

[31] C.-L. I and R. D. Gitlin, "Multi-code CDMA wireless personal communications networks," Proc. IEEE ICC'95, pp. 1060-1064, Jun. 1995.

[32] Z. Jiang and N. K. Shankaranarayana, "Channel Quality Dependent scheduling for flexible wireless resource management," Proc. of GLOBECOM'01, pp. 3633-3638, 2001.

[33] J. Y. Hui, "Resource allocation for broadband networks," IEEE J. Select. Areas Commun., vol. 6, no. 9, pp. 1598-1608, Dec. 1988.

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

[35] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive survey," IEEE Personal Commun., pp. 10-30, June 1996.

[36] Y. Y. Kim and S. Q. Li, "Capturing important statistics of a fading/shadowing channel for network performance analysis," IEEE J. Select. Areas Commun., Vol. 17, pp. 888-901, May 1999.

[37] W. C. Y. Lee, Mobile Cellular Telecommunications: Analog and Digital Systems, McGraw-Hill, 1995.

[38] 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.

[39] S. Lu and V. Bharghavan, "Fair scheduling in wireless packet networks," IEEE/ACM Trans. on networking, vol. 7, no. 4, pp. 473-489, Aug. 1999.

[40] 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.

[41] M. Naghshineh and M. Willebeek-LeMair, “End-to-end QoS provisioning in multimedia wireless/mobile networks using an adaptive framework,” IEEE Commun. Magazine, pp. 72-8, Nov. 1997.

[42] T. S. E. Ng, I. Stoica and H. Zhang, “Packet fair queueing algorithms for wireless networks with location-dependent errors,” Proc. of IEEE INFOCOM’98, pp. 1103-1111, March 1998.

[43] T. Ojanpera and R. Prasad, WCDMA: Towards IP mobility and mobile Internet, Artech House, 2001.

[44] 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.

[45] J. D. Parsons, The Mobile Radio Propagation Channel, John Wiley & Sons, Second Edition, 2000.

[46] J. G. Proakis, Digital Communications, McGraw-Hill, Third Edition, 1995

[47] J. Qiu and E. W. Knightly, ”Inter-class resource sharing using statistical service envelopes,” Proc. of IEEE INFOCOM’99, pp. 1404-1411, 1999.

[48] P. Ramanathan and P. Agrawal, “Adapting packet fair queueing algorithms to wireless networks,” ACM/IEEE MOBICOM’98, pp. 1 - 9, Dallas, TX, 1998.

[49] D. Reininger, R. Izmailov, B. Rajagopalan, M. Ott and D. Raychaudhuri, “Soft QoS control in the WATMnet broadband wireless system,” IEEE Personal Communications, pp. 34-43, Feb. 1999.

[50] O. Sallent, J. Perez-Romero, F. J. Casadevall, and R. Agusti, “An emulator framework for a new radio resource management for QoS guaranteed services in W-CDMA systems,” IEEE J. SELECT. AREAS COMMUN., vol. 19, no. 10, pp. 1893-1904, Oct. 2001.

[51] H. Sariowan and R. L. Cruz, “SCED: a generalized scheduling policy for guaranteeing quality-of-service,” IEEE/ACM Trans. Networking, vol. 7, no. 5, pp. 669-683, Oct. 1999.

[52] S. Shenker, “Fundamental design issues for future internet,” IEEE J. SELECT. AREAS COMMUN., vol. 13 no. 7, pp. 1176 -1188, Sep. 1995.

[53] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," Proc. of ACM SIGCOMM'95, pp. 231-242, Berkeley, CA, USA, 1995.

[54] S. Singh, "Quality of service guarantees in mobile computing," Computer Communications, vol. 9, pp3. 59-371, Apr. 1996.

[55] M. Srivastava and P. Mishra, "On quality of service in mobile wireless networks," Proc. of the IEEE 7th Workshop on Network and Operating System Support for Digital Audio and Video, pp. 147-158, 1997.

[56] 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.

[57] I. Stoica, H. Zhang, and T. S. E. Ng, "A hierarchical fair service curve algorithm for link-sharing, real-time, and priority services," Proc. of ACM SIGCOMM'97, pp. 249-262, Sep. 1997.

[58] H. Wang and N. Moayeri, "Finite state markov channel - a useful model for radio communication channels," IEEE Trans. Vehicular Tech., pp. 163-171, Feb. 1995.

[59] W. Winston, Introduction to Mathematical Programming, Applications and Algorithms, Second Edition, Duxbury Press, 1995.

[60] S. Yajnik, J. Sienicki and P. Agrawal, "Adaptive coding for packetized data in wireless networks," Proc. PIMRC'95, vol. 1, pp. 338-342, 1995.

[61] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," Proc. of IEEE, vol. 83, no. 10, Oct. 1995.

[62] L. Zhang, "VirtualClock: A new traffic control algorithm for packet-switched networks," ACM Trans. Computer Systems, vol. 9, no. 2, pp.101-124, May 1991.

[63] Q. Zhang and S. Kassam, "Finite-state Markov model for rayleigh fading channels," IEEE Trans. Communications, pp.1688-1692, Nov. 1999.

[64] L. Zhuge, Y. Cao and V. O. K. Li, "Adaptive Packet Scheduling in Cellular CDMA," under review, IEEE INFOCOM'03, 2002.

[65] L. Zhuge and V. O. K. Li, "Interference estimation for admission control in multi-service DS-CDMA systems," Proc. of IEEE GLOBECOM'00, pp. 1509-1514, Nov. 2000.

[66] L. Zhuge and V. O. K. Li, "Forward link capacity in multi-service DS-CDMA systems," Proc. of IEEE GLOBECOM'01, pp. 609-613, Nov. 2001.