Innovative Opportunistic Scheduling Algorithms for Networks with Packet-Level Dynamic

by

Lina Ma

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.
I understand that my thesis may be made electronically available to the public.

Abstract

Scheduling in wireless networks plays an important role. The undetermined nature of the wireless channel is usually considered as an undesirable property. Recently, the idea of opportunistic scheduling is introduced and it takes advantage of the time-varying channel for performance improvement such as throughput and delay.

Since the introduction of opportunistic scheduling, there are two main bodies of works. The first body of works assume that each user is greedy and has infinite backlog for transfer. With this assumption, fairness objective becomes an important factor in designing a scheduling algorithm to avoid severe starvation of certain users. Typical fairness involve processor sharing time fairness, proportional fairness, and minimum performance guarantee. On the other hand, delay performance is not a appropriate factor to evaluate the effectiveness of a scheduling algorithm because of the infinite backlog assumption. In reality, this assumption is not true as data arrives and leaves the network randomly in practice.

The second body of works deal with the relaxation of the infinite backlog assumption. Thus, the notion of stability region arises. The definition of stability is that the queue at each source node remains finite. Stability region can be defined as the set of traffic intensities which can all be stabilized by the network. The well known throughput optimal algorithm is proven capable of achieving the largest stability region.

In this thesis, two innovative opportunistic scheduling algorithms which aim to minimize the amount of resources used to stabilize the current traffics are proposed. The key feature of our algorithms is that the incoming traffic rates are available to the scheduler, whereas the throughput optimal algorithm has no such prior traffic knowledge. Performance comparisons are made by means of simulation to demonstrate that the proposed algorithms can achieve the same stability region as the throughput optimal algorithm. Moreover, the delay performance is better than that of the throughput optimal algorithm, especially under heavy traffic conditions.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Catherine Rosenberg for her constant guidance, support and encouragement. Without her help, this thesis would have never been possible.

I would also like to thank Prof P. H. Ho and Prof L-L. Xie for being the readers of this thesis and for their valuable suggestions and comments.

I am also grateful to my best friend, Kexin Ma, who has always been supportive and patient throughout the years. His knowledge and exceptional research skills have given me valuable insights into this work. Thanks to all my friends and colleagues for their friendship and helpful advices.

Special thanks to the University of Waterloo for the financial support and research facilities. Many thanks to the administrative support staff Wendy Boles.

Lastly, I would like to thank my family for their endless love, encouragement and support. Without them, I would not become who I am today. This thesis is dedicated to my father and mother.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In wired line networks, the link capacities are fixed and there is no interference between adjacent links. Scheduling has been extensively studied in operations research. However, these scheduling policies can not be directly applied to the wireless domain due to time-varying channels and multiuser-diversity. Traditionally, the time-varying property of the wireless channel is considered undesirable, and many research have been done trying to overcome this problematic feature. New scheduling solutions are specifically designed and tailored for the wireless networks, and a new name is given to these policies – opportunistic scheduling.

The formal definition of opportunistic scheduling is that it is a scheduling policy which exploits the channel variations of users to achieve higher throughput. In simple words, the policy assigns the channel to the user with the best channel condition at current time and postpone the users whose channel conditions are poor. In addition, opportunistic scheduling can also provide quality of service (QoS) at the cost of throughput. In most of the recent literatures, opportunistic scheduling is performed slot by slot and the QoS is established in terms of time average.

There are two major bodies of opportunistic scheduling algorithms. The first body of works [11],[12],[14], and [15] adopt the assumption of saturated traffics, which means that each individual user is greedy and has infinite backlog to transfer. The second body of works [1],[3], and [7] relax this assumption and address the issue of the stability region. In this body of work, the users are non greedy and the algorithms which are able to achieve

the largest stability region are preferred since more revenue can be generated from the service provider's point of view. However, the gain of stability region usually comes at the cost of delay performance degradation. In this thesis, a new scheduling algorithm is proposed with the aim of resources minimization and its stability region is investigated by means of simulation.

A well designed scheduling algorithm should address the following three critical factors: fairness, delay and throughput. Interestingly, fairness is essential in the context of saturated throughput and its purpose is to ensure no one will be starved for a long period of time. The reason is that every user demands more than the processing capacity of the system, if no fairness constraint is implemented, some users may be starved indefinitely. There are several different types of fairness such as processor sharing time, minimum performance guarantee [15] and proportional fairness [9],[19]. However, in case of unsaturated throughput, pre-defined fairness constraints such as minimum amount of processor sharing time is not necessarily appropriate. For example, if we allocate a pre-defined processor sharing time to a particular user, there will be potentially a waste of processor time. The argument is simple: data arrives dynamically and user may not require all the reserved processor time to empty its buffer. When this happens, the scarce network resources is wasted and the scheduling algorithm becomes inefficient. Therefore, we studied and compared the distribution of starvation period of our algorithm with existing scheduling algorithms via simulations to ensure that the proposed algorithm does not result in severe starvation and provides a comparable performance in terms of starvation period to other algorithms.

Secondly, delay is not a proper factor to be considered in saturated throughput case since all user has infinite backlog which implies that the delay of each user is also infinite. However, it is a very important performance measure in unsaturated case. It is clear that stability region and delay performance are two separate optimization objectives. In general, an algorithm cannot optimize both objectives at the same time without making necessary tradeoffs. In this thesis, the primary design goal is to minimize the amount of resource used to stabilize the current traffics, and the secondary goal is to investigate its delay performance and stability region. We will show through simulation results that our algorithms indeed can achieve the same stability region as the throughput optimal algorithm and provides a significant delay improvement by utilizing queue length information.

The last criterion is throughput. If one algorithm is able to maximize the average throughput in long term for any stable input configuration, it must be the algorithm which achieves the largest stability region. Hence, any algorithms which is capable of achieving the largest stability region can be considered as throughput optimal. Note that an algorithm which maximizes throughput on a short-term scale does not necessarily maximize the throughput in long term. We will use simulation to illustrate this simple fact.

The thesis is organized as follows: in Chapter 2 we will explore and study previous works on various opportunistic scheduling policies, which are designed for TDMA systems. Those policies are further divided into different categories based on the types of fairness constraints and the length of period over which the constraint is applied. In Chapter 3, two new opportunistic scheduling algorithms focusing on minimizing the resources used to stabilize the current traffics under the assumption that the traffic is known are proposed. Their performances are analyzed and compared with the well-known throughput optimal algorithm [3] and the Round Robin policy via simulations. Finally, we conclude in Chapter 4 with the main contributions of this work and suggestions on future research works.

Chapter 2

Previous Works

In this chapter, a brief review of the current literature on opportunistic scheduling will be presented. Typical works with and without infinite backlog assumption are discussed. The following algorithms all consider a time-slotted system model where time is the resource to be shared among all users, which includes TDMA systems and time-slotted CDMA systems. Both the uplink and the downlink scheduling of a wireless network are considered, and the base station serves as the central control station.

2.1 Long-term Constraints in Saturated Cases

Scheduling algorithms with long-term constraints try to guarantee certain QoS or fairness over the entire time horizon. Therefore, user-oriented constraint or requirement can only be satisfied on an average scale. There is no guarantee that the requirement will be met in a finite time interval; hence, a real-time user who experiences a bad channel condition for a long time will also experience a prolonged delay period. The scheduling schemes described in [15] assume that all users are greedy with saturated traffics by which the authors actually mean that each user always has data to send and the queues are infinite in lengths. It is important to note that a fairness criteria is essential to scheduling problems for greedy users in wireless system. Without a good fairness criterion, the system performance can be maximized by simply letting the user with the best channel condition to transmit all the time. This will cause the "poor" users to be starved infinitely. In the following

subsection, we study three types of fairness constraints for both the uplink and the downlink of a wireless network in detail: Resource Sharing Fairness Constraint, Performance Based Fairness Constraint, and Minimum Performance Fairness Constraint. The notations used in this chapter are listed below:

- N: denotes the set of users indexed by i.
- $\mu_i(t)$: denotes the channel rate for user i in timeslot t. Thus $\vec{\mu}(t) = [\mu_1(t), ..., \mu_N(t)]$ denotes the vector of the channel rates for all users at time t.
- $f_i(\cdot)$: denotes the system utility function, which is typically a function of the transmission rate (i.e. $f_i(\mu_i)$). We assume that f_i is an additive convex function.
- $Q(\vec{\mu}(t))$: denotes a scheduling policy which selects a user in timeslot t, given $\vec{\mu}(t)$.

2.1.1 Resource Sharing Fairness Constraint – Temporal Fairness

In essence, in a typical TDMA system, time is the resource shared by all users. Therefore, a natural fairness criterion is to assign each user a pre-defined share of the total processor time. Let r_i denote the minimum fraction of processor time that should be assigned to user i, where $r_i \geq 0$ and $\sum_{i=1}^{N} r_i \leq 1$. The value of r_i is the minimum fraction of time that a user should be allocated on the channel, which is usually determined by the user's class or the price a user is willing to pay. The scheduler decides which time slot should be assigned to which user, given the minimum requirements. This type of fairness ensures that on average each user gets a certain share of the time resources. The associated problem formulation is shown below:

$$\max_{Q} \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] \tag{2.1}$$

$$\text{Subject to} \qquad P\{Q(\vec{\mu}) = i\} \geq r_i \tag{2.2}$$

The feasibility is guaranteed by imposing the condition $\sum_{i\in N} r_i \leq 1$. The solution can be obtained by formulating the Lagrangian:

$$\begin{aligned} L(Q,\vec{\lambda}) &= \sum_{i\in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] + \sum_{i\in N} \lambda_i \{E[I_{\{Q(\vec{\mu})=i\}}] - r_i\} \\ &= \sum_{i\in N}\sum_{\vec{\mu}\in S} \pi_s f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}} + \sum_{i\in N} \lambda_i (\sum_{\vec{\mu}\in S} \pi_s I_{\{Q(\vec{\mu})=i\}} - r_i) \end{aligned}$$

where λ_i is the Lagrangian multiplier and S denotes the set of channel states $\vec{\mu}$. In [15], the authors have assumed that the channel is a discrete memoryless channel with finite number of states. Let $\pi_{\vec{s}}$ be the stationary distribution of the channel at state $\vec{s}$. For a given, $\vec{\lambda} = [\lambda_i]$, Q will maximize $L(Q,\vec{\lambda})$ if and only if

$$Q = \arg\max_i \{\sum_{\vec{\mu}\in S} \pi_{\vec{\mu}} \sum_{i\in N} (f_i(\mu_i) + \lambda_i) I_{\{Q(\vec{\mu})=i\}}\} \tag{2.3}$$

At the beginning of each time slot t, given the channel state, the solution is

$$Q^*(\vec{\mu}(t)) = \arg\max_i \{f_i(\mu_i(t)) + \lambda_i(t)\} \tag{2.4}$$

Using stochastic gradient algorithm, the Lagrangian multiplier is updated as:

$$\lambda_i(t+1) = [\lambda_i(t) + \varepsilon_t (r_i - I_{\{Q(\vec{\mu}(t))=i\}})]^+ \tag{2.5}$$

where $\varepsilon_t = 1/t$ is a positive decreasing sequence, and $[\cdot]^+$ denotes the positive projection.

The Lagrangian multiplier λ_i can be considered as an offset used to achieve the fairness requirement. Suppose we want to maximize the performance of the overall system without any constraint. It is clear that the scheduler should choose the user with the highest channel rate at each time slot (i.e. $Q^*(\vec{\mu}(t))) = \arg\max_i(\mu_i(t))$. However, such a scheme may be unfair to some users whose channels are poor for a long time. Hence, to satisfy the fairness requirement, the scheduler should pick the "relative-best" user to transmit. User i is "relative-best" if $f_i(\mu_i) + \lambda_i > f_j(\mu_j) + \lambda_j$. If $\lambda_i > 0$, then user i is an unfortunate user because its channel is poor and an offset must be used to ensure it can be served fairly. The dynamic of λ_i can be also seen from the dual update equation. If the required resource is satisfied, λ_i will decrease towards zero. Otherwise, it will increase towards $+\infty$. It is important to note that to maximize the performance of the overall system,

these unfortunate users can only receive the amount of time resource equivalent to their minimum requirements. On the other hand, users with better channel condition get more than their minimum requirements in cases where $\sum_{i=1}^{N} r_i < 1$.

The average system performance is maximized by Q^* even if the users' channel conditions are arbitrarily correlated, both in time and across users. If individual channel conditions are independent of each other, then the following proposition holds:

Proposition 2.1: If the performance values $f_i(\mu_i)$, $i \in N$, are independent, then for all i,

$$E[f_i(\mu_i)I_{\{Q^*(\vec{\mu})=i\}}] \geq P(Q^*(\vec{\mu}) = i)E[f(\mu_i)] \geq r_i E[f(\mu_i)] \tag{2.6}$$

Note that $E[f_i(\mu_i)I_{\{Q^*(\vec{\mu})=i\}}]$ is the average performance value of user i by using policy Q^*. $P(Q^*(\vec{\mu}) = i)E[f(\mu_i)]$ is the average performance of user i by using a non-opportunistic scheduling algorithm such as Round Robin, where $P(Q^*(\vec{\mu}) = i)$ is greater or equal to the fraction of time assigned to user i.

Proposition 2.1 guarantees that the average performance of each user by using Q^* will be no worse than that of any non-opportunistic scheduling algorithm that allocates the same fraction of service time to the user if users' channel conditions are independent of each other. This result can be interpreted as follows. When a user is experiencing good channel conditions, it has a higher chance to get transmission opportunity. When a user is experiencing bad channel conditions, it has less probability to get transmission opportunity. Hence, a user tends to transmit more often under good channel conditions. The net effect is that every user sees an effective channel distribution which has a higher mean transmission rate than the actual channel model. Finally, different users may experience different level of improvement depending on the physical channel model. Normally, the larger the variance of a user's channel condition, the greater the improvement. If a user has a constant channel condition, then the equality sign in (2.6) holds.

2.1.2 Performance Based Fairness Constraint – Utility-based Fairness

In wireless networks, since the channel conditions are always varying, the amount of time slots allocated to a certain user does not have a linear relationship with its performance. In

other words, allocating certain time slots to a user does not grant the user a certain amount of throughput value. Hence, from a user perspective, another performance guarantee is more preferred. In [15], the authors state that Utility-based fairness ensures that each user gets a certain share of the total system throughput. Note that under such fairness constraint, each user's performance will have an impact on the total system throughput. Therefore, although each user is guaranteed some throughput in terms of percentage of the total system throughput, if one user misbehaves, the rest of the users will have to be penalized in order to meet each individual's fairness constraint. The corresponding optimization problem is shown below:

$$\max_{Q} \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] \tag{2.7}$$

$$\text{Subject to} \qquad E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] \geq a_i \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] \tag{2.8}$$

where $a_i's$ are predetermined fairness parameters and the feasibility is guaranteed by imposing the condition $\sum_{i \in N} a_i \leq 1$. Let $\nu = \sum_i a_i$ be a tuning parameter. The smaller the ν is, the larger the opportunity to improve the system performance. The solution can be obtained through solving the Lagrangian equation:

$$L(Q, \vec{\lambda}) = \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] + \sum_{i \in N} \lambda_i \{E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] - a_i \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}]\}$$

where λ_i is the Lagrangian multiplier for user i. For a given $\vec{\lambda} = [\lambda_i]$, the policy Q will maximize $L(Q, \vec{\lambda})$ if and only if

$$Q^* = \arg\max \{\sum_{\vec{\mu} \in S} \pi_{\vec{\mu}} \sum_{i \in N} f_i(\mu_i)(1 + \lambda_i - \sum_{i \in N} \lambda_i a_i) I_{\{Q(\vec{\mu})=i\}}\}$$

At the beginning of each time slot, the channel condition is known and the optimal policy is given by

$$Q^*(\vec{\mu}(t)) = \arg\max_i \{f_i(\mu_i(t))(1 + \lambda_i(t) - \sum_{i \in N} \lambda_i(t) a_i)\}$$

λ_i is updated by using a stochastic approximation algorithm as follows:

$$\lambda_i(t+1) = [\lambda_i(t) + \varepsilon_t (a_i \sum_{i \in N} f_i(\mu_i(t)) I_{\{Q(\vec{\mu}(t))=i\}} - f_i(\mu_i(t)) I_{\{Q(\vec{\mu}(t))=i\}})]^+$$

where ε_t is positive decreasing sequence to ensure convergence.

Utilitarian scheduling schemes have certain notable features. First, any policy Q that satisfies the fairness constraint defined in (2.8) has the property that

$$\frac{a_i}{1-\nu+a_j} \leq \frac{E[f_i(\mu_i)I_{\{Q(\vec{\mu})=i\}}]}{E[f_j(\mu_j)I_{\{Q(\vec{\mu})=j\}}]} \leq \frac{1-\nu+a_i}{a_j},\ \forall\ i,j \in N$$

In other words, a utilitarian fairness constraint controls the maximum discrepancy of performance between users.

Secondly, utilitarian fairness ensures that a user is given at least a fraction of the total performance. Sometimes, this is a more suitable fairness constraint than temporal fairness. However, there is also a significant disadvantage of this type of fairness. A user experiencing poor channel conditions could have a large impact on the overall system performance. Rewriting (2.8), we have $E[f_i(\mu_i)I_{\{Q(\vec{\mu})=i\}}]/a_i \geq \sum_{i\in N} E[f_i(\mu_i)I_{\{Q(\vec{\mu})=i\}}]$. Therefore, if one user performs extremely bad and has a large value of a_i, then the overall system performance will be degraded significantly. To overcome this problem, the value of a_i will be decreased when:

$$\frac{E[f_i(\mu_i)I_{\{Q(\vec{\mu})=i\}}]}{P(Q(\vec{\mu})=i)(\sum_{i\in N} E[f_i(\mu_i)I_{\{Q(\vec{\mu})=i\}}])} \leq \beta$$

where β is a pre-determined threshold.

2.1.3 Minimum Performance Fairness Constraint – Absolute Minimum Capacity

The scheduling schemes discussed above provide fairness guarantees in terms of relative performance measures. On the other hand, the following policy presented in [15] guarantees each user an absolute performance value. Hence, the QoS is defined in terms of minimum performance guarantee. Unlike the previous two schemes, this one offers a more direct way of guaranteeing service quality. But since the total capacity is a finite value, it might not be always possible to satisfy all users' minimum data-rate requirements. Hence, feasibility is an issue in this type of scheduling policy. However, to illustrate, we assume the minimum capacities given in the problem can be realized by at least one policy. The corresponding

optimization problem is shown below:

$$\max_{Q} \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] \tag{2.9}$$

$$\text{Subject to} \qquad E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] \geq R_i \tag{2.10}$$

where R_i is the minimum data-rate requested by user i. The associated Lagrangian equation is:

$$L(Q, \vec{\lambda}) = \sum_{i \in N} E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] + \sum_{i \in N} \lambda_i \{E[f_i(\mu_i) I_{\{Q(\vec{\mu})=i\}}] - R_i\}$$

where $\lambda = [\lambda_i]$ is the Lagrangian multiplier. For a given $\vec{\lambda}$, the policy Q will maximize $L(Q, \vec{\lambda})$ if and only if

$$Q^* = \arg\max \{ \sum_{\vec{\mu} \in S} \pi_{\vec{\mu}} \sum_{i \in N} (f_i(\mu_i) + \lambda_i f_i(\mu_i)) I_{\{Q(\vec{\mu})=i\}} \}$$

At the beginning of each time slot, the channel condition is known and the optimal decision is:

$$Q^*(\vec{\mu}(t)) = \arg\max_{i} \{ f_i(\mu_i(t))(1 + \lambda_i(t)) \} \tag{2.11}$$

λ_i is updated by using a stochastic approximation algorithm as follows:

$$\lambda_i(t+1) = [\lambda_i(t) + \varepsilon_t (R_i - f_i(\mu_i(t)) I_{\{Q(\vec{\mu}(t))=i\}})]^+ \tag{2.12}$$

where ε_t is a positive decreasing sequence. If we further simplify the solution to make $Q^*(\vec{\mu}(t)) = \arg\max_i \{v_i(t) f_i(\mu_i(t))\}$ where $v_i(t) = 1 + \lambda_i(t)$, it turns out that a large group of opportunistic scheduling problems has the solution in this form. For example, the solution in Utility-based Fairness scheme can be converted to this form. Also, a very well known scheme - throughput optimal scheduling scheme [3] is also in this form and will be discussed in the next section.

As mentioned before, feasibility is an issue in this case. However, there are some natural settings where feasibility is not an issue. For instance, let $R_i = \rho_i E[f_i(\mu_i)]$, where $0 \leq \rho_i \leq 1$ and $\sum_{i \in N} \rho_i \leq 1$. This setting can be achieved by a non-opportunistic scheduling policy in which user i is selected for transmission with probability ρ_i. Therefore, it is feasible for opportunistic scheduling policies naturally.

2.2 Long-term Constraints in Unsaturated Cases

In this section, we remove the assumption that all users are greedy and assume that each user's queue is finite. By relaxing this assumption, the crucial performance factor – delay must be taken into consideration. In general, QoS can be defined in many different ways. One of the standard ways is that the delays of most of the data packets need to be kept below a certain threshold. For example, in [3], the authors have used the following requirement:

$$P(W_i > T_i) \leq \delta_i \tag{2.13}$$

where W_i is a packet delay for this user, and parameter T_i and δ_i are the delay threshold and the maximum probability of exceeding it respectively. Note that this requirement is established in long-term sense. There is no guarantee that every packet will have a delay less than T_i.

It has been shown that the above requirement can be achieved by throughput optimal scheduling schemes, in particular the Modified Largest Weighted Delay First ($M-LWDF$) scheme. The definition of *throughput optimal* scheduling algorithm is that a scheduling algorithm is called throughput optimal if it is able to keep all queues stable if this is feasible at all. $M - LWDF$ is throughput optimal and it is given by

$$Q^*(t) = \arg\max_i \{\alpha_i W_i(t)\mu_i(t)\}$$

where $W_i(t)$ is the head-of-line packet delay for queue i, $\mu_i(t)$ is the channel condition, and α_i is an arbitrary positive constant. The above algorithm has another equivalent form shown below:

$$Q^*(t) = \arg\max_i \{\alpha_i q_i(t)\mu_i(t)\} \tag{2.14}$$

where $W_i(t)$ is replace by $q_i(t)$ which represents the queue length. In the simulation section, we will compare the performance of the proposed algorithm with that of (2.14). From this point onwards, we will refer to the throughput optimal policy as the one given by (2.14).

There are three key features associated with the above policy:

- The policy uses both current channel conditions and queue lengths to make a scheduling decision.

- The policy is able to achieve the largest stability region without knowing the incoming traffic rates.
- The delay distribution of individual users can be controlled by tuning the parameters α_i.

The ability to control individual delay distribution is a powerful feature. Requirement (2.13) can be satisfied by selecting α_i as follows:

$$\alpha_i = \frac{-\log(\delta_i)}{T_i E[\mu_i]} \tag{2.15}$$

where $E[\mu_i]$ is the mean of user i's channel rate. Parameter α_i considers the QoS requirement, and provides QoS differentiation between the flows. For instance, if users A and B have the same delay threshold T, but $\delta_A = 4\delta_B$, which results in $\alpha_B = 2\alpha_A$. The net effect is that user B will be given priority in assigning transmission opportunity. Substitute (2.15) into (2.14), we have

$$Q^*(t) = \arg\max_i \{\frac{-\log(\delta_i) q_i(t)\mu_i(t)}{T_i E[\mu_i]}\}$$

It is easy to see that the chance of a user being scheduled is increased when the user has a higher QoS requirement and the values of its queue length and channel rate are increased. In all, this rule approximately "balances" different users' probabilities of dead-line violation relatively to their maximum allowed values δ_i. Therefore, the rule supports every user's desired QoS specified in (2.13) if this can be done at all with any other rule. This implies that there is a fundamental limit in specifying delay requirement. In other words, the delay bound cannot be made arbitrarily small. In fact, increasing the parameter α_i of user i, while keeping α_is of other users unchanged, reduces delays for this flow at the expense of a delay increase for other flows. Overall, there is no gain without loss. However, in this thesis, an algorithm which provides better delay performance, especially under heavy traffic conditions, than (2.14) is proposed and presented later.

We note that all the schemes discussed in section 2.1 deal with long-term performance measures. In reality, short-term performance guarantee is more important from a individual user's point of view. When a user's service is lagging behind its specified share or it

is experiencing significant delay, we want to increase its transmission opportunities to guarantee its service quality. Hence, we will consider opportunistic scheduling with short-term constraints in the following section.

2.3 Short-term Constraints in Saturated Cases

Short-term constraints guarantee fairness/QoS over finite time duration. This is a more appealing requirement in practice as most applications are delay sensitive to some extent. Since the time horizon is infinite in the long-term fairness case, the QoS can be met precisely on average. However, due to the lack of scheduling flexibility, short-term fairness is often difficult to be met precisely within a finite time duration. In [11], the authors present a scheme with short-term temporal fairness with the assumption that users are greedy, which means each of the users always has data to receive on the downlink). Let's assume that each user i has an associated weight ϕ_i such that $\sum_{i \in N} \phi_i \leq 1$. Let's group time slots into successive non-overlapping windows of M slots each. The problem formulation is shown below:

$$\max \sum_{t=0, i \in N}^{M-1} E[\mu_i(t) I_{Q(t)=i}] \tag{2.16}$$

$$\text{Subject to} \quad \sum_{t=0}^{M-1} I_{Q(t)=i} \geq \phi_i M \tag{2.17}$$

In words, the above optimization problem can be defined as follows: among all scheduling policies which select each user $\phi_i M$ times in M consecutive time slots, find the one which maximizes the system throughput. Clearly, this policy ensures that no user will be starved more than $2M-1$ slots, and every user will get its $M\phi_i$ number of slots in every successive non-overlapping window.

Since the problem involves dynamic channel conditions and usually it is quite difficult to estimate many parameters in the channel state model, the solution to the above problem cannot be obtained in the same way as the long-term problems and the Lagrangian multiplier will not be able to converge within the specified finite time duration. However, a few special cases have been worked out and a heuristic policy is proposed. We will give

the final results directly without showing the technical proof, which can be found in [12].

Case 1: $M = \infty$

This case is identical to the long-term temporal fairness in section 2.1.1. On average, user i must get a time share of the service time that is greater than or equal to ϕ_i. The corresponding problem formulation is

$$\begin{aligned} &\max \sum_{i \in N} E[\mu_i I_{\{Q(\vec{\mu})=i\}}] \\ \text{Subject to} \quad &E[I_{\{Q(\vec{\mu})=i\}}] \geq \phi_i \end{aligned}$$

The solution is given by

$$Q^*(\vec{\mu}(t)) = \arg\max_i \{\mu_i(t) + \lambda_i(t)\} \tag{2.18}$$

where λ_i is a non-negative Lagrangian multiplier.

Case 2: N *i.i.d* users and $\phi_i = 1/N$, $M = N$

It is clear that the window size in this case is the shortest possible. Every user gets exactly one slot in every M slots. The channel models for all users are identically and independently distributed across users and time. In the following case, this *i.i.d* assumption will be removed and a different solution will be obtained. The problem formulation for case 2 is:

$$\begin{aligned} &\max \sum_{t=0}^{M-1} \sum_{i \in N} E[\mu_i I_{\{Q(\vec{\mu})=i\}}] \\ \text{Subject to} \quad &\sum_{t=0}^{M-1} I_{\{Q(\vec{\mu})=i\}} = 1 \end{aligned}$$

and the optimal scheduling policy is

$$Q^*(t) = \arg \max_{i \in A^*(t)} \{\mu_i(t)\} \tag{2.19}$$

where $A^*(t)$ denotes the set of users who have not been served yet. The dynamic of $A^*(t)$ is $A^*(t) = A^*(t-1) - Q^*(t-1)$. This policy is given the name Opportunistic Round Robin

policy. Once the user is selected, this user will be considered as inactive until the start of the next non-overlapping window.

Case 3: N independent users and $\phi_i = 1/N$, $M = N$

The problem formulation is identical to Case 2, but users have different channel distributions. The channel rates for each user are still independent across time. The optimal scheduler is

$$\begin{aligned} Q^*(t) &= \arg \max_{i \in A^*(t)} \{\mu_i(t) + V^*_{A^*(t)-\{i\}}\} \\ A^*(0) = N, &\quad A^*(t+1) = A^*(t) - Q^*(t) \end{aligned} \tag{2.20}$$

where V^*_B is the total expected optimal reward which could be received by users in set B. In words, the above policy can be described as follows. Let $A^*(t)$ be the set of unserved users at the beginning of slot t. Assume that the scheduler picks user $i \in A^*(t)$ for service. The throughput in the current slot will be $\mu_i(t)$. The expected total throughput in the remaining window will be $V^*_{A^*(t)-\{i\}}$. Hence, the optimal policy in this time slot selects the user with the maximum $\mu_i(t) + V^*_{A^*(t)-\{i\}}$ to maximize the total expected throughput.

In fact, a careful thought should reveal that (2.20) is equivalent to the following policy:

$$\begin{aligned} Q^*(t) &= \arg \max_{i \in A^*(t)} \{\mu_i(t) - E[\mu_i]\} \\ A^*(0) = N, &\quad A^*(t+1) = A^*(t) - Q^*(t) \end{aligned}$$

The rational of this policy is very straightforward: schedule the user whose channel is the best comparing to its mean channel rate. Since the channel are not $i.i.d$, $E[\mu_i]$ will not be the same for all users. However, in Case 2, $E[\mu_i]$ is the same for all users and then the policy reduces to

$$Q^*(t) = \arg \max_{i \in A^*(t)} \{\mu_i(t)\}$$

Motivated by the solution in Case 1, a heuristic policy has been proposed to solve the original general problem (2.16):

$$\begin{gathered} Q^*(t) = \arg \max_{i \in A(t)} \{\mu_i(t) + \lambda_i\} \\ A(0) = N, \ N_i(0) = 0 \\ N_i(t) = N_i(t-1) + I_{\{Q(t-1)=i\}}, \ A(t) = A(t-1) - Q(t-1) I_{\{N_i(t) = M\phi_i\}} \end{gathered}$$

where λ_i denotes the value of the Lagrangian multiplier obtained in the long-term case (after convergence in steady state). $A(t)$ denotes the set of users whose constraints have not been met. Within a scheduling window, the scheme always picks the user having the largest $\mu_i(t) + \lambda_i$ from the active users set $A(t)$ at the beginning of each time slot. A counter, $N_i(t)$, is used to keep track of the number of time slots allocated to each user. If the counter is equal to the minimum fairness requirement of a user, then this user is removed from $A(t)$. We can easily see that the performance of this policy will converge to the long-term temporal fairness policy when M is large. If the sum of all requested resource is less than the total resource (i.e. $\sum_i \phi_i < 1$), this heuristic algorithm is not the optimal solution. In this case, the true solution requires accurate channel estimation, which is usually a complicated process. Therefore, the author does not mention this case in his original work.

We note that if the short-term constraints can be relaxed, then it can be solved by standard Lagrangian method as well. Hence, we can also draw a conclusion that the stricter the short-term constraint is, the less flexibility the system will have to improve the total system performance. There exists a tradeoff between total system performance values and each user's quality of service.

Furthermore, although this scheme which is capable of providing short-term fairness to guarantee each user's performance looks quite appealing, it still does not take queue length into consideration when scheduling is performed. The assumption that all users are greedy is not practical; hence, we want to propose new opportunistic scheduling policies such that both queue and channel information can to utilized to provide short-term performance guarantees without assuming all users are greedy. In this thesis, short-term performance refers to delay and maximum starvation period. Although these two performance factors are all user-level optimization objectives which are not the primary concern of the service provider, they should be analyzed to ensure that an "acceptable" level is achieved. In Chapter 3, we propose two new algorithms which minimize the resources used to stabilize the incoming traffics. Simulation results show that they can achieve the same stability region as the throughput optimal algorithm and provide better delay performance.

2.4 Optimal Fair Scheduling

Although opportunistic scheduling have attracted lots of researchers' attention recently, other resource management schemes which can provide fair and efficient resource allocations to a large number of users are also explored and investigated in many network studies. In this section, we briefly discuss an algorithm which provides fair allocation of capacity to large number of users of best-effort connections [1]. In this type of framework, the greedy-user assumption is removed and it does not provide explicit QoS in terms of delay and throughput. Its main objective is to allocate resources in a fair manner.

Unlike opportunistic scheduling policies, this type of algorithm do not explore channel variations. In other words, such scheduler is blind to any channel information. Normally, there always exists a tradeoff between "fairness" and "efficiency", and to allocate resources fairly among users is gaining wide interest both in wire-line and wireless networks. The notion of proportional fairness is used in [1]. This type of fairness is from the fairness axioms in game theory which was first introduced in a network context by Mazumdar *et al* [19]. Kelly termed it "proportional fairness" in later publications, which have been cited frequently.

Kelly has shown that if the utility function is in the form of $\log(\cdot)$, then the solution of the optimization problem satisfies proportional fairness. Indeed, Mazumdar proves that the solution obtained using this type of utility function is an instance of the Nash Bargaining Solution (NBS). The formal definition of NBS does not require log utility function at all. The features of the NBS solution are:

- The NBS is Pareto optimal.
- The solution is unchanged if the performance objectives are scaled in the form of $au + b$. (scale invariant property)
- The solution is not affected by enlarging the domain if agreement can be found on a restricted domain. (irrelevant-alternatives axiom)
- The solution does not depend on the specific labels. (symmetry property)

Pareto optimality refers to the fact that all resources are distributed in the sense that

the bandwidth allocated to any user cannot be further increased without sacrificing the bandwidth allocated to other users.

In [7], Girard *et al* present a network system where multiple users share the available resource on a demand basis, and the sum of all demands usually exceeds the available total bandwidth. The bandwidth requested by each user is not absolutely needed and the users can accept an allocation smaller than what they originally demanded. The optimization problem is formulated as

$$\begin{aligned} & max_{\vec{x}} \prod_{i=1}^{N} x_i \\ \text{Subject to } & \sum_{i=1}^{N} x_i = C \\ & 0 \leq x_i \leq d_i \end{aligned} \tag{2.21}$$

where N is the total number of connections, C is the total available bandwidth, $\vec{x} = [x_i]$ is the allocation vector and d_i represents the amount of bandwidth requested by i. And the objective in form of (2.21) is equivalent to

$$max_{\vec{x}} \sum_{i=1}^{N} \log(x_i)$$

As mentioned before, this type of utility function will lead us to an NBS solution. Since proportional fairness has powerful features, it is the most commonly used fairness objective in utility maximizing problems. Furthermore, $\log(\cdot)$ is a strictly concave function. This property will yield a unique global optimal solution.

In general, different utility functions will lead us to different operating point on the Pareto optimal surface. A more general form of the utility function is given in [13] as follows:

$$U_i(x_i) = w_i \frac{x_i^{1-\gamma}}{1-\gamma}, \gamma > 0$$

where x_i is the bandwidth allocated to user i, w_i is the weight, the maximization problem corresponds to a weighted *proportional fairness* as $\gamma \to 1$ and weighted *max-min* fairness as $\gamma \to \infty$.

Note that *max-min* fairness is also Pareto efficient and it allocates network resources in such a way that the bandwidth allocated to a user cannot be increased without decreasing the bandwidth allocated to another user. In other words, with *max-min* fairness, resources are allocated to users in the order of increasing demands, and no one will receive more than it requires. Users with unsatisfied demands will split the remaining resources evenly among themselves.

2.5 Summary

To sum up, we have studied different scheduling algorithms with different focuses in this chapter. In reality, there is no single algorithm can maximize multiple objectives at the same time without any trade-offs. Each scheduling policy has its own assumptions and limitations. Hence, different algorithms are tailored to different application areas. In the next chapter, we will present two general scheduling algorithms which aim to minimize the resources used to in stabilizing the incoming traffic. Meanwhile, their stability regions and delay performance are analyzed by simulations. These proposed algorithms relax the assumption of infinite backlog.

Chapter 3

An Innovative Opportunistic Scheduling Algorithm for Networks with Packet-Level Dynamic

3.1 Motivation and Basic Framework

In practice, all service providers want to serve as many users as possible for revenue maximization provided that the service is charged on subscription basis. However, the amount of incoming traffic that can be served is limited by the wireless channel and scheduling scheme. In this chapter, we will present two opportunistic scheduling algorithms which both aim to minimize the amount of resources used in order to stabilize the incoming traffic. Although both algorithms are developed with the same optimization goal, they have different delay performance. Simulations indicate that both algorithms can achieve a stability region almost identical as the throughput optimal algorithm [3]. The two algorithms work under any traffic model or traffic regulation method and assumes that the average incoming traffic rates are known. The traffic intensity information is assumed to be available to the system before scheduling decision is made.

The system model under investigation in this thesis is the following. The network consists of a single base station which connects to all end users as shown in Fig 3.1. In reality, the controller at the base station is the unit responsible for efficient allocation of

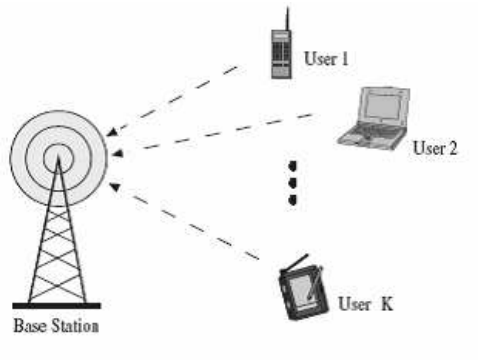

Figure 3.1: Network Topology

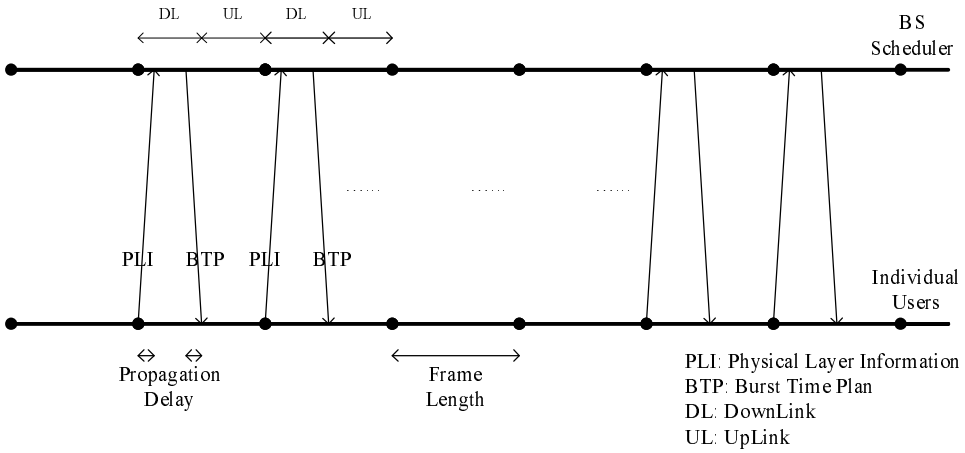

Figure 3.2: Timing of Events

the available resource to individual user as shown in Fig 3.2. Physical layer information (PLI) is reported to the base station at the beginning of each frame. The controller will then prepare a Burst Time Plan (BTP) showing which time slot is allocated to which user. For modeling purpose, we consider slot-by-slot scheduling where the system time is slotted with time-slot duration T. At the beginning of each time slot, the scheduler collects the information of individual channel rate and queue length as inputs to assign transmission opportunities. Only one user can be served in each time slot. We assume that the regulated traffic rates are available to the scheduler at the start of a session. The number of sessions is fixed in the network and corresponds to the number of users, and the only dynamic in

the system is packet-level dynamic.

Since a network under this model may not be stable, the notion of stability region arises. The well-known definition of network stability is

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t I_{\{\sum_{s=1}^{S} n_s(t) + \sum_{l=1}^{L} q_l(t) > M\}} dt \to 0, \ as \ M \to \infty \tag{3.1}$$

where $n_s(t)$ denotes the number of sessions at source node s, and $q_l(t)$ denotes the queue length at link l. In words, we say a network is stable if the number of sessions is finite and the queues at each link are finite as well. The stability region is defined as the set of arrival traffic intensities under which (3.1) is satisfied.

In this thesis, we consider uplink scheduling and the above general stability equation is interpreted as follows: there are S source nodes/users and S uplinks in total. q_l denotes the queue length at uplink l. We assume that each source node only initiates one session ($n_s = 1$). Therefore, the first term in (3.1) converges by definition. Since the network structure is a one-hop topology, the system is stable if the queue length at each source node is finite, which guarantees the convergence of the second term in (3.1).

Two of the criteria to design a good scheduling algorithm for networks with packet-level dynamic are: stability region and delay. In general, different scheduling algorithms will result in different stability regions. However, the fundamental limit of the stability region is governed by the nature of the channel. From the service provider's point of view, stability region is a system level performance. A larger stability region will accommodate more users/traffics and in turn generate more revenues. On the other hand, delay is a user level performance. Every user wants to transfer data with minimum delay. It is obvious that maximizing the stability region and minimizing delay cannot be achieved at the same time. In this thesis, we will propose two scheduling algorithms which try to minimize the resources being used to stabilize the current traffic. Although these two optimization problems are different from the optimization problem that would maximize the stability region, simulation results suggest that both algorithms can potentially achieve a stability region very close to the one of the throughput optimal algorithm. Moreover, they can also provide a better delay performance than the throughput optimal algorithm.

A natural question to ask is why fairness is not considered as one of the design criteria? In networks with greedy users, users may be starved indefinitely since they are competing

for limited resources or transmission opportunities. Recall that the definition of a greedy user is a user with infinite backlog to transfer. In our case, each user has a finite queue length and the system operates inside the stability region; hence, it is able to handle all the incoming traffics, which implies that no user will be starved indefinitely. Therefore, a strict fairness constraint is not required in case of non-greedy users. However, the starvation period should be analyzed to ensure no user is starved for a prolonged period. The starvation period of a user refers to the maximum number of time slots between two consecutive transmission opportunities for a user with non-empty queue.

In this thesis, we want to investigate and compare the throughput, delay performance and the starvation period of the proposed two algorithms with other well-known schemes.

3.2 Problem Formulation and Solution

In this section, we will develop two algorithms which try to minimize the resources being used to stabilize the incoming traffic.

To formulate the optimization problems, first let $s_i(Q)$ be the average processor time of user i under scheduling scheme Q given by

$$s_i(Q) = \lim_{t\to\infty} \frac{1}{t} \int_0^t I_{\{Q(\vec{\mu}(t))=i\}} dt \tag{3.2}$$

where $\vec{\mu}(t) = [\mu_1(t), ..., \mu_N(t)]$ is the channel rate vector at time t in unit of bits/second. This unit can be converted to packets/second if the number of bits in a packet is known. Although this thesis deals with packet-level dynamics, we will use bits/second as the unit. We assume that the channel distribution is stationary. Clearly, $0 \leq s_i(Q) \leq 1$. By ergodicity, (3.2) is equivalent to

$$s_i(Q) = E[I_{Q(\vec{\mu})=i}] \tag{3.3}$$

Now, we proceed to formulate the first optimization problem in the time domain which minimizes the resources used to stabilize the current traffic:

$$\min_{Q} \lim_{t\to\infty} \frac{1}{t} \int_0^t \sum_{i=1}^{N} I_{\{Q(\vec{\mu}(t))=i\}} dt$$

$$\text{subject to } \lim_{t\to\infty} \frac{1}{t} \int_0^t \mu_i(t) I_{\{Q(\vec{\mu}(t))=i\}} dt \geq \lambda_i, \forall i$$

where $\vec{\lambda} = [\lambda_1, ..., \lambda_N]$ is the average arrival rate vector with unit bits/second. By ergodicity, the above problem is equivalent to

$$\min_{Q} \sum_{i=1}^{N} E[I_{\{Q(\vec{\mu})=i\}}] \tag{3.4}$$

$$\text{subject to } E[\mu_i I_{\{Q_(\vec{\mu})=i\}}] \geq \lambda_i, \forall i \tag{3.5}$$

To change the problem to a maximization problem, we add a minus sign to the objective and obtain:

$$\max_{Q} -\sum_{i=1}^{N} E[I_{\{Q(\vec{\mu})=i\}}] \tag{3.6}$$

$$\text{subject to } E[\mu_i I_{\{Q_(\vec{\mu})=i\}}] \geq \lambda_i, \forall i \tag{3.7}$$

Standard Lagrangian method can be used to solve the above problem.

$$L(Q(\vec{\mu}), \vec{\gamma}) = -\sum_{i=1}^{N} \sum_{\vec{\mu} \in C} \pi_{\vec{\mu}} I_{\{Q(\vec{\mu})=i\}} + \sum_{i=1}^{N} \gamma_i (\sum_{\vec{\mu} \in C} \pi_{\vec{\mu}} \mu_i I_{\{Q(\vec{\mu})=i\}} - \lambda_i)$$

where $C = \otimes \mu_i, i \in N$ denotes the Cartesian product, and $\pi_{\vec{\mu}}$ is the stationary probability at state $\vec{\mu}$. For a given $\vec{\gamma}$, the corresponding optimal policy $Q(\vec{\mu})$ maximizing $L(Q(\vec{\mu}), \vec{\gamma})$ is given by

$$\begin{aligned} Q^*(\vec{\mu}) &= \arg\max L(Q(\vec{\mu}), \vec{\gamma}) \\ &= \arg\max \left(\sum_{\vec{\mu} \in C} \pi_{\vec{\mu}} \sum_{i=1}^{N} (-I_{\{Q(\vec{\mu})=i\}} + \gamma_i \mu_i I_{\{Q(\vec{\mu})=i\}}) \right) \\ &= \arg\max_{i} (\gamma_i \mu_i - 1) \end{aligned} \tag{3.8}$$

Thus, at the beginning of the tth time slot, the scheduling policy is

$$Q^*(\vec{\mu}(t), \vec{\gamma}(t)) = \arg\max_{i} (\gamma_i(t) \mu_i(t) - 1) \tag{3.9}$$

The dual variable is updated at the end of the tth slot using stochastic approximation which is given by

$$\gamma_i(t+1) = [\gamma_i(t) - \epsilon_t(\mu_i(t) I_{\{Q(\vec{\mu}(t), \vec{\gamma}(t))=i\}} - \lambda_i)]^+ \tag{3.10}$$

where ϵ_t is a positive sequence which goes to zero as $t \to \infty$.

The stability of the above solution is established in the long-term sense. In general, any policy which guarantees long-term performance cannot guarantee short-term performance. From a user's point of view, whenever the design only provides long-term stability, there is always a possibility that some users may be starved for an extended period.

To further improve short-term performance, we formulate our second optimization problem which utilizes the real-time queue information. The derivation and goal of the second problem are identical to the first problem. However, the difference is the addition of the real-time information. Thus, the second problem can be considered as a modified algorithm based on the first one. Let $q_i(t^-)$ be the queue length of user i at the beginning of tth slot. Let $q_i((t+1)^-)$ be its queue length at the beginning of the $(t+1)$th slot. Redefine constraint (3.5) as

$$E[\mu_i I_{\{Q_(\vec{\mu})=i\}}] \geq \lambda_i + \alpha_i I_{\{q_i((t+1)^-)\geq q_i(t^-)\}} - \beta_i I_{\{q_i((t+1)^-)<q_i(t^-)\}}, \forall i \tag{3.11}$$

where α_i represents a positive virtual drift of the traffic intensity, and β_i represents a negative virtual drift of the traffic intensity. Its value can be set to a fraction of the actual traffic intensity. Note that the actual traffic intensity remains the same. The temporary drift is added to combat the randomness of the channel and traffic arrivals for better delay response.

In equilibrium, the actual operating point oscillates around the true operating point. This operation can be illustrated in Fig 3.3. The trade-off of doing so is change in stability region. Specially, when the actual operating point is close to the boundary of the system capacity, a positive virtual drift will improve short-term performance at the cost of reducing stability region. A positive virtual drift will force the system to assign more service time to a user to reduce its queueing delay, whereas a negative drift will decrease the service time assigned to a user.

In addition, the real-time queue information can reduce time slot wasted by equation (3.9). At each scheduling instant, the system should only schedule users with nonzero queue. Therefore, a policy incorporating real-time information is proposed:

$$Q^*(\vec{\mu}(t), \vec{\gamma}(t)) = \arg\max_{i\in D}(\gamma_i(t)\mu_i(t) - 1) \tag{3.12}$$

$$\gamma_i(t+1) = [\gamma_i(t) - \epsilon(\mu_i(t)I_{\{Q(t)=i\}} - \lambda_i - \alpha_i I_{\{q_i((t+1)^-)\geq q_i(t^-)\}} + \beta_i I_{\{q_i((t+1)^-)<q_i(t^-)\}})]^+ \tag{3.13}$$

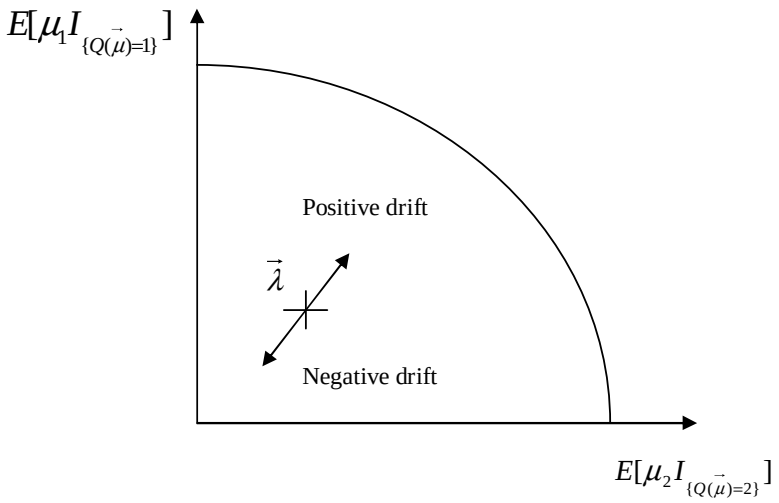

Figure 3.3: Stability Region and Operating Point with Virtual Drift

where D denotes the set of users with nonzero queues. It is important to note the fact that the introduction of real-time information does not affect the operating point in the long term. However, it improves the performance in short term at the cost of reduction of the stability region. In the next section, we will run simulation to illustrate that the trade-off is small. This results can be explained in this way: since the drift is temporary and the scheduler responds to the queueing dynamic rapidly, the net tradeoff is zero in the long term.

To further improve the performance, another parameter ϕ_i is introduced. ϕ_i represents the threshold of user i's queue length at which the drift mechanism will be activated. To be more precise, (3.13) is executed if user i's current queue length exceeds ϕ_i. Otherwise, its dual variable is not changed. The idea is that when two users both experience increase in queue length, but if one user's queue length is below a certain threshold, the other user should be given transmission priority.

Remark: In (3.13), the step-size is a constant instead of a decreasing sequence. This design is different comparing to the normal application of stochastic approximation algorithm. In general, to ensure convergence, a decreasing step-size must be used as in (3.10). The effect of using a constant step-size is that the output will oscillate around the equilibrium point forever. In our case, oscillation is a desirable property since we want to track

Table 3.1: Scenario 1 Channel Distribution

Channel rate (kbps)	38.4	76.8	102.6	153.6	204.8	307.2	614.4	921.6	1228.8	1843.2	2457.6
Distribution 1	0.05	0.05	0.05	0.05	0.1	0.1	0.2	0.2	0.1	0.05	0.05
Distribution 2	0.05	0.05	0.1	0.1	0.2	0.2	0.1	0.05	0.05	0.05	0.05

the dynamic of queues.

3.3 Simulation Analysis: First Scenario

This section presents the simulation results of the two proposed algorithms, and their performance are compared with three other benchmark algorithms. Let LS denote the scheduling algorithm corresponding to equations:

$$Q^*(\vec{\mu}(t), \vec{\gamma}(t)) = \arg\max_i(\gamma_i(t)\mu_i(t) - 1)$$
$$\gamma_i(t+1) = [\gamma_i(t) - \epsilon_t(\mu_i(t)I_{\{Q(\vec{\mu}(t),\vec{\gamma}(t))=i\}} - \lambda_i)]^+$$

and SS denote the scheduling algorithm corresponding to equations:

$$Q^*(\vec{\mu}(t), \vec{\gamma}(t)) = \arg\max_{i\in D}(\gamma_i(t)\mu_i(t) - 1)$$
$$\gamma_i(t+1) = [\gamma_i(t) - \epsilon(\mu_i(t)I_{\{Q(t)=i\}} - \lambda_i - \alpha_i I_{\{q_i((t+1)^-)\geq q_i(t^-)\}} + \beta_i I_{\{q_i((t+1)^-)<q_i(t^-)\}})]^+$$

We simulated the opportunistic scheduling policies for typical settings of a CDMA-HDR system in three different simulation scenarios. The first simulation scenario is the following. There are 10 users in the system, and there are two types of channel distributions shown in Table 3.1. The unit of the channel rate is kbps/second. User 1 to 5's channels are modeled as i.i.d channels with distribution 1. User 6 to 10's channels are modeled as i.i.d channels with distribution 2. The mean channel rates for distributions 1 and 2 are 714.89kbps and

517.78kbps respectively. Simulation has been run under this setting for many times. The results of all trials have been found to be consistent to ensure the validity of the simulation. Only one sample result for each test case will be presented below for illustration purpose.

Before we present the simulation results, we will introduce the three benchmark algorithms briefly.

Benchmark 1: `Local Throughput Maximization (`LTM`) Algorithm`

$$Q^*(\vec{q}(t), \vec{\mu}(t)) = \arg\max_i \{min(q_i(t), \mu_i(t))\} \tag{3.14}$$

The idea of this algorithm is to maximize the throughput in the current time slot. However, to achieve the largest stability region, long-term average throughput should be maximized. Intuitively, maximizing throughput in a given slot is not equivalent to maximizing average throughput in long term. We will illustrate that this algorithm has a smaller stability region than that of the proposed algorithm.

Benchmark 2: `Throughput Optimal (`TO`) Algorithm`

$$Q^*(\vec{q}(t), \vec{\mu}(t)) = \arg\max_i \{q_i(t)\mu_i(t)\} \tag{3.15}$$

This is the well known algorithm which achieves the largest stability region [3]. We will use simulation results to show that both the proposed algorithm in equations (3.9) and (3.10) and the modified algorithm in equations (3.12) and (3.13) can have better delay performance than the TO algorithm under different traffic intensities and the tradeoff in the stability region is very small and can almost be neglected.

Benchmark 3: `Round Robin (`RR`) Algorithm`
This algorithm picks user sequentially for transmission. If the scheduled user has no data to transmit, the algorithm searches for the next user in the sequence with pending data for transmission.

The strategy of the first simulation environment is the following. The five algorithms (LS, SS, LTM, TO, RR) will be simulated simultaneously in three different traffic inten-

Table 3.2: Scenario 1 Low Traffic Intensity

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	50	50	40	40	20	20	30	30	10	10

Table 3.3: Scenario 1 Parameters of Algorithm SS under Low Traffic Intensity

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	5	5	4	4	2	2	3	3	1	1
β (kbps)	2.5	2.5	2	2	1	1	1.5	1.5	0.5	0.5
ϕ (bits)	5	5	4	4	2	2	3	3	1	1

sity regions: low, high and saturated traffic region[1]. The reason to simulate under these three cases is that traffic rates vary constantly in reality and we want to investigate if the proposed algorithm perform consistently well in all scenarios. In the three cases, all algorithms see identical arrival process. The performance under investigations are average delay, average throughput and starvation period.

3.3.1 Performance Analysis with Low Traffic Intensity

In this section, we will show the performance comparison under low traffic intensity. The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.2 and 3.3.

The arrival traffic is modeled by Poisson process. α is chosen to be 10 percent of the arrival rates and β is chosen to be half of α. ϕ is 10 percent of the expected arrival in one slot. The slot length is 0.0005 second. The simulation runs over 10^6 slots with the first 10^5 slots removed to eliminate the transient effect. These setups will be used in all subsequent simulations, and will not be restated. Tables 3.4, 3.5 and 3.6 show the simulation results.

[1]The precise stability region is very difficult to compute theoretically. A rough boundary is found by running simulation with a small increment of traffic intensity each time until the system is not stable. Here, we pick a typical rate setting from each region and show the final results directly.

Table 3.4: Scenario 1 Average Delay Performance Comparison under Low Traffic Intensity

Delay (ms)	LS	SS	LTM	TO (baseline)	RR
User 1	2.1681 (+2.16%)	2.1645 (+1.93%)	1.9250 (-9.29%)	2.1222	7.0837 (+233.79%)
User 2	2.1715 (+2.04%)	2.1688 (+1.91%)	1.9268 (-9.46%)	2.1281	7.1870 (+237.72%)
User 3	2.4002 (+0.15%)	2.4025 (+0.25%)	2.2299(-6.95%)	2.3965	5.3280 (+122.32%)
User 4	2.4051 (+0.27%)	2.4066 (+0.33%)	2.2244 (-7.27%)	2.3987	5.2914 (+120.59%)
User 5	3.4341 (-5.24%)	3.4671 (-4.32%)	3.8099 (+5.14%)	3.6238	3.5728 (-1.41%)
User 6	4.1598 (-4.65%)	4.1871 (-4.02%)	4.0876 (-6.30%)	4.3626	4.4760 (+2.60%)
User 7	3.4424 (-2.64%)	3.4522 (-2.36%)	3.0582 (-13.51%)	3.5357	8.3096 (+135.02%)
User 8	3.4455 (-2.68%)	3.4538 (-2.45%)	3.0513 (-13.82%)	3.5404	8.2211 (+132.21%)
User 9	5.9828 (-7.45%)	6.0497 (-6.44%)	7.1389 (+10.44%)	6.4641	3.0390 (-52.99%)
User 10	5.9995 (-7.58%)	6.0720 (-6.47%)	7.1495 (+10.13%)	6.4918	3.0527 (-52.98%)

Table 3.5: Scenario 1 Average Throughput Comparison under Low Traffic Intensity

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	49.99	49.99	49.99	49.99	49.99
User 2	50.03	50.03	50.03	50.03	50.03
User 3	39.99	39.99	39.99	39.99	39.99
User 4	39.99	39.99	39.99	39.99	39.99
User 5	19.99	19.99	19.99	19.99	19.99
User 6	20.00	20.00	20.00	20.00	20.00
User 7	30.00	30.00	30.00	30.00	30.00
User 8	30.00	30.00	30.00	30.00	30.00
User 9	10.00	10.00	10.00	10.00	10.00
User 10	10.00	10.00	10.00	10.00	10.00
Total	300.00	300.00	300.00	300.00	300.00
Theoretical total	300	300	300	300	300

As we expected, the throughput of all algorithms are the same because the operating point is strictly inside of the stability region of all algorithms. In fact, this operating point is far away from the boundary. However, their delay performance are slightly different. First of all, the majority users under LS and SS have better delays than that of TO.

Interestingly, there is no concrete evidence that SS performs better than LS in this traffic region. This implies that the drifting mechanism is not giving us any substantial gain. This fact can be interpreted in the following way: since the drift mechanism is

Table 3.6: Scenario 1 Maximum Starvation Period under Low Traffic Intensity

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	28	28	32	30	9

Table 3.7: Scenario 1 High Traffic Intensity

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	190	190	180	180	200	200	160	160	170	170

Table 3.8: Scenario 1 Parameters of Algorithm SS under High Traffic Intensity

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	19	19	18	18	20	20	16	16	17	17
β (kbps)	9.5	9.5	9	9	10	10	8	8	8.5	8.5
ϕ (bits)	19	19	18	18	20	20	16	16	17	17

designed to improve the short-term delay performance by tracking the queue dynamic, its advantage will be obvious when the queue length swings up and down frequently and substantially. When the traffic is low, the queue length is very stable. Therefore, the effect of the drift mechanism is not significant. It should be expected that the performance of SS should stand out when traffic rate is high.

When it comes to starvation period, LS, SS, LTM and TO have almost identical maximum starvation period. However, RR has a starvation period of $N-1$, where N is the number of users in the system. This type of starvation behavior is a dual sword. In one way, it ensures that everyone gets their service in a fair manner, and no user will be starved more than $N-1$ slots. In the other way, this fair manner of assigning transmission opportunity could hurt users with higher traffic rate badly.

3.3.2 Performance Analysis with High Traffic Intensity

In this section, we will show the performance comparison under high traffic intensity. High traffic intensity refers to the arrival rates which are close to the boundary of the stability region of throughput optimal algorithm. In the next section, we will show that a small increment of the traffic rates will cause instability of all algorithms. The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.7 and 3.8. Tables 3.9, 3.10 and 3.11 show the simulation results.

Table 3.9: Scenario 1 Average Delay Performance Comparison under High Traffic Intensity

Delay (ms)	LS	SS	LTM	TO (baseline)	RR
User 1	40 (-33.3%)	30 (-50%)	20 (-66.7%)	60	unbounded
User 2	40 (-33.3%)	30 (-50%)	20 (-66.7%)	60	unbounded
User 3	50 (-16.7%)	30 (-50%)	20 (-66.7%)	60	unbounded
User 4	50 (-16.7%)	30 (-50%)	20 (-66.7%)	60	unbounded
User 5	40 (-33.3%)	30 (-50%)	40 (-33.3%)	60	unbounded
User 6	30 (-50%)	40 (-33.3%)	unbounded	60	unbounded
User 7	50 (-28.6%)	50 (-28.6%)	40 (-42.9%)	70	unbounded
User 8	60 (-14.3%)	50 (-28.6%)	40 (-42.9%)	70	unbounded
User 9	50 (-28.6%)	40 (-42.9%)	80 (+14.3%)	70	unbounded
User 10	50 (-28.6%)	40 (-42.9%)	80 (+14.3%)	70	unbounded

The above simulation results provide an illustration that the delay performance of SS outperforms other schemes except user 6. Moreover, LS's delay performance is also better than that of TO. This result can be explained by investigating the decision process of these schedulers. TO makes decision based on current queue length and channel condition, whereas LS only makes decision based on channel condition (assume the dual variable converges and serves as a scaling factor). LS knows the incoming traffic rates in advance and tries to meet this target by exercising the knowledge of the channel variation. This process does not suffer from the randomness of the queues. On the other hand, without the prior knowledge of the traffic rates, TO has to use queue process to ensure stability. The randomness of the queuing process will degrade the delay performance of TO. In conclusion, the prior knowledge of traffic rates does improve the delay performance of both LS and SS significantly. Additionally, SS outperforms LS for majority users due to the introduction of the drifting mechanism which utilizes the real-time queueing information.

The average throughput comparison provides evidence that LS, SS and TO have larger stability regions than that of LTM and RR. Specially, RR has the smallest stability

Table 3.10: Scenario 1 Average Throughput Comparison under High Traffic Intensity

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	190	190	190	190	71.4
User 2	190	190	190	190	71.2
User 3	180	180	180	180	71.6
User 4	180	180	180	180	71.2
User 5	200	200	200	200	71.8
User 6	200	200	174.8	200	52
User 7	160	160	160	160	51.6
User 8	160	160	160	160	52.1
User 9	170	170	170	170	51.8
User 10	170	170	169.9	170	52.1
Total	1800	1800	1774.9	1800	616.7
Theoretical total	1800	1800	1800	1800	1800

Table 3.11: Scenario 1 Maximum Starvation Period under High Traffic Intensity

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	30	30	33	30	9

Table 3.12: Scenario 1 Saturated Traffic

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	195	195	185	185	205	205	165	165	175	175

Table 3.13: Scenario 1 Parameters of Algorithm SS under Saturated Traffic

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	19.5	19.5	18.5	18.5	20.5	20.5	16.5	16.5	17.5	17.5
β (kbps)	9.75	9.75	9.25	9.25	10.25	10.25	8.25	8.25	8.75	8.75
ϕ (bits)	19.5	19.5	18.5	18.5	20.5	20.5	16.5	16.5	17.5	17.5

region resulting from its "fairness" manner of assigning resources. LTM has a smaller stability region because its intention is to maximize the throughput in the current time slot. In general, local throughput maximization is not equivalent to long-term throughput maximization. As mentioned in the derivation of algorithm SS, the stability region of SS may be sacrificed for delay performance improvement. However, this tradeoff is negligible since the drifting is temporary, and the positive drift cancels out the negative drift in long term.

In terms of maximum starvation period, LS, SS, LTM and TO still have almost identical values. RR's starvation period remains unchanged as we expected.

3.3.3 Performance Analysis with Saturated Traffics

In this section, we will show the performance comparison under saturated traffics. Saturated traffics refers to the arrival rates outside the stability region. Since the last section simulates the traffic rates which are close to the boundary of the stability region, we only need to increase the traffic rates by a small amount to obtain a saturated traffic setting. The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.12 and 3.13. Tables 3.14 and 3.15 show the simulation results. Note that the delay performances are not compared because the system is no longer stable and the queue length of each user is not bounded anymore. All algorithms will have unbounded delays for most of the users.

Table 3.14: Scenario 1 Average Throughput Comparison under Saturated Traffic

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	192.3	192.8	194.9	192.2	71.2
User 2	192.3	192.8	195.1	192.3	71.6
User 3	182.3	182.2	185	182.3	71.5
User 4	182.3	182.2	185	182.3	71.5
User 5	202.3	203.4	204.6	202.3	71.6
User 6	202.3	203.8	169.4	202.3	51.7
User 7	162.3	161.1	164.9	162.3	51.7
User 8	162.3	161.0	165	162.3	51.5
User 9	172.3	171.8	170.7	172.2	51.9
User 10	172.3	171.8	169.3	172.3	51.6
Total	1822.8	1822.8	1804	1822.8	615.9
Theoretical total	1850	1850	1850	1850	1850

Thus, only the average throughputs are compared to illustrate the processing capacity of each algorithm.

The delay performance is not included in this case because all delays will be unbounded. However, the average throughput comparison still provides us important insights about the stability region/processing ability of each algorithm. LS and TO have identical throughput for each user, and LS, SS and TO all have same total throughput. Comparing with the high traffic intensity case, the traffic rate of individual user is increased only by 5kbps.

Table 3.15: Scenario 1 Maximum Starvation Period under Saturated Traffic

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	30	30	33	30	9

Table 3.16: Scenario 2 Channel Distribution

Channel rate (kbps)	38.4	76.8	102.6	153.6	204.8	307.2	614.4	921.6	1228.8	1843.2	2457.6
Distribution 1	0.06	0.06	0.04	0.04	0.1	0.1	0.19	0.19	0.06	0.08	0.08
Distribution 2	0.15	0.15	0.1	0.1	0.1	0.1	0.02	0.05	0.05	0.09	0.09

However, this small amount of increase in traffic intensity has caused the system to become unstable under all schemes. Hence, this implies the three algorithms have similar processing ability and stability region.

3.4 Simulation Analysis: Second Scenario

To further validate our results, we will simulate the two algorithms under a different simulation scenario. Similar to the first scenario, there are also ten users in the system and there are two different types of channel distributions shown in Table 3.16. User 1 to 5's channels are modeled as i.i.d channels with distribution 1. User 6 to 10's channels are modeled with distribution 2. The mean channel rates for distributions 1 and 2 are 777.992kbps and 600.98kbps respectively.

3.4.1 Performance Analysis with Low Traffic Intensity

In this section, we will show the performance comparison under low traffic intensity. The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.17 and 3.18. Tables 3.19, 3.20 and 3.21 show the simulation results. The delay performance for most of the users have been improved under both proposed algorithms comparing to that of the throughput optimal policy. However, there is no obvious evidence showing which one of the two proposed algorithms is better in terms of delay. And since we are testing all

Table 3.17: Scenario 2 Low Traffic Intensity

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	50	50	40	40	20	20	30	30	10	10

Table 3.18: Scenario 2 Parameters of Algorithm SS under Low Traffic Intensity

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	5	5	4	4	2	2	3	3	1	1
β (kbps)	2.5	2.5	2	2	1	1	1.5	1.5	0.5	0.5
ϕ (bits)	5	5	4	4	2	2	3	3	1	1

algorithms with low traffic intensities, the system is stable under all schemes tested; hence, as expected, the average throughput of each user as well as total throughput are same for all five schemes.

Table 3.19: Scenario 2 Average Delay Performance Comparison under Low Traffic Intensity

Delay (ms)	LS	SS	LTM	TO (baseline)	RR
User 1	2.2516 (+1.55%)	2.2452 (+1.26%)	1.9080 (-13.95%)	2.2173	6.8414 (+208.55%)
User 2	2.2523 (+1.5%)	2.2482 (+1.32%)	1.9098 (-13.93%)	2.2189	6.8620 (+209.25%)
User 3	2.4874 (-0.35%)	2.4908 (-0.21%)	2.2012 (-11.82%)	2.4962	5.2862 (+111.77%)
User 4	2.4920 (-0.11%)	2.4966 (+0.08%)	2.2020 (-11.73%)	2.4947	5.2976 (+112.36%)
User 5	3.5422 (-5.04%)	3.5797 (-4.03%)	3.7438 (+0.37%)	3.7301	3.6214 (-2.92%)
User 6	3.8905 (-3.46%)	3.9177 (-2.79%)	4.2301 (+4.97%)	4.03	7.1362 (+77.07%)
User 7	3.2421 (-1.12%)	3.2475 (-0.96%)	3.2069 (-2.2%)	3.2790	10.278 (+213.45%)
User 8	3.2428 (-1.1%)	3.2418 (-1.13%)	3.2088 (-2.13%)	3.2788	10.4576 (+218.95%)
User 9	5.5641 (-7.1%)	5.6347 (-5.93%)	7.2764 (+21.48%)	5.9896	3.9265 (-34.44%)
User 10	5.5708 (-7.15%)	5.6431 (-5.94%)	7.2658 (+21.11%)	5.9996	3.9264 (-34.45%)

Table 3.20: Scenario 2 Average Throughput Comparison under Low Traffic Intensity

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	49.98	49.98	49.98	49.98	49.98
User 2	50.01	50.01	50.01	50.01	50.01
User 3	39.99	39.99	39.99	39.99	39.99
User 4	39.99	39.99	39.99	39.99	39.99
User 5	20.00	19.99	19.99	19.99	19.99
User 6	20.02	20.02	20.02	20.02	20.02
User 7	30.01	30.01	30.01	30.01	30.01
User 8	29.99	29.99	29.99	29.99	30.00
User 9	10.00	10.00	10.00	10.00	10.00
User 10	10.00	10.00	10.00	10.00	9.99
Total	299.99	299.99	299.99	299.99	299.99
Theoretical total	300	300	300	300	300

3.4.2 Performance Analysis with High Traffic Intensity

The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.22 and 3.23. The results in Tables 3.24, 3.25 and 3.26 illustrate that the delays of all users have been improved significantly using the proposed schemes and SS performs even better than LS for most of the users. Moreover, the system becomes unstable under schemes LTM and RR; hence, they must have smaller processing capacities than those of the proposed

Table 3.21: Scenario 2 Maximum Starvation Period under Low Traffic Intensity

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	28	28	32	28	9

Table 3.22: Scenario 2 High Traffic Intensity

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	215	215	205	205	225	225	185	185	195	195

Table 3.23: Scenario 2 Parameters of Algorithm SS under High Traffic Intensity

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	21.5	21.5	20.5	20.5	22.5	22.5	18.5	18.5	19.5	19.5
β (kbps)	10.75	10.75	10.25	10.25	11.25	11.25	9.25	9.25	9.75	9.75
ϕ (bits)	21.5	21.5	20.5	20.5	22.5	22.5	18.5	18.5	9.75	9.75

algorithms and TO.

3.4.3 Performance Analysis with Saturated Traffics

The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.27 and 3.28. Tables 3.29 and 3.30 show the simulation results. Again, in this case, each user's arrival intensity is increased by 5kbps comparing with the high traffic intensity case. The total throughput values under all schemes are smaller than the theoretical ones, and this means that the system is no longer stable. We can say that the processing capacity of our proposed algorithms are almost identical to that of the throughput optimal policy.

3.5 Simulation Analysis: Third Scenario

In this scenario, we increased the number of users in the system to fifteern and the two types of channel distributions are shown in Table 3.31. User 1 to 7's channels are modeled as i.i.d channels with distribution 1. User 8 to 15's channels are modeled with distribution 2. The means of distribution 1 and 2 are 714.89kbps and 517.78kbps respectively. Since the simulation results are quite consistent with our two previous scenarios, only the simulation results are presented below for illustration purpose. We will give a brief summary for all

Table 3.24: Scenario 2 Average Delay Performance Comparison under High Traffic Intensity

Delay (ms)	LS	SS	LTM	TO (baseline)	RR
User 1	22 (-45%)	21 (-47.5%)	83 (+107.5%)	40	unbounded
User 2	20 (-50%)	20 (-50%)	33.3 (-16.75%)	40	unbounded
User 3	21 (-48.78%)	21 (-48.78%)	72 (+75.6%)	41	unbounded
User 4	22 (-45%)	22 (-45%)	41 (+2.5%)	40	unbounded
User 5	21 (-46.15%)	20 (-48.72%)	unbounded	39	unbounded
User 6	21 (-44.74%)	19 (-50%)	56 (+47.37%)	38	unbounded
User 7	34 (-19.05%)	34 (-19.05%)	15 (-64.29%)	42	unbounded
User 8	27 (-35.71%)	32 (-23.81%)	14 (-66.67%)	42	unbounded
User 9	27 (-34.15%)	27 (-34.15%)	18 (-56.1%)	41	unbounded
User 10	24 (-41.46%)	26 (-36.59%)	19 (-53.66%)	41	unbounded

three test scenarios at the end of this chapter.

3.5.1 Performance Analysis with Low Traffic Intensity

The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.32 and 3.33. Tables 3.34, 3.35 and 3.36 show the simulation results.

Table 3.25: Scenario 2 Average Throughput Comparison under High Traffic Intensity

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	214.9	214.9	214.7	214.9	77.5
User 2	215.1	215.1	214.2	215.1	77.4
User 3	205.2	205.2	204.6	205.1	78.2
User 4	205.1	205.1	205	205.1	77.9
User 5	225.1	225.1	216.1	225.1	77.8
User 6	224.9	224.9	224.2	224.9	60.7
User 7	185.4	185.1	185	185.1	60.9
User 8	184.9	184.9	184.8	184.9	59.6
User 9	195.1	195	194.9	194.9	60.6
User 10	194.8	194.8	194.8	194.8	60.8
Total	2050.4	2049.9	2038.5	2049.9	691.5
Theoretical total	2050	2050	2050	2050	2050

Table 3.26: Scenario 2 Maximum Starvation Period under High Traffic Intensity

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	36	36	36	35	9

Table 3.27: Scenario 2 Saturated Traffic

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	220	220	210	210	230	230	190	190	200	200

Table 3.28: Scenario 2 Parameters of Algorithm SS under Saturated Traffic

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	22	22	21	21	23	23	19	19	20	20
β (kbps)	11	11	10.5	10.5	11.5	11.5	9.5	9.5	10	10
ϕ (bits)	22	22	21	21	23	23	19	19	20	20

Table 3.29: Scenario 2 Average Throughput Comparison under Saturated Traffic

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	218.1	218.6	210.9	218	78.5
User 2	218	218.6	210	217.9	77.5
User 3	208	207.9	208.8	208.1	78
User 4	207.9	207.8	209.2	208	77.6
User 5	228	229.2	208.7	227.9	76.1
User 6	227.9	229.4	225.6	227.9	59.8
User 7	188	186.8	190	188.1	61
User 8	188	186.7	190	188	61.7
User 9	198	197.4	200	198	61
User 10	197.9	197.4	200.1	198.1	60
Total	2079.8	2079.8	2053.4	2079.8	691.1
Theoretical total	2100	2100	2100	2100	2100

Table 3.30: Scenario 2 Maximum Starvation Period under Saturated Traffic

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	36	36	36	36	9

Table 3.31: Scenario 3 Channel Distribution

Channel rate (kbps)	38.4	76.8	102.6	153.6	204.8	307.2	614.4	921.6	1228.8	1843.2	2457.6
Distribution 1	0.05	0.05	0.05	0.05	0.1	0.1	0.2	0.2	0.1	0.05	0.05
Distribution 2	0.05	0.05	0.1	0.1	0.2	0.2	0.1	0.05	0.05	0.05	0.05

Table 3.32: Scenario 3 Low Traffic Intensity

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}
Traffic Intensity (kbps)	10	10	20	20	30	40	40	30	30	25	20	20	20	20	10

Table 3.33: Scenario 3 Parameters of Algorithm SS under Low Traffic Intensity

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α (kbps)	1	1	2	2	3	4	4	3	3	2.5	2	2	2	2	1
β (kbps)	0.5	0.5	1	1	1.5	2	2	1.5	1.5	1.25	1	1	1	1	0.5
ϕ (bits)	1	1	2	2	3	4	4	3	3	2.5	2	2	2	2	1

Table 3.34: Scenario 3 Average Delay Performance Comparison under Low Traffic Intensity

Delay (ms)	LS	SS	LTM	TO (baseline)	RR
User 1	6.7216 (-4.92%)	6.8033 (-3.77%)	8.0758 (+14.23%)	7.0697	4.8832 (-30.93%)
User 2	6.7120 (-4.41%)	6.7610 (-3.72%)	8.0858 (+15.15%)	7.0220	4.8521 (-30.90%)
User 3	4.3539 (-2.06%)	4.3618 (-1.88%)	4.4454 (-0%)	4.4454	6.1163 (+37.59%)
User 4	4.3082 (-2.23%)	4.3460 (-1.38%)	4.4466 (+0.91%)	4.4066	6.2609 (+42.08%)
User 5	3.4507 (+0.4%)	3.4610 (+0.7%)	3.2074 (-6.68%)	3.4368	9.0406 (+163.05%)
User 6	2.9661 (+2.65%)	2.9525 (+2.18%)	2.5724 (-10.98%)	2.8896	17.1992 (+495.22%)
User 7	2.9756 (+2.81%)	2.9700 (+2.62%)	2.5672 (-11.3%)	2.8943	19.0410 (+557.89%)
User 8	4.1520 (-0.75%)	4.1444 (-0.93%)	3.6404 (-12.98%)	4.1833	48.1911 (+1051.98%)
User 9	4.1623 (+0.01%)	4.1749 (+0.31%)	3.6485 (-12.34%)	4.1619	48.2847 (+1060.16%)
User 10	4.5989 (-0.61%)	4.6094 (-0.38%)	4.1909 (-9.43%)	4.6272	24.9216 (+438.60%)
User 11	5.0838 (-2.12%)	5.0907 (-1.99%)	4.9052 (-5.56%)	5.1939	13.0099 (+150.49%)
User 12	5.0960 (-1.67%)	5.1120 (-1.36%)	4.8916 (-5.62%)	5.1827	12.9829 (+150.5%)
User 13	5.1107 (-2.24%)	5.1285 (-1.90%)	4.8951 (-6.36%)	5.2278	12.5932 (+140.89%)
User 14	5.1120 (-1.53%)	5.1470 (-0.85%)	4.8824 (-5.95%)	5.1914	12.4662 (+140.13%)
User 15	7.5113 (-4.58%)	7.5603 (-3.96%)	8.5609 (+8.75%)	7.8720	5.2233 (-33.67%)

3.5.2 Performance Analysis with High Traffic Intensity

The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.37 and 3.38. Tables 3.39, 3.40 and 3.41 show the simulation results.

3.5.3 Performance Analysis with Saturated Traffics

The detailed traffic rates and parameters of algorithm SS are shown in Tables 3.42 and 3.43. Tables 3.44 and 3.45 show the simulation results.

3.6 Simulation Summary

The two proposed algorithms have been simulated in three different scenarios with different channel conditions, traffic arrival rates as well as number of users in the system. All numerical results are consistent and indicate that the delay performance of both proposed algorithms are better than other benchmark schemes for majority users under any traffic conditions. And the improvement is more significant when traffics are heavy. Moreover, simulation results also indicate that the processing abilities/stability regions of the two proposed algorithms are very close to that of the throughput optimal algorithm.

Table 3.35: Scenario 3 Average Throughput Comparison under Low Traffic Intensity

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	10.01	10.01	10.01	10.01	10.01
User 2	10.01	10.01	10.01	10.01	10.01
User 3	20.01	20	20	20.01	20.01
User 4	20.01	20.01	20.01	20	20.01
User 5	30.02	30.02	30.02	30.02	30
User 6	40.04	40.04	40.04	40.04	40.04
User 7	40.04	40.04	40.04	40.04	40.04
User 8	29.97	29.96	29.97	29.97	30
User 9	30.03	30.03	30.03	30.03	30.01
User 10	25.01	25.01	25.01	25.01	25
User 11	20.03	20.04	20.03	20.04	20.04
User 12	20	20	20.01	20	20.01
User 13	20.02	20.02	20.02	20.02	20.03
User 14	20.04	20.04	20.04	20.04	20.04
User 15	10	10	10.01	10	10
Total	345.25	345.25	345.25	345.25	345.26
Theoretical total	345	345	345	345	345

Table 3.36: Scenario 3 Maximum Starvation Period under Low Traffic Intensity

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	38	38	39	39	14

Table 3.37: Scenario 3 High Traffic Intensity

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	130	130	140	140	150	160	160	130	120	115

User	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}
Traffic Intensity (kbps)	120	120	120	130	140

Table 3.38: Scenario 3 Parameters of Algorithm SS under High Traffic Intensity

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	13	13	14	14	15	16	16	13	12	11.5
β (kbps)	6.5	6.5	7	7	7.5	8	8	6.5	6	5.75
ϕ (bits)	13	13	14	14	15	16	16	13	12	11.5

User	11	12	13	14	15
α(kbps)	12	12	12	13	14
β (kbps)	6	6	6	6.5	7
ϕ (bits)	12	12	12	13	14

Table 3.39: Scenario 3 Average Delay Performance Comparison under High Traffic Intensity

Delay (ms)	LS	SS	LTM	TO (baseline)	RR
User 1	45 (-43.84%)	46 (-43.19%)	28 (-64.91%)	80	unbounded
User 2	48 (-39.45%)	48 (-39.31%)	25 (-68.77%)	80	unbounded
User 3	41 (-46.68%)	37 (-52.26%)	44 (-42.95%)	77	unbounded
User 4	38 (-50.63%)	36 (-53.33%)	61 (-21.88%)	78	unbounded
User 5	37 (-49.46%)	31 (-57.49%)	unbounded	73	unbounded
User 6	33 (-52.56%)	27 (-62.18%)	unbounded	70	unbounded
User 7	29 (-58.67%)	25 (-65.17%)	unbounded	71	unbounded
User 8	46 (-44.74%)	46 (-44.54%)	67 (-20.33%)	84	unbounded
User 9	47 (-46.08%)	58 (-34.37%)	33 (-62.25%)	88	unbounded
User 10	64 (-27.36%)	86 (-3.34%)	26 (-70.23%)	89	unbounded
User 11	85 (-2.46%)	74 (-15.48%)	32 (-63.97%)	88	unbounded
User 12	43 (-51.42%)	52 (-40.42%)	40 (-54.28%)	88	unbounded
User 13	45 (-47.23%)	56 (-34.96%)	27 (-68.4%)	86	unbounded
User 14	46 (-45.26%)	48 (-42.6%)	73 (-13.37%)	84	unbounded
User 15	38 (-52.65%)	36 (-55.62%)	unbounded	80	unbounded

Table 3.40: Scenario 3 Average Throughput Comparison under High Traffic Intensity

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	130.1	130	130.1	129.99	47.2
User 2	130.2	130	130.1	130	48.6
User 3	140.1	140	140.1	140	47.3
User 4	140.1	140.1	139.8	140	48.00
User 5	150	150	149.6	149.8	48.2
User 6	160	160.1	154.7	159.9	47.8
User 7	159.9	159.9	150	159.9	47.1
User 8	129.8	129.7	129.9	129.8	34.6
User 9	120.1	120.1	119.9	119.9	34.1
User 10	115.1	114.8	115	114.9	34.6
User 11	120.5	120	120	120	34.2
User 12	119.9	119.9	119.9	119.8	34.3
User 13	120.1	119.9	120.1	119.9	34.2
User 14	130	130	129.8	129.9	34.4
User 15	139.9	139.9	135.9	139.9	33.9
Total	2005.7	2004.3	1984.8	2003.7	608.6
Theoretical total	2005	2005	2005	2005	2005

Table 3.41: Scenario 3 Maximum Starvation Period under High Traffic Intensity

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	45	45	45	45	14

Table 3.42: Scenario 3 Saturated Traffic

User	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8	λ_9	λ_{10}
Traffic Intensity (kbps)	135	135	145	145	155	165	165	135	125	120

User	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}
Traffic Intensity	125	125	125	135	140

Table 3.43: Scenario 3 Parameters of Algorithm SS under Saturated Traffic

User	1	2	3	4	5	6	7	8	9	10
α (kbps)	13.5	13.5	14.5	14.5	15.5	16.5	16.5	13.5	12.5	12
β (kbps)	6.75	6.75	7.25	7.25	7.75	8.25	8.25	6.75	6.25	6
ϕ (bits)	13.5	13.5	14.5	14.5	15.5	16.5	16.5	13.5	12.5	12

User	11	12	13	14	15
α (kbps)	12.5	12.5	12.5	13.5	14
β (kbps)	6.25	6.25	6.25	6.75	7
ϕ (bits)	12.5	12.5	12.5	13.5	14

Table 3.44: Scenario 3 Average Throughput Comparison under Saturated Traffic

Throughput (kbps)	LS	SS	LTM	TO	RR
User 1	131.4	131.2	134.7	131.5	48
User 2	131.4	131.1	134.9	131.5	47.4
User 3	141.5	142	144.7	141.6	47.9
User 4	141.5	142	144.1	141.6	47.6
User 5	151.5	152.7	143.1	151.4	47
User 6	161.5	163.3	145.8	161.3	46.9
User 7	161.4	163.3	146	161.4	47.8
User 8	131.4	131.2	129.9	131.3	34.2
User 9	121.4	120.4	125	121.4	35.1
User 10	116.4	115.1	120	116.5	35
User 11	121.3	120.3	124.8	121.3	35.5
User 12	121.5	120.5	125	121.5	35.2
User 13	121.4	120.5	125.1	121.5	34.4
User 14	131.5	131.2	133.4	131.4	36.1
User 15	141.6	142.2	132.5	141.7	35.1
Total	2027	2027	2008.9	2026.9	613.2
Theoretical total	2080	2080	2080	2080	2080

Table 3.45: Scenario 3 Maximum Starvation Period under Saturated Traffic

Scheduling Scheme	LS	SS	LTM	TO	RR
Maximum Starvation Period (slots)	47	47	47	47	14

Chapter 4

Conclusion

The goal of this thesis is to develop efficient algorithms which minimize the amount of resources to stabilize the incoming traffic. The stability regions of the proposed algorithms are investigated by using simulations. The results suggest that the proposed algorithms can achieve almost the same stability region as the throughput optimal algorithm. However, a strict mathematical derivation in proving that the proposed algorithms indeed achieve the largest stability region is left for future research.

The most important difference between the proposed algorithms and the well known throughput optimal algorithm is that instead of using the queue length, the incoming traffic rate is used when scheduling is done. The delay performance of the proposed algorithms are analyzed by means of simulation and their performances are compared with the throughput optimal algorithm. The simulation results show that the application of traffic arrival rates does improve the delay performance and the improvement is more significant when traffic intensity is high.

Simulations have been run under three different traffic regions. Especially under high traffic condition, the delay performance is significantly improved. Finally, in saturated traffics case, numerical results illustrate that our two schemes have comparable data processing ability as the throughput optimal algorithm. We can conclude that the proposed algorithms perform consistently well under all traffic conditions and is able to handle the constantly varying traffic intensities in reality.

For future work, it will also be interesting to analyze the impact of the tuning pa-

rameters (drifting parameters) in the modified algorithm. In addition, we would like to extend this work to WiMax system which has several intermediate stations called subscriber station between the end users and the central base stations. With the addition of this subscriber station, the new topology looks like a two-level point-to-multipoint structure. In this case, we would like to design the corresponding opportunistic algorithm which achieves the largest stability region with minimal communication overhead. Typically, the number of users in WiMax system is huge. It is impossible to upload the queue information of every user to the base station for scheduling decision to be made. Hence, information aggregation is necessary and it will be a interesting topic to study what kind of aggregated information the user should send.

Bibliography

[1] G. Acar and C. Rosenberg. Algorithms to Compute for Bandwidth on Demand Requests in a Satellite Access Unit. In *Proc. of 5th Ka Band Utilization Conference*, Oct. 1999.

[2] M. Andrews, S. Borst, F. Dominique, P. jelenkovic, K. Kumanaran, K. Ramakrishnan, and P. Whiting. Dynamic Bandwidth Aloocation Algorithms for High-Speed Data Wireless Networks. *Bell Labs Technical Report*, 2000.

[3] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, and P. Whiting. Providing Quality of Service over a Shared Wireless Link. *IEEE Communications Magzine*, 39(2):150–153, Feb. 2001.

[4] Y. Cao and V. Li. Scheduling Algorithms in Broad-Band Wireless Networks. 89(1).

[5] Y.J. Choi and S. Bahk. Delay-Senstitive Packet Scheduling for a Wireless Access Link. *IEEE Transactions on Mobile Computing*, 5(10):1374–1380, Oct. 2006.

[6] H. Fattah and C. Leung. An Overview of Scheduling Algorithms in Wireless Multimedia Networks. *IEEE Wireless Communications*, Oct. 2002.

[7] A. Girard, C. Rosenberg, and M. Khemiri. Fairness and Aggregation: A Primal Decomposition Study.

[8] M. Hu, J. Zhang, and J. Sadowsky. Size-Aided Opportunistic Scheduling in Wireless Networks. In *Proc. of IEEE GLOBECOM*, 2003.

[9] F.P. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[10] C. Koksal, H. Kassab, and H. Balakrishnan. An Analysis of Short-Term Fairness in Wireless Media Access Protocols. In *Proc. of ACM/SIGMETRICS*, June 2000.

[11] S.S. Kulkarni and C. Rosenberg. Opportunistic Scheduling for Wireless Systems with Multiple Interfaces and Multiple Constarints. In *Proc. of ACM/SIGCOM MSWiM*, Sep. 2003.

[12] S.S. Kulkarni and C. Rosenberg. Opportunistic Scheduling Policies for Wireless Systems with Short Term Fairness Constarints. In *Proc. of IEEE GLOBECOM*, Dec. 2003.

[13] X. Lin and N.B. Shrof. The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks. *IEEE/ACM Transactions on Networking*, 14(2):302–315, 2006.

[14] X. Liu, E.K.P. Chong, and N.B. Shroff. Opportunistic Transmission Scheduling with Resource-sharing Constraints in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 19(10), Oct. 2001.

[15] X. Liu, E.K.P. Chong, and N.B. Shroff. A Framework for Opportunistic Scheduling in Wireless Networks. *Computer Networks*, 41(4):451–474, Oct. 2003.

[16] S. Shakkottai and A. Stolyar. A Study of Scheduling Algorithms for a Mixture of Real- and Non-Real-Time Data in HDR. *Bell Labs Tech. Memo.*, Aug. 2000.

[17] T.C. Tsai, C.H. Jiang, and C.Y. Wang. CAC and Packet Scheduling Using Token Bucket for IEEE 802.16 Networks. *Journal of Communications*, 1(2):30–37, May 2006.

[18] K. Wongthavarawat and A. Ganz. Packet Scheduling for QoS support in IEEE 802.16 Broadband Wireless Access Systems. *International Journal of Communications Systems*, 16:81–96, 2003.

[19] H. Yaiche, R. Mazumdar, and C. Rosenberg. A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks. *IEEE/ACM Transactions on Networking*, 8(5):667–678, 2000.