University of Alberta

A Power Conserving Fair Queueing Protocol for Location-Dependent Errors

by

Ashutosh Muni ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2000

0-612-60155-2

University of Alberta

Library Release Form

Name of Author: Ashutosh Muni

Title of Thesis: A Power Conserving Fair Queueing Protocol for Location-Dependent Errors

Degree: Master of Science

Year this Degree Granted: 2000

Ashutosh Muni

Ashutosh Muni
615 GSB, University of Alberta
Edmonton,AB
Canada, T6G 2H1

Date: Dec. 21, 1999

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **A Power Conserving Fair Queueing Protocol for Location-Dependent Errors** submitted by Ashutosh Muni in partial fulfillment of the requirements for the degree of **Master of Science.**

[illegible]

Dr. I. Nikolaidis

[illegible]

Dr. M. H. MacGregor

[illegible]

Dr. X. Sun

Date: Dec. 20, 1999

Abstract

Advances in wireless networking technology and portable/mobile computers have engendered a relatively new area of research, *mobile computing.* Mobile computing has a promise to allow the user carrying a small sized portable computers to access a shared infrastructure independent of their physical location, thus providing a flexible communication and continuous access to networked services using wireless networking technology.

Providing *fair scheduling* is a critical requirement for sustaining the desired quality of service for the mobile users over scarce, dynamic and shared wireless channels. Most contemporary wire-line fair scheduling algorithms cannot be directly used for wireless networks because of the *location-dependent unreliable* nature of wireless channels. Further, limited size of mobile and portable computers poses another important issue of *power conservation* at the mobile nodes (MNs). Various schemes have been proposed to address fairness and power conservation issues [1, 2, 3, 4]. The proposals on fairness usually extend the operation of *Weighted Fair Queueing (WFQ)* or other guaranteed service wire-line scheduling algorithms, to wireless networks, and claim to provide *delay guarantees*, *short term fairness for error-free mobile(s)*, and *long term fairness for error prone mobile(s).* Separately from the research on fairness, proposals have been made to address the issue of power conservation from both the hardware and software perspectives.

In this thesis we present a new *Power Conserving Fair Queueing* (PCFQ) media access control (MAC) protocol, which addresses both the fairness and the power conservation issues while incorporating realistic assumptions regarding the lack of *a-priori* knowledge of wireless channel state. Simulation results are then presented to address the issues of service guarantees, fairness and power conservation at the MNs.

The best advice to those about to embark
on a very large simulation is often the same as
Punch's famous advice to those about to marry:
Don't
by Bratley, Fox, and Schrage (1986)

To my parents
Mr. & Mrs. J.M. Tyagi
and my sisters
Mrs. Nidhi Tyagi & Dr. Neeti Sharma

Acknowledgements

I would like to take this opportunity to thank my supervisor Dr. Ioannis Nikolaidis for his patience, persistence, constructive criticisms, and funding. I would like to acknowledge his guidance not only on technical aspects but also on effective writing.

I am also grateful to the members of my examining committee, Dr. X. Sun and Dr. M. H. MacGregor for taking the time to read and provide valuable comments for this thesis.

I would also like to thank all my colleagues in the Department of Computing Science for their support and encouragement. They include: Srinivas Padmanabhuni, Theodore Ono-Tesfaye, Wladek Olesinski, and Nehal Koticha.

Finally, my thanks go to my parents and my sisters for their encouragement and endurance.

Table of Contents

List of Figures

List of Tables

List of Symbols

η_{max} - Maximum achievable utilization.

μ_B - Mean period during which channel is in error state.

μ_G - Mean period during which channel is in error-free state.

$\phi(i)$ - Rate weight of flow i.

ρ_i - Rate allocated to flow i.

ρ_i^0 - Rate with which the flow i was admitted.

ρ_i^R - Recovery rate of flow i.

σ - Maximum burst size in WFQ.

$\hat{A}(t)$ - Set of active flows.

B_i - Lag of node i.

B_i^{max} - Maximum lag allowed for node i.

B_S - Maximum burst size in PCFQ.

C - Total channel capacity.

CF - Finish number of packet currently being served in SCFQ.

$Delay_C$ - Cycle delay.

$Delay_I$ - Insertion delay.

$Delay_P$ - Propagation delay.

$Delay_Q$ - Queueing delay.

d_{SCFQ} - Worst case delay in SCFQ.

d_{WFQ} - Worst case delay in WFQ.

$\hat{E}(t)$ - Set of flows experiencing channel error.

$F(i, k, t)$ - Finish number of kth packet of connection i at time t.

$F(p_i^k)$ - Finish number of kth packet of node i.

L_i - Lead of node i.

L_i^{max} - Maximum lead allowed for node i.

$\hat{N}(t)$ - Set of normalized flows.

N - Number of data transmission slots in a cycle.

O_C - Overheads due to cycle.

P_A - Length of acknowledgment packet.

P_B - Length of schedule packet.

P_D - Length of data packet.

P_I - Length of polling information packet.

P_P - Length of poll probe packet.

$P(i,k,t)$ - Size of kth packet of connection i.

P_{max} - Maximum length of data packet.

Q^i_{max} - Maximum buffer size for a flow i.

$\hat{R}(t)$ - Set of recovering flows.

$R(t)$ - Current round number in WFQ.

r - Service rate of a scheduler.

$S_i[t_1,t_2]$ - Service received by node i during the $[t_1,t_2]$ time interval.

$S(p^k_i)$ - Start time tag of kth packet of node i.

t_A - Time required to transmit an acknowledgment packet.

t_B - Time required to broadcast a cycle schedule.

t_C - Time required to transmit a cycle.

t_D - Time required to transmit a data packet.

t_G - Generation time of a packet.

t_I - Time required to transmit polling information.

t_P - Time required to carry out a poll probe.

t_{Poll_Ins} - Time period between the schedule broadcast and polling instance.

V_{t_k} - Virtual time of kth packet.

$W_i(t_1,t_2)$ - Service received by node i during (t_1,t_2) time interval.

Chapter 1

Introduction and Motivation

Recent developments in the areas of high-performance portable/mobile computers and wireless networks have triggered research in the area of *mobile computing.* Owing to the unreliable nature of wireless channels (especially when combined with the user mobility) and given the limited size of mobile and portable computers, two major issues in the area of mobile computing are the *fair distribution of wireless bandwidth* and the *conservation of power* at the mobile nodes (MNs).

Various schemes have been proposed to address the fairness and power conservation issues [1, 2, 3, 4]. The proposals on fairness usually extend the operation of *Weighted Fair Queueing (WFQ)* [8] or other guaranteed service wire-line scheduling algorithms to wireless networks and claim to provide *delay guarantees*, *short term fairness for error-free mobiles*, and *long term fairness for error prone mobiles.* Separately from the research on fairness, proposals have been made to address the issue of power conservation from both the hardware and software perspectives [9, 10, 11, 12, 6, 4, 13]. This thesis presents a new *Power Conserving Fair Queueing* (PCFQ) media access control (MAC) protocol, which addresses both the fairness and the power conservation issues while incorporating realistic assumptions regarding the lack of *a-priori* knowledge about the wireless channel's state.

1.1 Mobile System Architecture

The use of wireless networks in a mobile computing environment offers the potential of unrestricted and continuous network connectivity. Mobile wireless systems can thus be viewed as computing systems that have the ability to compute, communicate, and collaborate anytime and anywhere. Based on the topology and/or routing followed by the mobile systems, there are two major types of networks: *ad-hoc networks* and *infra-structured networks*.

In ad-hoc networks, colocated MNs establish peer-to-peer communication among themselves without the help of any infrastructure such as a wired/wireless backbone. Broadcasting/flooding and building a temporary infrastructure by following a hierarchical packet passing scheme are the two major routing approaches followed to implement ad-hoc networks [14].

Despite the possibility of ad-hoc networking, many applications require communication with services located in a pre-existing infrastructure [15] e.g., a high speed wired/wireless backbone. In such a case all MNs are linked to the fixed network via access points referred to as *base stations* (BS). The network traffic in such infra-structured networks is typically divided into two directions: *upstream* (into the backbone) and *downstream* (from the backbone).

Various terms have been introduced to describe such mobile systems [16], including *ubiquitous computing*, *nomadic computing*, *decoupled computing*, and *wireless computing* systems. *Ubiquitous computing* refers to an environment where many computing devices are available in the same limited space such as an office. To interconnect a large number of such devices, wireless communication is seen as a convenient solution. The characteristic of ubiquitous computing is the very high demand for aggregate bandwidth within a limited volume even though the per-device demands are modest (e.g. 64 Kbps or less). *Nomadic computing* refers to the ability to compute as the user relocates from one environment to another. In this model, it is expected that individual organizations will construct their own wireless infrastructures that are linked across organizations by wired networks. In a nomadic computing scenario a user would like to be able to use his or her device even in a foreign organization's wireless infrastructure. System support for nomadic computing must address among other things the issues of trust, security, and privacy as wireless computing devices move between organizational infrastructures. *Decoupled computing* refers to the ability to compute even when completely detached from the existing computing and communications infrastructure. It enables operations, such as file access from servers, even while disconnected. For example by caching file data, the cached files may be re-synchronized with the copies at the server when connection is re-established. It is also termed *disconnected computing*. *Wireless computing* refers to computing systems that are connected to their working environment via wireless links, using radio frequency (RF) or infrared (IR) devices. The term also applies to computing devices participating in a wireless local area network, with gateways allowing connectivity to traditional wired networks.

1.1.1 Wireless Architectures

Two popular wireless architectures are *cellular networks* and *wireless LANs*. In general, both architectures allow mobile users to communicate to the wired backbone and subsequently to other mobiles via a BS.

In a typical cellular network architecture, the network provider divides the service area into small zones or *cells*. Each cell contains a BS that maintains wireless communication with the cellular devices in its cell and that is wired to the central office. Various technologies are used to share the transmission medium between the cellular devices in a cell. For the sake of convenience we will consider only *time division multiple access* (TDMA) schemes in the scope of this thesis. In TDMA, transmission slots are allocated to various cellular devices and all of them use the same frequency to transmit the data. TDMA schemes can be further distinguished into *frequency division duplexing* (FDD) and *time division duplexing* (TDD) based on the number of carrier frequencies used between the BS and the MNs [3]. In FDD separate frequencies are used for upstream and downstream transmissions, allowing it to have almost immediate acknowledgments in both directions. In TDD only one carrier frequency is used to transmit data in both the upstream and the downstream directions. This, however, generally adds an extra delay due to turnover between transmitter and receiver modes.

A Wireless LAN (WLAN) is a network that can be used as an alternative for a wired LAN within a building or campus, using wireless transmission mediums such as radio frequency (RF) and infra-red (IR). In a typical WLAN environment all the MNs are connected to an access point, which for the sake of consistency is also called the BS. Usually, the BS is also entrusted with various other responsibilities such as scheduling upstream and downstream traffic, managing the MNs and providing the required quality of service (QoS) to the MNs. Popular MAC level standards proposed for WLANs are *IEEE Standard P 802.11* and *High Performance Radio Local Area Network* (HIPERLAN).

The IEEE standard P 802.11 [17] is a WLAN standard developed by IEEE committee P 802.11 in order to specify a wireless interface between a wireless node and a BS, as well as among wireless nodes. It proposes a multiple access scheme using a technique based on carrier sense multiple access/collision avoidance (CSMA/CA). The basic access method is to listen the channel before transmitting and if the medium is busy the transmitter backs off for a random period. It avoids collisions by sending a short Ready To Send (RTS) message which contains the destination address and the duration of message. The standard specifies the use of a 2.4 GHz Industrial Scientific and Medical (ISM) band with 1-2 Mbps throughput.

HIPERLAN [18] is a European Telecommunications Standards Institute (ETSI) standard for next generation wireless systems on digital high speed wireless communication in 5.15-5.3 GHz and 17.1-17.3 GHz spectrum. It is designed to work even without any infra-structure, and thus can be used for ad-hoc WLANs. The 5 GHz HIPERLAN standard specifies a raw bit rate of up to 24 Mbps.

The area of coverage of a cellular network or the infra-structured WLAN is limited to the range in which transmissions BS can be received by the mobile nodes. Such an area is often referred as a *cell.* In general, when a MN crosses a cell during a call, it must switch to the BS in the new cell. This is managed by *hand-off* mechanisms. Usually, all MNs in a cell monitor a signal called the *beacon*, which is broadcasted periodically by the BS. As the node moves between cells, it may receive multiple beacons from multiple BSs. If the strongest beacon is from the BS other than the one that the device currently belongs to, a hand-off is initiated. The MN is assigned to the new BS, upon completion of hand-off.

In the scope of this thesis we describe a uni-cell wireless LAN architecture using the same frequency for both upstream and downstream transmissions (TDD). All the hand-off related activities are assumed to be performed by the BS using a separate protocol outside the MAC protocol. In Chapter 4, we outline how the handoff mechanisms can be integrated with the presented protocol.

For the sake of handoffs and new 'incoming' mobiles a connection is reserved at a rate determined by the BS. The connection corresponds to time slots reserved for handoff requests and new incoming mobiles.

1.2 System Constraints

Unlike wired networks, the mobility of MNs and the wireless nature of mobile wireless computing system impose many constraints [19, 16, 20]. Two particular constraints considered in the scope of this thesis are *power consumption* and *fading.*

Power consumption

Battery-powered mobile nodes have limited energy for computing and communications. Power in mobile nodes is consumed by different components such as screen

Component	Power consumed (W)
Processor	1.1 - 3.3 W (14-27%)
Hard drive	1.0 - 1.5 W (12.1-12.25%)
Backlight	2.3 - 3.4 W (28-30%)
Display	0.2 - 0.8 W (2-7%)
Modem	0.5 - 0.6 W (5-7%)
Sound	0.15 - 0.2 W (1.5-2.5%)
Video	0.3 - 0.5 W (3.8-4.5%)
Power management	0.1 - 0.2 W (1-1.5%)
Memory	0.05 - 0.1 W (0.5-0.8%)
Misc.	1.7 - 2 W (13-23%)

Table 1.1: Power consumed if power is not conserved [5, 4]

back-lighting and display, CPU, memory, hard disk, wireless communication unit, and keyboard. An example of the power consumption of each component is shown in Table 1.1. Efforts are being made to save power by keeping the display off when not in use. Moreover, a new breed of energy efficient CPUs, which save energy when idle are becoming more popular [6, 7, 5]. Flash memories, which consumes less power than dynamic RAM are being used as well. In software design, energy conservation has led to new classes of energy efficient systems software, data access protocols, and algorithms [10].

Conserving battery power in mobiles is a crucial consideration in designing protocols for mobile computing. This issue should be considered through all layers of the protocol stack, including the application layer. In this thesis power conservation issues at the MAC layer will be analyzed.

The chief sources of energy consumption in the mobile node for MAC-related activities are the CPU and the transceiver. CPU usage may be reduced by moving CPU-intensive protocol operations to the BS. The transceiver can operate in either *transmit*, *receive*, or *sleep* mode. In general, a transceiver consumes more power in transmit mode than in receive mode and consumes least power in sleep mode. An attempt is made in this thesis to keep the mobile node in the sleep mode as often

Transceiver Modes	GEC Plessey DE6003 2.4GHz radio	Lucent's 15dBm 2.4GHz Wavelan radio
Transmit	1.8W	1.725W
Receive	0.6W	1.475W
Sleep	0.05	0.08W

Table 1.2: Power consumed by the transceiver

as possible. Table 1.2 shows the typical values for the power consumed [10] by the transceiver circuitry in different modes.

Fading

The *channel fading* phenomenon occurs in practically all wireless environments involving radio links. *Fading* refers to changes in the amplitude of the received signals due to changes in the characteristics of the transmission path and motion of the transmitter and receiver. The received signal may be blocked due to the dielectric properties of the obstructing objects such as hills, buildings, etc. Fading observed in such scenarios is termed *shadow fading*. In some cases multiple copies of the same signal are received at different times due to different paths being followed by the signal. If the signals received are displaced by an integer number of wavelengths they tend to add together and reinforce the signal, while if they are a half-integer number of wavelengths apart they tend to cancel and produce a fade, which is called *small scale (multi-path) fading*. Periods of perceived connection (error free state) and disconnection (error state) are observed due to the fading characteristics in a wireless network. The fading properties of wireless networks distinguish them from wired networks. Another factor that adds to the fading characteristics is the user mobility. Since channel fading is a location-dependent phenomenon, protocols developed for such systems should account for user mobility and the location dependent nature of channel fading.

In this thesis an attempt is made to provide a MAC protocol for the mobile wireless environment which does not assume advance knowledge of location dependent channel fading.

1.3 Guaranteed Service Scheduling

Applications that require a strict bound on performance by demanding a guarantee of service quality are called *guaranteed service applications.* Scheduling disciplines that satisfy these applications are called *guaranteed service scheduling-schemes.* Four common performance parameters that are widely used to describe relevant guarantees are: *bandwidth*, *delay*, *delay-jitter*, and *loss*.

A bandwidth bound requires that a connection receives a minimum bandwidth from the network. A delay bound is a deterministic or statistical bound on some parameter of the delay distribution, such as the worst-case delay, the mean delay, or the 99-percentile delay. A delay jitter bound requires that the network provides a bound for the difference between the largest and smallest delays received by packets. Finally, the loss bound requires that the percentage of packets lost on a connection be bounded.

In general the guaranteed service scheduling schemes in wire-line networks determine the order in which the scheduler should serve packets from various connections. Two fundamental choices are to serve the packets in the order of their arrival, or to serve them out of order according to a per-packet *service tag.* The *First Come First Serve* (FCFS) algorithm use the former approach, and services the packets in the order of their arrival. FCFS restricts the scheduler from scheduling connections with varying performance parameters as it will not allow packets that demand a better QoS (for instance lower delay) to skip to the head of the queue. Thus various strategies have been adapted to identify a suitable service order that could provide service tags in accordance with the service quality and the performance parameter desired by the connections. In other words service tags should be assigned such that the performance guarantees of all the connections be fairly met according to their desired

QoS. For instance, weighted fair queueing (WFQ) [8, 21] uses a service tag called the *finish number* to identify the service order of packets from various connections. WFQ thus provides connections with guaranteed bandwidth and worst case delays. The worst case delay of WFQ over K series of connections all using WFQ is:

$$D_i = \frac{\sigma_i}{\rho_i} + \sum_{k=1}^{K-1} \frac{Pmax_k}{\rho_k} + \frac{Pmax}{C}, \tag{1.1}$$

where σ_i is the maximum burst size at connection i, ρ_i is the rate allocated to connection i, $Pmax_k$ is the maximum packet size of connection k, while $Pmax$ is the maximum packet size over all connections and C is the total channel capacity or the peak rate at which the scheduler can service the packets.

The packet loss rate is practically zero in fiber-optic networks. Thus all the wire-line fair queueing algorithms assume that there will be no losses due to channel error. Up-to-date information is available at the scheduler regarding the backlogs in wire-line networks, allowing them to determine the accurate service order of packets, thus meeting the desired performance bounds. However in the case of wireless networks, due to location-dependent channel fading, the channel [1] state is neither perfect nor instantaneously known. Up-to-date information regarding the backlogs is also not available at the scheduler. This restricts the direct usage of wire-line scheduling schemes in a wireless environment.

1.4 Thesis Motivation

1.4.1 Problem Statement

Figure 1.1 shows a typical mobile wireless environment considered in the scope of this thesis. Due to the presence of location dependent channel fading and subsequently the out-of-date information regarding the traffic, the following issues should be resolved in a mobile wireless environment, in order to provide guaranteed service:

[1] We use the term *channel* to describe the logical link between a MN and the BS.

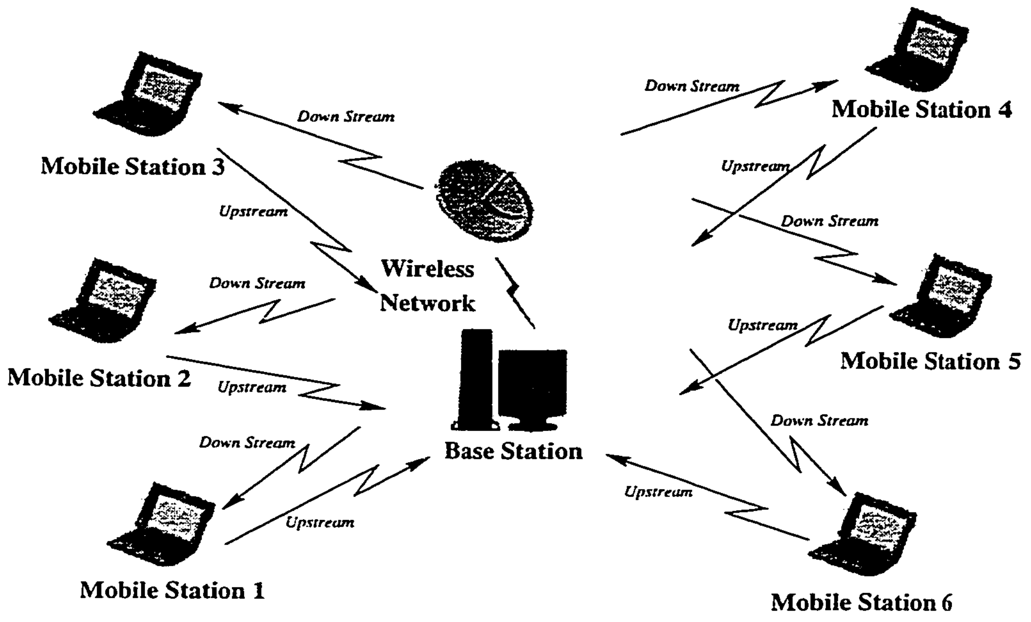

Figure 1.1: Mobile wireless environment

1. The high probability of packet error in a wireless environment necessitates the use of confirmations (acknowledgments) in order to identify whether or not the transmission was successful, thus reducing the maximum efficiency of the channel. Therefore all guaranteed service scheduling algorithms for the mobile wireless environment should account for the acknowledgment of each packet.

2. If an MN perceives channel errors for a long period, it might not get the QoS it was guaranteed at the time of connection establishment. Fairness in the wireless environment is thus redefined. We rename wire-line fairness as *short-term fairness*, and *long-term fairness* as wire-line fairness provided in the long run or over a period of time. Since there is no way to instantly provide the amount of service lost during the period that the channel was in the error state, *short-term fairness* for connections perceiving channel error is not possible. In the mobile wireless environment, we can achieve short-term fairness for error-free connections only. For MNs perceiving channel errors, *long-term fairness* can be achieved by providing a bound on the time it takes to recover from the

service lost during the channel error.

3. Channel fading is a location dependent phenomenon. In a mobile environment *a-priori* knowledge of channel state is not available. In order to 'learn' whether or not the channel is in the error state a MN has to wait for acknowledgments, thus consuming more power and wasting bandwidth.

4. In order to conserve energy at the MNs, the BS station is entrusted with all the complex and power consuming operations, such as scheduling and handoff. However, a BS does not have up-to-date information regarding the backlogs at the MNs, thus creating a need for frequent *polling* in order to update the information at the BS.

Considering these issues, we propose a MAC protocol which uses a TDMA/TDD scheme for mobile wireless environment. The BS accounts for channel fading without assuming a-priori knowledge of channel state. It also accounts for the bandwidth lost during polling, receiving acknowledgments, and while identifying the channel state. The protocol reduces the energy utilization at the MNs while ensuring long-term fairness for error prone MNs, and short-term fairness for error-free MNs.

1.4.2 Previous Work

In this section we report on the literature in the areas of WFQ for wireless links and power conserving MAC protocols.

Wireless Fair Queueing

Various protocols have been proposed for mobile wireless computing [22, 23]. In order to cope with the channel fading phenomena, the proposed schemes operate through a direct characterization of the channel between a base and a mobile, or between mobiles, as in either the *'error free'* or *'error'* state. Successful data transmission is only possible while the corresponding channel is in the 'error free' state. The proposed schemes assume that errors are location-dependent. This fact is exploited by communicating with mobiles for which the channels is known to be in the error free state, instead of the mobiles for which the channels is in the error state. A separate model captures the amount of excess service received (*lead*) by the mobiles with error free channels, at the cost of service lost (*lag*) by mobiles with channels in the error state. Mobiles with lead are termed *leading*, while the mobiles with lag are termed *lagging*. Lagging mobiles are later compensated when the leading mobiles, experience error free channels state.

Some of the assumptions made in the proposed schemes, in particular the assumption that the success of a data transmission is known *a-priori*, is clearly *unrealistic*. Any practical implementation of a fair access protocol will be unable to accurately determine whether the state of the channel is error free by any means other than actually attempting to make use of the channel through data transmission. Even the ability to detect the presence of noise on the channel is only evidence, but not a guarantee, that a data transmission will fail. Moreover, the potential delay between detecting the state of the channel and making use of the state information may be sufficient for the state of the channel to change, possibly several times. Therefore, the state of the channel must be probed through attempted transmissions. However, in previous proposals transmissions are scheduled only if the channel is determined to be in the 'error free' state. Hence, a chicken-and-egg problem arises that is ignored by the current literature. Some part of the allocated bandwidth may be wasted due to transmissions made to determine the channel state. In a guaranteed service environment, the bandwidth must not be wasted at the expense of the guarantees provided to the mobiles during call admission. In other words, the scheduler and the CAC (Connection Admission Control) algorithms should also account for the probing overhead.

Power Conservation

Various MAC protocols have been proposed for the mobile wireless computing environment [24, 17, 25] in order to conserve power at the mobile nodes. A common strategy employed is to reduce the number of re-transmissions due to collisions. Thus, almost all of the proposed protocols use some form of collision avoidance. Hence, the MAC protocols of wire-line networks using random access mechanism are ruled out because of the high probability of collisions at moderate loads. At the same time these protocols do not provide service guarantees.

Some protocols use contention techniques to avoid collisions. They require each mobile node to listen to the wireless medium before transmitting. Each transmitting node is required to transmit a signal (RTS) indicating the duration for which it will be using the channel. All the other mobile nodes willing to transmit during that period will back-off, thus avoiding possible collisions. Yet, the per packet overheads incurred due to the RTS and the fact that one has to listen before transmitting a packet imposes a burden on the mobile nodes in terms of power consumption. Further, the power consumption in such protocols is a function of the traffic load of the other mobile node(s).

Another approach used is to allow the mobile nodes to send transmission requests during a certain request for access (RA) slot. Subsequently, scheduling is done on a slot-by-slot basis and an explicit announcement at the beginning of each slot is made to identify the owner for that slot. The explicit slot-by-slot announcement allows the BS to implement a collisions free, 'optimal', and 'just in time' scheduling. However, an additional burden is placed on the receiver subsystem of the mobiles to receive and decode the announcements during every slot.

Another dimension of the problem arises from the fact that probing the channel to determine the channel state can be a wasteful operation in terms of power consumed at the mobile. Because they simply assume a-priori knowledge of the channel state, none of the proposed protocols account for the power used while probing the channel state.

The protocol proposed in this thesis, PCFQ, replaces a-priori with *post-facto* channel state recognition. In doing so, it accepts, but limits, the extent to which the transmission capacity will be wasted in determining the state of the channel. PCFQ is a low power consumption protocol in which the BS assigns the slots in TDM frames based on backlog information that is locally available for the downstream direction, or information that is collected for the upstream direction through polling. The overheads due to polling in PCFQ are limited to be per frame rather than being per packet. PCFQ allows the error-prone mobiles to recover more quickly by using the spare capacity which is either currently unallocated or was allocated to nodes that are currently in the 'error' state.

PCFQ achieves *short term* as well as *long term* fairness for mobile nodes which never experience errors. *Long term* fairness can be achieved for error prone nodes provided that the CAC algorithm leaves enough spare bandwidth to account for bandwidth lost due to bad channel states. Using PCFQ, the mobile remains in sleep mode most of the time, in order to conserve energy.

1.5 Thesis Outline

The structure of the thesis is as follows. Chapter 2 is a brief presentation of fair queueing algorithms in wire-line and wireless networks. Chapter 3 contains a literature survey regarding the research performed in the area of power conservation. In Chapter 4 a detailed description of PCFQ is presented. Chapter 5 and 6 contain operational and implementation issues, results, and a few related theorems and finally in Chapter 7, conclusions and scope of future work are provided.

Chapter 2

Guaranteed Service Scheduling

Given a set of packet arrivals in the service queue, a server uses a *scheduling discipline* to decide which request to serve next. Scheduling disciplines are important because they are the key to *sharing* network resources *fairly* and to providing performance-critical applications, such as telephony and interactive multi-participant games, with *performance guarantees.* A scheduling discipline has the following two orthogonal components: deciding the order in which requests are to be serviced and managing the queue of requests awaiting service. To cater to these components, guaranteed scheduling disciplines must fulfill the following four, sometimes contradictory, requirements:

Ease of implementation

In a high-speed network, a server may need to pick the next packet for transmission every time a packet departs, which can be once every few milliseconds or less. Thus a scheduling discipline for high-speed network should require only a few simple operations; preferably, it should be implementable inexpensively in hardware.

Performance bound

Another major requirement of a scheduling discipline is to provide arbitrary per-connection *performance bounds*, restricted only by the network resources. Performance bounds can be expressed either *deterministically* or *statistically.* Deterministic bounds hold for every packet of the connection, while a statistical bound is a probabilistic bound on performance of the connection over a period of time.

Ease and efficiency of admission control

A switch controller should be able to decide, given the current set of connections and the traffic descriptors for a new connection, whether it is possible to meet the new connection's performance requirements without jeopardizing the performance of existing connections. This mechanism is called *admission control.* A scheduling discipline is expected to permit easy admission control and at the same time utilize the network efficiently.

Fairness and protection

A scheduling discipline allocates a share of the link capacity and buffers to each connection it serves. Ideally, allocation is called *fair* if it satisfies the following inequality stated by one of the many *fairness queuing* algorithms.

$$\left|\frac{W_i(t_1,t_2)}{\rho_i}\right| = \left|\frac{W_j(t_1,t_2)}{\rho_j}\right| , \tag{2.1}$$

where nodes i and j are assumed to be continually backlogged over a real time interval $[t_1, t_2]$, and $W_i(t_1, t_2)$ and $W_j(t_1, t_2)$ denote the services received (in bits) by nodes i and j respectively.

Fair resource allocation to a set of connections is a *global* objective, while a scheduling discipline takes only *local* resource allocation decisions. Detailed descriptions of some key fair queueing algorithms for wired networks will be presented in the next section.

Protection means that misbehavior by one connection by sending packets at a higher rate than its fair share should not affect the performance of other connections. For example, first-come-first-served (FCFS) queueing does not provide protection, because the mean delay of a source may increase if the sum of the arrival rates over all sources increases. Thus a misbehaving source, by sending too quickly, increases the mean delay of all other connections. In short a fair scheduler automatically provides protection, because it limits the misbehaving connection to its fair share. However the converse may not be true. That is, if the scheduler provides protection to any connection, it may or may not be fair.

2.1 Wire-line Fair Queueing Algorithms

Many fair queueing algorithms have been proposed for the wire-line computing environment. This section describes some of the more popular ones.

2.1.1 Max-min Fair Share

Max-min fair share is one common technique for sharing scarce resources among a given set of users *fairly*, with possibly varying demands. The fairness in the max-min fair share queueing algorithm is defined by allocating the user with a small demand for a resource the amount it requests, and evenly distributing the unused resources to the users with large demands. In other words,

- Resources are allocated in increasing order of demand;
- No source is allocated more than its demand;
- Sources with unsatisfied demands get an equal share of the leftover resource.

Let us consider a set of sources $1, ..., n$ that have resource demands $d_1, d_2, ..., d_n$, such that $d_1 \leq d_2 ... \leq d_n$. Let C be the total server capacity. Then the source with least demand d_1, is given $\frac{C}{n}$ of the resource. If $d_i < \frac{C}{n}$, then the excess resource, $(\frac{C}{n} - d_i)$ is evenly distributed amongst the remaining $(n-1)$ sources. Thus each of them gets an extra share of $(\frac{C}{n} - d_i)/(n-1)$. This way any excess resource is equally distributed among the remaining unsatisfied sources. Such an allocation is called a *max-min fair* allocation, as it maximizes the minimum share of a source whose demand is not fully satisfied.

Very often sources have priorities attached to them identifying their rights to seek resources. In such a case weights $w_1, w_2, ..., w_n$ are associated with the sources $1, 2, ..., n$, which reflect their relative resource share. The concept of max-min fair share extended to include such weights is called *max-min weighted fair share allocation*. This is accomplished as follows:

- Resources are allocated in order of increasing demand, normalized by the weights for each source.
- No source gets a resource share larger than its demand.
- Sources with unsatisfied demands get resource shares in proportion to their weights.

The max-min fair allocation can be achieved with an *ideal* and un-implementable work-conserving scheduling discipline called *Generalized Processor Sharing (GPS)* [26]. GPS serves packets as if they are in separate logical queues, visiting each non-empty queue in turn and serving an infinitesimally small amount of data from each queue. Thus in any finite interval, GPS visits all logical queues atleast once. In the case of weighted connections, sources are served in proportion to their weights whenever there is any data in their queues. If a queue is empty, the scheduler skips to the next non-empty queue, thus achieving the max-min fair share allocation.

2.1.2 Weighted Fair Queueing (WFQ)

Weighted fair queueing, like *Packet-by-packet Generalized Processor Sharing (PGPS)* [8, 21] is an approximation of GPS scheduling. Unlike GPS, packet size is not infinitesimally small, and in the case of variable packet length, it does not need to know the connection's mean packet size in advance. WFQ computes the time a packet would complete service had we been serving packets with a GPS server, and then serves the packets in order of these finishing times. WFQ simulates GPS "on the side" and uses the results of this simulation to determine service order. A packet's finishing time under GPS is called its *finish number* rather than finish time, to emphasize that it is only a service tag indicating the relative order in which the packet is to be served. In order to calculate the finish number for a packet, WFQ makes use of another variable called the *round number*. The round number is the number of rounds of service a bit-by-bit (or packet-by-packet) round-robin scheduler has completed at a given time. In general, the finish number is calculated as follows

$$F(i,k,t) = max[F(i,k-1,t), R(t)] + \frac{P(i,k,t)}{\phi(i)}\ , \tag{2.2}$$

where $F(i,k,t)$ is the finish number for the kth packet that arrives on connection i at time t, $F(i,k-1,t)$ is the finish number for the $(k-1)$th packet, $R(t)$ is the current round number, $P(i,k,t)$ is the size of kth packet, and $\phi(i)$ is the rate weight allocated to connection i.

Weighted fair queueing (or PGPS) thus has following three desirable properties:

- By approximating GPS, it protects connections from each other.
- It provides a worst-case end-to-end queueing delay for connections irrespective of other connections and the number of hops traversed and provides real-time performance guarantees for the connections.
- WFQ yields better results for sources with intelligent flow control mechanisms and tends to drop packets for sources that send data at a rate consistently higher than their fair-share.

2.1.3 Start-time Fair Queueing (SFQ)

The major drawback in the implementation of WFQ is the updating of round numbers on packet arrival, as it might lead to *iterative deletion* of connections [27]. This problem arises when a connection becomes inactive (deleted from the list of active connections) thus increasing the effective rate of service for other active connections. This increase in the rate of service for active connections might enable them to become inactive earlier, which in turn further increases the rate of service for the rest of the active connections and thus produces a cascading effect. This cascading effect leads to completion of more rounds than estimated initially, and thereby complicates correct estimation of the current round number.

Self Clocked Fair Queueing (SCFQ) is an algorithm which speeds up the computation of round numbers. In SCFQ, on arrival of a new packet, the finish number of the packet *currently in service* is used instead of using the round number to compute the finish number. Thus the round number updates (as in Equation 2.2 for WFQ) will be performed as follows

$$F(i,k,t) = max[F(i,k-1,t), CF] + \frac{P(i,k,t)}{\phi(i)} \ , \tag{2.3}$$

where CF is the finish number of packet currently being served and $\phi(i)$ is the rate weight allocated to the connection i.

Although the SCFQ round number update rule is easy to implement, it is shown to be unfair [28] over short time scales. SCFQ also leads to larger worst-case latencies than WFQ, and, consequently, has a potential greater unfairness at shorter time scales. If the size of the largest packet allowed in the network is P_{max}, the total number of active connections are N, and the service rate of the scheduler is r, then the worst case latencies for SCFQ and WFQ will be

$$d_{SCFQ} = \frac{P_{max}}{\rho_i} + \frac{(N-1)P_{max}}{r} \ , \tag{2.4}$$

$$d_{WFQ} = \frac{P_{max}}{\rho_i} + \frac{P_{max}}{r}. \tag{2.5}$$

Another variant of SCFQ, called *Start Time Fair Queuing (SFQ)*, has the same computational benefits of SCFQ without being penalized for large worst-case delay. In SFQ, both the finish number and the *start number* of each packet are computed. The start number of the packet arriving at an inactive connection is initialized to the current round number. If the connection is active, the start number is the finish number of the previous packet. The finish number of packet is calculated as sum of its start number and its packet size divided by its allocated rate. The round number is set to the start number of the packet currently in service. If there are no packets to send, the round number is set to the largest of the finish numbers of any packet sent until that time. The scheduler services packets in increasing order of start number.

SFQ has an implementation complexity similar to SCFQ. Moreover, the worst case delay of SFQ is much lower than that of SCFQ. As shown in [27], the difference in the maximum delay SCFQ and SFQ, for a channel capacity C is

$$\frac{P_{max}}{\rho_i} - \frac{P_{max}}{C}, \tag{2.6}$$

2.1.4 Virtual Clock

The virtual clock scheduling discipline is another variant of WFQ. A virtual clock scheduler stamps packets with a tag, and packets are served in the order of their tags, as in WFQ. However, the tag values are not computed to emulate the GPS scheduling. Instead, a virtual clock scheduler emulates time-division multiplexing in the same way that a fair queueing scheduler emulates GPS. As in Equation 2.2 the finish number is thus computed as

$$F(i,k,t) = max[F(i,k,t), real\ time] + \frac{P(i,k,t)}{\phi(i)}. \tag{2.7}$$

Since the round number in WFQ was hard to calculate, real time is used instead of round number, thus simplifying the calculation of the virtual clock finish number. As shown in [29, 30], when all connections are backlogged, virtual clock and WFQ provide identical service and identical worst-case end-to-end delay bounds. However, it has been shown in [30], that for best-effort connections, the relative fairness bound for virtual clock is infinity, i.e:, if two connections are backlogged, one of them may receive infinitely more throughput than the other. As a result to implement one scheduler at a switch, WFQ may be better.

2.2 Limitations

The basic assumptions made in all the fair queueing algorithms discussed so far are:

- Perfect channel conditions (i.e., no bit errors).
- Up-to-date information at the scheduler regarding the backlog.

Both these assumptions fail to hold in wireless networks and therefore none of the fair queueing models discussed so far can be directly used in the wireless environment. The following examples further strengthen the argument that the channel fading phenomenon and out-of-date information at the scheduler in wireless networks violate the fairness and service guarantees of wire-line fair queueing algorithms.

Example 1 Channel fading. As shown in [2], consider three backlogged MNs during the time interval $[t_i, t_{i+2}]$ with equal rates, i.e: $r_1 = r_2 = r_3$. Assume that nodes 1 and 2 perceive an error-free channel while node 3 perceives channel errors during the interval $[t_i, t_{i+1}]$. Using the wire-line fair queueing algorithms such as WFQ, the channel capacity for the two intervals will be allocated as follows

$$S_1[t_i, t_{i+1}] = S_2[t_i, t_{i+1}] = \frac{1}{2},\ S_3[t_i, t_{i+1}] = 0\ , \tag{2.8}$$

$$S_1[t_{i+1}, t_{i+2}] = S_2[t_{i+1}, t_{i+2}] = S_3[t_{i+1}, t_{i+2}] = \frac{1}{3}\ , \tag{2.9}$$

$$S_1[t_i, t_{i+2}] = S_2[t_i, t_{i+2}] = \frac{5}{6},\ S_3[t_i, t_{i+2}] = \frac{1}{3}\ , \tag{2.10}$$

where $S_n[t_{i+j}, t_{i+k}]$ is the service received by node n during the $[t_{i+j}, t_{i+k}]$ interval.

Equation 2.10 shows that nodes S_1 and S_2 received more service during the time interval $[t_i, t_{i+2}]$ than the node S_3, thus violating the fairness property of fair queueing algorithms.

Example 2 (Out-of-date information) Consider a scenario where the information regarding the upstream traffic of the MN A is updated at the BS at time instants t_i and t_k, where $t_i < t_k$. Further, suppose a packet P_A is generated at node A, at time t_j, where $t_i < t_j < t_k$. Then the packet P_A will be not be considered for scheduling until the time instant t_k. Hence the delay values for packet P_A will be inflated by $t_k - t_j$ time units, which may even violate the guaranteed delay bounds.

2.3 Wireless Fair Queuing

Many wireless fair scheduling techniques have been proposed [31, 32, 22, 23]. A generic framework for wireless fair queuing along with the key components, alternative policies and mechanisms for each of them, will be presented in this section. Five main components [2] can be identified as *error-free service model*, *lead and lag model*, *compensation model*, *channel monitoring and prediction model*, and *slot queue and packet queue decoupling model*.

As shown in Figure 2.1, different components in the framework interact as follows The *error-free service model* is used as the service that each connection should receive. The *lead and lag model* specifies how much additional service any node has received due to the channel error state of others. It also keeps track of how much additional service a lagging node is eligible to receive, and how much service a leading node will have to relinquish in the future. The *compensation model* identifies the instant and the amount of service that a leading node should relinquish to any lagging node. It tries to make error periods transparent to lagging nodes while maintaining graceful degradation for the leading nodes. The *channel monitoring and prediction* model is used to determine whether or not a node perceives an error-free channel state during

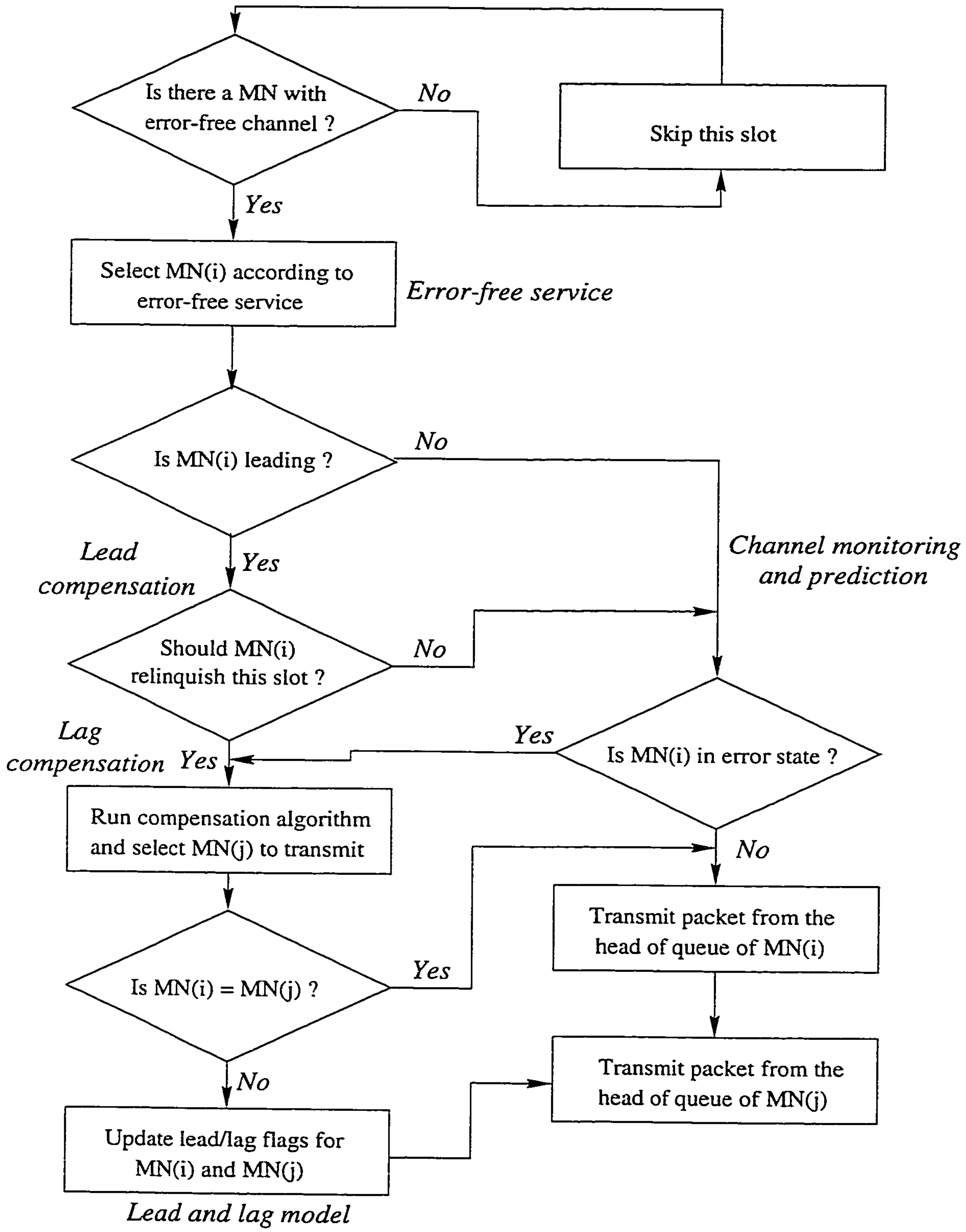
Is there a MN with error-free channel ?
No
Skip this slot
Yes
Select MN(i) according to error-free service
Error-free service
Is MN(i) leading ?
No
Lead compensation
Yes
Channel monitoring and prediction
Should MN(i) relinquish this slot ?
No
Lag compensation
Yes
Yes
Is MN(i) in error state ?
Run compensation algorithm and select MN(j) to transmit
No
Transmit packet from the head of queue of MN(i)
Is MN(i) = MN(j) ?
Yes
No
Transmit packet from the head of queue of MN(j)
Update lead/lag flags for MN(i) and MN(j)
Lead and lag model

Figure 2.1: Framework model for wireless fair queuing [2]

a slot. It is thus coupled with the compensation model to identify the instants when a lagging node will start receiving the additional service. Finally, in cases where issues related to delay-sensitive versus error-sensitive connections have to be dealt with, the *slot queue and packet queue decoupling* model is used. Slots in the slot queue denote the slots for nodes and not the specific packets for the nodes. This allows another level of abstraction to accommodate applications with varying performance parameters.

Error-free Service Model

This model assumes that there are no channel errors and defines an *ideal* fair queueing service model. It provides a reference for how much service a node should receive in an ideal error-free environment. Most contemporary wireless fair queueing algorithms use well known wire-line fair queueing algorithms, as described in the previous chapter, for their error free service model. The wire-line fair queueing algorithms that have been used are:

- wire-line fair queuing algorithms such as WFQ or PGPS [8, 21], in which the rate of change of virtual time (dV/dt) is explicitly simulated. Idealized Wireless Fair Queueing (IWFQ) [22] uses WFQ or Worst-case Fair Weighted Fair Queueing (WF2Q) [33] as its error-free service model.

- wire-line fair queueing algorithms such as Start-time Fair Queueing (SFQ) [27], in which the virtual time is not explicitly simulated. In SFQ, the virtual time is set to the start time of the packet that is currently being served. Channel-condition Independent Fair Queueing (CIF-Q) [23], uses SFQ as its error-free service model.

- a variation of fluid fair queueing that allows decoupling of delay and rate in the scheduler is used by Wireless Fair Service (WFS) [32]. This is achieved by allocating a rate weight r_i and a delay weight d_i for each node i, and modifying the tagging mechanism as follows:

$$S(p_i^k) = max[V(A(p_i^k)), S(p_i^{k-1}) + \frac{L_i^{k-1}}{r_i}] \,, \tag{2.11}$$

$$F(p_i^k) = S(p_i^k) + \frac{L_i^k}{d_i} \,, \tag{2.12}$$

where $S(p_i^k)$, $V(t)$, and $F(p_i^k)$ stand for the start time tag, virtual time tag and finish time tag for packet k of node i. $A(p_i^k)$ is the arrival time of packet k at node i.

Lead and Lag Model

Lagging nodes are those nodes that have received channel allocations less than their error-free service allocation. Likewise, *leading* nodes are those nodes that have received channel allocations in excess of their error-free service allocation at the cost of lagging nodes. Nodes which receive a channel allocation equal to their error-free service allocation are referred as *in-sync* nodes. The *lag* of lagging nodes denotes the amount of additional service to which a node is entitled in the future so as to compensate for the lost service in past, while the *lead* of leading nodes is the amount of service that the node must relinquish in the future so as to compensate for the additional service it received in the past. Two distinct approaches used to compute the lag and lead are:

- The difference between the error-free service and actual service received by a MN is called its lag or lead. In this approach the lagging node is compensated for its lost service irrespective of whether or not its lost slots were utilized by other nodes. For instance, assume that MN i perceives channel errors while all other nodes that perceive an error free channel state have zero backlog. In this case, although node i has lost its service, no other node has taken advantage of it. The approach of compensating even for such MN i, is used by the Server Based Fairness Approach (SBFA) [31].

- Under the second approach, the lag of a node is the number of slots allocated to the node during which it could not transmit due to channel error, and another backlogged node that had no channel error transmitted in its place and increased its lead. Thus, the lag of a node is incremented upon a lost slot only if another node is prepared to relinquish a slot in the future. In other words, in this type of system the sum of lead and lag over all MNs in a cell at all instants will be zero. IWFQ [22], WFS [32], and CIF-Q [23] follow this approach.

A common strategy of bounding the lead and lag parameters has been used by most of the wireless fair queueing algorithms. The upper bounds for the lead and lag parameters are determined using node-specific parameters. Bounding lag would thus limit the maximum error burst that can be made transparent to the node. Likewise an upper bound on lead limits the maximum number of slots a leading node must relinquish in the future in order to compensate for additional service it received in the past.

Compensation Model

To establish long term fairness for lagging MNs, it is necessary to compensate for their lost service while the channel was in the error state. This model thus identifies how the lagging mobile nodes that now perceive an error free channel will be compensated. In other words the compensation model makes error periods transparent to lagging nodes, while maintaining a graceful degradation for the leading nodes. It thus determines the way a lagging node makes up for its lag and the way a leading node gives up its lead. Following are the three main issues addressed by this model:

Compensation style

It determines how a leading node relinquishes the slots to lagging nodes. There are three possible options for the leading node to relinquish lead:

- As in IWFQ [22], a leading node relinquishes all slots until it becomes in-sync with its error-free service. This implies that any node that has accumulated a large lead may end up being starved at a later stage while relinquishing slots for lagging nodes.

- Leading nodes may also be made to relinquish a fraction of their lead. This fraction again may either be a constant as in CIF-Q [23], or a function of lead as in WFS [32]. This implies that the degradation in service of leading nodes will be graceful.

- Yet another approach is to reserve a portion of channel bandwidth for compensating the lagging nodes. This implies that leading nodes will never be forced to relinquish their lead. This approach is used in SBFA [31].

Compensation instants

It identifies the instants when a lagging node will be compensated for its lost slots. There are three options for allocating these *compensating slots* to the lagging nodes:

- In the first approach, the lagging nodes are given a priority over the in-sync and leading nodes. All compensating slots are preferentially allocated to the lagging nodes, as in IWFQ [22]. This preferential treatment of lagging nodes may starve the in-sync and leading nodes, thus adversely affecting their service.

- Another approach, as used in CIF-Q [23] and WFS [32], is to compensate the lagging node only when the leading nodes relinquish slots. This approach ensures that the lagging node does not adversely affect the in-sync nodes.

- Lastly, slots for compensating the lagging nodes may be allocated from the reserved fraction of channel bandwidth set aside for them, as in SBFA [31].

Amount of compensation

In order to fairly distribute the compensating slots among lagging nodes three design choices have been explored in the contemporary algorithms.

- Compensation slots are allotted to lagging nodes in the increasing order of their lag, as in CIF-Q [23].
- Lagging nodes are compensated by the order in which they became backlogged or lagging. This approach was used in IWFQ [22] and SBFA [31].
- Lagging nodes are compensated fairly, in proportion to their lag as in WFS [32].

Channel monitoring and prediction

The limited knowledge of the channel state at any instant makes perfect channel-dependent scheduling impossible. It has been suggested that all nodes should monitor their channel state continuously, so as to be able to predict their channel state at any time instant. This channel state information should then be transferred to the scheduler for better channel-dependent scheduling. This model tries to provide an approximate estimation of the channel states at any time instant for every backlogged MN.

Most contemporary algorithms use a Markov model to predict the channel state in the future, as in WFS [32] and CIF-Q [23]. They assume an upper bound on the number of errors during any time interval. The delay and throughput properties derived for the wireless fair queueing algorithms so far are typically *channel-conditioned*. That is, they are conditioned on the fact that node i perceives no more than n errors in any time interval T_i.

Slot queues and packet queues

Unlike wire-line fair queueing algorithms, in a wireless channel the number of retransmissions due to channel errors imposes a need for separation between, "when to send the next packet" and "which packet to send next". The former is decided by the scheduler while the latter may be application specific. For example, which packet to send next may resolve to different choices, depending on whether the connection is error-sensitive or delay-sensitive. Thus in order to have an additional level of abstraction, a *slot* is used as a unit of channel allocation, while a *packet* is used as a unit of data transmission. Thus, when a packet arrives at the node queue, a corresponding slot is generated in the *slot queue* of the node, and tagged according to the wireless fair queueing algorithm. This limits the scope of the scheduler as it need not worry about which packet the node wishes to transmit. The approach of segregating error-sensitive and delay-sensitive applications is used in WFS [32].

2.3.1 Idealized Wireless Fair Queueing (IWFQ)

The error-free service in IWFQ [22] is simulated using WFQ [8, 21] or WF2Q [33]. The start time tag and finish time tag are assigned based on WFQ. The service tag of an arriving packet is set to the finish tag of the packet at the head of its queue. The scheduler then selects the node with the minimum service tag among the backlogged nodes that perceive an error-free channel state. Each node i has a lead bound of L_i and a lag bound of B_i. In other words, the service tag of node i is not allowed to be more than L_i above, or be more than B_i below, the service tag of its error-free service.

If a node perceives a channel error, it retains its service tag. Likewise, if a node receives additional service, its service tag increases. Consequently, the service tag of the lagging nodes falls behind the tags of the in-sync or leading nodes. Since the scheduler selects nodes in increasing order of service tags, lagging nodes will receive preferential treatment over the in-sync and leading nodes. Thus the compensation model is implicit in the algorithm.

A node lagging for a long time would thus be able to seize the channel as soon as it becomes error-free. This can potentially create a scenario where the leading nodes and sometimes even the in-sync nodes would be starved for channel access for a long period. Thus the IWFQ model does not support graceful degradation of service for leading nodes. IWFQ is also computationally expensive, since it uses WFQ to simulate the error-free service model, and as stated in the previous chapter, it is difficult to compute the round number for WFQ.

2.3.2 Channel-condition Independent Fair Queueing (CIF-Q)

In CIF-Q [23], the error-free service is simulated using SFQ [27]. Each node has a flag called the *lag* parameter which distinguishes between leading, lagging and in-sync nodes. If the lag is positive the node is lagging, if it is negative the node is leading and if it is zero the node is in-sync. When a lagging or in-sync node i perceiving an error-free channel is allocated the channel, it transmits a packet. Otherwise, a backlogged node j which perceives an error-free channel state is allowed to transmit instead of node i. Following that, the lag of node i is incremented and the lag of node j is decremented. Thus at any instant sum of lag over all nodes in the system is zero.

For the compensation model, the leading node i retains a fraction α of its service, and relinquishes $(1 - \alpha)$ for lagging nodes. The parameter α is governs the service degradation of leading nodes. Compensation slots are allocated in order of decreasing lag for the backlogged lagging nodes.

Depending on the value of α, CIF-Q provides a linear degradation in service of leading nodes. It also performs compensation of lagging nodes by preferentially swapping slots with leading nodes thus ensuring that in-sync nodes are not significantly affected. This algorithm can be considered as an improvement over IWFQ for two reasons: CIF-Q provides a graceful degradation of leading nodes without affecting the in-sync nodes, and it is less complex as it uses SFQ instead of WFQ for its error-free service model.

CIF-Q, however, unrealistically assumes a-priori knowledge of the channel state. Thus it does not account for any potential delay and probing overheads required to determine the channel state. The error model used by CIF-Q assumes that channel error is perceived for a fixed duration (for the first 45 seconds in a 200 second simulation). Following that the channel is believed to perceive an error-free channel long enough to clear the accumulated backlog thus providing long-term fairness to error-prone nodes. However, such error models do not capture the general channel behavior in a practical implementation.

2.3.3 Server Based Fairness Approach (SBFA)

In SBFA [31], a framework for using different wire-line scheduling algorithms in the wireless domain is provided. Any wire-line scheduling algorithm that needs to be adapted to the wireless domain can be used as an error-free service model for SBFA. In SBFA a fraction of bandwidth is reserved for the compensation of lagging nodes. Thus, unlike other wireless scheduling algorithms, it tries to compensate the lagging nodes using *reserved* bandwidth.

A long-term fairness server (LTFS) is created in SBFA to provide compensation. LTFS is allocated a rate which reflects the amount of bandwidth reserved for compensation. SBFA does not explicitly identify any leading node, and does not explicitly bound the lag of any node. The order of compensation among lagging nodes is the order in which the packets are queued in the LTFS.

SBFA provides long-term fairness, but not short-term fairness or worst-case delay bounds. The rate of compensation is bounded by the amount of bandwidth reserved for compensation. In-sync nodes may be affected and the service degradation of leading and in-sync nodes is not graceful. The available service is also bounded below by the minimum service contracts established during connection admission.

2.3.4 Wireless Fair Service (WFS)

In WFS [32], the error-free service model is computed using a modified fluid fair queuing algorithm. Unlike other scheduling algorithms, WFS can provide nodes with stringent bandwidth and delay bounds.

Each node i has a lead bound L_i^{max} and a lag bound B_i^{max}. A node with lead of L_i relinquishes a fraction L_i/L_i^{max} of its slots, while a node with lag of B_i receives a fraction $B_i / \Sigma_{j \epsilon S} B_j$ of all the relinquished slots, where S is the set of backlogged nodes. Leading nodes relinquish slots in proportion to their lead and relinquished slots are fairly distributed amongst the lagging nodes.

This implies that the service degradation is graceful for leading nodes and the fraction of slots relinquished by the leading nodes decreases exponentially with lead. The compensation model also ensures that the compensation among the lagging nodes is fair. WFS thus achieves both short-term and long-term fairness, as well as providing delay and throughput bounds. The error-free service model of WFS allows it to decouple delay and bandwidth requirements as well. WFS can be considered as an improvement over the CIF-Q algorithm, as it compensates the lagging flows fairly and provides slot queue and packet queue decoupling.

Like CIF-Q, WFS also assumes a-priori knowledge of the channel state and it does not account for any potential delay and probing overheads required to determine the channel state. However, as stated earlier, any practical implementation of a fair access protocol will be unable to accurately determine the channel state by any means other than actually attempting to make use of the channel through data transmission. Hence the algorithm should account for probing overheads involved in determining the channel state accurately.

2.4 A New Fair Queuing Algorithm (PCFQ)

Though many wireless fair queueing algorithms have been proposed, none seem to be capable of resolving all the issues associated with wireless mobile communication subject to location-dependent errors. In this section an attempt is made to highlight the key parameters involved in the design of the PCFQ queueing algorithm.

2.4.1 Error-free Service Model

All previous algorithms invariably use one of the popular wire-line fair queueing algorithms as their *ideal* error-free service model. Because the computation of the round number in SFQ is faster than in WFQ, we use SFQ as the error-free service model in the protocol (PCFQ) proposed in this thesis. A start time termed the generation (t_G), is attached to every packet. Based on the rate (ρ_i), allocated to node i, a finish time, referred to as virtual time (V_t), is calculated as follows:

$$V_{t_K} = max[V_{t_{K-1}}, t_G + \frac{P_D}{\rho_i}] \, , \tag{2.13}$$

where V_{t_K} and $V_{t_{K-1}}$ are the virtual times for the kth and $(k-1)$th packet, P_D is the length of data packets.

2.4.2 Lead and Lag Model

None of the algorithms discussed earlier actually account for the bandwidth used to determine the state of the channel. Most of them [32, 22, 23] assume instant knowledge of channel errors and thus end up having a situation where another node j, perceiving an error-free channel can utilize the channel, thus gaining lead. This lead is then later relinquished for the lagging nodes, and thus at any instant of time the sum of lead and lag over all nodes is zero. This is un-realistic as there is no instant knowledge of the channel state available. PCFQ, on the other hand, accounts for the bandwidth used in determining the actual channel state.

There is no explicit concept of *lead* and *lag* model in our protocol. But it still captures the idea in the form of *recovering* and *normalized* nodes. Any node with an error-free channel state, having the virtual time of the packet at the head of the queue greater than the virtual time for the corresponding packet in the error-free model, is called a *recovering* node. All the nodes that are not recovering and perceive an error-free channel state are called *normalized* nodes. In other words, any node trying to recover from the loss as suffered due to channel errors is referred to as a recovering node. Together, the recovering nodes and normalized nodes are called the *active nodes*, while the nodes perceiving channel errors are treated as temporarily inactive nodes (although their backlog builds up). Recovering set denoted by $\hat{R}(t)$ constitutes of all recovering nodes, normalized set denoted by $\hat{N}(t)$ constitutes of all normalized nodes, and active set denoted by $\hat{A}(t)$ constitutes of all recovering and normalized nodes (i.e., $\hat{A}(t) = \hat{N}(t) \cup \hat{R}(t)$).

A flag is set each time a node loses its share of service. This flag is used to identify nodes that have lost service in the past due to errors in the channel. If the node develops a backlog by the time the channel becomes error-free, the node must recover, and the modified rate determined by the compensation model by which it recovers is called its *recovery rate.* When the backlog of a recovering node drops to zero, it is again regarded as a normalized node, and the time that it takes to revert back to the normalized stage is called its *recovery time.* Thus PCFQ does not keep track of the actual lead and lag a node has acquired.

2.4.3 Compensation Model

As in IWFQ [22], the compensation model is implicit in the PCFQ protocol. When a nodes starts to recover, its virtual time tag would be smaller than the virtual time tag of normalized nodes. The virtual time of a recovering node is then modified as follows:

$$V_t = max[V_t, min_{i\epsilon\hat{A}(t)}(V_{t_i})] \ , \tag{2.14}$$

where $\hat{A}(t)$ denotes the set of active MNs.

This ensures that the recovering node does not starve the normalized nodes, which unlike in IWFQ, continue to get their fair share of the service. The amount of compensation is also implicit in PCFQ. Packets are scheduled in the increasing order of virtual time until either all the packets from the active nodes have been scheduled or the maximum limit for data transmission slots in a cycle has been reached. This implies that all the transmission slots left over by the normalized nodes will be filled by the recovering nodes allowing them to recover. If there is more than one recovering node, slots will again be allocated on the basis of the virtual time. Since virtual time is a function of the corresponding rate of the node, the number of slots will be proportional to the rate that has been allocated to the node during call setup. In short, recovery rate for node i is equal to the sum of its admitted rate and a fraction of the available excess bandwidth. Available excess bandwidth denotes the amount of bandwidth left over after accounting for overheads involved and allocating guaranteed rates to all MNs in the active set. The rate of recovery ρ_i^R, for any node i, is thus *re-allocated* for each cycle based on number of connections in the active $(\hat{A}(t))$ set, and is represented mathematically as follows:

$$\rho_i^R = \rho_i^0 + \left[\frac{C - (O_C + \sum_{k \epsilon \hat{A}(t)} \rho_k^0)}{\sum_{k \epsilon \hat{R}(t)} \rho_k^0} \right] * \rho_i^0 \ , \tag{2.15}$$

where ρ_i^R is the recovery rate for node i, C is the channel capacity, $\hat{A}(t)$ denotes the set of active nodes, $\hat{R}(t)$ denotes the set of nodes in the recovery state, ρ_k^0 is the rate allocated to node k at the time of call admission, and O_C is the overall overhead involved due to cycle. In short, recovery rate is equal to the sum of the rate allocated to the node during call admission and a fraction of excess bandwidth. Excess bandwidth or the bandwidth available after allocating all nodes in active set is distributed between the recovering nodes in proportion of their allocated rates.

Because of the finite buffer space Q_{max}^i for the ith flow and assuming a (σ, ρ) source model, the total traffic to be serviced during the recovery interval R_i is no more than $Q_{max}^i + \sigma_i + (\rho_i^0 * R_i)$. Assuming a bandwidth allocation of ρ_i^0 for the ith flow and the worse case with respect to recovery time (i.e., all other flows are recovering), the recovery time can be found to be:

$$R_i = \frac{Q_{Max}^i + \sigma_i}{(\rho_i^R - \rho_i^0)} \ , \tag{2.16}$$

where Q_{Max}^i is the maximum buffer size, ρ_i^R is the worst case recovery rate (i.e. when all the nodes are recovering or $|\hat{N}(t)| = |\hat{E}(t)| = 0$), σ_i is the maximum burst size, and ρ_i^0 is guaranteed rate of the recovering node i.

Notice that the bound given by Equation 2.16 can be observed only if the flow does not experience any further error state while recovering. Moreover, the recovery time can be shorter if arrivals find the buffer full and are subsequently discarded. Equation 2.16 assumes no loss, in order to provide an upper bound regardless of the exact timing of arrivals.

Bounding the maximum backlog at a node introduces the possibility of packet *losses* in our scheme. In other words, any new packet that arrives at a node with a backlog of Q_{Max} will be dropped. The possibility of packet loss exists in the previous queueing algorithms such as IWFQ, CIF-Q, and WFS as well. Like PCFQ, IWFQ also proposes a limit on the maximum backlog that can accumulated at any node. The error model of CIF-Q assumes that a channel experiences errors for a certain fixed period (for the first 45 second in a 200 seconds simulation period). CIF-Q does not explicitly bound the maximum backlog, but uses an error model which indirectly restricts the amount backlog at any node by limiting the error period. In WFS, the lag of each node is bounded to a maximum value, B_i^{max}, which in turn is equivalent to a bound on the maximum backlog a node could perceive.

2.4.4 Channel Monitoring and Prediction

Since neither the BS nor the MN has instantaneous knowledge of the channel state, unlike IWFQ, CIF-Q and WFS, PCFQ employs a more realistic approach of channel monitoring. Both the BS and the MN realize that channel errors have occurred when they do not receive the data they were supposed to according to the schedule. When the channel for a mobile has been detected to be in error, the scheduler does not schedule any event for that mobile until a successful polling event for that mobile takes place. A successful poll identifies that the channel has recovered from the error state. In short, no prediction mechanism is used to predict channel errors, which in turn implies that PCFQ can be used on any random channel error model.

2.4.5 Slot Queue and Packet Queue Decoupling

PCFQ does not explicitly decouple the slot queue and packet queue, but an approach similar to that in WFS [32] can be applied to achieve slot queue and packet queue decoupling. In order to allow the decoupling, another level of abstraction can easily be added to PCFQ. Various applications demand different types of performance guarantees. If the application is delay-sensitive or error-sensitive, the mobile nodes and the BS are responsible for identifying which packet should be transmitted in the allocated slots, in the upstream and the downstream directions, respectively.

In short, unlike the previous protocols, the protocol proposed in this thesis does not assume instant or a-priori knowledge of the channel state. It also accounts for the bandwidth used in channel state identification, and considers the out-of-date information available at the BS to provide fair service to all the MNs. It thus provides short-term fairness for all the error-free nodes and long-term fairness for the error-prone nodes. The rate of recovery (ρ_i^R) and the time it takes to recover from the service lost (R_i) due to channel error for MN i are also bounded.

Chapter 3

Power Conservation

Limiting energy consumption of mobile computers is important owing to the limited battery capacities. Various efforts have been, and are being, made to conserve energy at the *hardware*, *software* and *wireless communication unit* levels.

3.1 Hardware Level

Table 1.1 shows the approximate power consumption in various hardware components of the mobile computer. With these values a mobile computer or laptop computer can operate for 3 - 6 hours only. Two major hardware strategies used to increase battery lifetime are:

- Design and use components that consume less power [6, 7].
- Keep most of the hardware components in low-power states most of the time.

Significant reduction in power consumption was observed using these techniques [6, 7, 5]. The tradeoff involved in taking advantage of a low power state is reduction in performance. Table 3.1 depicts the typical values of power consumption for hardware components when power conservation techniques are used.

Component	Power consumed with conservation techniques (W)	Power consumed without conservation techniques (W)
Processor	0.8 - 2.0 W (16-25%)	1.1 - 3.3 W (14-27%)
Hard drive	0.2 - 0.6 W (4.5-8.6%)	1.0 - 1.5 W (12.1-12.25%)
Backlight	1.1 - 2.0 W (24-26%)	2.3 - 3.4 W (28-30%)
Display	0.2 - 0.75 W (4.5-17%)	0.2 - 0.8 W (2-7%)
Modem	0.01 - 0.4 W (0.1-5%)	0.5 - 0.6 W (5-7%)
Sound	0.01 - 0.02 W (0.02-0.4%)	0.15 - 0.2 W (1.5-2.5%)
Video	0.3 - 0.5 W (5.8-10.5%)	0.3 - 0.5 W (3.8-4.5%)
Power management	0.1 - 0.2 W (1.5-2.6%)	0.1 - 0.2 W (1-1.5%)
Memory	0.05 - 0.06 W (0.7-1.4%)	0.05 - 0.1 W (0.5-0.8%)
Misc.	1.1 - 1.5 W (19-31%)	1.7 - 2 W (13-23%)

Table 3.1: Power consumed in various hardware components of mobile computer [6, 7, 5].

3.2 Software Level

Exploiting low power states of the hardware is one of the major strategies to conserve power. In this section, software strategies will be discussed for keeping the hardware in those states, as often as possible. Three different software strategies can be employed:

- **Transition strategy:** Try to identify the *trigger* point for switching the hardware to low power consuming mode.
- **Load-change strategy:** Attempt to modify the load on a hardware component in order to increase its use of lower power mode.
- **Adaption strategy:** Identify the modifications needed in the software in order to use novel, power conserving hardware components.

These strategies are being used by software designers at the *component level*, *operating system level*, *application level* and at *the user level* [34, 12, 4]. The application layer can conserve power by employing an application-aware adaptation strategy. If the application layer can predict its future requirements, the load-change strategy can be used effectively. User-level power conservation strategies are not considered to be

Transition States	Latency Involved (μ sec)
Transmit $\rightarrow$ Receive	6 - 30
Transmit/Receive $\rightarrow$ Sleep	4 - 10
Sleep $\rightarrow$ Transmit/Receive	20 - 40

Table 3.2: Latency involved in switching modes [6, 5].

too promising, since the user lacks knowledge about the power consumption of each component and is unable to make decisions fast enough to switch the component to a low power state.

3.3 Wireless Communication Unit Level

At the wireless communication unit level, the operating modes of a typical MN are *transmit*, *receive*, *idle*, *sleep* and *off*, in decreasing order of power consumption [7]. In *transmit* mode, mobiles are transmitting data; in *receive* mode, they are receiving data; in *idle* mode they are doing neither, but the transceiver is still powered and ready to receive or transmit; in *sleep* mode, the transceiver circuitry is powered down, except occasionally the circuitry wakes up to listen for incoming transmissions; finally in the *off* mode the transceiver circuitry is powered down and there is no occasional listening for incoming transmissions. Transitions between modes involve some latency [35]. These latency periods vary and range from six to forty microseconds. Table 3.2 shows the typical values for the latency periods involved in transitions from one mode to another.

Some devices provide the ability to dynamically modify their transmission power, thus reducing the power consumption in the transmit mode. This also helps in reducing the noise level for neighboring wireless devices, but the disadvantage is that reducing the transmission power also reduces the signal to noise ratio of its transmissions, thus increasing the bit error rate. The transmission power required for wireless devices also depends on the distance to the receiver. For instance in a wireless LAN environment where the distances are short, transmissions typically require 1.5 - 2.0 W.

Another strategy to reduce power consumption at the mobile device is to increase the time an MN remains in the sleep mode. One way to do that is to reduce the data transmission rate or stop data transmission altogether when the channel is bad. That is, stop transmitting when the probability of a dropped packet is high, so that less transmission time is wasted in re-transmissions [13]. If transmission ceases altogether, probing packets must be sent to determine when the channel regains the error-free state. Another approach is to use a medium access protocol that dictates in advance when each wireless device may receive data. This allows the device to switch to sleep mode during the period that it will not receive or transmit any data. Yet another way of conserving power is to reduce the transmission power and add correction codes to the data to mitigate the effects of increased bit error rate due to weaker transmission signals [36]. Using correction codes, however, has a side effect of reducing effective data bandwidth by consuming extra bandwidth for the additional code bits.

In the protocol proposed in this thesis, the MNs remain in either transmit, receive or sleep mode. The following issues must be considered in order to conserve energy at the wireless communication unit level [10].

Collision elimination

Collisions while nodes are willing to transmit data leads to unnecessary power consumption. Furthermore it also contributes to inflating delays. The number of re-transmissions will also increase if any kind of *collision detection* technique is used for wireless fair queuing.

Use of broadcast mode should be minimized

In *broadcast mode* the BS transmits data meant for a group of MNs requiring all the MNs to remain in *receive mode*. This leads to significant cumulative energy consumption. On the other hand, if each individual mobile was addressed separately more energy would be consumed at the BS but power would be saved at the MNs.

Mobiles should not switch states frequently

Significant time and energy is wasted by the transceiver unit of MN in switching between various states, such as from transmit to receive mode and vice-versa. Since there is a latency involved to toggle between the two states, efforts should be made to minimize switching between such states.

Delay vs Power Tradeoff

The IEEE standard P802.11 recommends that MNs should remain in *sleep mode* to conserve energy and keep buffering packets [17]. Then after buffering a certain number of packets the MN should transmit them. This way energy can be conserved at the cost of additional delay that might be incurred due to the buffering of packets.

Overhead should be reduced

Overheads in a protocol can be per packet such as destination and source address, or amortized such as acknowledgments and control packets. The more the overheads the longer it will take to transmit and/or receive the packet, thus increasing the time a mobile has to be in *transmit* or *receive mode.*

Centralized scheduling

Finally, studies have shown that *centralized scheduling* yields better results in terms of energy conservation than *distributing scheduling* algorithms [9, 25]. Distributed scheduling algorithms are less desirable as MNs might not receive reservation requests from all other nodes and further schedule computation consumes energy. Also, since a BS can be assumed to have more computational power and a better knowledge of the cell and the MNs in the cell, it is better to let the base do scheduling for both upstream and downstream traffic.

In the scope of this thesis we focus on conserving energy in MNs at the wireless communication unit level. The PCFQ protocol, is a collision-free protocol. It attempts to keep the broadcast mode on a per cycle basis and attempts to minimize the number of times a mobile has to switch states. The BS does the centralized scheduling, and the overheads are kept to a per cycle rather than per packet basis. Unlike some algorithms, PCFQ does not require continuous listening to the channel to identify the channel state, thus consuming less power [23, 2].

3.4 Power Conservation Protocols

In this following section, some MAC protocols commonly used in the mobile wireless environment will be presented and compared on the basis of their power conservation.

3.4.1 IEEE Standard P802.11

The IEEE standard P802.11 [17] for wireless LANs defines multiple access using a technique based on Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA). The basic idea of using collision avoidance was to remove the contention slots and consequently reduce the number of transmissions involved. As per the protocol, any backlogged mobile may transmit immediately if it detects the channel to be free for more than the DIFS (Distributed Coordinate Function Inter-Frame Space) period. If the mobile finds the channel busy it goes to the *backoff state* and defers its transmission. An unsuccessful transmission is followed by a *contention window* with a pre-determined number of slots. A mobile in backoff state then randomly picks a contention slot and detects transmission from other nodes. If transmission is detected it again enters the backoff state, while if no transmission is detected it has captured the medium. Clearly collisions increase power consumption as well as the obligatory sojourn time between transmit and/or receive states for an amount of time equal to the DIFS.

3.4.2 Packet Reservation Multiple Access (PRMA)

PRMA was proposed [24] to facilitate the integration of the voice and data traffic. In PRMA, packets are grouped into periodic information and random information packets. If a mobile with periodic information transmits a packet successfully in an available slot, that slot would be reserved for that mobile in the future. Mobiles with random information, however, will have to contend for an available slot each time. In an ideal error-free environment, PRMA would allow voice packets to reserve slots in advance while data packets would have to contend before each transmission. However, if the channel is error-prone, than each time a voice packet is corrupted due to channel error, the connection has to contend with other active connections. Thus, the power conservation measures taken in PRMA are susceptible to traffic and channel errors.

3.4.3 Multi-services Dynamic Reservation-TDMA (MDR-TDMA)

The MDR-TDMA protocol [37], divides a stream of TDMA frames into sub-streams for different types of traffic. MDR-TDMA supports CBR, VBR, and ABR traffic. Frames are dynamically allocated to different types of traffic. Time is divided into N_r request slots and N_m message slots. Each message slot is used for the transmission of a packet. Requests are comparatively short in length and are used for initial access during the contention mode. Out of the N_m message slots, a maximum of $N_v < N_m$ slots in each frame are assigned to CBR type voice traffic. VBR and ABR packet data are dynamically allocated after all the CBR slots have been allocated. The channel access scheme used by MDR-TDMA is a combination of circuit mode reservation of slots for CBR traffic and dynamic assignment of the remaining bandwidth for VBR and ABR traffic. Apart from traditional first-come-first-served (FCFS) scheduling, time-of-expiry (TOE) is also implemented to improve the delay performance of the protocol. In MDR-TDMA each time a mobile needs to transmit VBR or ABR traffic, it has to contend with other nodes, thereby increasing power consumption at the MNs. In the case of CBR traffic, power is conserved by reserving the slots in advance. However, as in PRMA, the nodes with CBR traffic also have to contend when a slot is missed due to a bad channel state.

3.4.4 Distributed Queueing Request Update Multiple Access (DQRUMA)

In DQRUMA [25], the BS employs a random access protocol and packet scheduling policy based on the traffic and service requirements of the MNs. When a packet joins the mobile's empty queue, the MN sends a transmission request to the base station. The BS is not explicitly informed for all subsequent packets: piggy-backing of transmission requests is used instead. DQRUMA uses a round-robin scheduling policy, and ALOHA random access protocol for medium access. From the power conservation point of view, the requirement that the mobile should listen during every slot places a significant burden on the MN's battery.

3.5 Power Conserving Fair Queueing (PCFQ)

PCFQ attempts to maximize the duration a mobile can stay in the sleep mode, thus conserving energy.

In PCFQ the BS is entrusted with the scheduling responsibilities for all the MNs. The power consumption at each MN for distributed scheduling is thus avoided due to the centralized scheduling at the base. The drawback of using centralized scheduling is the lack of up-to-date backlog information for the upstream traffic. To ensure fair scheduling, each MN thus needs to be polled frequently to refresh the upstream traffic information at the BS.

The BS schedules each MN in advance informing them when to transmit and/or receive data, in order to avoid data collisions. This is done by broadcasting a schedule for an interval which identifies the slots in which each MN is scheduled to transmit

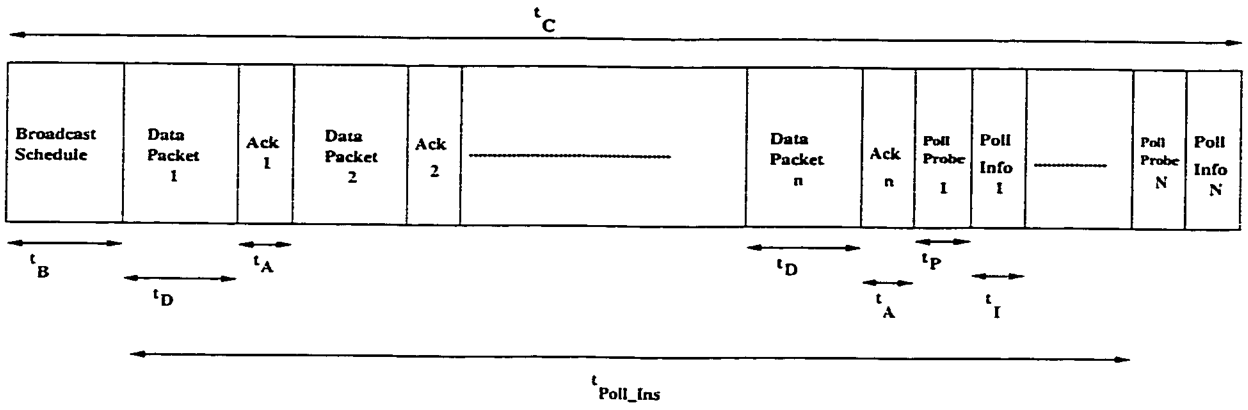

Figure 3.1: Typical Cycle Structure of PCFQ

and/or receive. This allows the MNs to switch to sleep mode whenever they are not able to receive or transmit any data. All the MNs are thus required to be synchronized either by a global broadcast clock or closely synchronized local clocks.

We propose to reduce the data transmissions made by an MN over a channel in error state, until the channel is known to be error-free from the base station's perspective. This helps to minimize bandwidth wasted on re-transmission(s) of dropped packets. However, the channel is probed occasionally once every cycle to determine, when it becomes error free. Further, due to the high probability of bit error, acknowledgments are required to determine the success or failure of data transmission. If data transmission is successful, the channel is considered to be in the error-free state while if the transmission fails the channel is in the error state.

PCFQ thus makes use of a cycle, as shown in Figure 3.1. A cycle consists of slots for *schedule transmission*, *data transmission*, their *acknowledgments*, and slots for *polling* each MN. Polling is further divided into two events, *poll probe* and *poll info*. During the poll probe the BS probes the MN and if the MN receives a probe successfully it transmits the poll information to the BS. The BS then generates the schedule for the cycle and broadcasts it in the slot for schedule transmission thus informing all the MNs in advance about their schedule. MNs are allowed to transmit and/or receive data during the data transmission slots. Each data transmission slot

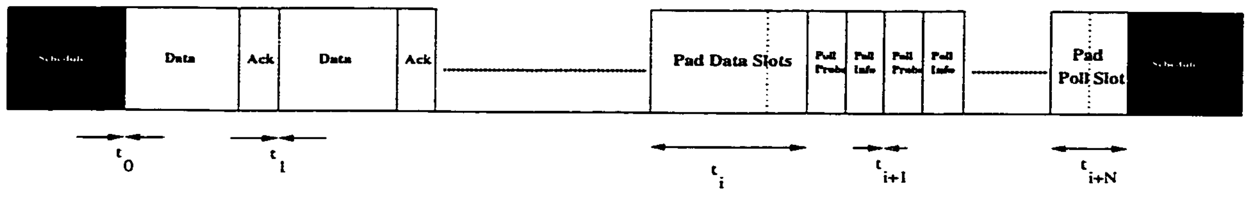

Figure 3.2: Base station

is followed by an acknowledgment slot to determine the outcome of the transmission, which is equivalent to determining the state of the channel. Polling slots are appended at the end of the cycle in order to update the upstream traffic information at the BS and also to probe for the current channel state. A successful poll implies that the channel is error-free. On the other hand if a mobile does not respond to a poll, then it is not considered for the next cycle as the channel is believed to be in the error state. This reduces bandwidth wasted on dropped packets, and possibly subsequent unsuccessful re-transmissions.

PCFQ considers the length of the cycle to be fixed. The fixed length of the cycle allows the MN to be in *sleep mode* until the next polling instant, if it either does not hear the schedule or is not scheduled in the current cycle. The drawback however is that if there is not enough available backlog to fill up all the transmission slots, than dummy slots have to be inserted in cycle to keep the cycle length fixed as shown in Figure 3.2. If the cycle length was variable then a mobile that did not hear the schedule would be required to stay in *receive mode* until it receives the schedule or poll probe from the BS. Hence the tradeoff by fixing the cycle length is between introducing dummy slots under *light* load thus reducing the overall utilization, and conserving more power at the MNs. The fixed cycle length provides advance knowledge of the polling and schedule instants and consequently helps in conserving power at mobiles.

Chapter 4

A Power Conserving Fair Queueing Protocol (PCFQ)

The basic goal of this protocol is to provide *fair* service to all the MNs and make an attempt to minimize the *power consumption* at the mobiles. In this chapter, a detailed description regarding the operation of the protocol is presented.

The protocol proposed in this thesis allows every mobile to have multiple flows (upstream and/or downstream). Each flow belongs to one of the three sets: *recovering*, *normalized*, or *error*. The recovering set $\hat{R}(t)$ is constituted of flows that received less service than their ideal fair share due to channel error in the past. The normalized set $\hat{N}(t)$ contains the flows that have received their ideal fair share of service. Recovering set and the normalized set combined together are referred as active set $\hat{A}(t)$. The flows that are currently experiencing channel errors belong to the error set, $\hat{E}(t)$. Together, these sets compose the set of all flows in a cell.

$$\hat{R}(t) \cup \hat{N}(t) = \hat{A}(t) \quad where, \tag{4.1}$$

$$|\hat{A}(t) \cup \hat{E}(t)| = n \ , \tag{4.2}$$

where n denotes the total number of flows in the cell at any point in time.

Figure 4.1 shows the transition of flows from one set to another. All the nodes that are successfully polled are considered to be currently perceiving an *error-free* channel. A polling slot is comprised of a poll probe by the BS and transmission

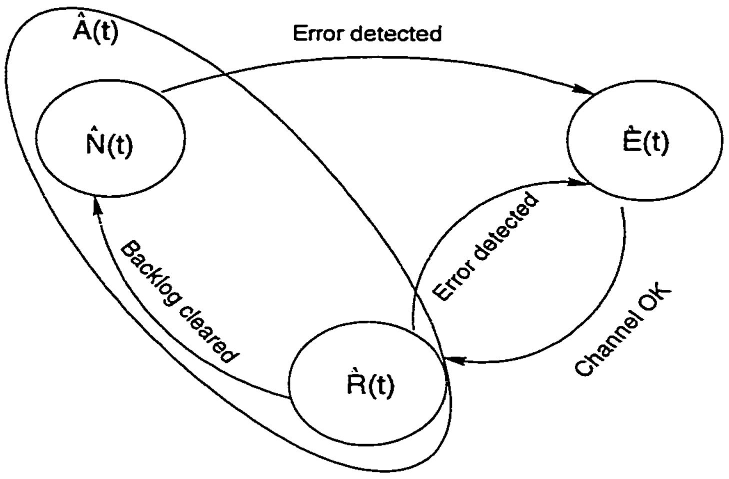

Figure 4.1: Set transition diagram for flows

of polling information from the mobile to base. The polling information includes details about the backlog at the MN for all flows, and details about the successful downstream transmissions. Based on the polling information from the MNs, the schedule for the next cycle is calculated at the base. This scheme is one of *collision avoidance* as slots for each transmission are pre-assigned ensuring *zero* collisions. It also *minimizes the broadcast time* which is limited only to the schedule broadcast in the beginning of each cycle. This *limits the overheads to a per-cycle* basis rather than to be per-transmission. As in other algorithms [22, 23] the *BS performs the scheduling* for both upstream and downstream traffic. In this protocol the BS tries to capture the *fairness* properties of SFQ while scheduling the packets from various flows.

The operating modes of the transceiver in a MN are *transmit*, *receive*, *idle*, *sleep* and *off*. When using PCFQ the mobile will be:

1. in *transmit* mode only when the mobile has been allocated a slot to transmit data, or while transmitting the polling information, or to acknowledge that a downstream packet has been received. Advance knowledge of the cycle schedule ensures that there is only one mobile transmitting at any instant and consequently eliminates the need for re-transmissions due to collisions. There might still be a need for re-transmission owing to corruption of packets due to bit

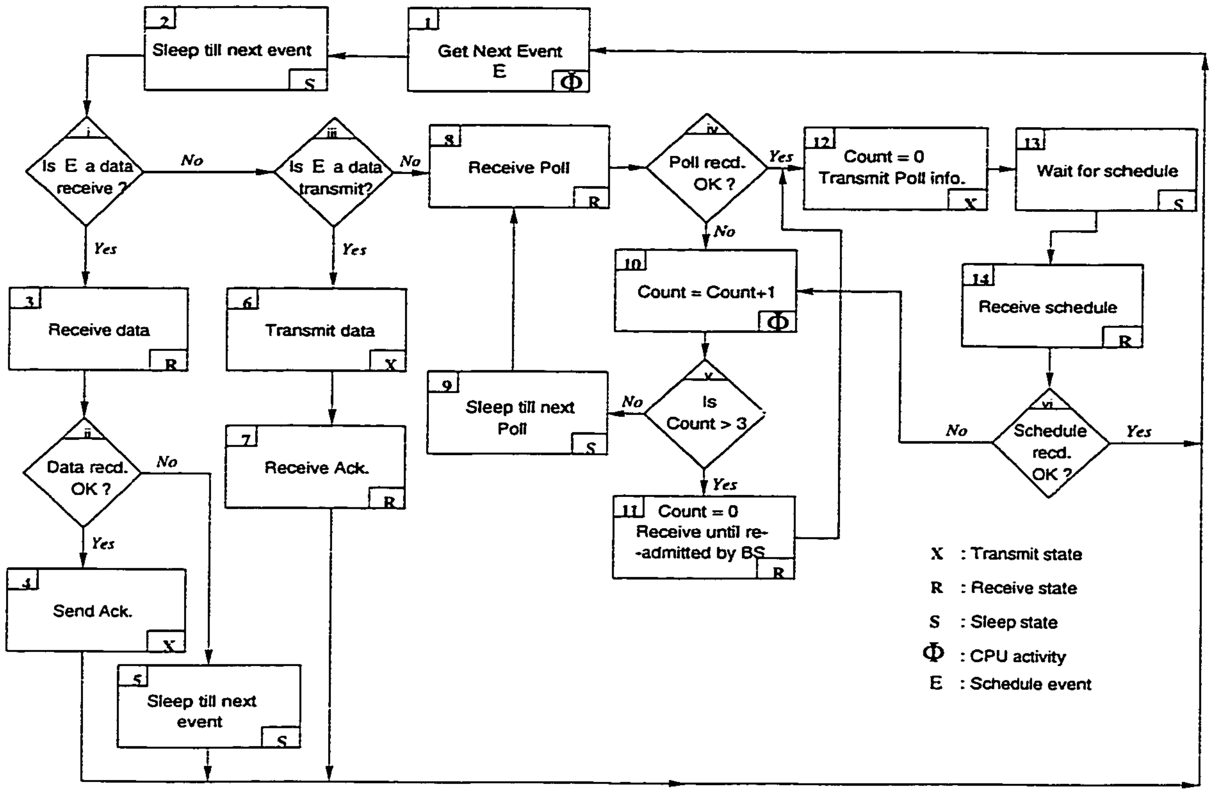

Figure 4.2: State transition diagram for a mobile node

errors.

2. in *receive* mode, when the schedule for the cycle is being broadcast, or when a MN is probed for polling, or when the mobile is receiving downstream data, or when acknowledgments for the upstream data are being received. Advance knowledge of the cycle schedule enables the MN to know when it is supposed to be in transmit or receive mode and when the next schedule will be broadcast.

3. in *sleep* mode, the transceiver circuitry of the mobile is switched off to conserve energy. In PCFQ, the mobile will be in sleep mode all the time, except when it is in *transmit* or *receive* mode.

4. in *idle* mode although, the mobile is neither transmitting nor receiving any data yet, the transceiver is powered on and ready to receive or transmit. PCFQ does not require the mobile to be in the *idle* state at any point of time.

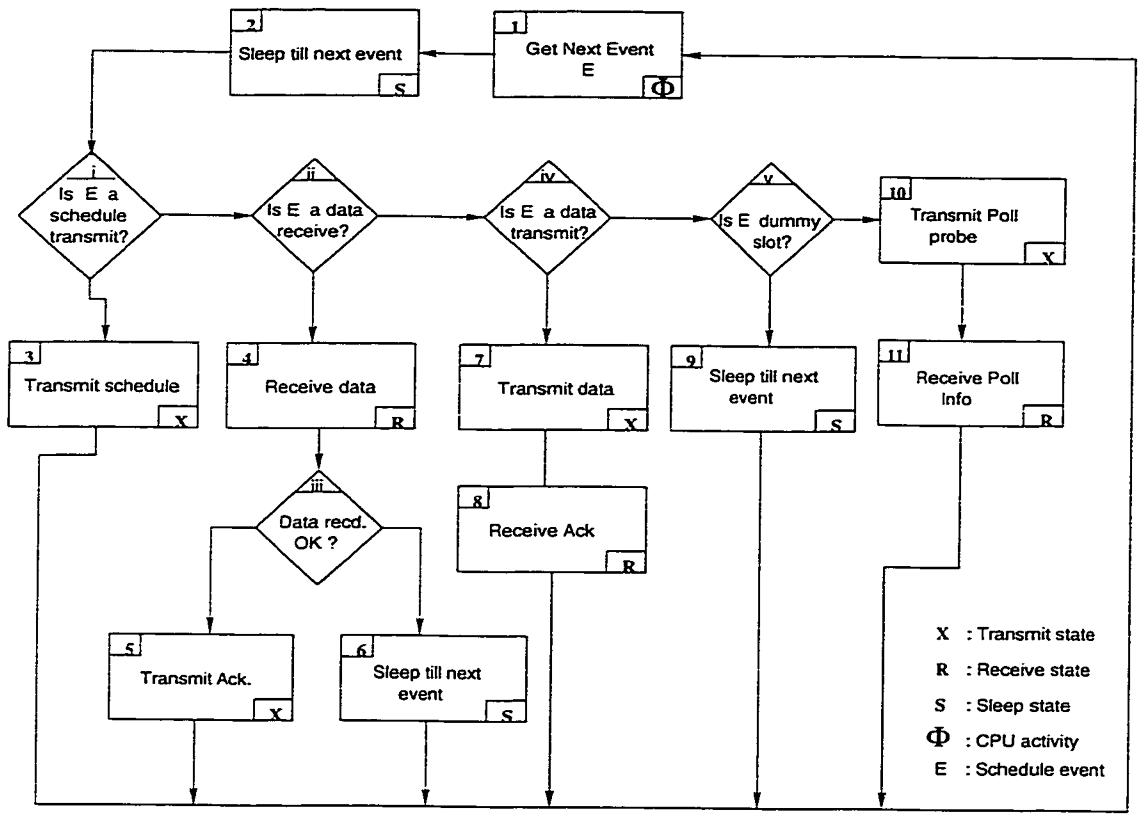

Figure 4.3: State transition diagram for the base station

5. in the *off* mode when the mobile wishes to terminate its connection.

Figure 4.2 shows the state transition diagram for a MN. A mobile is in receive state (R) when the transceiver is receiving data, in transmit (X) state when the transceiver is transmitting data, and in sleep state (S) when the transceiver is switched off until the next event. A state ϕ denotes a CPU activity in progress at the MN. Various events, denoted by E, correspond to the slots in the transmission cycle. A mobile strips the event from the schedule and switches to sleep state until the next slot allocated for it (see Figure 4.2 box 1-2). If the event is a data receive (see Figure 4.2 conditional box i), the MN switches to receive state, to receive the data packet (box 3). If the data is received successfully (conditional box ii), the mobile switches to transmit state to transmit the acknowledgment (box 4). Otherwise it goes to sleep (box 5) until the next event. If the event E is a data transmit event (upstream transmission, conditional box *iii*), the MN switches to transmit state to transmit the data (box 6). After the data transmission, the mobile returns to receive state to receive the acknowledgment (box 7). The MN switches to the receive state to receive the poll probe (box 8) during the polling event. If the probe is received successfully (conditional box *iv*), the MN transmits the polling information (box 12) and then sleeps until the schedule for next cycle is broadcasted (box 13). If the poll probe was not successfully received a counter is incremented (box 10) which keeps track of the number of unsuccessful polling events. If the poll is unsuccessful more than three times in succession (conditional box *v*), the MN is considered an *inactive node*, and switches to receive state (box 11) until it is re-admitted by the BS. If the number of failed polling attempts is less than three, the mobile sleeps until the next polling instant (box 9). Finally, if the mobile is not able to receive the schedule (conditional box *vi*), the counter is incremented and the mobile sleeps until the next successful poll.

Figure 4.3 shows the state transition diagram for the BS. Power conservation, is not an issue at the BS, yet PCFQ tries to minimize the energy consumption at the BS. Much like the MN, the BS also switches between transceiver states based on scheduled events. If the event is a scheduled transmit event (conditional box *i*), the BS generates the schedule and transmits it (box 3). If the event is a data receive event (conditional box *ii*), the BS switches to receive and receives data from the mobile (box 4). If the data was received successfully (conditional box *iii*), it switches to transmit and transmits the acknowledgment (box 5), otherwise it sleeps until the next scheduled event (box 6). If the event is a data transmit event (conditional box *iv*), the BS transmits data (box 7) and switches to receive the acknowledgment from the mobile (box 8). However, if the event is a dummy slot, used to pad the cycle in order to keep a fixed cycle length, (conditional box *v*), the BS sleeps until the next scheduled event (box 9). Finally, if the event is a poll event it transmits the poll probe for a MN (box 10) and switches to receive in order to receive the poll information

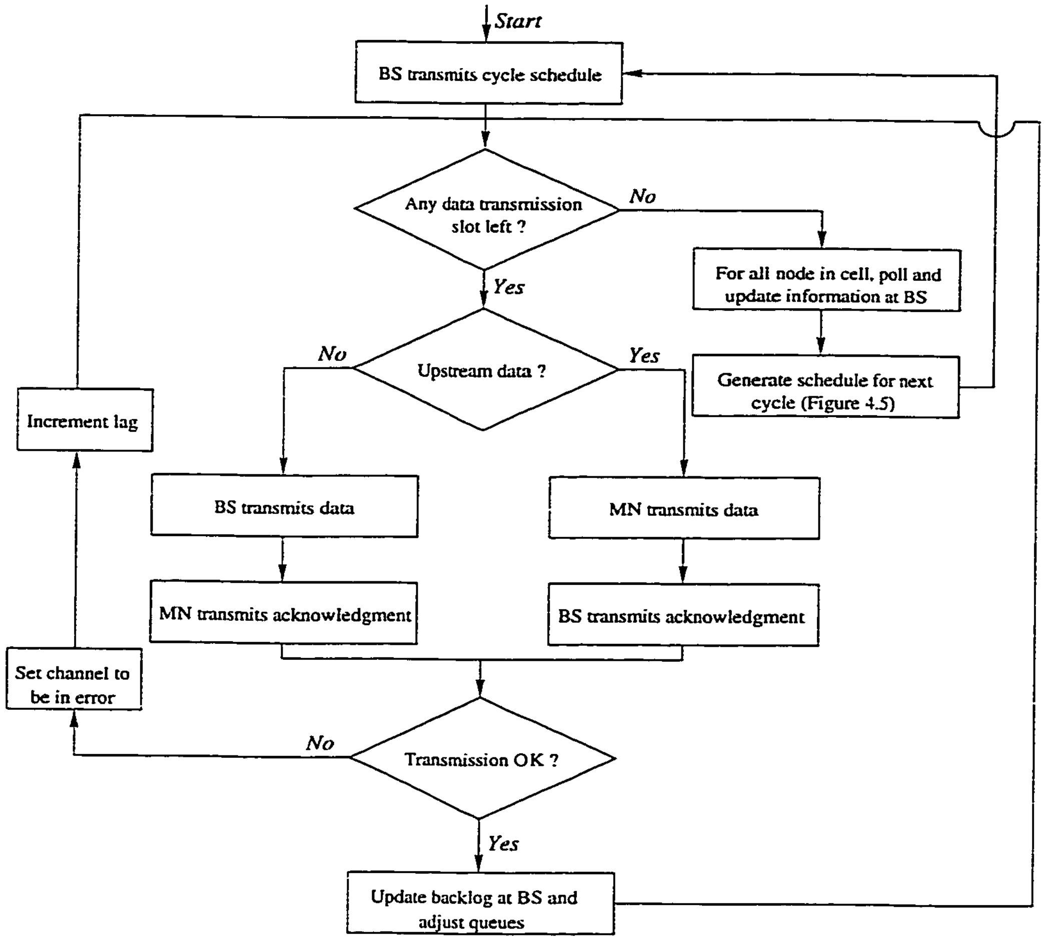
Start
BS transmits cycle schedule
Any data transmission slot left ?
No
For all node in cell, poll and update information at BS
Generate schedule for next cycle (Figure 4.5)
Yes
Upstream data ?
No
Yes
Increment lag
BS transmits data
MN transmits data
MN transmits acknowledgment
BS transmits acknowledgment
Set channel to be in error
No
Transmission OK ?
Yes
Update backlog at BS and adjust queues

Figure 4.4: Flow chart – I for PCFQ

from the MN (box 11).

The protocol makes its best effort to conserve energy at the MNs by keeping it in *sleep* mode unless it is required to be in *transmit* or *receive* mode. In order for mobiles to transmit and receive data accurately during the assigned slots, their clocks must be synchronized with the BS. In the present scope of work, we assume that the MNs have sufficiently accurate and synchronized clocks. We also propose that the clocks can be synchronized once every cycle, from the reception of the schedule.

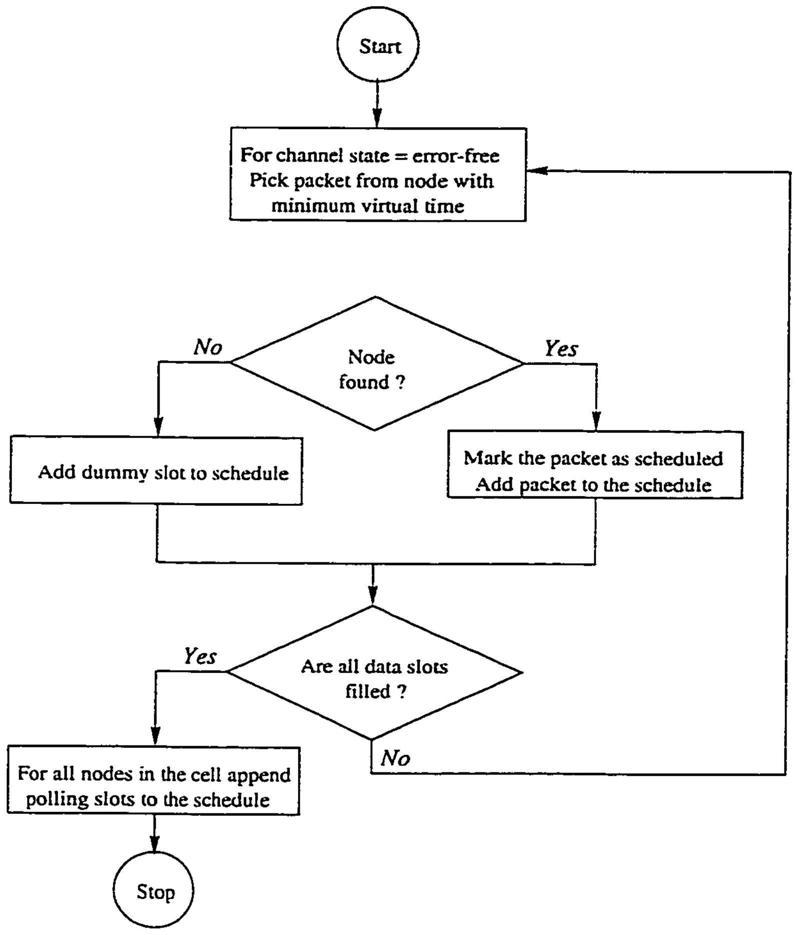
Start
For channel state = error-free
Pick packet from node with
minimum virtual time
Node
found ?
No
Yes
Add dummy slot to schedule
Mark the packet as scheduled
Add packet to the schedule
Are all data slots
filled ?
Yes
No
For all nodes in the cell append
polling slots to the schedule
Stop

Figure 4.5: Flow chart – II for PCFQ (Schedule generation)

4.1 Protocol Description

The protocol generates a schedule for both upstream and downstream traffic. The basic difference between the two is that the BS has *out-of-date* information for the upstream traffic, while it has current information for the downstream traffic. The BS has *out-of-date channel state* information for both upstream and downstream traffic. The BS keeps track of the channel states for individual nodes based on the last information exchange with each node. In between two information exchange events, BS has out-of-date information of the channel state. An error in the channel state for a flow implies a bad channel state for all flows of this node, in that particular direction.

Figures 4.4 and 4.5 present a general working model of PCFQ protocol. As shown in Figure 4.4, BS first transmits the cycle schedule and checks if there is any data transmission slot left in the cycle to be serviced. If it is an upstream data transmission slot, MN transmits the data and awaits an acknowledgment from BS, otherwise the BS transmits data packet and awaits an acknowledgment from the MN. If the transmission was successful backlog and queues at the BS and MN are adjusted, otherwise the channel is set to be in the error state and lag for the MN is incremented. After all the data transmission slots have been serviced, the BS polls each MN and updates its information to generate the schedule for the next cycle. As shown in Figure 4.5, BS generates the schedule by selecting a packet with minimum virtual time from the flows experiencing error-free channel. It then marks the packet to be scheduled and adds it to the cycle schedule. If no such packet was found it adds dummy slots to the schedule in order to keep the cycle length fixed. After all the data transmission slots have been filled, BS appends the polling slots for each MN to the schedule.

The protocol requires the algorithm shown in Figure 4.6 to be executed on every single MN and the algorithms shown in Figures 4.7 and 4.9 to be executed on the BS.

4.1.1 Mobile

As illustrated in Figure 4.10, each mobile maintains a queue of data packets for each admitted flow. A data packet is identified by a `<flow-id,packet-seq>` pair. The generation time (t_G) and status of each packet are also recorded. The mobile is in the *receive* state until it receives the first cycle schedule. As suggested in the algorithm shown in Figure 4.6, lines 3-6, if the mobile is not able to receive the schedule, it switches to the *sleep* state until its next polling instant. This will be a time period of t_{Poll_Ins} as shown in Figure 3.1, which is known in advance due to the fixed cycle length. If the mobile is able to receive the schedule successfully, it switches to the *sleep* state until the next slot allotted to it in the cycle, for a time period of t_0 as shown in Figure 4.8.

```
/* Constants used: (refer Figure 3.1)
      $t_{Poll\_Ins}$: Time for next polling instance.
      $t_C$: Time required to transmit complete cycle.
      $t_A$: Time required to transfer acknowledgment packet.
      $t_I$: Time required to transfer poll information.
      $t_i$: Time till next transmit/receive/poll slot. */

while (!done)
  i = 0, t = 0;
  if (receive(schedule) == OK)
    sleep($t_i$);
  else
    sleep($t_{Poll\_Ins}$);
  while ($t < t_C$)
    switch(schedule→Type) {
      case Transmit:// Slot for data transmission.
        transmit(packet);
        if (receive(ack) == OK)
          delete(packet); // If packet transmitted delete packet from queue.
      case Receive:// Slot for receiving data from base.
        if (receive(packet) == OK)
          transmit(ack); // Transmit an acknowledgment if packet is received.
        else
          sleep($t_A$); // Sleep for $t_A$ time units .
      case Poll:// Slot for polling.
        if (receive(poll_probe) == OK)
          transmit(poll_info); // Transmit poll information.
        else
          sleep($t_I$); // Sleep for $t_I$ time units.
    end switch
    sleep($t_i$);
    i++;
  endwhile
endwhile
```

Figure 4.6: Algorithm for mobile node

```
1. base()
2.   i = 0;
3.   generate(schedule);
4.   transmit(schedule);
5.   while (!done)
6.     switch (schedule→Type)
7.       case Schedule:// Slot for schedule broadcast.
8.         i = 0;
9.         generate(schedule);// Generates schedule for next cycle.
10.        transmit(schedule); // Broadcasts schedule for the cycle.
11.      case Transmit:// Slot for transmitting packet.
12.        transmit(packet,i);
13.        if (receive(ack,i) == OK)
14.          delete(packet,i);
15.        else
16.          Update_Vt[i] = YES;
17.      case Receive:// Slot for receiving data packets.
18.        if (receive(packet,i) == OK)
19.          transmit(ack,i);
20.        else
21.          sleep($t_A$);
22.          Update_Vt[i] = YES;
23.      case Poll:// Slot for probing and polling nodes.
24.        transmit(poll_probe,i);
25.        if (receive(poll_info,i) == OK)
26.          update_base();
27.        else
28.          Update_Vt[i] = YES;
29.    end switch
30.    sleep($t_i$);
31.    i++;
32.  endwhile
33. end base()
```

Figure 4.7: Algorithm for the base station–I

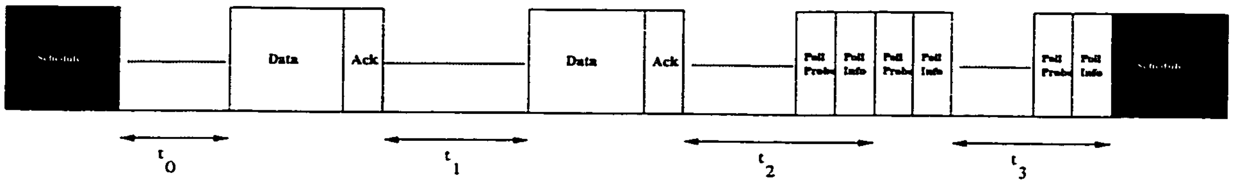

Figure 4.8: Mobile node

Transmit, receive and polling events of the cycle are handled in an MN as shown in Figure 4.6, lines 7-26. If a mobile is scheduled to transmit an upstream data packet, it switches to transmit state. The mobile transmits the upstream packet at the head of its queue and then remains in *receive* state for the acknowledgment (Figure 4.6, lines 9-12). If the acknowledgment is received the MN deletes the packet from its queue. It then switches to *sleep* state for a period of t_1 time units as shown in Figure 4.8.

If a mobile is scheduled to receive a downstream data packet it switches to the receive state. As shown in Figure 4.6, lines 14-17, the mobile receives the downstream packet from the base and if the packet was successfully received it transmits the acknowledgment for the packet. Otherwise the mobile sleeps for an interval equivalent to the acknowledgment packet transfer time of t_A time units. It then switches to the *sleep* state for t_2 time units as shown in Figure 4.8.

During the poll slot (Figure 4.6, lines 18-22), the mobile first waits to receive the poll probe from the BS, and if the probe is successfully received, it transmits the poll information to the base. Otherwise, the mobile sleeps for an interval equivalent to the poll information transfer of t_I time units. The mobile then switches to *sleep* state for t_3 time units as shown in Figure 4.8.

4.1.2 Base Station

As shown in Figure 4.10, the BS maintains one queue for each admitted downstream traffic flow. For upstream traffic it maintains the *virtual time* of the packet at the head of queue of the corresponding MN, the total backlog of each flow and a flag for

```
generate(schedule s) // Generates the schedule for next cycle
    for each node i // Update virtual time for nodes that experienced error.
        if (Update_Vt[i] == Yes)
            change_vt(i);
    end for
    while ((cycle not full) && (packets left))
        $i$ = find_node();// Finds the node with least $V_t$.
        if (node $i$ Polled) // If node was successfully polled.
            insert(packet,i);// Schedule packet at the head of queue for transmission.
    endwhile
    while (cycle not full)
        insert($Pad_{Data}$); // If cycle is not full, stuff in dummy packets.
    endwhile
    for each node $i$
        if (node $i$ alive) // If node is known to be in cell.
            insert(Poll,i);// Schedule poll events for all nodes.
        else
            insert($Pad_{Poll}$);
    end for
end generate()


change_vt(node i) // Change the virtual time for node i.
    $V_t = max[V_t, min_{k \in \hat{A}(t)}(V_{t_K})]$
    Update_Vt[i] = No;// Reset the Update_Vt flag.
end change_vt()
```

Figure 4.9: Algorithm for the base station–II

updating the *virtual time* when any node loses its share of bandwidth due to channel errors. The *virtual time* tag for any node is the time by which the packet should be serviced using the fair queuing algorithm (SFQ) for the given quality of service under error-free service. The virtual time tag for an upstream flow is assigned after successfully receiving poll information from the corresponding MN. The virtual time for each packet j is set to:

$$V_t^j = max[(V_t^{j-1} + (P_D/\rho_i)), min_{k \in \hat{A}(t)}(V_{t_k}), (t_G^j + (P_D/\rho_i))] \, , \tag{4.3}$$

where V_t^j denotes the virtual time of the jth packet to arrive in the queue, P_D is the data packet length, ρ_i is the rate allocated to node i, $\hat{A}(t)$ denotes all the flows in the active set, and t_G^j is the time instant when the new packet j was generated. In short, term $V_t^{j-1} + (P_D/\rho_i)$ denotes the virtual time of an already backlogged flow, while if the flow is not backlogged, the virtual time is assigned to the maximum of virtual time of the packet currently in service, and the virtual time based on the time packet was generated.

When the system starts up, the BS generates empty schedules. Based on the information gathered for the upstream and downstream traffic during the dummy cycle it generates a schedule for the next cycle and broadcasts it as shown in Figure 4.7, lines 3-10.

Cycle Construction

In order to generate the cycle, as shown in Figure 4.9 (lines 1-20) the BS selects the packet with the minimum virtual time and schedules it for the next cycle. It keeps scheduling by increasing order of virtual time until all packets have been scheduled or until the maximum number of slots for transmission have been used (Figure 4.9, lines 6-10).

A maximum limit on the number of transmission slots is enforced to allow error prone nodes to re-transmit lost packets within a fixed time frame. A large cycle would increase the turn-around time for error prone nodes to re-transmit thus inflating the delay for the packet. Likewise, a very small value for the cycle length would increase the amortized cost of the schedule broadcast and MN polling. If some transmission slots are unused, dummy packets are introduced (Figure 4.9, lines 11-13) in order to keep the cycle length fixed. Once all data slots have been scheduled, the polling events are scheduled for all nodes. Once again if some polling slots are left unused, dummy polling slots are inserted shown in Figure 4.9, lines 14-19 to fill up the cycle. Different cells may have different cycle lengths, which can be made known to the MNs through the schedule broadcast.

To summarize, the intuition behind setting the maximum number of slots in a cycle is as follows:

1. In order to determine the fading duration and current channel state, polling is done. However, polling too frequently would increase the associated overheads.

2. Since the cycle length is fixed, MNs are able to update the backlog information more accurately at the BS.

3. The fixed timing of polling instants allows the mobile to sleep and thus conserve energy.

Error State Detection

When the BS is scheduled to transmit a data packet, it transmits the downstream packet at the head of the node queue (Figure 4.7, lines 11-12), and then the BS reverts to *receive* state to await acknowledgment. If the acknowledgment is received the BS deletes the packet from the queue, otherwise it sets the update virtual time flag for the node (Figure 4.7, lines 13-16). By setting the update virtual time flag, the node is marked to be in the error set. That is, the node is moved from the active set to the error set.

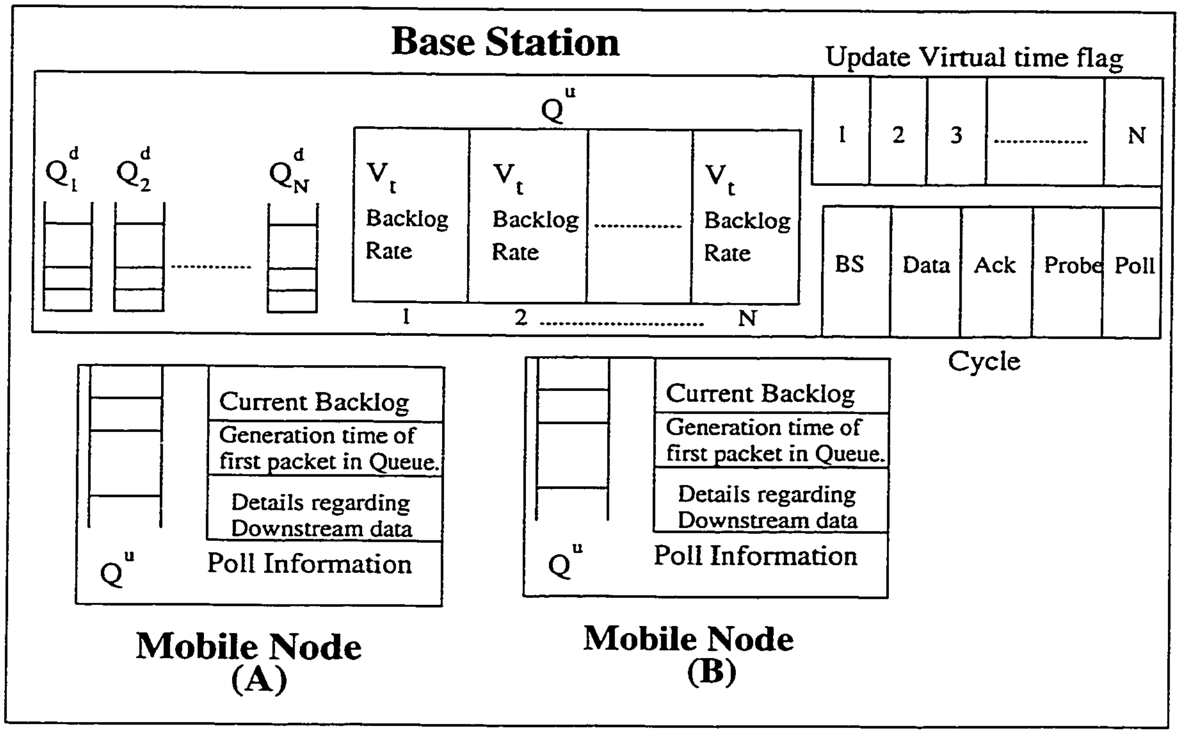

Figure 4.10: Data structure at the base station and mobile nodes

When the BS is scheduled to receive an upstream packet, it waits to receive the packet from the MN. If the packet was successfully received it transmits an acknowledgment as shown in Figure 4.7, lines 17-19. Otherwise it sleeps for time equivalent to the acknowledgment packet transfer of t_A time units, and sets the update virtual time flag for the node (Figure 4.7, lines 20-22), thus transitioning the MN from the active set to the error set.

Each time a packet is successfully transmitted the virtual time of the node is advanced by (P_D/ρ_i). If an MN loses service due to errors in the channel, its virtual time will not be advanced. Thus a mobile experiencing a long period of channel error will retain a smaller virtual time tag.

We note that when an MN experiences channel errors, and if it retains its virtual time for the packet at the head of the queue, then it could potentially fill up all transmission slots and thus starve the error free nodes when it eventually returns to good state. To avoid this scenario, in this case the virtual time is updated, thus preserving fairness for *error-free nodes*. According to the protocol, the *update time* flag is set whenever any node loses service due to channel errors. When the node returns to the error free state or transitions from the error set to the active set (Figure 4.9 lines 2-5), the virtual time of the node is updated as follows (Figure 4.9 lines 21-24):

$$V_t = max[V_t, min_{k \in \hat{A}(t)}(V_{t_K})] \ , \tag{4.4}$$

where V_t denotes the virtual time of the node regaining the error-free state and V_{t_K} denotes the virtual time of the packet at the heads of the queues at the other nodes in the active set $\hat{A}(t)$.

Updating the virtual time tag as shown in Figure 4.9 lines 21-24 for nodes ensures that the order of error free nodes is not adversely affected by nodes experiencing errors. In other words when an error-prone MN returns to the error free state it has an unbounded backlog, provided sufficient buffer space exists. Updating the virtual time creates a scenario where the backlog accumulated while in the error state is equivalent to a burst of instantaneous arrivals. This is because the virtual time for the packet at the head of the queue of any node in $\hat{A}(t)$ would be as if the packet has just arrived. Thus the error-free nodes are isolated from the error-prone nodes thereby ensuring *short term* fairness for the error-free nodes. A maximum limit is also enforced on the amount of backlog (Q_{Max}), that can be accumulated while the channel is perceived to be in error. This allows advance knowledge of the worst-case recovery period for any node, once it perceives an error-free channel and starts recovering.

Finally, during the poll slot, the BS transmits the poll probe to each MN and then waits to receive the polling information (Figure 4.7 lines 23-25). The polling information is comprised of the amount of backlog (in data slots) for each upstream flow from the mobiles, and acknowledgments for the packets successfully received for the downstream flows. If the polling information is received it updates the information for the node, else the update virtual time flag is set for the node (Figure 4.7 lines

26-28).

In summary the protocol uses a *virtual time tag* to identify the order of packet transmission and a *fixed cycle* to ensure that the mobile is not switched on all the time. The protocol also ensures that a recovering MN does not starve any normalized MN.

Chapter 5

Operational and Implementation Issues

5.1 Operational Issues

In the following section we present the operational issues of *call admission*, *network throughput*, *packet delay*, *recovery rate*, *recovery bound*, and *power bounds* for MNs involved in our protocol.

5.1.1 Call Admission

The BS is responsible for admitting the flows from various mobile nodes in a cell. Any new flow willing to join the cell has to contend with other such flows in order to gain access to the channel. Flows which are able to access the channel successfully then inform the BS about their desired quality of service (QOS). Based on this information, the BS decides whether or not the resource requirement of the flows could be met without jeopardizing the existing flows. This process of admitting or rejecting flows in a cell is known as *call admission*. Call admission is employed in the following two scenarios:

1. When a new node is entering from the adjacent cell. In such cases the BS of the cell catering to the node earlier would contact the BS of the target location in order to transfer the control of the node to the new BS. This is called *handoff*. Call admission is employed to determine whether or not the new node can

be admitted by the new BS, thereby resulting in a successful or unsuccessful handoff.

2. When a node from the same cell wishes to initiate a new flow. In such a case the BS would use the call admission process to either admit or reject the new flow. This process is called *join* operation.

5.1.2 Network Throughput

The percentage utilization of the network was considered as a function of rates demanded by wireless nodes and total channel capacity. i.e.

$$\%Utilization = (\frac{\sum_{i=1}^{N} \rho_i^0}{C}) * 100 \ , \tag{5.1}$$

where N is the number of flows, C is the total channel capacity and ρ_i^0 is the rate reserved for MN i.

Owing to the additional payload of acknowledgments, polling, and schedule broadcast in a cycle, 100% channel utilization is not possible using PCFQ. The maximum achievable utilization for given parameters can be derived as follows.

A cycle consists of one slot of duration t_B for broadcasting the schedule, N slots for data transmissions and their acknowledgments, followed by polling events for each node. The total overhead for a cycle can thus be defined as follows:

$$O_C = \frac{t_B + (N * t_A) + (n * (t_P + t_I))}{t_B + (N * t_D) + (N * t_A) + (n * (t_P + t_I))} \ , \tag{5.2}$$

where O_C is the overall cycle overhead, t_B is time taken to broadcast the cycle schedule, N is the number of data transmission slots, t_A is the time taken to transmit an acknowledgment packet, n is the total number of nodes, t_P is the time taken for the poll probe packet, t_I is the time it takes to transmit the poll information packet and

t_D is the time taken to transmit a data packet. In short, the denominator denotes the time required to transmit a cycle, t_C.

The maximum achievable utilization is:

$$\begin{aligned} \eta_{max} &= 1 - O_C \ , \\ \eta_{max} &= \frac{(N * t_D)}{t_B + (N * t_D) + (N * t_A) + (n * (t_P + t_I))} . \end{aligned} \tag{5.3}$$

The BS performs call admission knowing that η_{max} is the maximum network utilization and allocates rates for flows in both upstream and downstream directions.

5.1.3 Packet Delay

In order to serve the *guaranteed-service* applications a strict performance bound is required. PCFQ, in an attempt to cater to these applications, provides three levels of performance bounds: a *delay bound*, a bound on *power consumption*, and a bound on the *recovery period* for lagging nodes.

The delay bound has four components: insertion delay ($Delay_I$), propagation delay ($Delay_P$), queueing delay ($Delay_Q$), and delay introduced due the cycle ($Delay_C$). For the sake of convenience we assume the propagation delay to be zero.

$$\begin{aligned} Delay_I &= \frac{(P_D + P_A)}{C} \ , \\ Delay_P &= 0 \ , \\ Delay_Q &= \frac{B_s}{\rho_i} \ , \\ Delay_C &= \frac{(2 * P_B) + (N * P_D) + (n * (P_P + P_I)) + (N * P_A)}{C} \quad or \ , \\ &= t_C + t_B \ , \\ delay\ bound &= Delay_I \ + \ Delay_P \ + \ Delay_C \ + \ Delay_Q \ , \end{aligned} \tag{5.4}$$

where C is the channel capacity, P_D is the data packet length, P_A is the acknowledgment packet length, P_P is the poll probe length, P_I is the length of polling information packet, P_B is the length of the schedule broadcast packet, N is the total number of data transmission slots, n is the total number of flows, B_s is the burst size and ρ_i is the rate allocated to node i.

5.1.4 Recovery

An important goal of this protocol is to maintain guaranteed service for the error-free flows irrespective of the flows experiencing channel errors. Any flow receiving service as per the error-free service model is said to be receiving *normalized service.*

Any flow which is receiving less than its normalized service is said to be a *lagging* flow. Likewise, any flow receiving more service than its normalized service is called a *leading* flow. When a lagging flow tries to clear its backlog generated during the period that the channel was in error state, the flow is said to be *recovering.* In other words, even in situations when a flow observes few corrupted slots in a cycle and an error-free channel during poll, the mobile will be considered to be recovering. Various terms involved with the recovering state of a flow are defined as follows:

Let j be a flow in transition from the no-error state to error state. As long as the flow j remains in the error state, backlog will be generated at the MN. Flow j will try to clear this backlog as soon as it returns to the error-free state again. The period during which the flow j tries to clear its backlog is termed its *recovery period.* During the recovery period the flow is said to be in *recovery state*, or recovering.

Consider a flow k to be in the recovery state. The amount of excess service given to flow k in addition to what was guaranteed to it determines its *recovery rate.*

For recovery, the protocol re-allocates the excess bandwidth left over from that which was either not allocated or allocated to flows in error state. The recovery rate ρ_j^R of flow j is:

$$\rho_j^R = \rho_j^0 + \left[\frac{C - (O_C + \sum_{i \epsilon \hat{A}(t)} \rho_i^0)}{\sum_{i \epsilon \hat{R}(t)} \rho_i^0} \right] * \rho_j^0 \ , \tag{5.5}$$

where ρ_j^R is the recovery rate for flow j, C is the channel capacity, $\hat{A}(t)$ denotes the set of active flows, $\hat{R}(t)$ denotes the set of flows in the recovery state, ρ_i^0 is the rate allocated to flow i at the time of call admission, and O_C is the overall overhead involved due to cycle. The time it takes to recover can also be bounded if the flow experiences an error-free channel throughout the recovery period as in Equation 2.16.

5.1.5 Power Bounds

A mobile can be in one of the three modes: transmit, receive, or sleep. Bounds for time spent in each mode can be calculated using the rate allocated to the node and the maximum number of transmission slots in a cycle.

A mobile will switch into transmit mode, to transmit the upstream data packet, to transmit acknowledgment for the downstream data packet and to transmit the polling information:

$$t_T = (\frac{\rho_r * N * (t_D + t_A)}{C}) + t_I \ , \tag{5.6}$$

where ρ_r is the worst case recovery rate from Equation 5.5 when all the nodes are recovering, N is the maximum number of transmission slots in a cycle, t_D is the time taken to transmit the data packet as shown in Figure 3.1, t_A is the time taken to transmit an acknowledgment packet, C is the channel capacity and t_I is the time taken to transmit the polling information to the base.

Likewise the mobile will switch to receive mode to receive downstream data, to receive an acknowledgment for the upstream data packet, to receive the poll probe from the BS and to receive the schedule for the cycle:

$$t_R = (\frac{\rho_r * N * (t_D + t_A)}{C}) + t_P + t_B \ , \tag{5.7}$$

where t_B is the time required to receive the schedule for the cycle and t_P is the time required to probe a node for polling as shown in Figure 3.1.

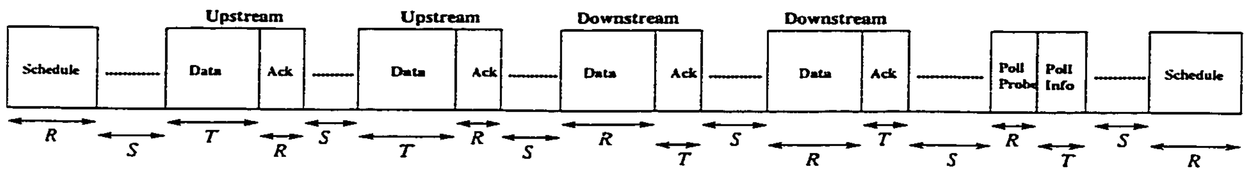

Figure 5.1: Turn around time–I

The mobile remains in the sleep mode whenever it is not in either the receive or transmit mode.

$$t_S = t_C - (t_T + t_R) \ , \tag{5.8}$$

where t_C is the time it takes to transmit a cycle as shown in Figure 3.1.

5.2 Implementation Issues

The two main implementation issues that need to be addressed for the PCFQ protocol are *join/leave and handoff control* and *power conservation.*

5.2.1 Join/Leave and Handoff Control

The PCFQ protocol deals with flows that have already been admitted to the cell. For any new flow willing to join this cell, it is proposed that it should contend with other such flows before being admitted. A *contention period* is the period during which all the new flows willing to join a cell contend to get the channel and inform the BS about their presence and the quality of service (QoS) they would like. Once a flow succeeds in seizing the channel, the BS decides whether or not the required quality of service can be met without jeopardizing the QoS of existing flows. The base is thus responsible for advertising the contention instant and its period in order to allow the new flows to contend. It is proposed that the advertising of the contention period can be done using a separate virtual channel. The contention periods are empty slots grouped together at the end of the cycle as shown in Figure 5.2. Any flow willing to

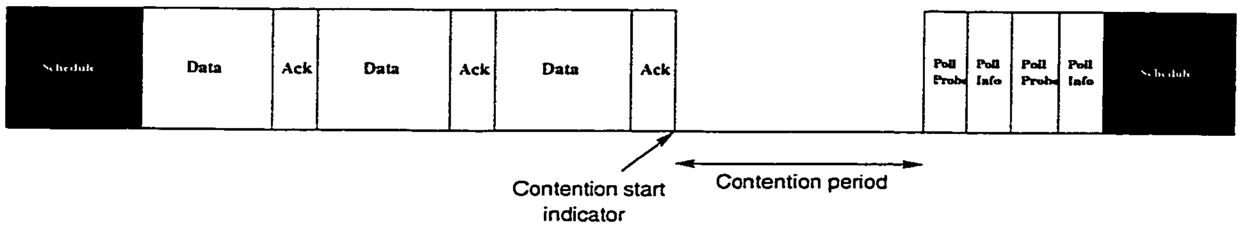

Figure 5.2: Contention period in a cycle

join will be listening to the broadcast messages regarding the contention periods. As stated in Section 4.1, dummy transmission slots are inserted in the cycle whenever the cycle is not full. These dummy slots can be used as contention slots for the new flows. In order to avoid a large latency in connection admission, we propose that the BS should periodically insert these contention slots in the cycle, if it can afford to accept a new connection. Using this scheme implies higher power consumption for the MN until it is admitted in the cell. However, such operations should be infrequent.

In addition to the admission of new flows we propose that the BS should also deal with the issues related to dropping a flow from the cell. We propose that if a flow is inactive for a certain period of time, that it be dropped from the set of admitted flows. The time threshold which determines when to drop a flow can be derived on the basis of number of unsuccessful polling attempts for the flow.

Another interesting scenario is when an MN moves from one cell to another while the connection is active. The BS catering to the node earlier would then have to transfer the control of the node to the new BS in order to allow the node to communicate in the new cell. This mechanism of transferring the control of a MN from one BS to other is called *handoff*. There are two major types of handoff operations, *hard handoff* and *soft handoff*. We prefer soft handoff over hard handoff because the former does not suffer from the problem of disconnection periods during handoff. A detailed description of these are outside the scope of this work. We propose to employ the following soft handoff scheme in PCFQ. When the mobile approaches the boundary of a neighboring cell, it receives transmissions from both BSs and a request for handoff

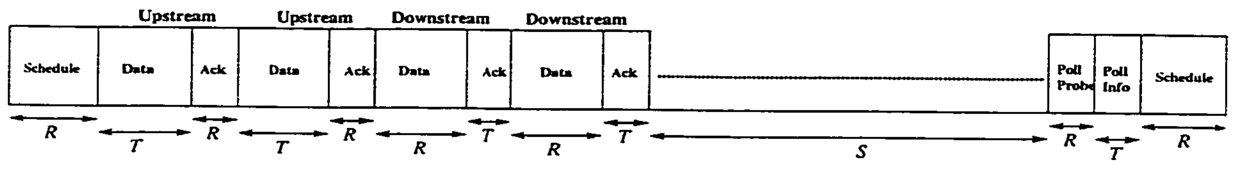

Figure 5.3: Turn around time–II

is then generated. The BS of the new cell applies its call admission procedure and subsequently accepts or rejects the MN. On successful completion of the handoff, the old BS drops the MN from its service queue, thereby allowing the new BS to take complete charge of the MN.

Another scenario can be when an inactive node moves from one cell to another. In such cases when the MN becomes active in the new cell, it waits to hear the *beacon* from the BS identifying its current location. A beacon is nothing but a signal from the BS announcing its presence in the cell. The MN has to wait for the schedule in order to synchronize its clock and then it waits for the contention period, in order to contend and gain access to the channel. The information regarding the start time and duration of contention period is supplied by the BS on a separate channel.

Yet another issue is to determine what happens to the backlog at the BS when an inactive node moves to another cell and for how long should the BS store such a backlog. In such cases, one option is to discard all the backlog at the BS once the MN has moved to another cell. The other option is to transfer all the backlog to the new BS during or after the handoff. In either case no service guarantees and/or delay bounds can be provided to a node during the handoff period, as the behaviour of the node cannot be safely predicted (mobile may even be powered down).

5.2.2 Power Consumption Control

Power consumption is a major concern for mobile computers. The PCFQ protocol requires the node to be in one of the three modes: *transmit*, *receive* or *sleep*. Time is consumed while the mobile switches from receive to transmit state or vice-versa. The typical values for such *turnaround times* varies from 6 to 30 microseconds. Within the context of this study the effects of turnaround times are assumed to be accounted for in the derivation of slots.

PCFQ tries to address power conservation by keeping the MN in sleep mode as long as possible. Turnaround times can be reduced significantly by carefully designing the cycle such that multiple packets are transmitted and/or received back to back, followed by an acknowledgment. Such a technique, can reduce the turnaround times, however, the susceptibility of a flow to error is increased if all of its transmissions are grouped together. This depends ultimately on the derivation of a bad period. As a compromise to provide the general solution we propose that the transmission slots from the same MN should be distributed over a cycle. This will reduce the number of times a node has to switch states between transmit and receive mode considerably, thus reducing latency.

To illustrate this, assume a cycle of 24 data transmission slots, with identical upstream and downstream traffic rates for six nodes admitted with equal rates and utilizing the channel to its capacity. If each node has two upstream data packets and two downstream data packets scheduled for this cycle, the cycle will be fully utilized. If these packets are grouped together as shown in Figure 5.3, each mobile will have 10 transition states from receive mode to transmit mode or vice-versa. However, if they are interspersed as shown in Figure 5.1 within the cycle, a mobile will have only 5 such transition states. Thus we save 50% of the transition states and latency per mobile, by interspersing the transmission slots.

Chapter 6

Simulation Results

Several sets of simulation experiments were performed, in order to quantify the performance of PCFQ. The focus was on the wireless MNs connected to the network via a BS. The bandwidth of the wireless channel is assumed to be fixed at 10Mbps. A total of six wireless MNs were used in the simulation study. Various experiments were conducted for different values of overall network demand, traffic pattern and outage rate. In this chapter the key models used during simulation will be explained first, followed by observations from the simulations.

6.1 Simulation Model

6.1.1 Channel State Model

The *error* and *no-error* periods in the channel are considered to be exponentially distributed with means of μ_B and μ_G respectively. We define the *Outage rate* as the percentage of time the channel is in the error state:

$$\textit{Outage rate} = \frac{\mu_B}{\mu_B + \mu_G}. \tag{6.1}$$

Simulations were run for three values of outage rate: 0.1%, 1% and 10%, with μ_B of 10 msec and 1 msec. To show short term fairness for error free nodes half of the nodes were considered to be error free all the time and the rest of the nodes were considered to be experiencing errors at the given outage rate.

6.2 Traffic Model

Both CBR and VBR traffic are modeled in these simulation experiments. In the case of CBR traffic, packet arrivals are assumed to be at a constant rate agreed upon during call admission. VBR traffic on the other hand, was based on the "token bucket" model [38]. This model allows a maximum burst of σ bits, and services the packets in a queue at a rate of ρ bits per unit time only. The peak arrival rate was assumed to be equal to the channel capacity.

A total of 120 seconds of protocol activity were simulated using six nodes: three nodes are always error-free and the others experience errors throughout the simulation. All nodes take part in both upstream and downstream transmissions, with one flow in each direction. All nodes are assumed to be allocated identical rates for convenience. Several rounds of simulation were used for various values of overall channel utilization. The total channel capacity is assumed to be 10Mbps. The packet size for both upstream and downstream transmissions are considered to be fixed at 1024 bytes. Acknowledgment packets, poll probe, and polling information packets are considered to be 10 bytes each. The size of the cycle schedule is assumed to be fixed at 1024 bytes. Fixed sizes for the schedule, polling, acknowledgment packets, and data packets are assumed to obtain a fixed cycle length, thereby conserving power at the MNs. The burst size (B_S) for each flow was fixed at 100 packets. B_S corresponds to maximum burst of traffic. Q_{Max} for each node is considered to be twice or five times the burst size: $Q_{Max} \in \{2B_S, 5B_S\}$. Since equal traffic is considered for both the upstream and the downstream channel, total utilization is double the utilization in one direction.

The simulation results will be presented into three categories: *delay bound guarantees*, *recovery bound*, and *power conservation*. Each of these categories will be discussed separately using the graphs presented in the end of this chapter.

6.3 Delay Bound Guarantees

We measure the *peak* delay values for both the error-free and error-prone nodes, noting that the average delay values are much less. These values are then compared with the delay bounds calculated using Equation 5.4. The maximum utilization that can be achieved in these simulations, derived using Equation 5.3 is approximately 94%, or 47% in each direction. We not that maximum utilization could be achieved for error-free nodes. Figure 6.1 shows peak delay values of the error-free MNs, for downstream transmission only, however similar results were obtained for the upstream transmissions as well. The X-axis in the figure shows the rate ρ_i allocated to each MN and the Y-axis is the maximum delay experienced by the node in milli seconds. Three of the four curves plotted in the graph give the delay values for outage rates of 0.1%, 1%, and 10%. The fourth curve gives the delay bounds obtained from the Equation 5.4. The graph in Figure 6.1 suggests that delay bounds are not violated by the error-free MNs, until the maximum achievable utilization of 47% in the downstream direction is achieved. Thus we claim that the delay for error-free nodes was not affected by the error-prone nodes, and thus delay guarantees for error-free nodes can be provided. As expected the delay was also not inflated by the outage period.

Likewise, in Figure 6.2, the peak delay values for the error-prone MNs are plotted on the Y-axis against the rate ρ_i allocated to each node. We note, that in case of error-prone nodes a fraction of bandwidth has to be reserved to allow them to recover from lost services accumulated during periods of channel error. The graph in Figure 6.2 shows that the delay bounds are violated for utilization greater than 40%. This implies that spare bandwidth is needed in order to compensate for the bandwidth lost when a node was in the error state. Moreover, the delay depends crucially on the outage rate. In order to achieve higher utilization for error-prone flows, we propose that the connection admission control (CAC) algorithm should reserve spare bandwidth, allowing even the error-prone nodes to obey the delay guarantees. Please note that in real system a-priori knowledge of error-free and error-prone flows is not available, so CAC has to be enforced instead of simply separating the two flows.

We also obtained results related to the service received by error-free and error-prone nodes at different outage rates. The X-axis in Figure 6.3 is the rate allocated to the nodes, and the Y-axis is the actual service received by the node in Kbps. Services received by the error-free nodes were identical irrespective of the outage rates, represented by one curve in the graph. The other three curves plotted in the graph show the services received by error-prone nodes experiencing outage rate of 0.1%, 1%, and 10%. Figure 6.3 suggests that error-free nodes receive the amount of service allocated to them, irrespective of the outage rate. This implies that the service of error-free nodes is not affected by the error-prone behaviour of other MNs.

6.4 Recovery Period

We also measured the recovery period as defined in Section 5.1.4 for the error-prone nodes. Figures 6.4 and 6.5, show the recovery period for 1% and 10% outage rates respectively. Here the X-axis is the time required to recover in milli seconds, while the Y-axis is the relative frequency of recovery. The graphs are plotted for down-stream traffic with an overall utilization of 35%. Similar observations were made for other utilization values as well. A generic bound on recovery period for recovering flows cannot be obtained due to the stochastic nature of channel state. However, a bound on recovery period is calculated in Equation 2.16 assuming that the flow remains error-free while recovering and that the buffer for a flow is at most Q_{max}. Figures 6.4 and 6.5 show that the error-prone nodes were able to recover within the recovery bound calculated using Equation 2.16. This behavior was only observed for an overall utilization of 40% or less. Again, we propose that spare bandwidth should be reserved in order to allow the error-prone nodes to recover more quickly. This can be accomplished by employing an appropriate CAC algorithm.

The X-axis in Figure 6.6 is the time required to recover in milli seconds, and

Y-axis is the cumulative frequency of recovery. The two curves plotted in the graph correspond to the outage rates of 1% and 10%. Please note that the recovery period observed for the outage rate of 1%, is larger than the recovery period observed for the 10% outage rate. This is because the higher the outage rate, the probability that a MN is in the error state at any given instant is higher. Thus for a given utilization, higher outage rates imply that there will be fewer nodes recovering thereby increasing the rate of recovery for recovering nodes, as given by Equation 5.5.

As shown in Figures 6.1 and 6.3, PCFQ provides delay and service guarantees for error-free nodes. Thus we claim that PCFQ provides short-term fairness for error-free nodes. However, for the error-prone nodes (Figures 6.2, 6.4, 6.5, and 6.6), long-term fairness can only be achieved by reserving spare bandwidth.

6.5 Power Conservation

One of the major contributions of the PCFQ protocol is that it tries to maximize energy conservation at the MN. In addition to that, although power conservation is not an issue at the BS, efforts have been made to conserve energy at the BS as well.

Figures 6.7 and 6.8, show the amount of time an MN spends in the transmit state for a CBR and VBR traffic respectively. The curves plotted in the graphs correspond to outage rates of 0.1%, 1%, and 10%. We observe that the amount of time an MN remains in transmit mode is less than the theoretical bound calculated using Equation 5.6, thus conserving energy. We also observe that the time spent in the transmit state increases with outage rate. This is because the number of re-transmissions of upstream data packets increases due to more frequent bad channel states for higher outage rates, thus increasing the amount of time an MN spends in the transmit state.

Likewise Figures 6.9 and 6.10 show the amount of time an MN spends in the receive state for CBR and VBR traffic respectively. The curves plotted in the graphs correspond to outage rates of 0.1%, 1%, and 10%. The amount of time an MN remains in receive mode is less than the theoretical bound calculated using Equation 5.7, thus conserving energy. We also notice that the time spent in the receive state is greater for an outage rate of 10%. This is because the number of re-transmissions of downstream data packets increases due to bad channel states for a higher outage rate, thus increasing the amount of time an MN spends in the receive state.

Figures 6.11 and 6.12, show the amount of time a MN spends in sleep state for CBR and VBR traffic respectively. The curves plotted in the graphs correspond to outage rates of 0.1%, 1%, and 10%. Unlike the transmit and receive state graphs, the amount of time a MN spends in the sleep state is more than the theoretical bounds calculated using Equation 5.8. However, since the sleep state is the least power consuming state, it is desirable to keep the MN in the sleep state more often. We also notice that the time spent in sleep state is less for an outage rate of 10%. This is because an MN spends more time in the transmit/receive state due to more frequent channel errors, thereby reducing the sleep time.

Figure 6.13 shows the relative time error-free mobiles spend in each of the three states for an outage rate of 1%. Likewise, Figure 6.14, shows the relative time error-prone mobiles spend in the three states for an outage rate of 1%. These graphs are presented for VBR traffic, however identical results were obtained for CBR traffic as well. In both of these graphs, we observe that the amount of time an MN spends in the sleep state is much greater than the time it stays in the receive or transmit states, thus conserving energy. We also notice that the receive time is greater than the transmit time. This is because an MN stays in receive mode to receive a downstream packet, to receive an acknowledgment for an upstream packet and to receive the cycle schedule, while it remains in transmit state only to transmit upstream data packets and to acknowledge downstream data packets. Also, the difference between the time an MN spends in receive and transmit state is constant. This is because we assume equal traffic for both upstream and downstream transmissions, and an MN remains in receive state to receive the cycle schedule.

We measure the power consumed at the BS throughout the simulation and present it for outage rates of 1% and 10% in Figures 6.15 and 6.16 respectively. Other routine work of the BS that involves transceiver usage, like advertising for contention slots, interacting with neighboring base cells during handoff, etc. are not taken into account for these graphs. The X-axis in these graphs is the overall channel utilization and the Y-axis is the percentage of time the BS spends in any of the three states. The amount of time the BS remains in the transmit, receive or sleep states is almost identical for all outage rates. This suggests that the outage rate in a certain cell does not influence the power consumption at the BS significantly. This observation may be useful for ad-hoc mobile wireless networks, where even the BS is power sensitive. Another observation that can be made from these graphs is that the amount of time the BS spends in the sleep state decreases with the increase in overall utilization. This is because an increase in utilization implies fewer dummy slots in the cycle, thus decreasing the amount of time a BS remains in the sleep state. Note that the percentage of time the BS remains in the transmit state is always greater than the amount of time it spends in the receive state. This is because the BS is in the transmit state when it transmits a downstream data packet, acknowledges an upstream packet and when it broadcasts the cycle schedule, while it remains in the receive state only to receive upstream data packets and acknowledgments for downstream packets. Also, the difference between the time the BS spends in transmit and receive is constant. This is because we assume equal traffic for both upstream and downstream transmissions, and the BS remains in the transmit state to broadcast the cycle schedule.

Based on the observations presented above, we claim that PCFQ attempts to minimize energy consumption at the MN by maximizing the time an MN spends in the least power consuming sleep state.

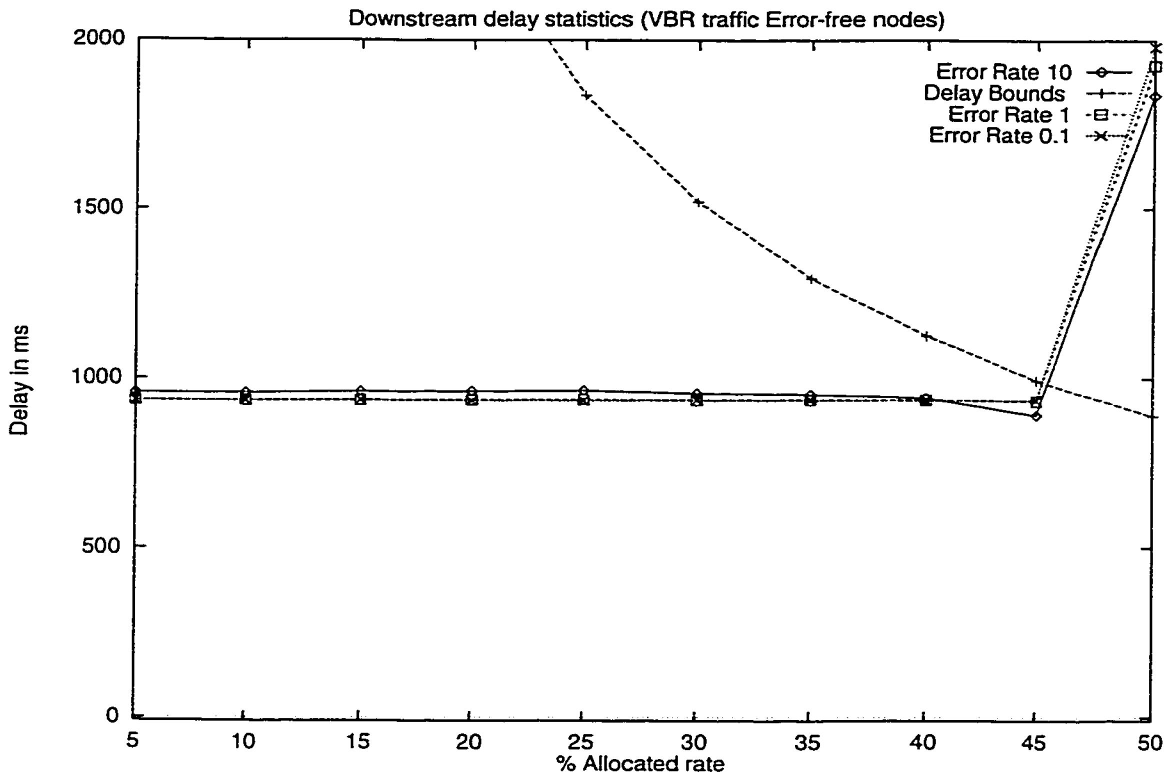

Figure 6.1: Allocated rate vs Delay (downstream traffic). Error-free nodes

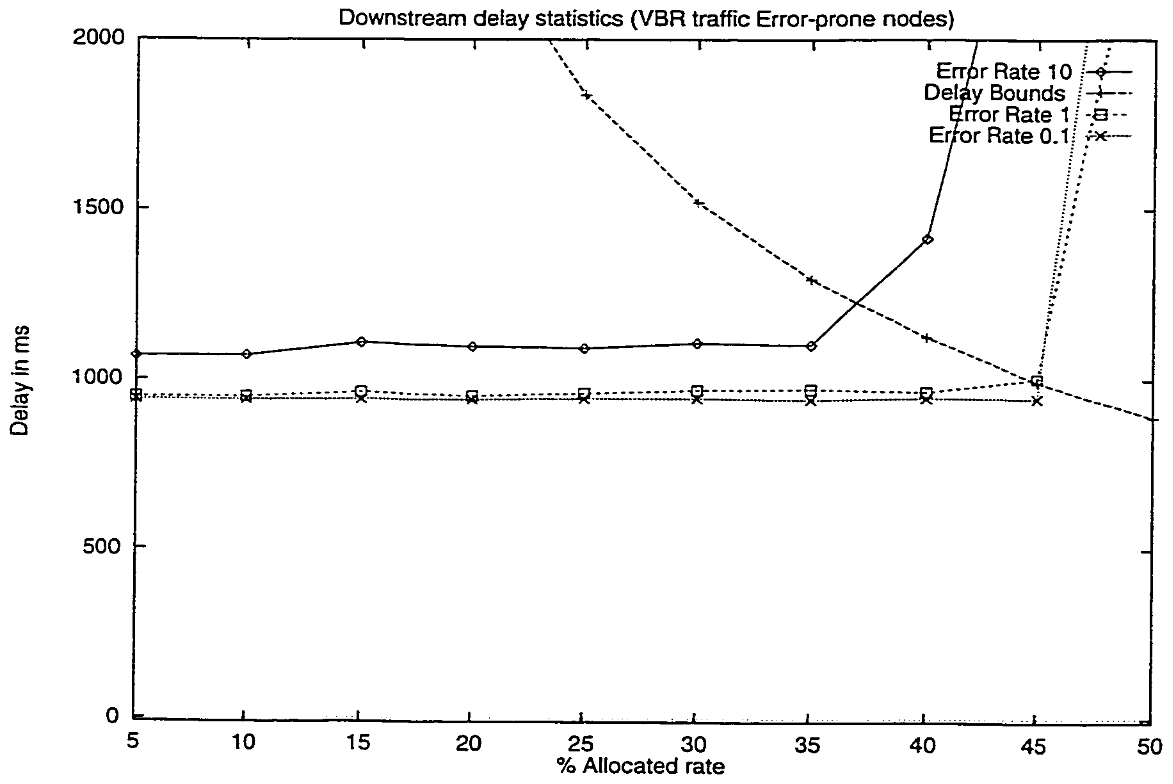

Figure 6.2: Allocated rate vs Delay (downstream traffic). Error-prone nodes

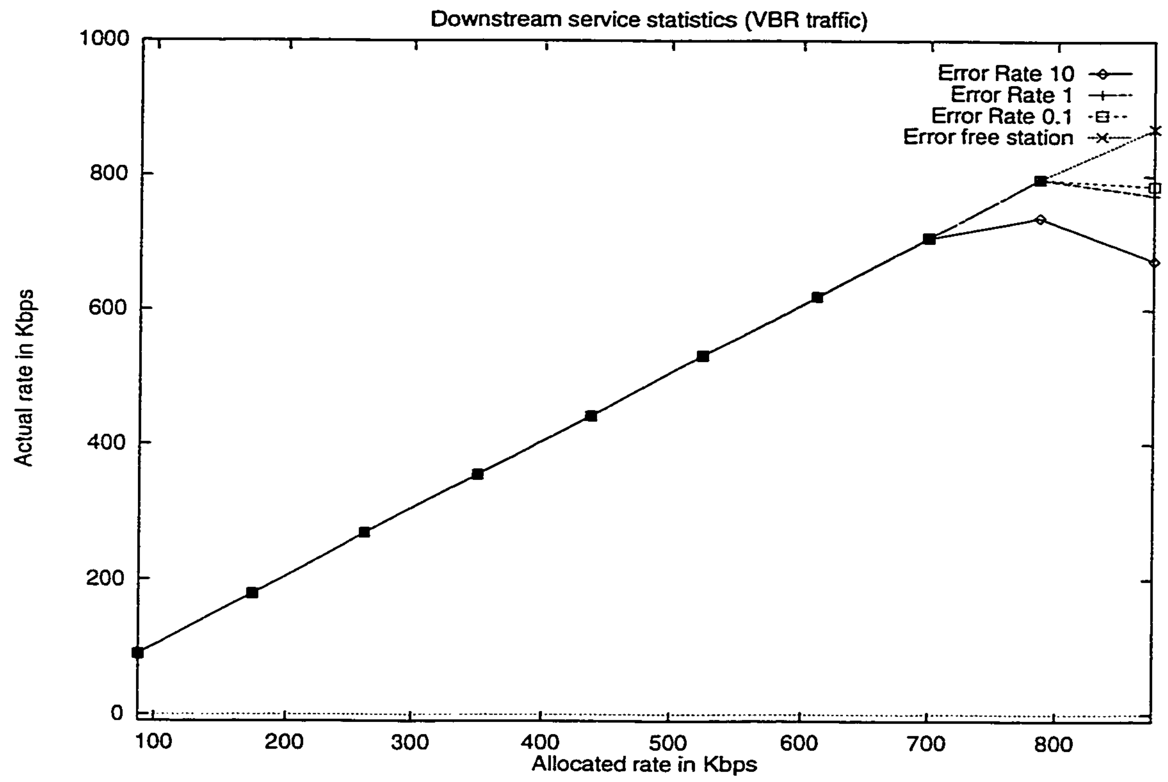

Figure 6.3: Allocated rate vs Actual rate in Kbps (downstream traffic)

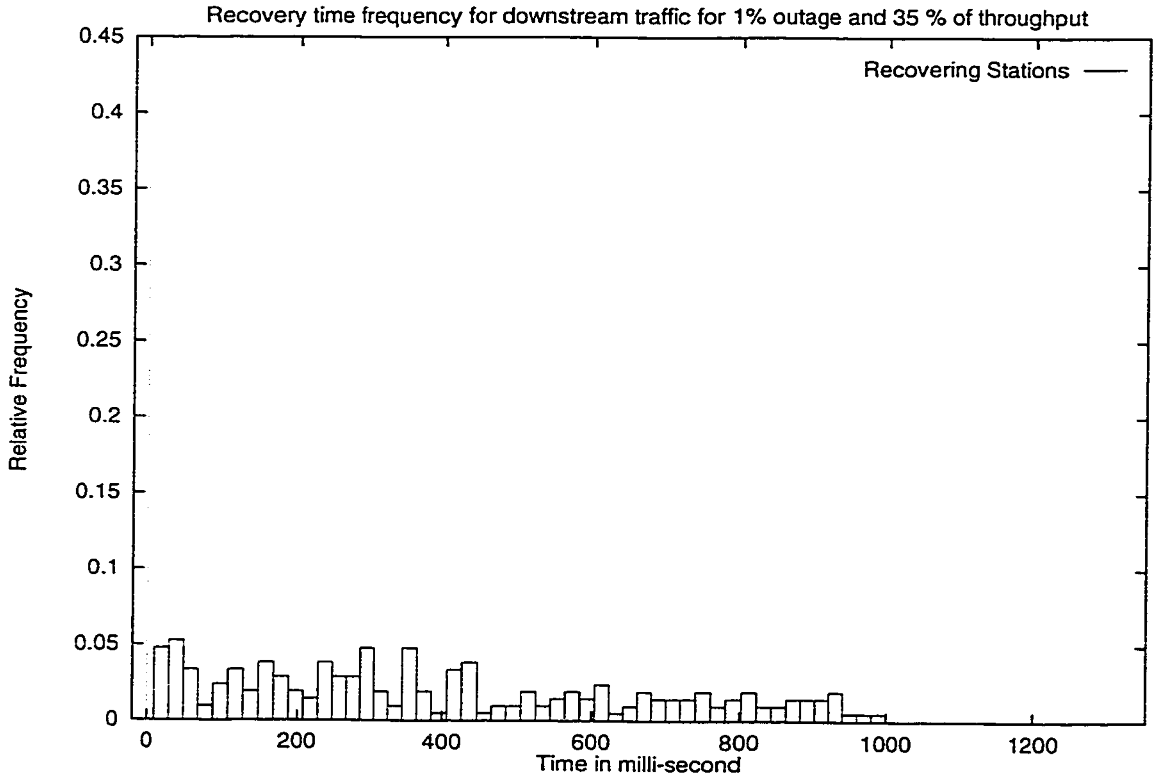

Figure 6.4: Recovery for 1% outage rate (Bound : 1339.28 msec)

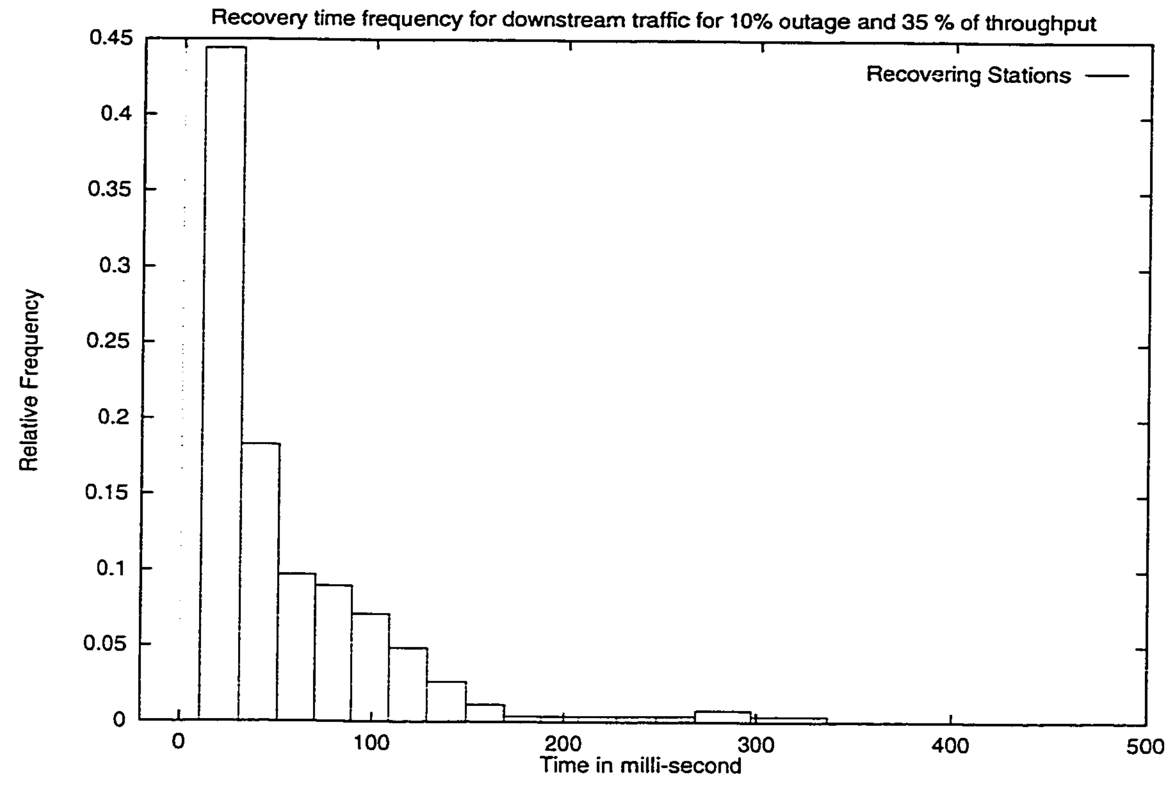

Figure 6.5: Recovery for 10% outage rate (Bound : 1339.28 msec)

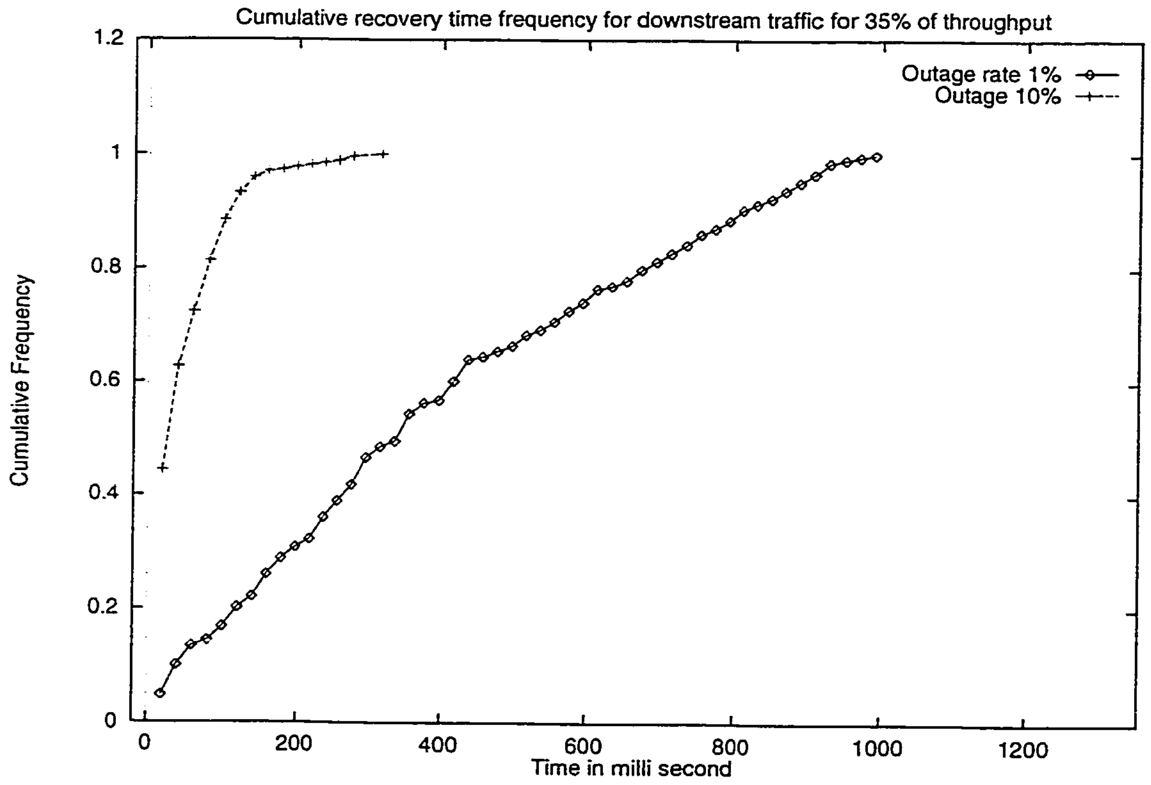

Figure 6.6: Cumulative recovery (Bound : 1339.28 msec)

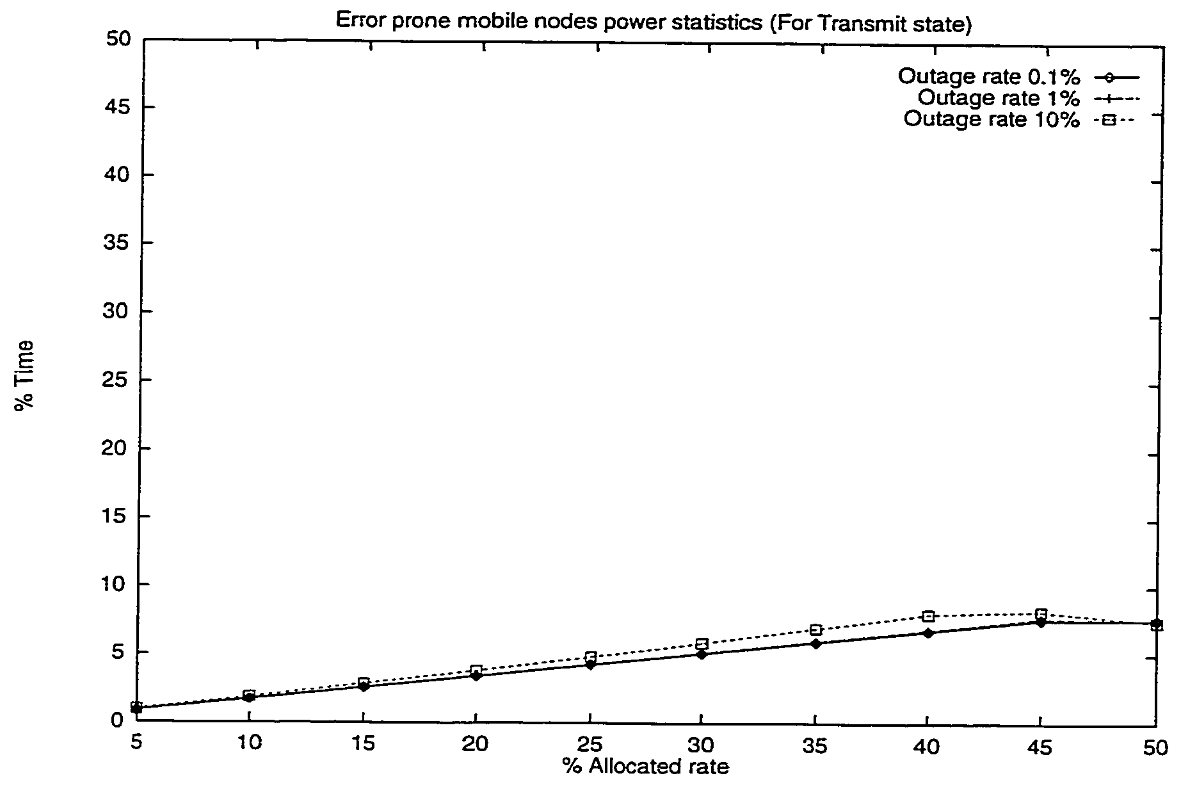

Figure 6.7: Mobile power consumption (Transmit mode) Error-prone nodes (CBR)

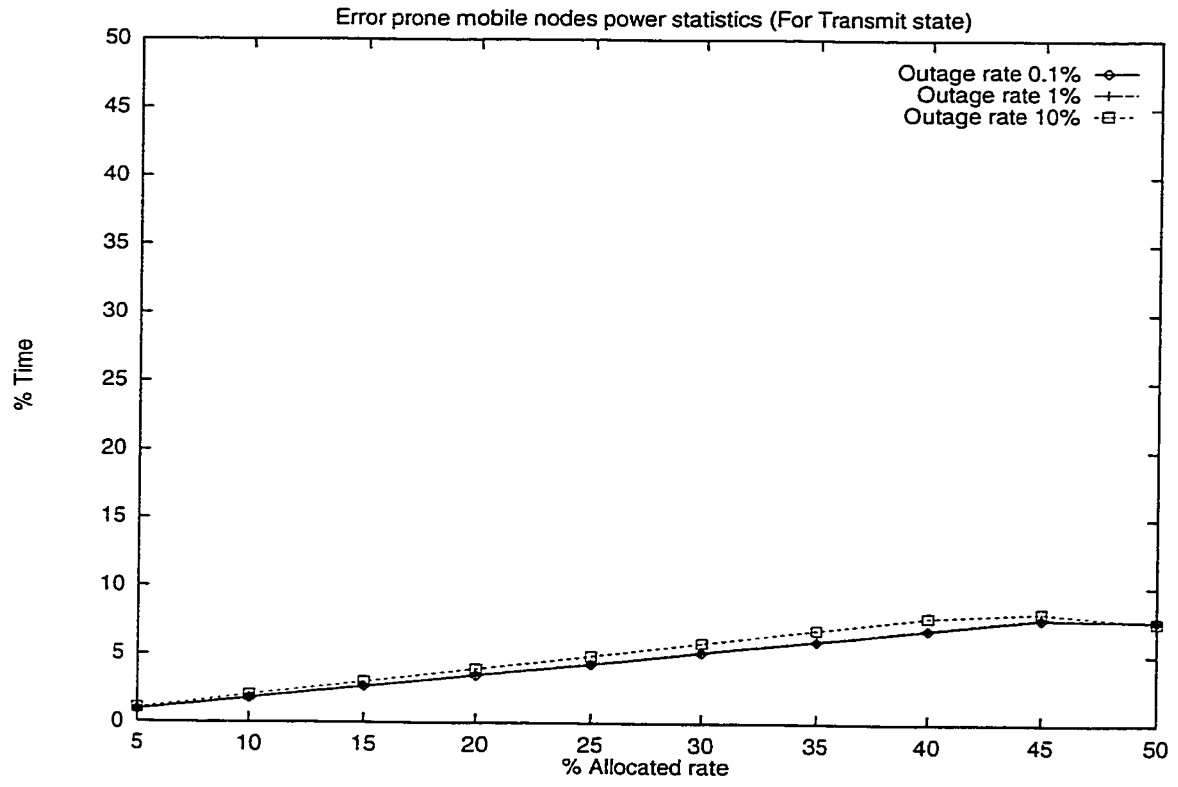

Figure 6.8: Mobile power consumption (Transmit mode) Error-prone nodes (VBR)

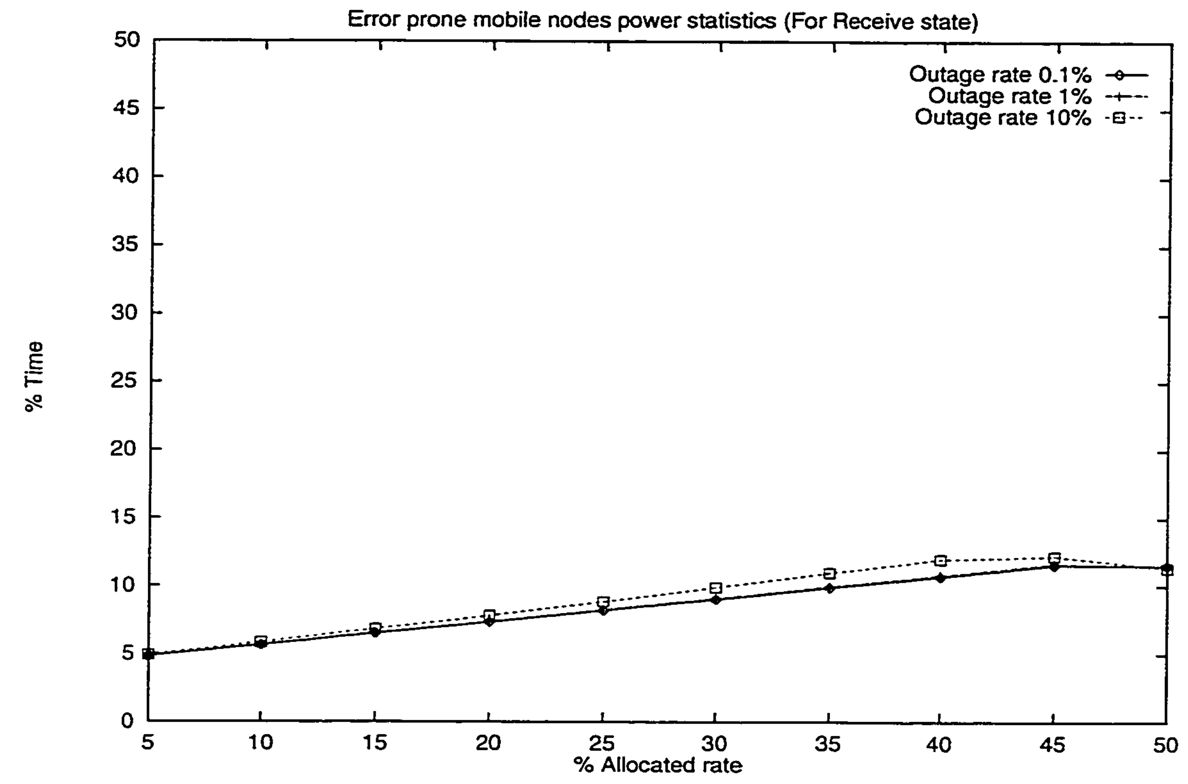

Figure 6.9: Mobile power consumption (Receive mode) Error-prone nodes (CBR)

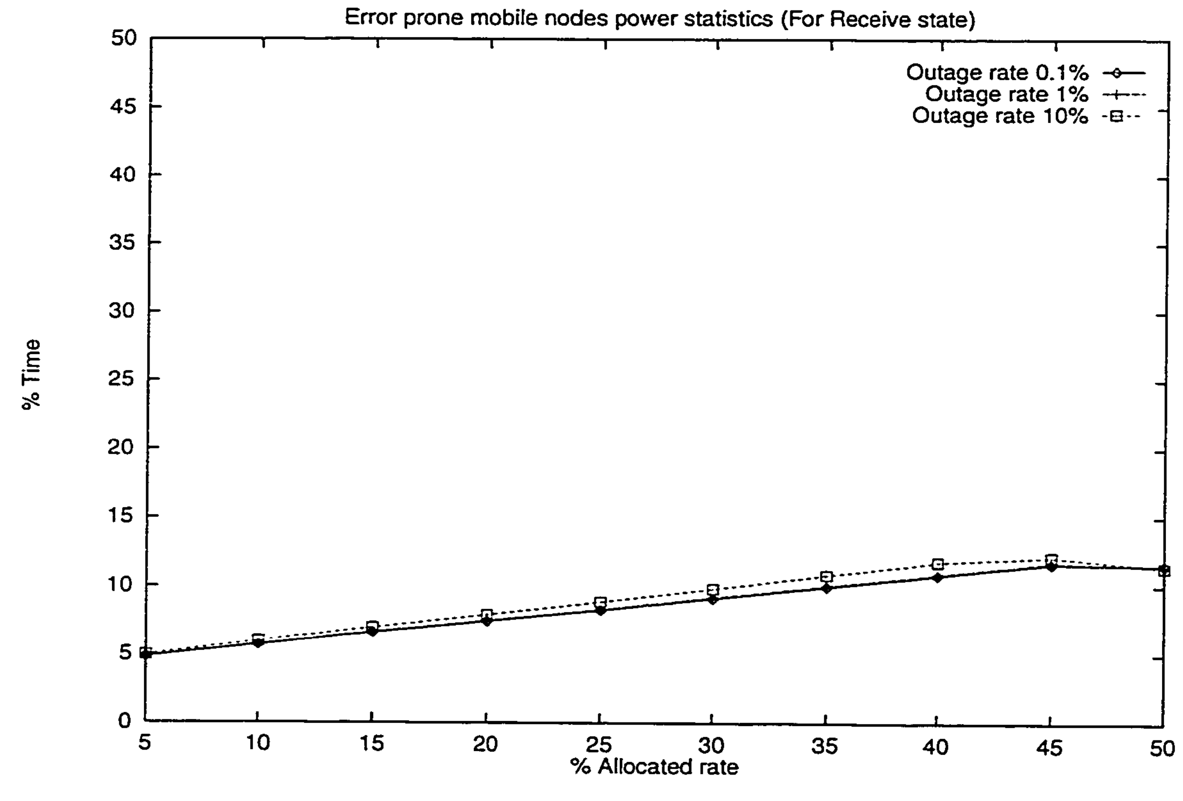

Figure 6.10: Mobile power consumption (Receive mode) Error-prone nodes (VBR)

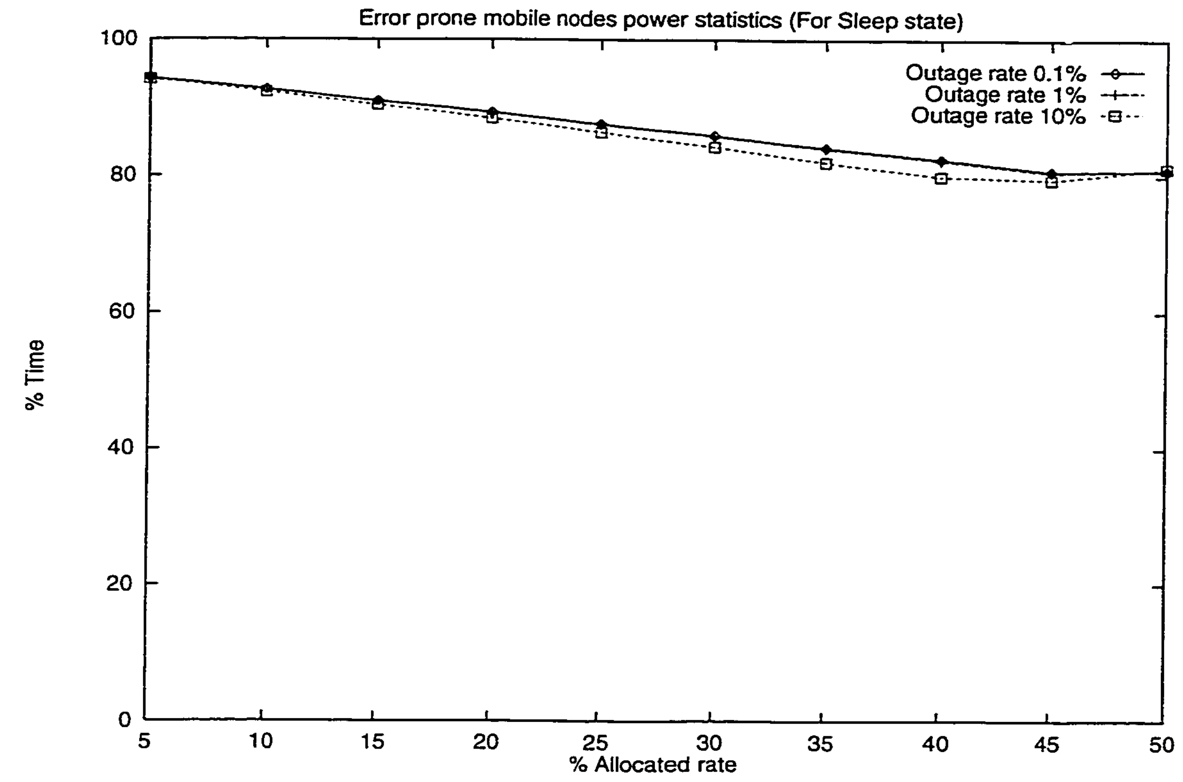

Figure 6.11: Mobile power consumption (Sleep mode) Error-prone nodes (CBR)

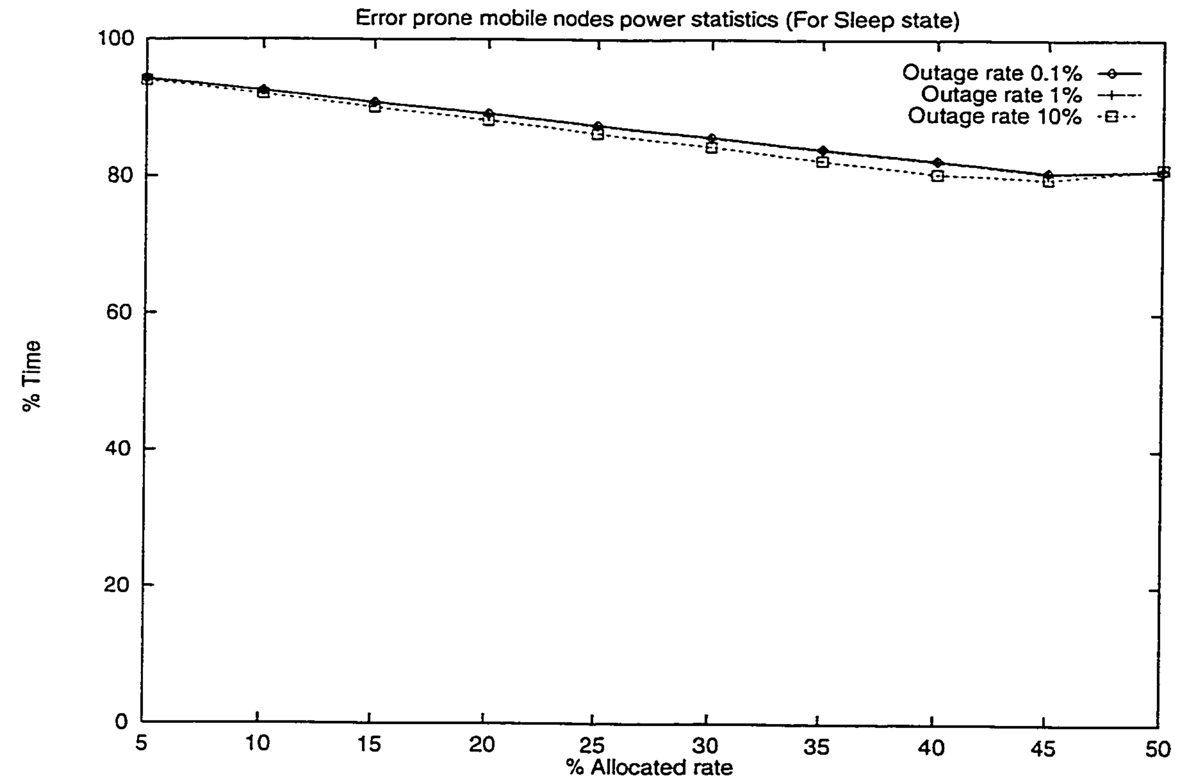

Figure 6.12: Mobile power consumption (Sleep mode) Error-prone nodes (VBR)

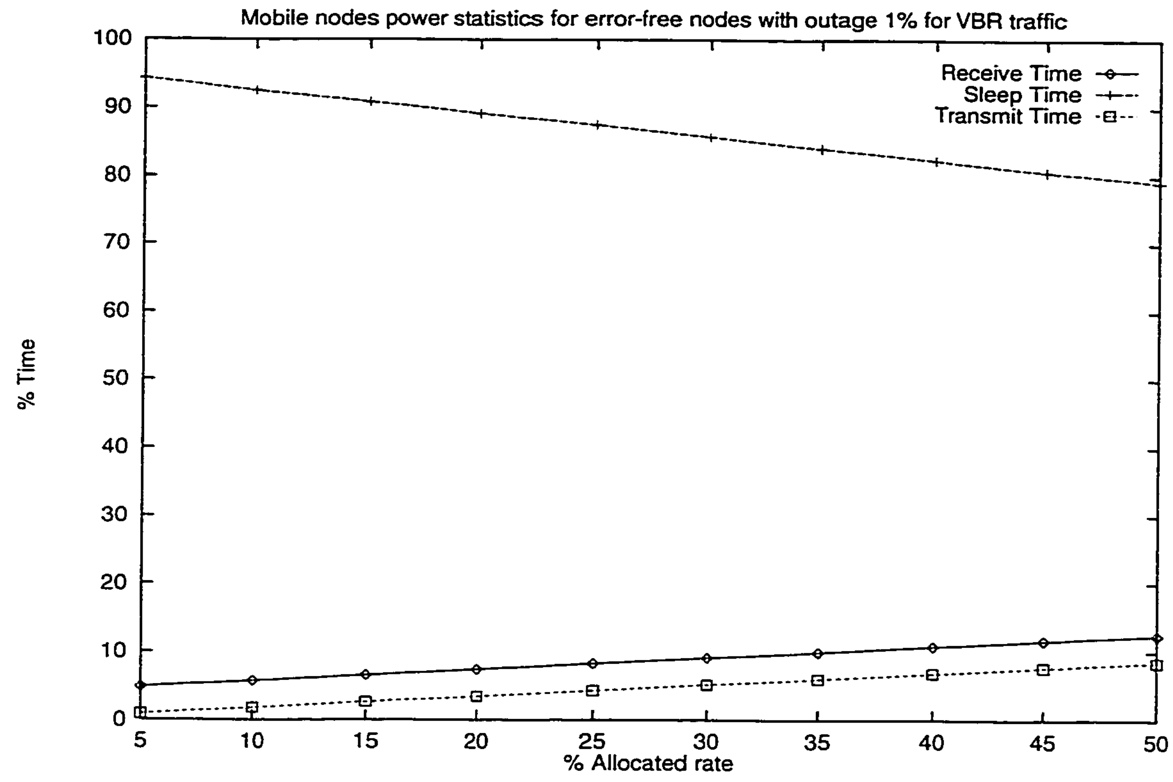

Figure 6.13: Mobile power consumption (Outage rate:1%) Error-free nodes

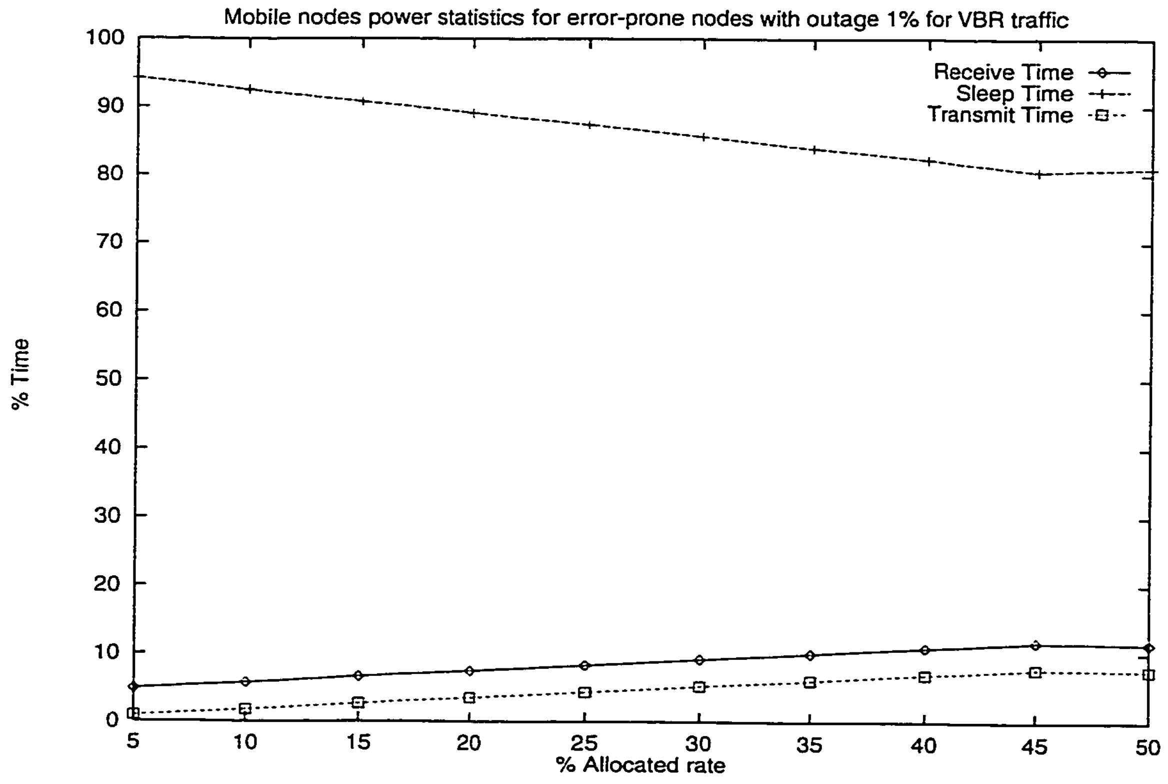

Figure 6.14: Mobile power consumption (Outage rate:1%) Error-prone nodes

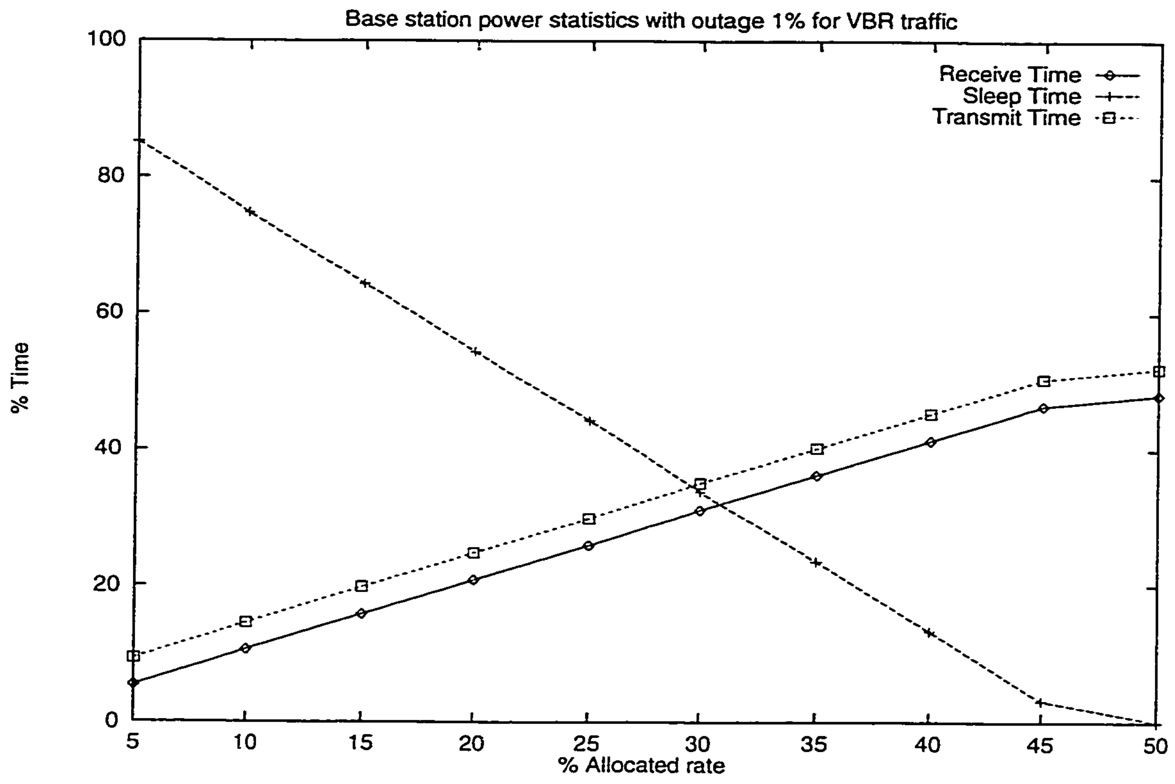

Figure 6.15: Base power consumption for 1% outage

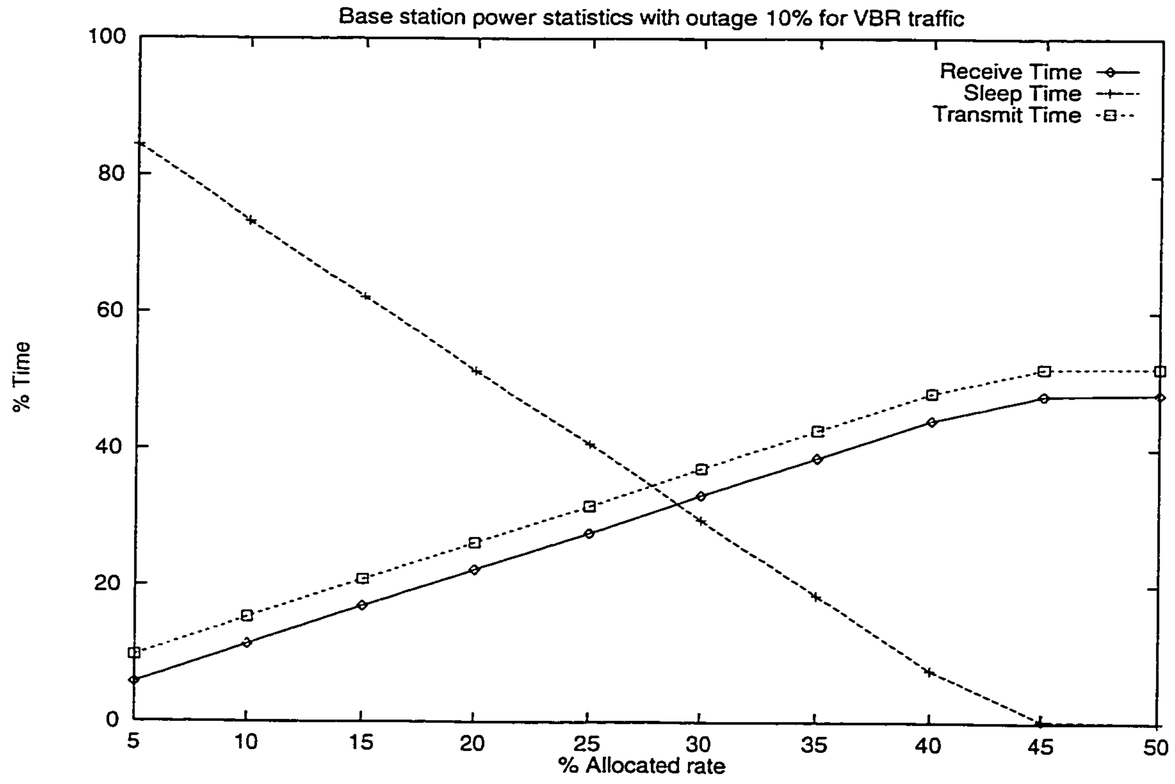

Figure 6.16: Base power consumption for 10% outage

Chapter 7

Conclusions and Scope for Future Work

7.1 Contributions

In the scope of this thesis we had two goals: to provide a realistic fair queueing algorithm and to conserve energy at the MNs.

7.1.1 Fair Queueing

Fair queueing algorithms for mobile wireless environment have been studied in this thesis. Various proposals made in the past have been presented and reviewed, with their merits and limitations.

PCFQ, provides *short-term* fairness (wire-line fairness) for error-free nodes and *long-term fairness* for error-prone nodes. PCFQ ensures that the error-free nodes are not affected by the error-prone nature of other nodes. Unlike the proposals in the literature [32, 22, 2, 31, 23], no assumption whatsoever is made regarding a-priori knowledge of the channel state, and the probing overheads involved to determine the channel state are accounted for in the protocol.

Although we are not able to provide analytical proofs, but the simulation results presented in this thesis demonstrate that the protocol is fair. The results show that strict delay guarantees can be achieved for the error-free nodes, while maintaining a bound on the recovery time for the error-prone nodes. Maximum utilization could be achieved for error-free nodes but in order to allow the error-prone nodes to recover from lost service accumulated during periods of channel error, a fraction of bandwidth has to be reserved. Since the advanced knowledge of the nature of wireless channel is not available, CAC algorithm has to account for spare bandwidth for recovery, thus reducing the overall channel utilization. During simulations, it has been observed that a high overall utilization can be achieved, while allowing error-prone nodes to recover from lost service.

7.1.2 Power Conservation

Separately from the fair queueing models, proposals have been made for conserving power in the mobile wireless environment. We studied the protocols for conserving power with their merits and limitations. An attempt has been made in this thesis to combine the features for power conservation and fair queueing.

PCFQ attempts to minimize power consumption at the MNs by introducing the notion of *cycles*. By using cycles of fixed length an MN switches to sleep state and wakes up to receive the schedule of the next cycle if it either did not hear the schedule or has nothing scheduled for the cycle. The complexity level of the protocol executed at the MN is also kept low in order to conserve energy. PCFQ is a pure contention free protocol for all the active nodes in the cell that have been accepted by the BS. Unlike the proposals in the literature [24, 17, 25, 37], PCFQ also minimizes losses due to the state transition of the wireless transceiver unit. The BS is entrusted to perform all the scheduling related activities thus reducing the workload of the MNs.

The simulation results presented in this thesis show that a MN remains in the sleep state most of the time, thus conserving energy. Due to the contention-free nature of the protocol, power consumption for re-transmission of packets is required only when channel error occurs.

7.2 Future Work

The scope of this thesis was limited to present a protocol capable of providing fairness and power conservation features in a mobile wireless environment. There exist various other dimensions of mobile wireless computing where the presented work can be extended. Even for achieving fairness and conserving power at MNs in a real testbed, improvements will be needed in the PCFQ protocol itself. This section covers various such areas where the present work can be extended.

7.2.1 Fair Queueing

The future work and extensions to what is proposed in this thesis can be grouped as follows:

Fairness model

The fairness model discussed in this thesis addresses wireless fair queueing in the context of a uni-cell wireless network. Though PCFQ is equally applicable across several neighboring cells, a complete solution requires substantial work on the MAC layer. Issues related to channel errors due to *interference* and the upstream/downstream frequencies allocated to adjoining cells would then be of concern. Also, providing fair packet scheduling in ad-hoc networks is a challenging task. A direct application of PCFQ in ad-hoc networks would necessitate a BS-like centralized unit.

Also, the ability to simultaneously provide different classes of services over a single physical infrastructure is one of the fundamental promises of high speed packet-switched integrated services networks. Although the protocol presented in this thesis does not in any way restrict the usage of PCFQ in such a *hierarchical link-sharing service* environment, there is a need to evaluate the fairness based on multiple quality of service (QoS) requirements.

Error model

The error model used for simulation in this thesis is very general and does not employ any kind of *channel prediction* strategies. However, if channel state prediction was possible a significant amount of probing overhead and loss due to channel error could be reduced. In other words, effective channel prediction strategies would help in developing more efficient fair queueing protocols for a mobile wireless environment. The development of effective channel prediction strategies is hence a promising research direction.

Another alternative to minimize the effect of channel errors or channel fading is using *error correction codes*. Error correction codes inflate the packet size and thereby reduce the overall channel utilization. However, to some extent they are capable of making channel fading transparent to the mobile user. Thus the tradeoff involved with error correction codes is between reduced overall channel utilization and the capability to transmit even when channel fading is observed. In the context of PCFQ, it might be worthwhile to see if there is any significant gain in using error correction codes for the control packets such as the cycle schedule, acknowledgment for upstream packets, and polling.

7.2.2 Power Conservation

Power at an MN is usually conserved by using hardware techniques and by employing software techniques to take advantage of lower power states of the component. In the scope of this thesis we focus only on the wireless communication unit of the MN and try to maximize the time the transceiver stays in the sleep state. Efforts should also be made to employ other software strategies as discussed in [11, 6, 7] to conserve more power at the MN.

7.2.3 Additional Considerations

Various other issues which are not directly related to fairness and/or power conservation, but offer a promising direction for research in the area of mobile wireless computing should also be considered.

Since we assumed that the BS will be responsible for locating the MN in cell, *location management* was not explicitly considered as an issue in the scope of our work. However, in all practical implementations an effective mechanism for location management would be necessary. *Mobile motion prediction* might also further improve upon the location management schemes [39, 40].

Another alternative to utilize the wireless channel more effectively is to design a new class of *adaptive applications.* These applications are able to exploit the changes in the QoS of the underlying communications infrastructure. Application specific rate adaption schemes and adaptive computing [41, 16] thus offer a potential to improve overall wireless network throughput, by exposing the time-varying nature of the communication channel to the applications.

Bibliography

[1] K. C. Chen, "Medium access control of wireless LANs for mobile computing," *IEEE Network*, vol. 8, pp. 50–63, Sept. 1994.

[2] V. Bharghavan, S. Lu, and T. Nandagopal, "Fair queuing in wireless networks: Issues and approaches," in *IEEE Personal Communications*, pp. 44–53, Feb. 1999.

[3] J. Sanchez, R. Martinez, and M. W. Marcellin, "A survey of MAC protocols proposed for wireless ATM," in *IEEE Network*, pp. 11:(6)52–62, Nov. 1997.

[4] G. Welch, "A survey of power management techniques in mobile computing operating systems," *ACM SIGOPS-OSR (Operating Systems Review)*, vol. 29, pp. 47–56, Oct. 1995.

[5] T. Martin and D. Siewiorek, "A power metric for mobile systems," in *Proceedings of the 1996 International Symposium on Lower Power Electronics and Design*, (Monterey, CA), pp. 37–42, 1996.

[6] J. R. Lorch, "A complete picture of the eneregy consumption of a portable computer," Master's thesis, University of California at Berkeley, Berkeley, Dec 1995.

[7] J. R. Lorch and A. J. Smith, "Software strategies for portable computer energy management," Technical Report CSD-97-949, University of California, Berkeley, June 1997.

[8] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” in *Proc. ACM SIGCOMM ’89*, (Austin, TX), pp. 1–12, Sept. 1989.

[9] K. M. Sivalingam, M. B. Srivastava, P. Agarwal, and J. C. Chen, “Low power access protocols based on scheduling for wireless and mobile ATM networks,” in *IEEE International Conference on Universal Personal Communications (ICUPC)*, (San Diego, CA), pp. 429–433, Oct. 1997.

[10] J. C. Chen, K. M. Sivalingam, P. Agrawal, and S. Kishore, “A comparison of MAC protocols for wireless local networks based on battery power consumption,” in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM’98)*, vol. 1, pp. 150–157, Mar 1998.

[11] M. Weiser, B. Welch, A. Demers, and S. Shenker, “Scheduling for reduced CPU energy,” in *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI): November 14–17, 1994, Monterey, California, U.S.A*, (Berkeley, CA, U.S.A), pp. 13–23, Nov. 1994.

[12] K. Li, R. Kumpf, P. Horton, and T. Anderson, “A quantitative analysis of disk drive power management in portable computers,” in *Proceedings of the Winter 1994 USENIX Conference: January 17–21, 1994, San Francisco, California, U.S.A* (USENIX Association, ed.), (Berkeley, CA, USA), pp. 279–291, Oct 1994.

[13] M. Zorzi and R. R. Rao, “Error control and energy consumption in communications for nomadic computing,” in *IEEE Transactions on Computers*, pp. 46(3):279–289, Mar. 1997.

[14] D. B. Johnson, “Routing in ad-hoc networks of mobile hosts,” in *Proceedings of the Workshop on Mobile Computing Systems and Applications. IEEE Computer Society*, (Santa Cruz, CA), pp. 158–163, December 1994.

[15] P. T. Davis and C. R. McGuffin, *Wireless Local Area Networks: Technology Issues and Strategies.* New York, NY, U.S.A: McGraw-Hill, 1995.

[16] R. H. Katz, "Adaptation and mobility in wireless information systems," *Personal Communications*, vol. 1, pp. 6–17, Mar 1994.

[17] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specification," *IEEE Standard*, vol. P802.11/D5, May 1996.

[18] J. Kruys, "HIPERLAN, applications and requirements," *In Proceedings of EFOC and N '93 June 30 - July 2*, vol. PON, pp. 104–108, July 1993.

[19] T. Imielinski and B. R. Badrinath, "Mobile wireless computing : Solutions and challenges in data management," Tech. Rep. TR-49, Wireless Information Network Laboratory (WinLAB) at Rutgers University, 1996.

[20] L. Kleinrock, "Nomadicity: Anytime, anywhere in a disconnected world," *Invited paper, Mobile Networks and Applications*, vol. 1, no. 4, pp. 351–357, 1997.

[21] S. Keshav, *An engineering approach to computer networking: ATM networks, the Internet, and the telephone network.* MA, U.S.A: Addison-Wesley, 1997.

[22] S. Lu, V. Bharghavan, and R. Srikant, "Fair queueing in wireless packet networks," in *Proceedings of the ACM SIGCOMM Conference : Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM-97)*, vol. 27(4) of *Computer Communication Review*, (New York), pp. 63–76, Sept. 14–18 1997.

[23] T. S. E. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM'98)*, vol. 3, (Pittsburg), pp. 1103–1111, Mar 1998.

[24] D. J. Goodman, R. A. Velenzuela, K. T. Gayliard, and B. Ramamurthi, "Packet reservation multiple access protocol for local wireless communications," *IEEE Transactions on Communications*, vol. 37, pp. 885–890, Aug. 1989.

[25] M. J. Karol, Z. Liu, and K. Y. Eng, "An efficient demand assignment multiple access protocol for wireless packet (ATM) networks," *ACM/Baltzer Wireless Networks*, vol. 1, no. 3, pp. 267–279, 1995.

[26] Z. L. Zhang, D. Towsley, and J. Kurose, "Statistical analysis of generalized processor sharing scheduling discipline," in *Proceedings, 1994 SIGCOMM Conference*, (London, UK), pp. 68–77, Aug. 31 - Sept. 2 1994.

[27] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 690–704, Oct. 1997.

[28] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of the 13th Annual Joint Conference of the IEEE Computer and Communications Societies on Networking for Global Communciation. Volume 2*, (Los Alamitos, CA, U.S.A), pp. 636–646, June 1994.

[29] H. Zhang and S. Keshav, "Comparison of rate-based service disciplines," in *Proceedings, of ACM SIGCOMM 1991*, (Zurich, Switzerland), pp. 113–122, Sept. 1991.

[30] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," in *Proceedings, of IEEE 1995*, vol. 83(10), pp. 1374–1399, October 1995.

[31] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks," in *Proceedings of the 4th Annual ACM/IEEE International*

Conference on Mobile Computing and Networking (MOBICOM-98), (New York), pp. 1–9, Oct. 25–30 1998.

[32] S. Lu, V. Bharghavan, and T. Nandagopal, "A wireless fair service algorithm for packet cellular networks," in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM-98)*, (New York), pp. 10–20, Oct. 25–30 1998.

[33] J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair-weighted fair queueing," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM'96)*, (San Francisco, CA, U.S.), pp. 120–128, Mar. 1996.

[34] R. Caceres, F. Douglis, K. Li, and B. Marsh, "Operating system implications of solid-state mobile computers," Tech. Rep. MITL-TR-56-93, Matasushia Information Technology Laboratory, May 1993.

[35] M. Stemm and R. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," in *Institute of Electronics, Information, and Communication Engineers Transaction on Communications*, pp. E80–B(8): 1125–1131, Aug. 1997.

[36] Y. Lu and R. Brodersen, "Unified power control, error correction coding and scheduling for a cdma downlink system," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM'96)*, (San Francisco, CA), pp. 1125–1132, Mar. 1996.

[37] D. Raychaudhuri and N. D. Wilson, "ATM-based transport architecture for multiservice wireless personal communications networks," *IEEE journal on selected areas in communications*, vol. 12, pp. 1401–1413, Oct. 1992.

[38] R. Cruz, "A calculus for network delay, Part I: Network elements in isolation," in *IEEE Transcations on Information Theory*, pp. 37(1):114–121, January 1991.

[39] P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Static and dynamic location management in distributed mobile environments," Tech. Rep. 94-030, Dept. of Comp. Sci, Texas A&M University and College Station, TX, June 1994.

[40] G. Liu and G. Maguire, "A predictive mobility management scheme for supporting wireless mobile computing," Tech. Rep. TRITA-IT R 95:04, Royal Institute of Technology, Feb. 1995.

[41] V. Bharghavan, "Challenges and solutions to adaptive computing and seamless mobility over heterogeneous wireless networks," in *IEEE Personal Communications*, vol. 4, pp. 1103–1111, Mar. 1997.