SCHEDULING MULTICAST AND UNICAST TRANSMISSIONS IN AN INFOSTATION ENVIRONMENT

DISSERTATION

Submitted in Partial Fulfillment

of the REQUIREMENTS for the

Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

POLYTECHNIC UNIVERSITY

By

Thomas J. Cortina

June 2003

Approved:

_____
Department Head

_____
Date

Copy No. _____

Approved by the Guidance Committee:

Major: Computer Science

_____

Phyllis G. Frankl

Professor of Computer and Information Science

_____
Date

Major: Computer Science

_____

Alex Delis

Associate Professor of Computer and Information Science

_____
Date

Minor: Electrical Engineering

_____

David J. Goodman

Professor of Electrical and Computer Engineering

_____
Date

VITA

Thomas J. Cortina was born in New York City in April, 1965. He was the valedictorian of the Class of 1987 at Polytechnic University, earning a Bachelor's degree in Computer Science with honors. In January 1989, he earned the Master's degree in Computer Science, and he has completed the requirements for the Doctor of Philosophy degree in Computer Science in June 2003. His research work described in this dissertation was completed during the period of June 2000- June 2003 as a member of the Wireless Internet Center for Advanced Technology (WICAT) at Polytechnic University working with its director, Dr. Phyllis G. Frankl. He was a lecturer at Polytechnic University from 1988-1991 and 1994-1997. During the period of 1992-1994, he worked as a research associate in the Center for Advanced Large-scale Computing (CALC) at Polytechnic University with its director, Dr. Donald Hockney. In 1998, he became a senior lecturer at the State University of New York at Stony Brook. He has won two awards for his teaching, and he has published several papers in the area of computer science education. In addition to his interest in education, his other research interests include mobile computing, programming methodology, and computer music.

ACKNOWLEDGMENTS

I wish to thank Dr. Phyllis G. Frankl for her unwavering support during this research endeavor. Despite the high workload that this author assumed, Dr. Frankl was extremely helpful in keeping the research work focused, providing positive feedback and useful critiques when necessary. This dissertation could not be accomplished without her assistance.

I also wish to thank Dr. David J. Goodman for his knowledge of wireless systems and his initial ideas about infostations that gave birth to this work. His words of wisdom and insightful comments about the derivations and simulation will continue to guide me in future research work.

I would like to thank Dr. Alex Delis for providing valuable feedback on my research as well as allowing me to take up space in his lab while this work was completed.

To Vlad Shkapenyuk, Mike McGrath and Timothy Rodolico who worked on undergraduate projects supporting this research work, I extend my sincerest thanks.

My family and friends have stood behind me during this long and tiring process, and I could not have made it without their support and love. Thank you to Dad, Mom (watching from above), my brother Lou, and the entire family for believing in me.

Most importantly, my deepest gratitude goes to Arno, who went far beyond the call of duty to help me with computer problems and to give me encouragement, strength and friendship during the really tough days. Arno, without you, this dissertation would not be complete. Thanks to you, it finally is.

AN ABSTRACT

SCHEDULING MULTICAST AND UNICAST TRANSMISSIONS
IN AN INFOSTATION ENVIRONMENT

by

Thomas J. Cortina

Advisor: Phyllis G. Frankl

Co-Advisor: David J. Goodman

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy (Computer Science)

June 2003

An infostation is a wireless data disseminator that provides data to mobile terminals within its limited range at high transmission speeds. Users of an infostation subscribe to information categories and receive relevant data from the infostation automatically as they pass through the infostation's transmission range. Data from an infostation is meant to be transmitted quickly to the mobile terminal and consumed later in an offline setting. Due to the limited amount of time that a terminal is within range of an infostation, an important goal is to minimize the delivery time for a data item in order to maximize the likelihood that the item will be transferred successfully to the mobile terminal before its departure. This dissertation uses mathematical analysis and simulation to investigate how to schedule data transmissions using multicast and unicast transmission techniques.

A single-channel model of an infostation environment is analyzed mathematically using M/D/1 queueing principles, and simulations are performed on this model to verify the analysis. In addition, an alternate broadcast model for the infostation is presented and

analyzed through simulation, and a simulation of the effect of packet errors on the standard model using a basic IEEE 802.11b packet error model is presented. We show that for light terminal traffic in the infostation range, unicast transmission is adequate to handle all requests, but as the traffic increases, the system should begin to use multicast, however all of the data items should not be multicast. Typically, only a small number of the most popular items should be multicast in order to achieve optimal performance of the infostation.

By analyzing the arrival rate of terminals into the infostation area and the distribution of items downloaded, the infostation designer can use these results to determine the specific number of multicast items to transmit per cycle to maximize the likelihood that a terminal will receive its desired item before it leaves the infostation's transmission range.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

In the past two decades, data broadcasting has been a popular area of research in the database and telecommunications fields. The major goal of data broadcasting is to provide end users with their required information in an optimal way. Optimality can be defined in different ways in these environments. A data broadcast scheduler may want to minimize the waiting time for a data item over all users, or the scheduler may want to minimize the number of requests made to the data server in order to reduce the load on the server. In the past, these studies have been performed assuming a wire-based network, where all terminals are constantly connected to the data transmission network and have unlimited power. Also, due to the development of transmission protocols like TCP/IP, data transmission errors and their effect on the data broadcast scheduling algorithms generally has not been studied for wired networks.

With the increased popularity of wireless communication networks for mobile users, the focus on data broadcast scheduling has increased, but the research has focused on some new challenges. In a wireless environment with mobile end users, constant connection to the communication network is no longer assumed. Therefore, schedules must be designed that maximize the number of users that receive their data items in a system where users are connected for short, unpredictable periods of time. Additionally, transmissions over wireless networks suffer from a much higher rate of loss due to interference, fading and other side effects when transmitting over the air to a wireless terminal. New research must focus on how to effectively utilize the transmission bandwidth in the presence of high transmission error rates. Besides these issues, power management plays a key role in the development of scheduling algorithms, since mobile terminals require the use of battery power which is extremely limited. End users not only use battery

power to run their terminals for ordinary computing tasks, but they use additional power during wireless communication which drains the available battery power further. An additional goal of the scheduler is to minimize the amount of power usage during a wireless data transfer in such an environment.

All of the tasks outlined above compete against each other as the wireless broadcast scheduler builds an optimized broadcast for the end users. In order for the scheduler to know what to broadcast, end users must send wireless messages to the server to indicate their presence and their desired transactions, draining some battery power. As the mobile users move around, the data transmission may be interrupted due to dropped data packets, and additional requests may need to be made to complete a data transaction. The scheduler will receive competing requests from a number of end users and must determine how to serve these different requests in a fair and equitable manner.

New research in mobile wireless data broadcasting differs from traditional data broadcasting research in several ways. Mobile wireless broadcasting must take into account the limited connection time of end users. It must consider the power usage of the end users in developing a viable schedule by minimizing the number of wireless requests from a user and by limiting the amount of time a user has to monitor the wireless broadcast channel for its data. Finally, it must also analyze the incoming requests for data to determine an optimal schedule that meets the needs of as many end users as possible.

This study analyzes mobile wireless data broadcasting in the context of an infostation, which is a wireless localized high-speed data dissemination system for mobile users. Users in the vicinity of an infostation can download or upload information at a high data rate for use at a later time offline. In an infostation system, mobile terminals are able to detect the presence of an infostation automatically without user intervention and receive only data pertinent to that mobile user based on the user's preferences.

In this study, we look at the issue of comparing the use of multicast and unicast transmissions in such a wireless environment. Multicast transmission can reach multiple

users at once, while unicast transmissions are one-to-one transmissions between the infostation server and the individual end user's terminal. Multicast transmissions can reach many users at the same time, reducing their waiting time for information, but works well only if many of the users want the same data items. Also, multicast transmission requires additional time by the end user during setup, and users must know the required multicast address to monitor which may vary from one infostation to another. Unicast transmission, on the other hand, reaches only one user at a time.

The major goal of this research is to study the conditions that are necessary in an infostation environment for using multicast, unicast or a combination of both transmission strategies. In our study, we generally assume items that are the same size and a transmission schedule that is cyclic and of fixed length (i.e. a multicast item is transmitted on a periodic basis at equal time intervals from one transmission to the next).

We have found that, in general, for infostations with a low arrival rate of new mobile terminals, unicast transmission is the optimal choice to minimize the expected waiting time and therefore maximize the probability that the terminal will receive its desired item(s) before leaving the infostation coverage area. As the arrival rate of new terminals increases, we show that the expected waiting time can be minimized by multicasting selected items from the database. The number of items to multicast depends on the arrival rate, the distribution of requests from the mobile users, and the size of the database. We also show that multicasting all items of the database does not minimize the expected waiting time for the end users in general.

The results in this study are determined by a combination of mathematical analysis and computer simulation. Mathematical analysis of several infostation models is discussed using standard queuing theory. Computer simulation is performed on these models and on additional models that are harder to analyze mathematically. One model introduces transmission errors into the infostation system to see how these affect the expected waiting time for mobile users. The computer simulation results support the conclusions derived

using queuing theory and suggest how many items of the database need to be multicast in order to maximize the likelihood that a mobile user will receive the desired data item before leaving the infostation area.

An infostation system designer can use the expected arrival rate of terminals in the infostation coverage area, the expected request distribution of these terminals, and the size of the database to determine the optimal schedule for the mobile users. By constantly monitoring the incoming requests, the designer can adjust the broadcast schedule dynamically to changing conditions such as a high arrival rate (e.g. during a "rush hour") or a highly skewed request distribution (e.g. during a "public emergency").

The organization of this dissertation is as follows:

- Chapter 2 discusses the infostation model in general and defines the terminology associated with infostations.

- Chapter 3 reviews previous data broadcast scheduling studies and briefly discusses how these studies differ from the study presented in this dissertation.

- Chapter 4 presents a mathematical analysis of an infostation model where the broadcast channel is used to multicast and unicast items alternately using the entire bandwidth available. Queuing theory is used to analyze this model, and graphs are presented to show the optimal number of items to be multicast from the database given the database size, arrival rate of terminals and expected request distribution of those terminals.

- Chapter 5 presents results from a computer simulation of the infostation model presented in chapter 4. The results of the simulation are compared to the mathematically derived results from chapter 4 to show that the results generally agree with the mathematical model. In addition to these results,

two additional infostation models are presented, one in which all items are multicast (some periodically and some only on demand by users), and another in which data transmission errors are introduced into the model. In each of these cases, we compare how these changes in the model affect the expected waiting time and how they would affect the power usage of the mobile terminals.

- Chapter 6 summarizes the results of this study and outlines future areas of related research for this infostation model and simulator.

- An appendix covering more detailed aspects of previous work used in the error modeling presented in Chapter 5 and a set of references conclude the dissertation.

Chapter 2

THE INFOSTATION ENVIRONMENT

Cellular technologies are designed with the assumption that service is required at all locations and at all times within the cell. Cells are placed so that mobile users can maintain uninterrupted service when moving from one cell to another. However, in such systems, the achievable data rates are often quite low, and even 3G technology will provide low data rates compared to wireless LAN technology.

In contrast, an infostation is a high-speed data disseminator that has a very small range of service but offers high data rates [9]. An infostation based on IEEE 802.11b technology could have a maximum range of approximately 100 meters at a maximum transmission rate of 11 Mbps [25].

2.1 Introduction

The basic model of an infostation is shown in Figure 2.1. For purposes of this paper, it is assumed that the coverage area of the infostation is circular in nature, although in real-world environments, this is not the case.

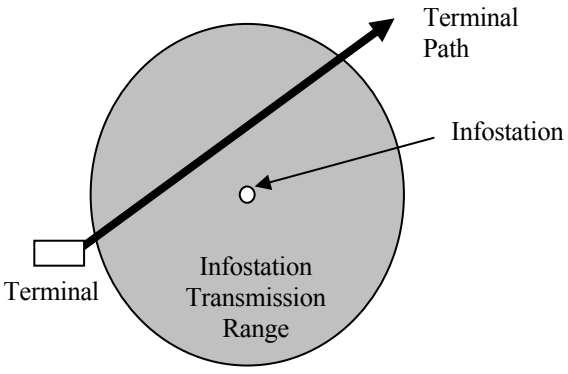

Figure 2.1 The infostation model.

The infostation emits a “beacon” periodically to alert mobile terminals of its presence while it transmits data to users within range of the infostation. As a mobile terminal enters the transmission range, the terminal detects the beacon and listens for any relevant data it requires. Our infostation transmits in cycles, with each cycle containing an index with information about the data items to be transmitted during that cycle. Terminals process the index, determine when required data is to be transmitted, listen for the data, and download data at the proper times.

The data that is transmitted from the infostation is classified into two categories: *multicast* and *unicast*. Multicast items are transmitted regularly without requiring any requests from the users in the infostation area. This type of item is multicast to users in order to save bandwidth by serving as many terminals that want the item as possible. On the other hand, a unicast item is not transmitted unless a terminal requests it. This transfer is performed using a one-to-one exclusive connection from the infostation to the terminal.

Since the terminals are mobile, they will be in the range of the infostation for a limited time. Typically, infostations fall into two modes: *drive-through* and *walk-through*. In the drive-through mode, the terminal is in an automobile or other high-speed vehicle, and the time in the range of broadcast can be several seconds, whereas in the walk-through mode of operation, the terminal is carried by a pedestrian and would be in the broadcast range for approximately 100 seconds or more typically, assuming that the terminal is traveling at a constant rate of speed of 1 m/sec across the diameter of the infostation area and the infostation is transmitting at a maximum rate of 11Mbps using IEEE802.11b. Our study considers the walk-through mode only.

The key to the infostation environment as compared to a cellular network is that this reception and transfer of data is automatic and requires no human intervention with the exception of some initial configuration of the mobile terminal by the user. The infostation and its server will need to determine what items need to be transmitted at any given time based on such factors as anticipated user arrivals, priority of users requiring service,

previous distribution of requests for various data items, amount of bandwidth available, and external conditions such as interference. In addition, the terminals will need to keep track of the data items only partially received due to the limited amount of time in the infostation area or the effect of transmission errors.

2.2 Infostation System Components

Most wireless data broadcast models are based on the system diagram shown in Figure 2.2. The system consists of a database and broadcast server connected to an infostation. The infostation transmits information from the server over the air to mobile terminals in its vicinity. These terminals can also communicate to the infostation (and thus, to the broadcast server) using an uplink channel. Each of these elements is described below.

- Database

  The database contains *N* data items, not necessarily of equal length. In this study, we will assume that the items are of equal length, but future work will consider items of varying lengths.

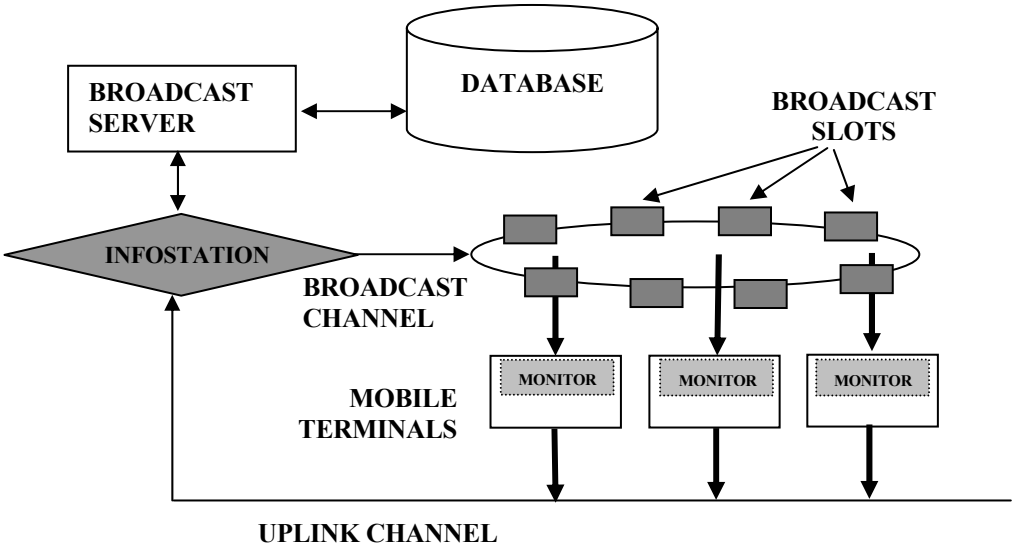

Figure 2.2: System diagram for the infostation model used in this study.

When the server broadcasts an item, the item is chopped up into packets (with padding in the last packet if necessary to make all packets the same size).

Additionally, a fixed number of packets can be grouped together into a chunk, so that a terminal that misses part of a data item can request and/or listen for the relevant chunk(s) rather than the entire item. In this dissertation, we do not describe the use of chunks further since we assume in most of this study that there are no transmission errors; therefore, a terminal does not need to request an item more than once when it enters the infostation transmission range. Future work that studies the effect of transmission errors should consider the benefits of subdividing an item up into chunks to improve the overall performance of the infostation.

Each data item has a demand probability which indicates the likelihood that an incoming terminal will want that item transmitted. We assume in this study that the demand probability of each item in the database is known *a priori*. In general, software in the broadcast server can monitor the number of requests for each item and adjust the demand probabilities accordingly. In the simulations described in chapter 5, we assume that the length of time studied is short enough that the demand probabilities of each item are constant.

- Broadcast Server

  The broadcast server has a connection to the database. This server runs in cycles, where each cycle has four phases as shown in Figure 2.3.

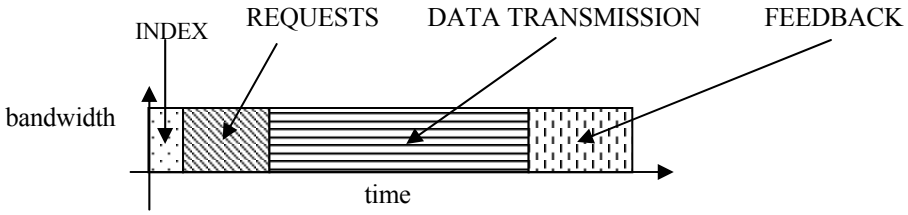

Figure 2.3: The broadcast server cycle (not drawn to scale).

The server starts by building an index of the items to be transmitted during that cycle which is sent to the infostation for transmission to the terminals within range. The server then listens for and processes any requests made by terminals in the infostation area that are captured by the infostation. These requests will determine the data items transmitted in future cycles. Once the request period is over, the server transmits the scheduled data items (or parts thereof) through the infostation one at a time based on a specific scheduling algorithm.

Finally, the server pauses again to receive feedback via the infostation from the terminals that received complete items. This information will be used by the server in the future to adjust its scheduling policy for changing terminal requirements.

The length of each cycle may be fixed or variable. A fixed-length cycle has an advantage in that arriving terminals will know approximately when the next index will be transmitted and can "sleep" for a fixed period of time if there is nothing of interest in the current cycle. However, the length of the cycle may be artificially long in some cases, delaying terminals from getting their data as quickly as possible. On the other hand, a variable-length cycle can define its cycle length based only on the amount of data to be transmitted at the given time (up to some limit). This minimizes the amount of wasted time in the transmission cycle, but it makes terminal management more complicated since terminals need to estimate the length of each cycle individually in order to reduce power consumption effectively.

Additionally, newly arriving terminals will have no information concerning the time of the next index and must monitor the wireless channel for the entire time, using valuable battery power.

- Infostation

  The data items and index are sent from the broadcast server to the infostation for broadcast over the air on the assigned broadcast channel. Conceptually, the infostation could be the server itself, but in our model we choose to separate it logically from the server and its functions, since several servers could utilize a single infostation for data transmission.

  The infostation can also receive specific requests for data items and feedback from the mobile terminals through an uplink channel. These two channels may be distinct communication channels, or they may be the same channel multiplexed for each function. In our model, we assume that requests, feedback and data transmissions are multiplexed on a single channel.

  Typically, an infostation uses an access point connected to a local area network. An important concern is that this access point may be used for other applications (for example, web surfing, email downloads, etc.) in addition to the data dissemination application described in this study. For this study, we assume that the amount of wireless bandwidth dedicated to the data delivery application is constant through time.

- Mobile Terminals and Monitors

  There are mobile terminals within the broadcast range of the infostation. In this study, we assume each terminal needs one item from the database at a time; this is common in most data broadcasting studies, although this restriction is not necessary in general.

Each terminal contains a monitor that keeps track of the current state of each item that the terminal needs. The monitor listens to the transmission for the next occurrence of the index to determine if a data item in the new cycle needs to be downloaded by the terminal. If the monitor determines that the terminal needs an item that is not being transmitted by the infostation in the current cycle, the monitor can transmit a request to the infostation for the item during the request phase of the cycle. We do not consider collisions on the transmission channel during the request phase in this simulation because the expected number of requests and size of the requests is small enough that a standard back-off algorithm can be used to handle collisions such as the algorithms used in the 802.11 protocol. [18]

During the data transmission phase of the current cycle, we assume each terminal can "sleep" (i.e. run in low-power mode) until there is data to download, in order to save power optionally. At the end of the data transmission phase, a terminal that has received a complete item can transmit feedback to the server (ignoring the effect of collisions). The server can use this feedback to adjust its scheduling algorithm when the needs of a terminal changes over time.

2.3 Key Objective Functions

The most common objective function cited in broadcast system studies is the average response time for a terminal, where response time is defined as the difference between the time that a terminal first needs a data item and the time that the terminal retrieves the item successfully from the broadcast. In an infostation environment, data items are typically not time-critical, as long as the terminal receives the data item while within range. We consider a terminal's trip through the coverage area successful if the terminal receives its desired data item completely while it is in the infostation transmission range.

Since terminals are in the infostation area for only a limited period of time, we wish to design a scheduling technique that satisfies as many terminals as possible. In other words, we wish to minimize the *probability of loss*, denoted P_l, which is defined as the

probability that a terminal will not receive the entire item by the time it leaves the infostation area. We assume in this study that a terminal must receive an item completely in order to be able to use it later after the terminal leaves the infostation area. Future work can consider cases where a terminal only needs to receive a given percentage of the item in order to be able to use it.

In addition to the probability of loss, we would like to examine how long it takes on average for a terminal to receive its item completely, denoted the *expected waiting time*, $E[W]$. This measure begins with the time the terminal has to wait for the first broadcast index after it enters the infostation area and ends when the terminal receives its item entirely. A study of the expected waiting time can help us understand how long an average terminal needs to remain in the coverage area in order to receive a successful transmission of its desired item.

2.4 Summary of Recent Infostation Research

The infostation concept is a relatively new design for a wireless system, developed initially at the Wireless Information Network Laboratory (WINLAB) of Rutgers University of New Jersey and is described in [9]. This section outlines a few major contributions in this area.

Iacono and Rose studied the delay in delivering file data for drive-through and walk-through modes of an infostation system [13]. The system that is mathematically analyzed consists of a number of infostations set up in linear and grid patterns. Data items that need to be transferred are split into segments that are distributed amongst the relevant infostations in order to serve high-speed terminals that pass by a sequence of infostations. Each infostation broadcasts its segment so that the terminal can receive the entire item. The authors present a near-optimal algorithm for file delivery that minimizes the overall delay for the terminals.

More recently at WINLAB, a study has been published that considers a model that contains a single infostation and mobile terminals that can also transfer data to other mobile terminals [29]. This model behaves somewhat like an ad-hoc network, such that each terminal acts like an infostation to relay information beyond the infostation's regular broadcast range. The authors assume unicast transmission between terminals, and only one transmission can occur from a terminal to another terminal at any given position in the model (i.e. unicast). By using analysis and simulation, the authors show that the performance of this model depends on the density of the terminals, the mobility of the terminals, and the number of data files that need to be disseminated to all terminals from the infostation.

In [7], the result of a simulation study is presented for an environment similar to an infostation where all items are transmitted using broadcast technology so all terminals can be served with a single broadcast of an item. Some items are broadcast automatically based on their anticipated popularity, and others are broadcast only on demand based on a terminal request. The parameters for the simulation are geared for a PCS system of mobile phone users which require typically small data items, whereas our study encompasses the transfer of very large data items such as MP3 music files or AVI movies to mobile computers.

Additional research using the infostation model has included the development of specialized Media Access Control (MAC) protocols [27] and the study of a mobile-aware application for tourism [28].

Chapter 3

DATA BROADCAST RESEARCH

A significant amount of research has been performed in the past decade that has focused on optimal ways to broadcast data wirelessly from servers to clients based on various optimality criteria. This chapter summarizes some of the most important research results from previous work, and shows how this research differs or influences our research work.

3.1 Data broadcast (push-based) scheduling

The simplest type of data broadcast is a pure push-based broadcast, also known as a flat broadcast, where data items are broadcast in a round-robin fashion without regard to the popularity of the items or the arrival rate of the terminals near the broadcast point. Obviously, this technique is very simple to implement, but it suffers from a very poor expected waiting time for the delivery of an item to a terminal.

If the probability that an item is requested is known, then a simple adjustment can be made to the flat broadcast to form a skewed broadcast [26]. Assume the probability that item i is requested is given by w_i, $1 \le i \le N$, for a database of N items. In this technique, item i is broadcast next from the server with probability p_i where

$$p_i = \frac{\sqrt{w_i}}{\sum_{j=1}^{N} \sqrt{w_j}} \tag{3.1}$$

This improves the overall expected waiting time for a terminal, but it can lead to starvation where a nearby terminal must wait an extremely long period of time before it receives its item in the broadcast.

The techniques described below take the skewed-broadcast technique further in clever ways to improve the expected waiting time for various broadcast scenarios.

A key result comes from [1] which describes the broadcast disks scheduling algorithm. In this algorithm, all of the data items are classified hierarchically based on their expected request rates from the terminals. Each item is placed on a virtual disk which spins at a rate proportional to the popularity of the items on the disk. Some disks "spin" faster and are broadcast more often, while other disks "spin" slower and are broadcast infrequently. One property of the broadcast disks scheduling algorithm is that it is periodic.

The broadcast disks algorithm can be tuned by setting the number of disks, the relative rotational speed of the disks with respect to one another, and the ranges of request probabilities that each disk represents. The broadcast disks algorithm may also introduce gaps in the broadcast schedule in order to maintain the proper rotational speeds of the disks. In the broadcast disks algorithm, all items are transmitted, whether there are terminals there to receive them or not, and terminals must monitor the broadcast channel for their item, since no indexing mechanism is described in this algorithm. In our study, we partition our database based on the popularity of the items, where items that are considered "more popular" (i.e. have greater demand probabilities) will be transmitted using multicast, while other items will be transmitted on request using unicast.

Vaidya and Hameed designed several broadcast algorithms for optimal push-based scheduling [24]. They assume all data items have the same size. Their results are based on the following rules to create a schedule that minimizes the expected waiting time for a terminal:

- Instances of each item in the broadcast schedule are equally spaced.
- Spacing of each item is proportional to the square root of its length and inversely proportional to the square root of its demand probability.

They present an online scheduling algorithm that tries to satisfy both rules as much as possible to minimize the expected waiting time for terminals. Another algorithm proposed by the same researchers in [11] derives the online broadcast schedule using an approach similar to broadcast disks, except that the rotational speed of each disk is calculated analytically (rather than set arbitrarily as in broadcast disks) to optimize the expected waiting time for all terminals.

Comparing these two algorithms to the models we present in this work, these algorithms assume that terminals are in a fixed position near the wireless transmitter and never leave the broadcast area, whereas in our work we assume that terminals are mobile and are within the coverage area for only a limited period of time. On the other hand, these algorithms and our analysis both assume that data items are equal-sized, although our simulator can handle items of varying sizes as well.

Su, Tassiulas, and Tsotras [22] designed a push-based system modeled as a deterministic Markov Decision. In their study, they derive a similar result as in [24], where the next item to be broadcast is given as a function of parameter γ:

$$w_i^{\gamma}\left(T - R_i\right) \tag{3.2}$$

where T is the current time, R_i is the time of the most recent broadcast of item i, and w_i is the probability of request for item i (we assume a unit transmission time here). The next item to be scheduled is the item with the largest value of equation 3.2 at each scheduling instance. The authors examine this criteria for various values of γ and conclude that $\gamma = 0.5$ yields the best results. This agrees with the work described in the previous section since $\gamma = 0.5$ means that the spacing between subsequent broadcast of the same item is proportional to the square root of the inverse of the request probability.

In addition to these results, they found that the availability of a feedback channel for requests does not improve the performance significantly at the server during heavy

loads. They assume that the probability of access of individual pages is known *a priori* as we have done. An extension to their model to handle varying file sizes is proposed in [21].

3.2 On-demand (pull-based) Scheduling

Instead of broadcasting all items from the server without any feedback from the terminals that are being served, on-demand scheduling considers schedules that are computed based solely on the demands of the waiting terminals. Research in this area usually assumes that there is an uplink channel available for terminals to send requests to the server for processing.

The uplink channel may be wired as in the case where terminals are at fixed locations in an office environment and have a wired connection to the local area network. On the other hand, the uplink channel could be wireless as in the case with mobile laptops and PDAs. In much of the research, the type of uplink channel is not critical to the results derived. This assumption is often invalid for wireless uplink channels that suffer from transmission losses due to common wireless channel anomalies such as interference, fading, etc.

This section summarizes the major results in on-demand scheduling and compares these models with the work described in this dissertation.

In [26], four fundamental scheduling algorithms were described for transmission of data items in an on-demand environment based on a satellite-based data delivery service:

- FCFS (First Come First Served): The common queuing discipline where data items are broadcast in the order of the requests received at the server.

- MRF (Most Requests First): The next item to be broadcast is the item that has the most pending requests in the server queue.

- MRFL (Most Requests First – Low): The same as MRF, except ties are broken by choosing the item with the lower request probability.

- LWF (Longest Wait First): Each item has a set of requests issued at various times. Each of these requests has a waiting time from the time of the request until the current scheduling time. The item with the highest sum of these waiting times is scheduled next.

If the request rates are very low, any of these methods will suffice, but as the request rates increase, MRF works best if the requests are uniformly distributed, whereas LWF has the best performance with respect to expected waiting time if the request distribution is Zipf (see Chapter 4 for a description of this distribution).

This work differs from our work, because this work assumes that a request that is made is broadcast (via satellite) to all users. In our work, we assume that individual requests can also be directed to specific terminals in order to save power on other terminals that do not need the information being transmitted.

Aksoy and Franklin [4] analyzed the results of the paper presented in the previous section and developed a new technique that is scalable and more practical for computing schedules based on the incoming requests. This technique, called R x W, maintains two lists R and W. The list R contains the number of requests for each item, ordered from highest number of requests to lowest. The list W contains the waiting time for the oldest pending request for each item, ordered from longest waiting time to shortest. At each scheduling decision point, the algorithm chooses the item that has the largest product of number of requests and waiting time. Items that are scheduled tend to have a large number of requests (close to the head of the R list) or have a very large waiting time since the last broadcast of the item (close to the head of the W list). Their results show that this technique performs similarly to LWF but is computationally more efficient.

In order to avoid searching the entire lists for the maximal product, this research also defines a parameterized version of the algorithm that finds the first product that exceeds some fraction of a threshold, where this fraction can be arbitrarily set by the system designer.

Our work does not consider the length of the waiting time for an outstanding request since requests are handled in a FIFO manner in our studies. Additionally, the number of requests in our study will influence the request distribution which will determine if an item is scheduled using multicast or unicast transmission.

Acharya and Muthukrishnan [3] focus on the stretch of a request as their objective function. Stretch is defined as the ratio of the expected waiting time for an item after it is request to the transmission time of the item. Because they are considering a measure that depends on the transmission time, their research does not assume equal-sized items as the previous studies have. However, they do assume that these items are split into equal-sized pages, and the pages are transmitted individually.

They present several algorithms that are applicable when items are not the same size:

- PLWF (Preemptive Longest Wait First): Similar to LWF of [26] except this decision is made after each page is broadcast.

- SRTF (Shortest Remaining Time First): The data item with the shortest remaining transmission time is scheduled next for broadcast.

- LTSF (Longest Total Stretch First): The data item that has the longest current stretch is scheduled next for broadcast.

- MAX: Each request has a deadline for transmission, and the item that has the earliest deadline is scheduled next for broadcast. The deadline is a function of the arrival time and the transmission time.

Of these algorithms, MAX performs very well for most test cases.

In this work, they consider non-uniform data sizes. We have not considered stretch as a metric in this study since the sizes of our data items are the same. However, the simulator

described in chapter 5 can handle data items of varying sizes quite easily, but it is unclear how the sizes of these items should be determined (i.e. using what random distribution).

3.3 Hybrid Data Scheduling

A pure push-based or pull-based system is generally not the most effective way to transmit data to wireless terminals except in rare situations, such as when the arrival rate of terminals (represented as requests) is very low and the database size is relatively low. In these situations, the server can handle the individual requests without much regard to the scheduling technique used.

Hybrid approaches have been proposed that combine push-based and pull-based strategies into one scheduling algorithm. In these algorithms, certain items are denoted as "popular" and are pushed to terminals automatically on a regular basis, while items that are not broadcast automatically must be requested by terminals in order to cause these items to be scheduled for future broadcast. The popularity of an item is a function of the number of terminals that want the data item per unit time. Note that this is not the same as a function of the number of requests a terminal receives since popular items will not generate requests from terminals in general.

The goal of hybrid broadcasting is to determine how much of the database should be automatically pushed to the terminals on a regular basis and how much of the database should be broadcast only upon request. This section highlights several research papers in this area and explains how this research is different than the work presented in this dissertation.

In [2], an extension to the broadcast disks called interleaved push and pull (IPP) is described that allows for terminal requests mixed with the push-based data broadcast. This technique is similar to our transmission techniques described in the next chapters, except that the authors were not studying the effects of multicast vs. unicast transmission in this algorithm. All data transmissions were assumed to be available to all terminals during

broadcast, even if one terminal makes the request for an item. This paper also describes a study testing the impact of terminal requests on steady-state and warm-up performance, techniques to maximize the benefit of terminal requests while avoiding server congestion, and the sensitivity of the performance to the variance in terminal access patterns. In their study, they assume that the broadcast and uplink channels were independent, which is different than what we have assumed in our implementation of the infostation. They also assume the broadcast program is static and data is read-only. They found that a pure push strategy is good if the server is saturated and the probability of an uplink request being dropped is extremely high, while a pure pull strategy is good is the server has little or no contention. This is consistent with our results. A merge of these techniques is good for reasonably consistent performance (for example, 30% pull for dynamic loads is reasonable in their study). Their study did not address items of different lengths or dealing with errors in transmission. Our simulation study described in Chapter 5 looks at a simple model for transmission errors and their effect on the overall system performance.

Another interesting model for hybrid broadcast systems is described in a paper by Stathatos, Roussopoulos and Baras [20]. The authors examined the issue of popular items in a unique way by assigning "temperatures" to items based on the frequency of access by terminals. "Vapor" items are heavily requested and put into general broadcast. "Liquid" items are requested moderately but not enough for general broadcast. "Frigid" items are not requested very much and are not fetched from the database on an as-needed basis only. Requests for liquid and frigid items adjust their temperatures slightly upward. Vapor items are not requested (since they are broadcast regularly) and slowly experience temperature drops. They propose an algorithm to handle the temperature assignment and broadcast scheduling. They state that the algorithm performs very close to optimal and depends only on the size of the hot spot, not on the volume of the workload. Their algorithm works well under dynamic, rapidly changing workloads. In our study, we also consider the effect of having popular items in the broadcast schedule although we do not use such a temperature analogy. Since our analysis assumes a relatively small time window, we do not consider changes in the request distribution in our simulations.

3.4 Indexing Methods

In our study, we utilize a simple indexing scheme where the index contains a list of the items to be transmitted in the current cycle. Terminals arrive in the infostation area and wait for the next index to determine if the items they want are scheduled for broadcast or if they need to make requests for the items. Three techniques described in this section examine alternate means of broadcasting the index in order to reduce the waiting time for the terminals.

The first technique involves the use of a signature, which is simply an identifier for a data item that is superimposed on the data item and broadcast as one unit. Terminals use a bit operations to compare the signature of a received data item with its list of desired item signatures and ignores those items that do not match the required signatures. Signatures can be used in multiple levels to allow a terminal to ignore sequences of data items in one group.

Hu, Lee, and Lee [12, 16] performed two studies that integrated point-to-point channels, broadcast channels and data caching. Various indexing methods were investigated including cached index, signature, and non-indexed methods. Flat broadcast scheduling was compared with broadcast disks as well. In these studies, only steady-state performance was measured and all terminals have similar hot spots and change access demand at the same time. The authors found that broadcast disks and flat broadcast have better performance than pure pull when data access patterns are skewed. Additionally, broadcast disks outperforms flat broadcast for skewed data access patterns and less dynamic workloads, whereas pure pull is a better choice for random access patterns or highly dynamic workloads. Compared to this work, we did not change the terminal access patterns dynamically through the simulation, since we assumed the window of time that we are studying is short enough to consider this a constant. An interesting point in their study is that they use $M/M/1$ as the underlying discipline, implying that service times are exponential, whereas we assume a constant service time.

A second technique to transmit indices is proposed in [14], which splits the index information into a tree structure. The upper levels of the tree are broadcast multiple times during one cycle for terminals that arrive during a cycle; these terminals can start to process the major structure of the index before the next cycle begins. In addition, during each index broadcast, only lower levels of the index tree that apply to the data items to be broadcast next (before the next index transmission) are also broadcast to the terminals.

If a terminal sees the item it desires in this section of the broadcast, it can capture it immediately instead of waiting for the next complete cycle as in standard indexing techniques. The index tree described in this research includes a mechanism to allow a terminal to determine when an item will be broadcast in the next cycle if it is missed in the current cycle (e.g. the terminal arrives just after the broadcast of its item in the current cycle). This scheme assumes a push strategy which means that the missed item will be available again in the next cycle. We do not assume this in general in our work, since we incorporate terminal requests in our index as well as push-based "popular" items.

A third technique studied for indexing in wireless broadcast systems is the use of hashing as described in [15]. Each data item also includes a control segment that allows the terminal to find its desired data item quickly. The control segment contains a hash function and a second function to handle collisions. The data terminal uses the hash function on the information in the control segment to determine the next broadcast time for the item it requires. This technique eliminates the need for a terminal to wait until the start of the next cycle for the index as we have done in our work. However, it also adds extra information to each data packet in the broadcast. Also, it is unclear how this technique would handle terminal requests for items not in the broadcast cycle.

3.5 Other Simulation Studies of Multicast Transmission

A number of research results have been published recently which use simulation as we have done to study various aspects of transmission in a wireless data broadcast system.

A simulation study is presented in [10] that is similar in its objective to the study presented here. The authors study a hybrid broadcast algorithm with the goal of finding the optimal number of items that should be pushed to terminals per cycle. The optimality criteria presented in [24] is used as a basis for the mathematical analysis that determines the optimal number of pushed items, while we use queuing theory to analyze the expected waiting time. The analysis is performed using a Zipf distribution as we have done, but for only one value for the distribution parameter θ, whereas we study a range of Zipf request distributions. Another difference in their study is that they assume that there are two broadcast channels, one for the index and one for the data items. They also assume that other terminals can receive a broadcast generated by a request from a terminal, whereas we study the situation where requests are served using unicast and are unavailable to other terminals who want the same item.

In [6], simulations are used to study a security issues in such a system. In this study, the authors propose a specialized scalable protocol that allows transmitted data to be readable by only select user groups. In [23], simulations are used to study a newly proposed algorithm for scheduling that is a function of the terminal's time to hear the index, the time it takes to receive its item after the index, and a threshold on how long a terminal or server will wait before the transmission is considered a failure. Simulation is used in [5] to analyze a set of scheduling algorithms for broadcast cycles that deal with updates of the data items in the cyclic broadcast. In [17], the author observes that in many instances, data requests for files are related. For example, in requesting an HTML file, additional files like images and audio may also need to be downloaded; a set of heuristics are presented that take advantage of these file dependencies to improve previously presented data broadcast scheduling algorithms.

3.6 Summary

All of these studies illustrate that wireless data broadcast systems have generated a great deal of interest in recent years. Some of these results are based on reducing the overall waiting time for a terminal to receive a data item creating effective broadcast schedules that

may incorporate both push-based and pull-based data transmission. Other results use clever indexing strategies to minimize the power usage of terminals by allowing them to determine when their data items will be broadcast so they can conserve power until the data transmission occurs. In most of this research, simulation was the key method to analyze the various algorithms and techniques for their effectiveness.

In the next two chapters, we examine wireless data broadcast systems in the context of an infostation environment, where terminals are within range of the wireless transmitter for a limited period of time. We use a combination of mathematical derivation and simulation to study schedules that consist of both multicast and unicast transmission in order to minimize the effective waiting time for the arriving terminals in the infostation area.

Chapter 4

MATHEMATICAL ANALYSIS OF THE INFOSTATION MODEL

One of the goals of an infostation system is to maximize the number of terminals that receive their desired data when they are within range of the infostation. This may be done using multicast transmission, unicast transmission or some combination of both techniques.

In this chapter, we present an analysis of a model based on an infostation system being developed at Polytechnic University. In this model, the transmission channel is used to transmit one item at a time using either multicast or unicast. This chapter will provide an analysis of the expected waiting time for terminals in such an environment and will illustrate conditions under which an optimal schedule can be determined as a function of the arrival rate and the request distribution of the terminals. What we show in this chapter is that as the arrival rate of terminals increases and the request distribution skews toward a smaller set of "popular" items, it becomes advantageous to multicast a small percentage of the database in order to minimize the expected waiting time for a terminal to receive its desired item, and therefore maximize the probability that any terminal will receive its item before departure from the infostation area.

4.1 Parameters of the model and assumptions

In this section, we describe the parameters of the infostation model described in chapter 2. We also present our simplifying assumptions on the general infostation model that are used in the subsequent analysis.

Terminals arrive according to a Poisson distribution at a mean arrival rate of λ terminals/second. Each terminal that enters within range of the infostation desires exactly

one item. Each terminal is within range for exactly the same amount of time t (in seconds). A terminal must receive all packets of an item to consider the wireless download successful. Each terminal makes a request or sends feedback to the server on the same channel during the respective request and feedback periods of a broadcast cycle.

In this analysis, we assume that the probability that a terminal wants item i ($1 \le i \le n$) is given by a Zipf distribution with parameter $\theta \ge 0$:

$$w_i = \frac{(1/i)^\theta}{\sum_{j=1}^{n} (1/j)^\theta}. \tag{4.1}$$

Figure 4.1 shows the values of w_i for $n = 100$. Note that when $\theta = 0$, the Zipf distribution is a uniform distribution. As θ increases, the requests tend to skew more and more toward the first few items of the database. Use of the Zipf distribution for terminal requests is well-established in the literature [1, 2, 4, 11, 12, 16, 21, 22, 24].

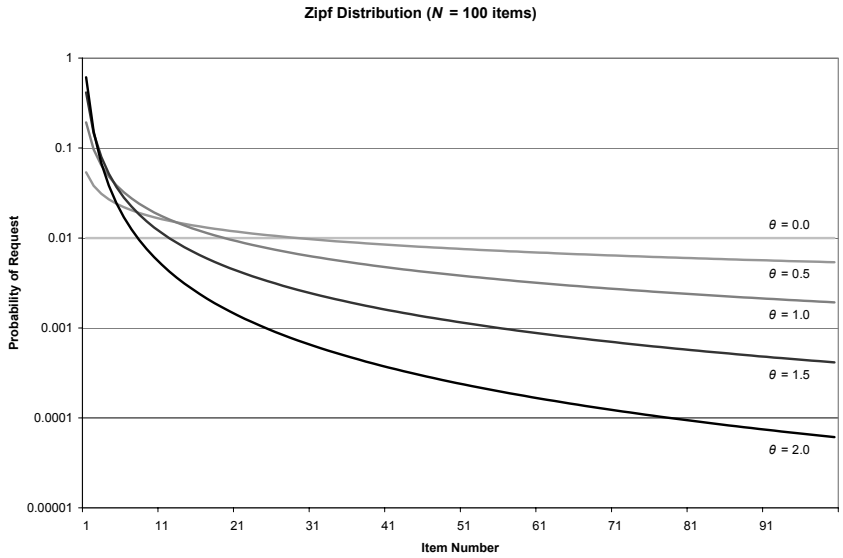

Figure 4.1: Probability that a terminal will request a specific item based on the Zipf distribution with parameter θ for a database of $n = 100$ items.

Each item is transmitted in one of two modes, multicast (one-to-many) or unicast (one-to-one). In multicast mode, we assume any terminal within range that wants the multicast item will receive the item when it is transmitted. Also, each multicast item is transmitted in its entirety once per cycle. In unicast mode, we assume only the terminal that requests the given item will receive it, even if other terminals want the same item. Unicast items are only scheduled for transmission once for each terminal request.

The database contains n items and is partitioned into two groups: n_m items that will be multicast to terminals regularly, and n_u items that will be unicast to terminals on request. Thus, $0 \le n_m \le n$, $0 \le n_u \le n$, and $n = n_m + n_u$. Each item requires D seconds to be transmitted (i.e. all items are the same size). No transmission errors occur when an item is broadcast wirelessly.

A broadcast cycle takes the form of Figure 4.2. Each cycle begins with an index that contains information about the items that will be transmitted in the current cycle. Terminals can use this information to determine if the desired item is scheduled. If so, the terminal can wait for the transmission of that item. Otherwise, the terminal makes a request for the item and then conserves power by entering into a "sleep mode" until the next cycle. We assume that the cycle is a fixed length, so a terminal will be able to determine when to "wake" for the next index.

After the index, there are C slots for transmission of data items to terminals. The first M slots are dedicated to transmission of multicast items. Multicast items do not need to be requested and are transmitted automatically by the infostation. The items that are transmitted using multicast are the most "popular" items (i.e. items with the highest request probability). We assume that the popularity of items is known ahead of time. For example, the feedback mechanism explained in section 2.2 can be used to estimate these probabilities. In our analysis, we assume $M = n_m$; therefore, each multicast item of the database is transmitted once per cycle.

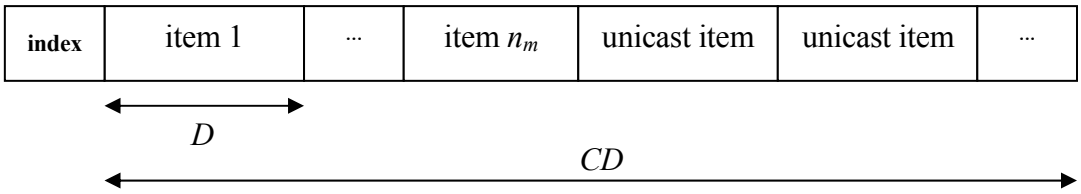

Figure 4.2 Broadcast cycle for the simplified infostation model.

After the multicast items are transmitted, any unicast items that were requested from previous cycles can be transmitted. Because the cycle size is a constant length, the number of unicast items that can be transmitted in each cycle is bounded. Let U = the number of unicast slots per cycle, where $U = C - M$. (Note: U is not necessarily equal to n_u.) We assume in this analysis that we will be able to transmit all multicast items in one cycle, but we may not be able to satisfy all pending requests for unicast items in one cycle. If the number of pending unicast requests is less than $C - n_m$, then the infostation will remain idle until the next index, in order to keep the cycle length constant for terminals that are in power-conservation mode.

Finally, we assume that requests and feedback are made using a separate uplink channel, so they do not extend the length of the cycle. It is important to keep in mind that if these segments use the same channel as the index and data broadcast, expected waiting times for terminals will increase.

4.2 Total waiting time

In this section, we will derive an expression for the expected waiting time for a terminal that arrives and wants one item of the database.

Let the random variable W be the waiting time for a terminal from its arrival in the infostation area until it receives its desired item (assuming the terminal does not leave the infostation area after arrival). Then $W = W_{index} + W_{item}$, where W_{index} is the terminal's waiting time from arrival until the terminal receives its first index, and W_{item} is the

terminal's waiting time from the index transmission until the terminal finally receives the item from the infostation.[1]

The waiting time for an item after the terminal receives its first index depends on whether the item is multicast or unicast. We write $W_{item} = W_{multicast}$ if the item is multicast, and $W_{item} = W_{unicast}$ if the item is unicast.

Let B = the probability that an item is multicast, where

$$B = \sum_{i=1}^{M} w_i \tag{4.3}$$

Then the expected waiting time for any newly arriving terminal that wants item j, $1 \le j \le n$, with M items transmitted each cycle using multicast, $0 \le M \le n$, is given by the equation

$$\begin{aligned} E[W] &= E[W_{index}] + E[W_{item}] \\ &= E[W_{index}] + E[W_{multicast}] \cdot B + E[W_{unicast}] \cdot (1 - B) \\ &= E[W_{index}] + E[W_{multicast}]\left(\sum_{i=1}^{M} w_i\right) + E[W_{unicast}]\left(1 - \sum_{i=1}^{M} w_i\right) \end{aligned} \quad . \tag{4.4}$$

The expression above depends on the arrival rate of terminals into the infostation area. Since each item takes a constant D seconds to transmit (i.e. to serve an item), one would think that an arrival rate of $1/D$ terminals/second or greater will lead to an unstable system. This is true when all items are unicast. However, when some items are multicast, fewer terminals want unicast items, so the arrival rate can exceed $1/D$ terminals/second as long as the arrival rate for terminals wanting unicast items stays below $1/D$ terminals/second. In other words, this means that the equation 4.4 holds for all M, $0 \le M \le C$, such that

[1] The random variables W_{index} and W_{item} are dependent random variables, but this will not affect the results presented in this chapter since we are examining the expected value of the sum of these variables which is the sum of the expected values of the variables, regardless of their dependence.

$$\lambda_u = \left(1 - \sum_{i=1}^{M} w_i\right)\lambda < \frac{1}{D}. \tag{4.5}$$

Keep in mind that up to this point, the expected waiting time does not depend on the request distribution that is used in the analysis, as long as the w_i values are non-increasing (i.e. $w_1 \geq w_2 \geq \ldots \geq w_n$). The Zipf distribution used in our analysis later in this chapter has this property.

Let us consider the amount of time the terminal waits for its first index after its arrival in the infostation coverage area. We will assume that the time to transmit the index is negligible, because the index is much smaller than a data item typically. Since a terminal can arrive randomly at any time within a cycle, we can see that

$$E[W_{index}] = \frac{CD}{2} \tag{4.6}$$

The amount of time the terminal waits for the item after it receives its first index depends on whether the item is transmitted using multicast or unicast. If the item is transmitted using multicast, the terminal is guaranteed to receive the item during the next full cycle after its arrival. If the item is transmitted using unicast, the terminal may have to wait through multiple cycles until the item is transmitted, depending on the number of terminals before this terminal that are awaiting unicast items when this cycle starts. If the number of waiting terminals is greater than U, then this terminal will have to wait for more than one cycle before it will receive its item. The next two sections will present derivations of $E[W_{item}]$ for multicast and unicast items.

4.3 Expected waiting time for multicast items

Consider a terminal that wants a multicast item. Clearly, it makes sense to transmit the multicast items in order of decreasing popularity. In other words, we transmit items in

order from 1 to M (= n_m), where $w_1 \geq w_2 \geq \ldots \geq w_M$. Let w_i' = the probability that a terminal wants item i, given that the item is multicast. Then

$$w_i' = w_i / (w_1 + w_2 + \ldots + w_M) = w_i / B. \tag{4.7}$$

Note that the item ordering in the broadcast cycle remains the same since $w_1' \geq w_2' \geq \ldots \geq w_M'$.

The expected waiting time for a terminal that wants a multicast item after it receives the index is given by

$$E[W_{multicast}] = D\sum_{i=1}^{M} i w_i' . \tag{4.8}$$

So, the total expected waiting time for a terminal that wants a multicast item is simply

$$E[W \mid multicast] = E[W_{index}] + E[W_{multicast}] = D\left(\frac{C}{2} + \sum_{i=1}^{M} i w_i'\right). \tag{4.9}$$

4.4 Expected waiting time for unicast items

We now consider a terminal that wants a unicast item. In order to determine the expected waiting time for an item by a terminal in this case, we need to determine how many terminals arrived before the new terminal and are still waiting for their items at the start of the first full cycle after the new terminal arrives. Since we assume the infostation handles unicast requests in the order that they are received (i.e. FIFO), a new terminal must wait until all prior requests have been satisfied before it can receive its data item.

We can model the terminals that want unicast items as a simple FIFO queue, where items are transmitted to these terminals in the order of their arrivals. If λ represents the average arrival rate of terminals in general, then let λ_u represent the average arrival rate of

terminals that want unicast items only. Since the number of terminals that want unicast items only depends on the request distribution, we have

$$\lambda_u = (1 - \sum_{i=1}^{M} w_i)\lambda = (1 - B)\lambda \tag{4.10}$$

Note that if $n_m = 0$, $\lambda_u = \lambda$.

Additionally, let μ_u = the average service rate of terminals that want unicast items only. Since, in general, we will not be broadcasting unicast items continuously throughout each cycle, we amortize the service rate such that it represents the rate at which we serve the same number of terminals if we were to transmit unicast items continuously throughout the cycle. Therefore,

$$\mu_u = \frac{U}{CD}. \tag{4.11}$$

Note that when $n_m = 0$, $U = C$, and $\mu_u = 1/D$ as expected (i.e. D seconds per item). Finally, we define

$$\rho = \frac{\lambda_u}{\mu_u}. \tag{4.12}$$

Consider a new terminal T that wants a unicast item. Assume the terminal arrives during cycle 1, as shown in Figure 4.3. At the start of cycle 1, there are K_0 terminals that are still waiting for their unicast items that were not transmitted in cycle 0. During cycle 0, up to U terminals of the K_0 terminals are served by the infostation. Thus, the number of residual terminals that are still waiting for their data items at the end of cycle 1 is given by K_1, where

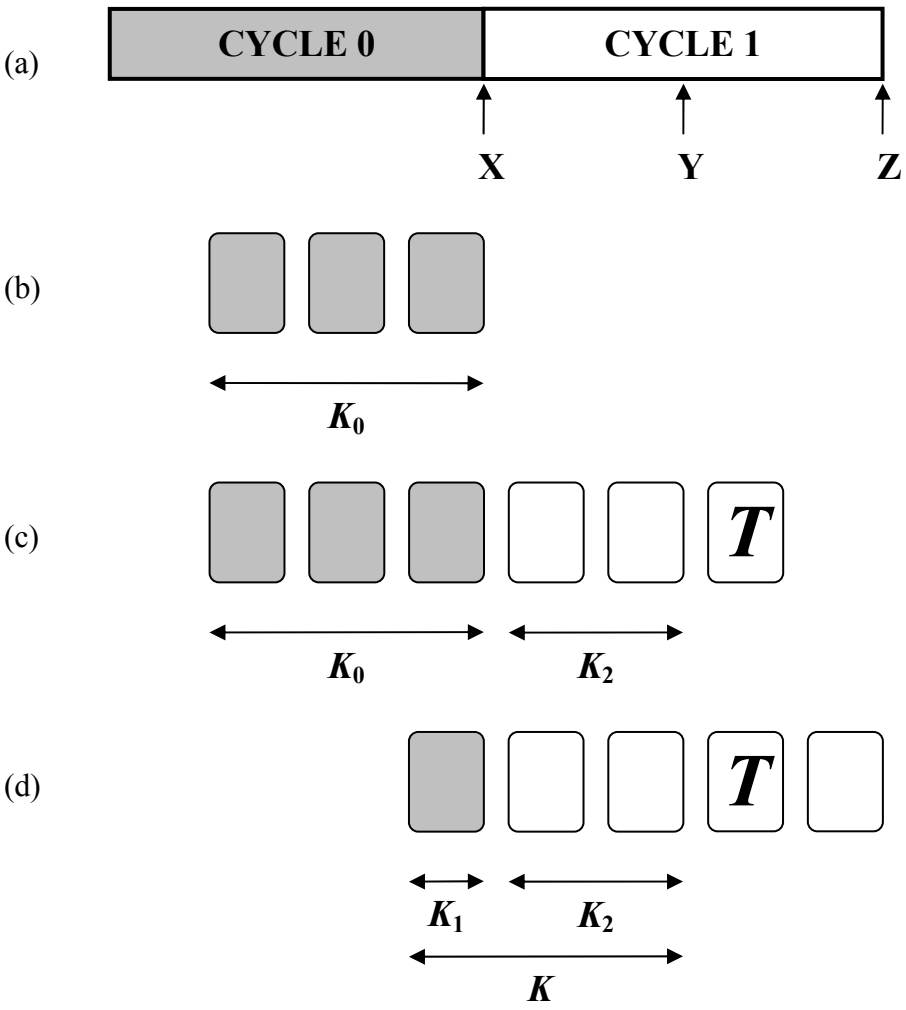

Figure 4.3 Determining the number of unicast slots that a terminal needs to wait through before it has received its item, assuming the item is unicast. (a) At the start of the cycle at time X, there are K_0 terminals still waiting for unicast items from previous cycle(s). (b) During this cycle, a new terminal *T* arrives at time Y, wanting a new item. There are K_2 terminals that have arrived ahead of terminal *T* during cycle 1. (c) By the end of this cycle at time Z, up to *U* terminals of the K_0 terminals are served, leaving K_1 terminals remaining from that group of terminals. Thus, there are $K= K_1+ K_2$ terminals ahead of the new terminal *T* when *T* receives its first index at the start of the next cycle.

$$K_1 = \begin{cases} K_0 - U, & K_0 > U \\ 0, & K_0 \le U \end{cases} \tag{4.13}$$

In addition to the residual terminals, there are K_2 terminal arrivals up to and including the arrival of the new terminal during the current cycle. Thus the total number of terminals waiting for their items at the start of the next cycle (the first index that the new terminal receives) is

$$K = K_1 + K_2. \tag{4.14}$$

If $0 \le K \le U - 1$, then the new terminal will receive its desired item in the next cycle and $W_{item} = (M + K + 1)D$. Consider $U \le K \le 2U - 1$, as shown in figure 4.4(a). By rearranging the transmission schedule time, it becomes clear that $W_{item} = (2M + K + 1)D$, as shown in figure 4.4(b).

Using a similar argument for larger values of K, we can write

$$W_{item} \mid unicast = W_{unicast}(K) = \begin{cases} (M + K + 1)D & if \quad 0 \le K < U \\ (2M + K + 1)D & if \quad U \le K < 2U \\ (3M + K + 1)D & if \quad 2U \le K < 3U \\ \cdots \end{cases} \tag{4.15a}$$

or, in general,

$$W_{unicast}(K) = (jM + K + 1)D \quad if \quad (j - 1)U \le K < jU. \tag{4.15b}$$

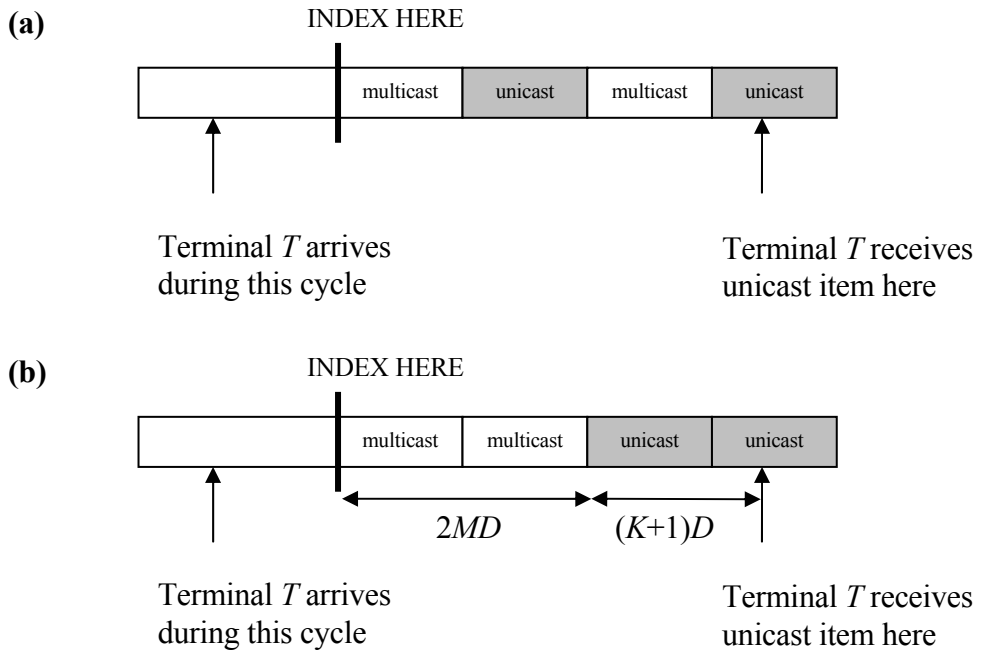

Figure 4.4 Finding $W_{item}|unicast$ when $U \leq K \leq 2U$-1.

Another way to look at equation 4.15b is as follows. If there are K terminals requesting unicast items ahead of a newly arriving terminal when it hears its first index, equation 4.15b shows that the new terminal must wait through $(\lfloor K/U \rfloor + 1)M$ multicast slots in addition to K slots for the terminals ahead of it before it will receive its item. (The additional slot of length D in equation 4.15b is for the new terminal itself.) Therefore, we can also write

$$W_{unicast}(K) = ((\lfloor K/U \rfloor + 1)M + K + 1)D. \quad (4.15c)$$

When the newly arriving terminal hears its first index, there will be K terminals ahead of it in the system queue waiting for unicast items. Therefore, the new terminal must

wait for these K terminals to be served before it can be served. Using equation 4.15c, the expected waiting time for a unicast item once the index is received is given by the formula

$$\begin{aligned} E[W_{unicast}] &= \sum_{i=0}^{\infty} W_{unicast}(K=i)\cdot P[K=i] \\ &= D\left(1+E[K]+\sum_{i=0}^{\infty}(\lfloor i/U \rfloor+1)M\cdot P[K=i]\right) \\ &= D\left(1+E[K]+\sum_{j=1}^{\infty} jM\cdot P[(j-1)U \le K < jU]\right) \end{aligned} \tag{4.16}$$

Equation 4.16 shows that the waiting time for a newly-arriving terminal that wants a unicast item is 1 slot for itself, K slots for the terminals ahead of it when the next cycle starts and jM slots for the time taken up by j multicast segments that from the start of the next cycle until the time the terminal's item is transmitted, where j depends on the number of terminals ahead of the newly-arriving terminal (i.e. K).

Finally, we can write that the total expected waiting time for a terminal that wants a unicast item is

$$\begin{aligned} E[W \mid unicast] &= E[W_{index}] + E[W_{unicast}] \\ &= \frac{CD}{2} + \sum_{i=0}^{\infty} W_{unicast}(K=i)\cdot P[K=i] \end{aligned} \tag{4.17}$$

In order to determine $\mathrm{E}[W_{unicast}] = \mathrm{E}[W_{item} \mid unicast]$, we need to determine $P[K = i]$, $i \ge 0$. To accomplish this, note that (using equation (4.14))

$$P[K=i] = P[K_1 + K_2 = i] = \sum_{j=0}^{i} P[K_1 = j]P[K_2 = i-j], \quad i \ge 0. \tag{4.15}$$

First, we consider $P[K_1 = j]$, the probability that there are j terminals waiting for service at the end of the cycle where the new terminal arrives. In a traditional $M/D/1$ queueing system, service occurs as long as there are terminals waiting for service. There

are no self-imposed idle times for the server, as illustrated in figure 4.5(a). However, in our system, the transmission of the unicast items is delayed in each cycle to accommodate the transmission of the multicast items first, as shown in figure 4.5(b). Note that at the start of each cycle and only at the start of each cycle, both systems behave as *M*/*D*/1. Therefore, we will approximate $P[K_1 = j]$ with the steady-state probability that there are j terminals in an *M*/*D*/1 system:

$$P[K_1 = j] = \begin{cases} 1-\rho & , j = 0 \\ (1-\rho)(e^{\rho}-1) & , j = 1 \\ (1-\rho)\left(e^{j\rho} + \sum_{k=1}^{j-1} (-1)^{j-k} e^{k\rho} \frac{(k\rho)^{j-k-1}}{(j-k)!} (j - k(1-\rho)) \right) & , j \geq 2 \end{cases} \tag{4.16}$$

Next, we consider $P[K_2 = j]$, the probability that there are j arrivals during the current cycle before the new terminal arrives. Assuming that the new terminal arrives in the middle of the cycle and all arrivals are Poisson distributed with parameter λ_u, we have

$$P[K_2 = j] = \frac{(\lambda_u CD/2)^j}{j!} e^{-\lambda_u CD/2}, \; j \geq 0. \tag{4.17}$$

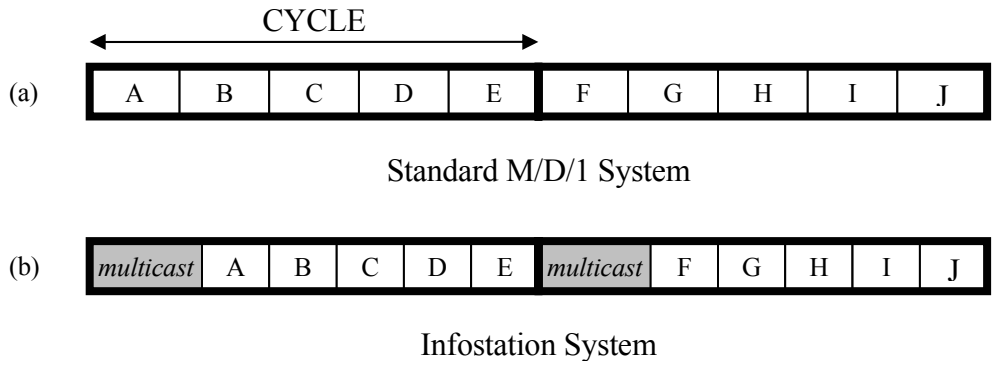

Figure 4.5 Comparing an M/D/1 queueing system with the infostation system. At the start of each cycle, the number of terminals in the imfostation system will be the same as in the standard M/D/1 system in steady state.

4.5 Analysis of the infostation model

The scheduler determines for each cycle what to transmit and in what order based on the number of multicast items and the individual requests for unicast items received from terminals entering the infostation area. Our goal is to determine how many items of the repository to multicast as a function of the arrival rate and the request distribution of the terminals. Using equation 4.4 and the derivations in the previous two sections, we calculated the expected waiting time for a terminal in an infostation environment, assuming that D = 1 second.[2] So our results will be expressed in terms of slots instead of seconds for this section. In this analysis, we assume that items are requested based on a Zipf distribution with parameter θ (see section 4.1).

In all results shown in this section, we assume a cycle consists of 100 slots (i.e. C = 100). Since a terminal can arrive at any time during a cycle with equal probability, $E[W_{index}] = CD/2$ = 50 seconds = 50 slots. This is a lower bound to the expected waiting time regardless of the number of items that are multicast.

4.5.1 Expected Waiting Time as a function of request distribution

In this section, we will study the expected waiting time for a terminal as a function of the request distribution (i.e. how skewed the requests are toward a small subset of the database) for various arrival rates. In each analysis, we vary the size of the database, starting with a small database (that can be completely transmitted in one cycle using multicast) and going up to a very large repository (that requires many cycles to transmit each item of the database once).

---

[2] For a transmission rate of 4Mbps, this corresponds to a data item of size 1MB which is equivalent to approximately 1 minute of mp3 music encoded at 128kbps, assuming no packet error loss.

4.5.1.1 Number of repository items and number of cycle slots are equal (C=100 and n = 100)

Figure 4.6(a) shows the expected waiting time as a function of M and θ for arrival rate $\lambda = 0.5$ arrivals/slot and repository size $n = 100$ items. Note that λ is the arrival rate for all terminals, not just those requesting unicast items.

When the number of multicast items is 0, all items are unicast, so the expected waiting time consists of the time to wait for the index plus the expected waiting time for the item that is unicast. In this case, $\lambda_u = \lambda$ and $U = C$, so $\rho = \lambda_u/\mu_u = \lambda/(U/C) = \lambda/1 = \lambda$. As an example, consider $C = 100$, $N = 100$ and $M = 0$ (i.e. pure unicast transmission) and an arrival rate of 0.5 terminals/slot. For this arrival rate, if we assume a new terminal arrives in the middle of a cycle on average, we would expect the waiting time to be 50 slots until the index is transmitted in the next cycle, plus 26 slots, since there would be 25 new arrivals before the new terminal which will be served first and then the new terminal would be served in the 26^{th} slot in the next cycle. Thus, we would expect the waiting time to be 50+26=76 slots as shown in figure 4.6(a). Note that this value is independent of θ since all requests are being served using unicast transmissions, so the request distribution is irrelevant.

At $\lambda = 0.5$ arrivals/slot, the infostation can easily keep up with the requests from the arriving terminals. For $\theta = 0$, the requests are uniformly distributed and are best serviced by unicasting each request in the order received. As the number of multicast items increases, terminals will wait longer on average to receive the same item since they will have to wait for the appropriate slot to receive the item even if it could have been unicast directly before then. For example, if $M = 50$ and a terminal wants item #50, the terminal would have to wait through half of the cycle until item #50 is multicast (after items #1 through #49), causing an expected wait time of 50 slots after the index is broadcast. However, if all items were unicast, then the terminal would have to wait on average $\lambda(C+1)/2 \approx 25$ slots after the index to receive the same item.

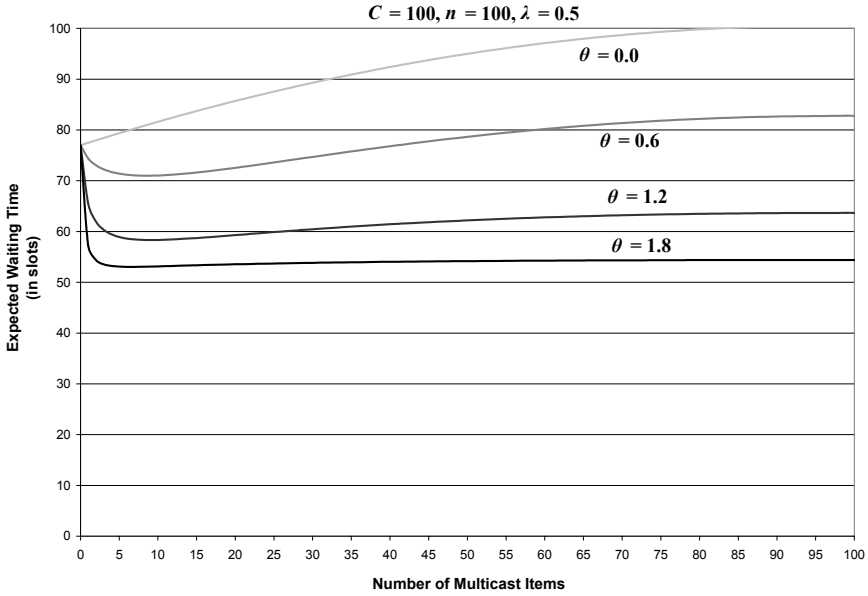

Figure 4.6(a) Expected waiting time as a function of number of multicast items per cycle with λ=0.5 arrivals/slot for various Zipf distributions with parameter θ.

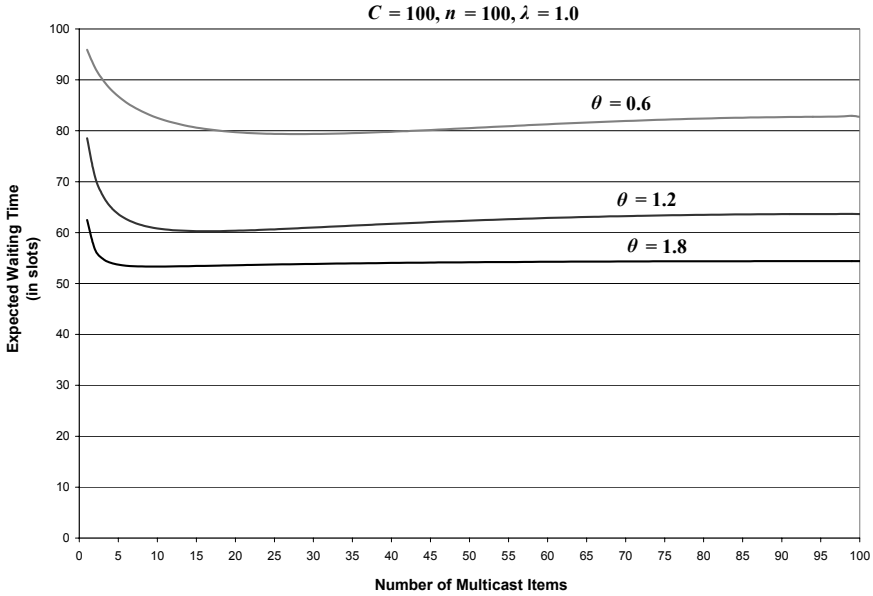

Figure 4.6(b) Expected waiting time as a function of number of multicast items per cycle with λ=1.0 arrivals/slot for various Zipf distributions with parameter θ.

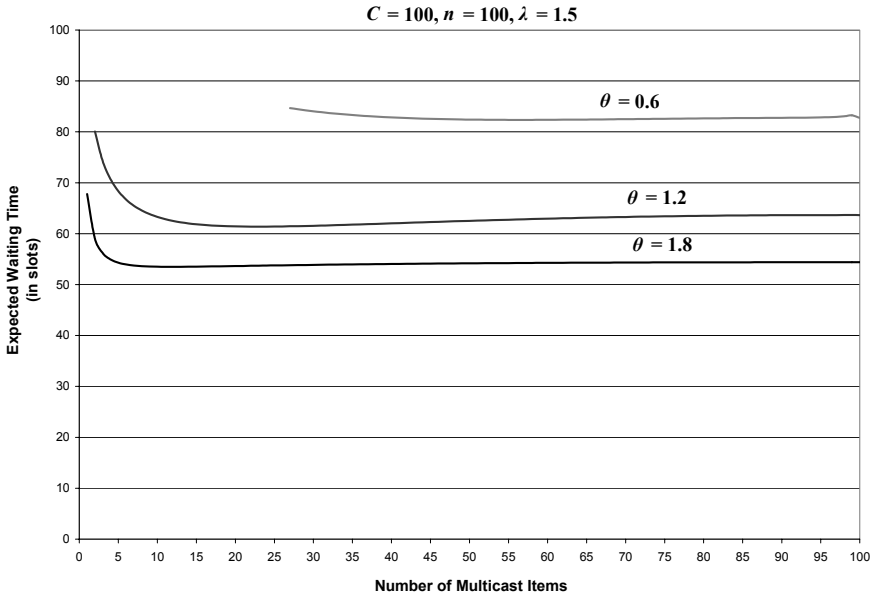

Figure 4.6(c) Expected waiting time as a function of number of multicast items per cycle with λ=1.5 arrivals/slot for various Zipf distributions with parameter θ.

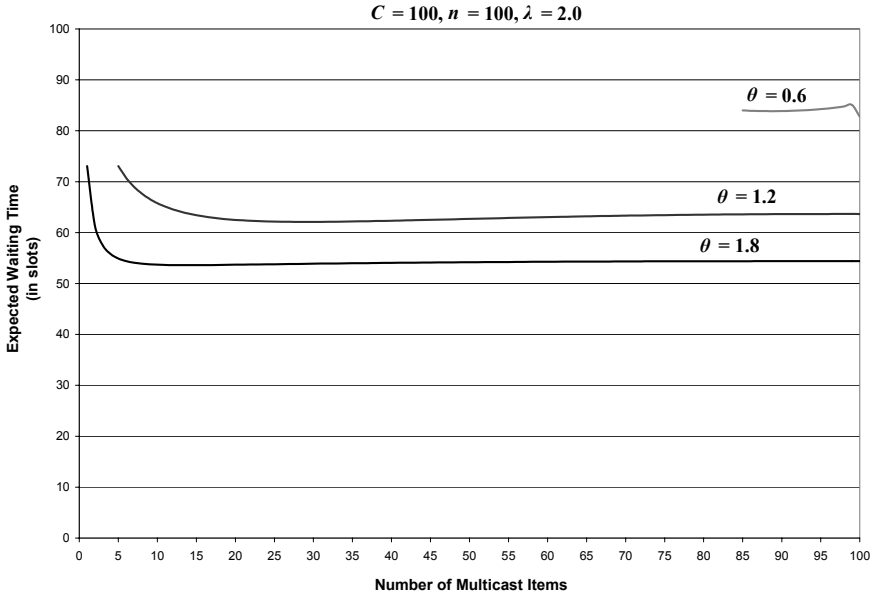

Figure 4.6(d) Expected waiting time as a function of number of multicast items per cycle with λ=2.0 arrivals/slot for various Zipf distributions with parameter θ.

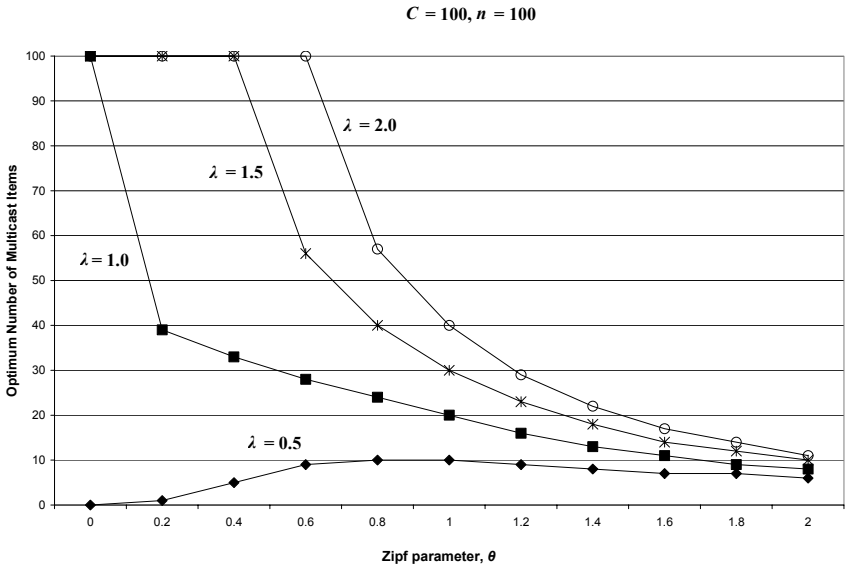

Figure 4.6(e) Optimal number of multicast items per cycle as a function of the Zipf request distribution with parameter θ for various arrival rates λ (in arrivals/slot).

As θ increases, however, the request distribution starts to shift toward a smaller subset of the items, with item #1 becoming more popular. As the figure shows, there is a benefit to multicast a subset of the database, up to the first twenty items for $\theta = 0.6$. At this value of θ, as we increase the number of multicast items from 0, we take advantage of the multiple requests for items with lower indices, using multicast to serve all waiting terminals for these items and reducing the overall expected waiting time for the terminals. However, if we multicast too much, we negate any savings we get from multicasting the "more popular" items by adding in longer waiting times for items with higher indices that are multicast near the end of the cycle. At $\theta = 0.6$, although the request distribution is skewing toward item #1, the other items still have a significant request probability and affect the overall expected waiting time.

As we approach $\theta = 2.0$, the request distribution is highly skewed toward item #1. In other words, most of the terminals want item #1. Therefore, we get a dramatic decrease in the expected waiting time as soon as we multicast the first item. However, if we multicast more items, the expected waiting time increases very slightly since there are few

terminals that have to wait a long period of time for their multicast items as with lower θ values.

Figure 4.6(b) shows the expected waiting time as a function of M and θ for arrival rate $\lambda = 1.0$ arrivals/second and repository size $n = 100$ items. At this arrival rate, saturation begins to occur in the infostation system. Recall that the analysis of the expected waiting time for terminals that want unicast items depends on a queueing model, so once ρ reaches 1, we can expect the waiting times to increase without bound. However, although $\lambda = 1$ in this case, this does not mean that ρ is necessarily 1, since some of the arriving terminals may want multicast items and these requests are not related to the queue size for the unicast requests.

Consider the case where $\lambda = 1.0$ and $\theta = 0.0$. Since $\theta = 0.0$, we know that all requests are uniformly distributed. This means that of the λ arrivals/second, the arrival rate for terminals that want unicast items is given by

$$\lambda_u = \frac{C - M}{n}\lambda = \frac{U}{n}\lambda \tag{4.18}$$

Thus, since $C = n$ in this analysis, and $\lambda = 1$:

$$\rho = \frac{\lambda_u}{\mu_u} = \frac{C}{n} = 1 \tag{4.19}$$

This shows that $\rho = 1$ when $\lambda = 1$, $\theta = 0$ and $C = n$, for all $M < C$. Therefore, the curve for $\theta = 0$ does not appear in figure 4.6(b) except when $M = C = 100$ (when the expected waiting time is 100.5) because the expected waiting time is unbounded.

If we consider higher values of θ, we see that in all of these cases, if $M = 0$, then $\rho = 1$ and the expected waiting time is unbounded (i.e. not shown in the graph). However, as M increases, the number of arrivals wanting unicast items decreases, which brings ρ below 1, since ρ only depends on the number of arrivals for unicast items, not multicast

items. In general, the minimum number of items that need to be multicast so that the expected waiting time is unbounded is given by *M'*, where

$$M' = \min_{M}\left(\sum_{i=1}^{M} w_i > 1 - \frac{1}{\lambda}\right), \quad \lambda \geq 1 \tag{4.20}$$

Figure 4.7 shows values for *M'* for various $\lambda \geq 1$ as a function of the Zipf θ parameter. When $\theta = 0$, recall that the request distribution is uniform. As the λ increases above 1, we need to shift more and more incoming terminals to multicast items until the arrival rate of terminals wanting unicast items falls below 1 arrival/second. For example, when $\lambda = 2.0$ and $\theta = 0$, at least half of the terminals must need multicast items in order to have $\lambda_u \leq 1$, so *M'* = 50 since the request distribution is uniform. When $\lambda = 2.0$ and $\theta = 2$, the request distribution is highly skewed toward item #1. Therefore, by multicasting item #1, we bring λ_u down below 1, resulting in *M'* = 1.

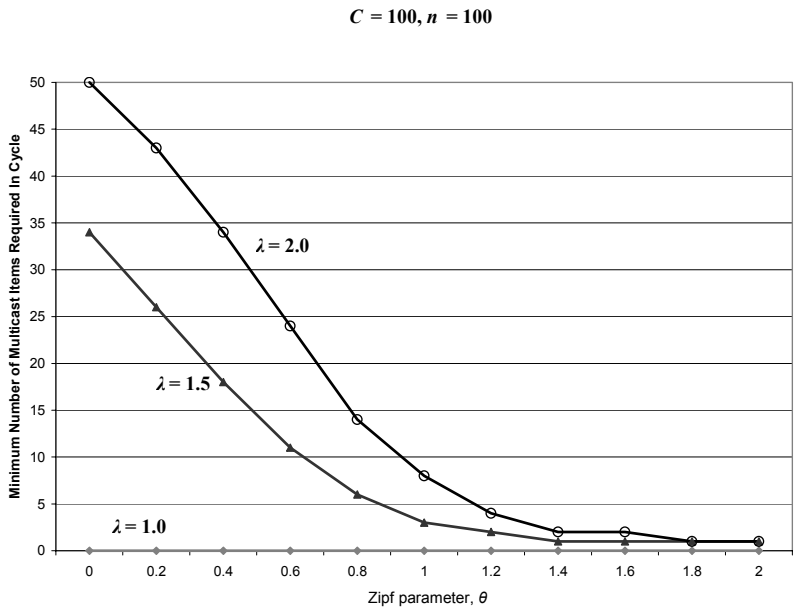

Figure 4.7 Minimum number of multicast items required per cycle as a function of the Zipf request distribution with parameter θ for various arrival rates $\lambda \geq 1$ (in arrivals/slot).

Examining figure 4.6(b) for higher θ values, we see clearly that there is an optimal number of multicast items to transmit each cycle in order to minimize the overall expected waiting time. As θ approaches 2.0, the graphs suggest that we should transmit a smaller subset of multicast items since most requests are skewed toward the first item of the repository and multicasting many items does not provide any useful benefit.

Figure 4.6(c) shows the expected waiting time as a function of M and θ for arrival rate $\lambda = 1.5$ arrivals/slot and repository size $n = 100$ items. As expected, as the arrival rate increases beyond saturation (for pure unicast), the request distribution must skew more toward a few items of the repository before the expected waiting times become reasonable. In this figure, we see that the expected waiting time is unbounded for $\theta = 0.0$ and $M < C$. At $\theta = 0.4$, the expected waiting time is bounded only if a large subset of the repository is multicast each cycle in order to keep ρ below 1. As θ increases further, fewer items in the repository need to be multicast in order to keep the unicast request queue from overflowing. As in the previous graphs, it is clear that there is a specific number of multicast items that minimizes the expected waiting time for the terminals which depends on θ.

Figure 4.6(d) shows the expected waiting time as a function of n_m and θ for arrival rate $\lambda = 2.0$ arrivals/slot and repository size $n = 100$ items. In comparison with the previous graphs, this graph shows that as λ increases, the request distribution must skew more toward the first item in order to remove enough requests from the unicast queue in order to prevent it from overflowing. In this case, this occurs when $\theta = 0.6$.

Figure 4.6(e) shows the optimal number of multicast items needed per cycle (M^*) to minimize the expected waiting time for any terminal, based on the previous figures. For arrival rates that do not saturate the infostation system (i.e. $\lambda < 1$), the graph shows that the minimum expected waiting time is achieved by transmitting all items using unicast, when the request distribution is uniform and there is no clearly "popular" item. As the requests

begin to skew toward a small subset of the items (i.e. as θ increases), the number of items that should be multicast increases to a maximum around θ = 1.0 and then begins to decrease again since the request distribution becomes very skewed toward item #1 and there is no need to multicast many of the items of the repository.

For arrival rates that begin to saturate the infostation system (i.e. $\lambda > 1$), the graph shows that for low θ values, we cannot keep up with the high request rate for unicast items so we have no option but to multicast all of the items to satisfy all requests in a finite amount of time. However, as θ increases, and the request distribution skews strongly toward item #1, many items no longer need to be multicast since few if any terminals will be in the area to receive them. As θ approaches ∞, we would expect all of these curves in this graph to converge to M^*=1.

4.5.1.2 Number of repository items is larger than the number of cycle slots (C=100 and n = 1000)

Figure 4.8(a) shows the expected waiting time as a function of M and θ for arrival rate λ = 0.5 arrivals/slot and database size n = 1000 items. Note that in this scenario, the repository is 10 times the size of a cycle, since a cycle can only hold 100 items. This means that the maximum number of multicast items we can have is limited to (the first) $1/10^{th}$ of the database.

Notice figure 4.8(a) that all of the curves start at 75 slots when M = 0, just as in figure 4.6(a). This is due to the fact that $E[W]$ does not depend on n when M = 0. As in the previous set of graphs, as θ increases, the optimal number of multicast items needed to minimize the expected waiting time begins to increase.

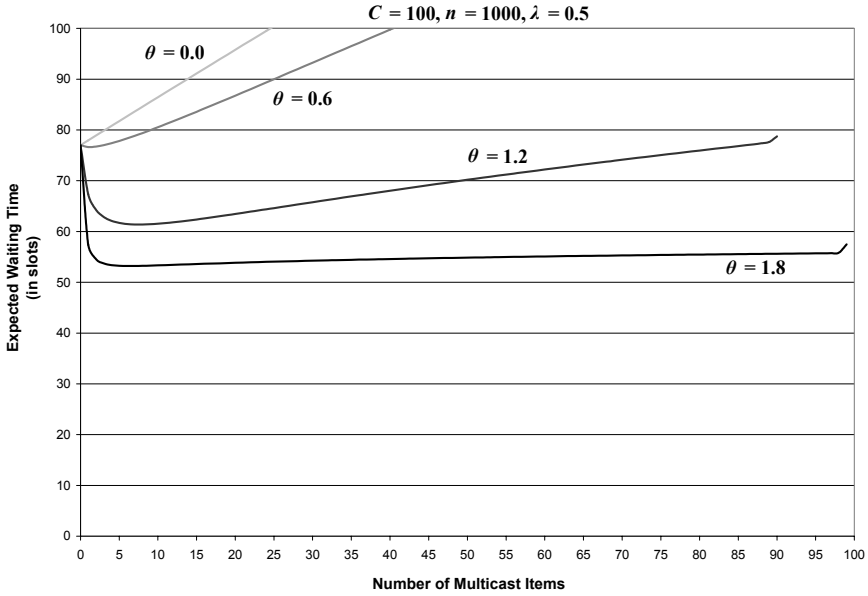

Figure 4.8(a) Expected waiting time as a function of number of multicast items per cycle with λ=0.5 arrivals/slot for various Zipf distributions with parameter θ.

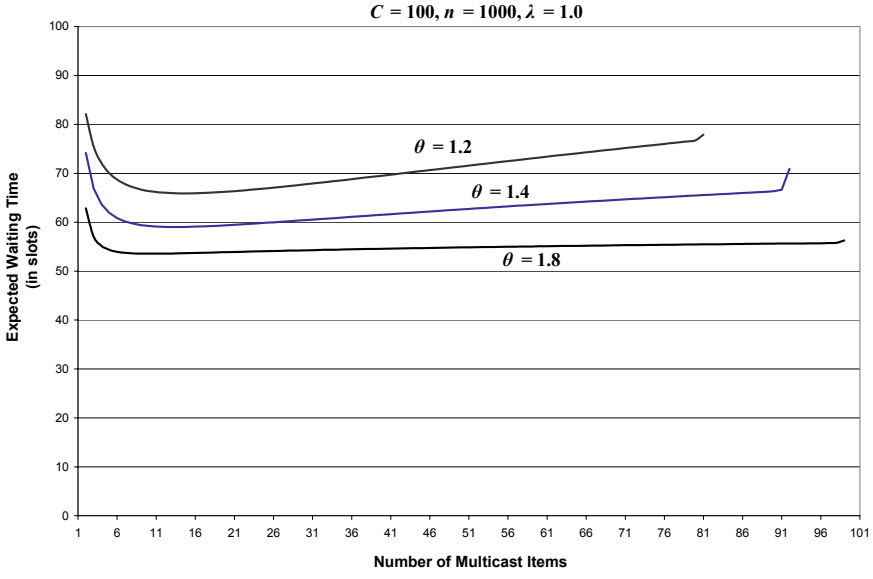

Figure 4.8(b) Expected waiting time as a function of number of multicast items per cycle with λ=1.0 arrivals/slot for various Zipf distributions with parameter θ.

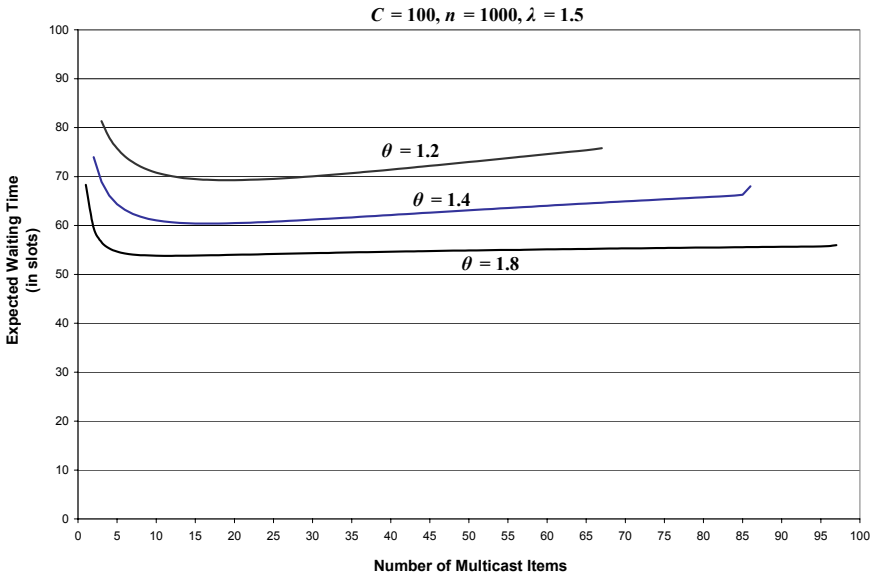

Figure 4.8(c) Expected waiting time as a function of number of multicast items per cycle with λ=1.5 arrivals/slot for various Zipf distributions with parameter θ.

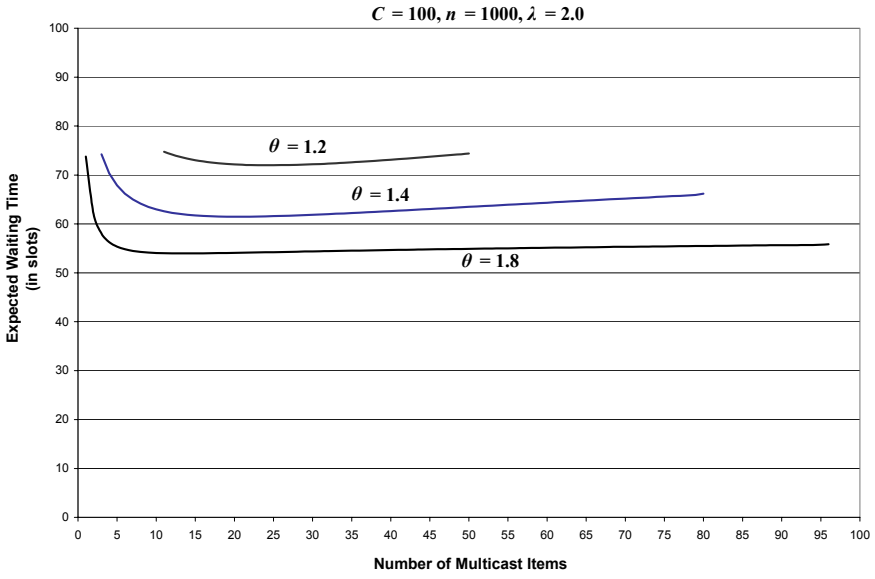

Figure 4.8(d) Expected waiting time as a function of number of multicast items per cycle with λ=2.0 arrivals/slot for various Zipf distributions with parameter θ.

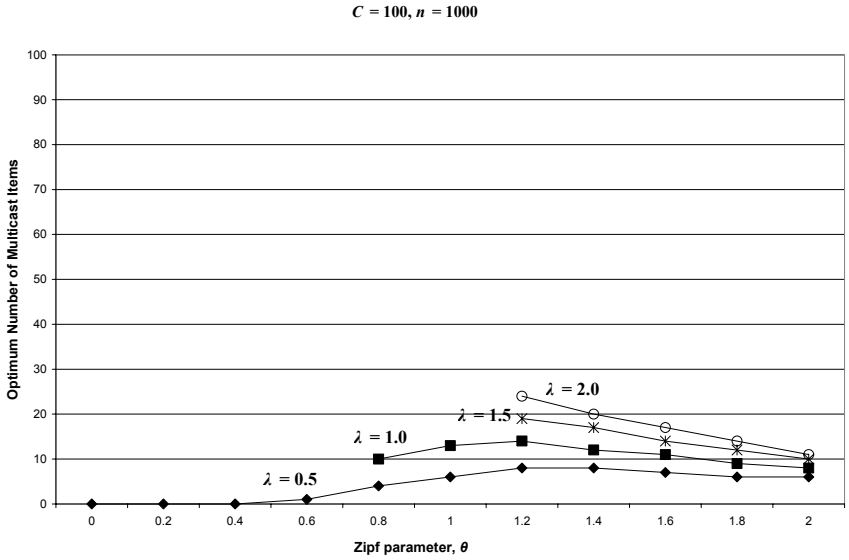

Figure 4.8(e) Optimal number of multicast items per cycle as a function of the Zipf request distribution with parameter θ for various arrival rates λ (in arrivals/slot).

In figure 4.8(a), we see a new feature in that the expected waiting time is unbounded when the number of multicast items M approaches C. The reason for this is because the entire repository cannot be transmitted in one cycle anymore. As M increases, there are fewer slots available to transmit unicast items per cycle. Once $M = C$, only the first M items of the repository can be transmitted; all other items will never be transmitted since there are no slots left in each cycle to utilize. Thus, any one terminal that wants a unicast item (numbered item n_m +1 to n) will cause the expected waiting time to increase to infinity. Even if the number of multicast items is less than C, but close to C, the number of slots remaining for unicast items is quite small and requests for unicast items will remain on the infostation scheduling queue for many cycles, causing the expected waiting time to explode.

If there are too many multicast items in the cycle, then the queue that holds the requests for unicast items will grow without bound, and the effective waiting time cannot be computed. In general, the effective waiting time will be undefined if

$$\frac{(1-B)\lambda C}{U} > D \tag{4.21}$$

In the cases where $C \neq N$, when equation 4.21 holds, the curves are not plotted since $\rho > 1$.[3]

Figure 4.8(b) shows the expected waiting time as a function of M and θ for arrival rate λ = 1.0 arrivals/slot and repository size n = 1000 items. As in the previous set of graphs, as the arrival rate reaches the saturation point, the expected waiting time becomes unbounded for $\theta = 0.0$.

By examining the curve for $\theta = 0.6$, we see that the number of multicast items we need to transmit in order to get a reasonable waiting time is very limited: smaller number of multicast items quickly flood the infostation scheduling queue with unicast requests, while a larger number of multicast items causes the large number terminals waiting for unicast requests to wait indefinitely.

Figures 4.8(c,d) show the expected waiting time as a function of M and θ for arrival rates λ = 1.5 and 2.0 arrivals/slot, respectively, and repository size n = 1000 items. In both cases, clear minimums are seen in the curves, indicating that there is indeed an optimal number of multicast items to transmit even when the repository size is much larger than the cycle size, as long as the request distribution is significantly skewed toward a very small subset of the repository items.

Figure 4.8(e) shows the number of multicast items needed per cycle to minimize the expected waiting time for any terminal. Comparing this figure with figure 4.6(e), we see one important difference. For higher arrival rates ($\lambda > 1$), there is no optimal number of multicast items that will minimize the expected waiting time for request distributions close to uniform (θ approaches 0). It is impossible for the infostation to keep up with the large number of varied requests coming from incoming terminals with such a uniform

[3] In some cases, points on the curve that yield ρ close to 1 as M approaches C are also not plotted due to numerical instability present when computing the probability of having a given number of terminals in the unicast queue when ρ is slightly less than 1.

distribution. As θ increases and the requests begin to skew toward the first few items of the repository, an optimal number of multicast items per cycle emerges. In addition, as θ increases toward ∞, the requests will all skew toward item #1 and the optimal number of multicast items per cycle should approach 1.

4.5.1.3 Number of repository items is much larger than the number of cycle slots (C=100 and n = 10000)

Figure 4.9(a) shows the expected waiting time as a function of M and θ for arrival rate λ = 0.5 arrivals/second and repository size n = 10000 items. In this scenario, the repository is 100 times the size of a cycle, since a cycle can only hold 100 items. This means that the maximum number of multicast items we can have is limited to (the first) $1/100^{th}$ of the repository, and most items must be unicast due to the very small cycle size. As before, we can see for low θ values, because of the low arrival rate, we can unicast all items and minimize the expected waiting time, whereas multicast some of the items will begin to cause significant delay for terminals wanting unicast items since these follow the multicast items and receive a smaller percentage of the overall bandwidth per cycle. Notice that due to the large repository size and small cycle size, as nm approaches C, the expected waiting time explodes regardless of θ. In fact, if $M = C$, it is impossible to transmit most of the repository so the expected waiting time will be unbounded. Thus, even for θ = 1.8, where M can range from 1 to 99 in order to keep the expected waiting time low (since most terminals want one of the first few items), once M = 100, the expected waiting time goes to infinity.

Figures 4.9(b,c,d) show the expected waiting time as a function of M and θ for arrival rate λ = 1.0, 1.5 and 2.0 arrivals/second, respectively, and repository size n = 10000 items. Just as in the previous scenarios, reaching the saturation point in terms of arrivals causes the expected waiting time to explode for request distributions that are close to uniform. In this case, for $\theta \leq 0.6$, we get expected waiting times that easily exceed 100 seconds. For higher θ values, even though the request skew more toward item #1, the cycle can only accommodate $1/100^{th}$ of the database, so the system easily becomes overwhelmed. For example, in figure 4.9(b) for λ = 1.0 and θ = 1.2, transmitting 73 multicast items causes

the expected waiting time to increase without bound because $\rho = \lambda_u/\mu_u = 0.27599/0.27 > 1$ (see equation 4.21).

Figure 4.9(e) shows the number of multicast items needed per cycle to minimize the expected waiting time for any terminal. For higher arrival rates and low θ (more uniform requests), there is no optimal number of multicast items per cycle that will minimize the expected waiting time, since the expected waiting time is unbounded in these cases. However, for higher θ, we find that there is an optimal number of multicast items that will minimize the expected waiting time for the terminals, even for a scenario where the database size is much larger than the number of items that can be transmitted per cycle.

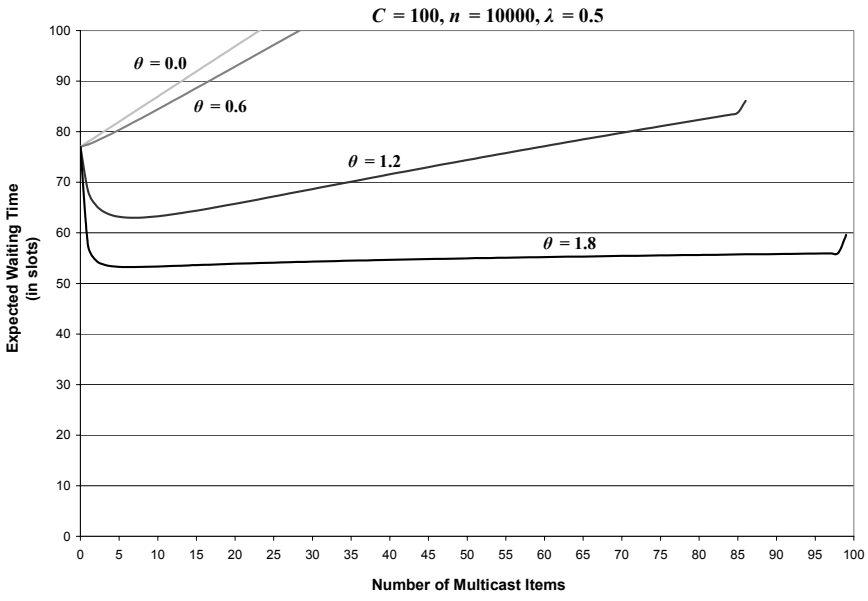

Figure 4.9(a) Expected waiting time as a function of number of multicast items per cycle with λ=0.5 arrivals/slot for various Zipf distributions with parameter θ.

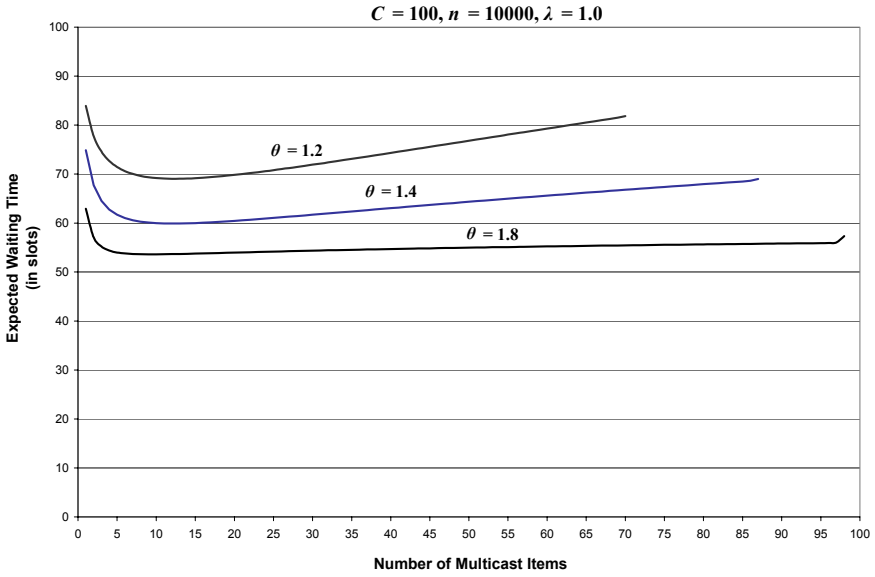

Figure 4.9(b) Expected waiting time as a function of number of multicast items per cycle with λ=1.0 arrivals/slot for various Zipf distributions with parameter θ.

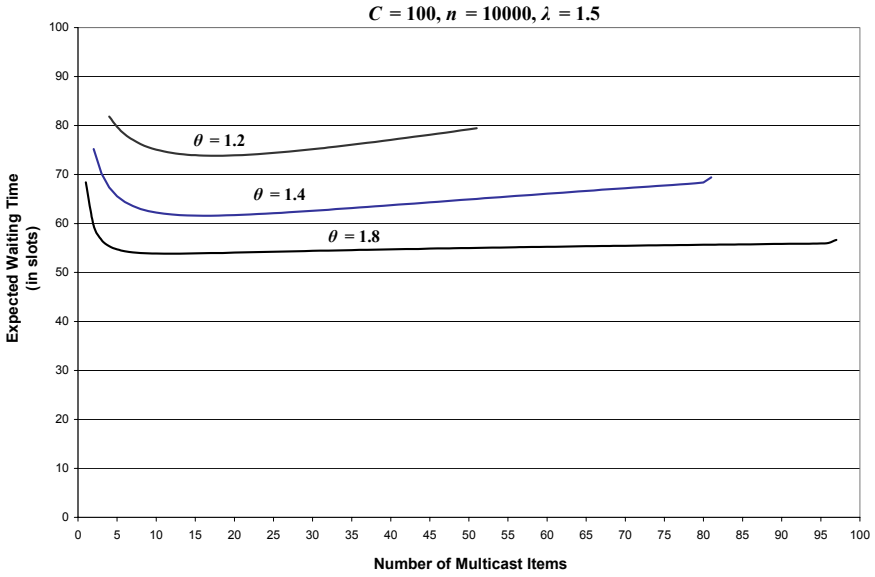

Figure 4.9(c) Expected waiting time as a function of number of multicast items per cycle with λ=1.5 arrivals/slot for various Zipf distributions with parameter θ.

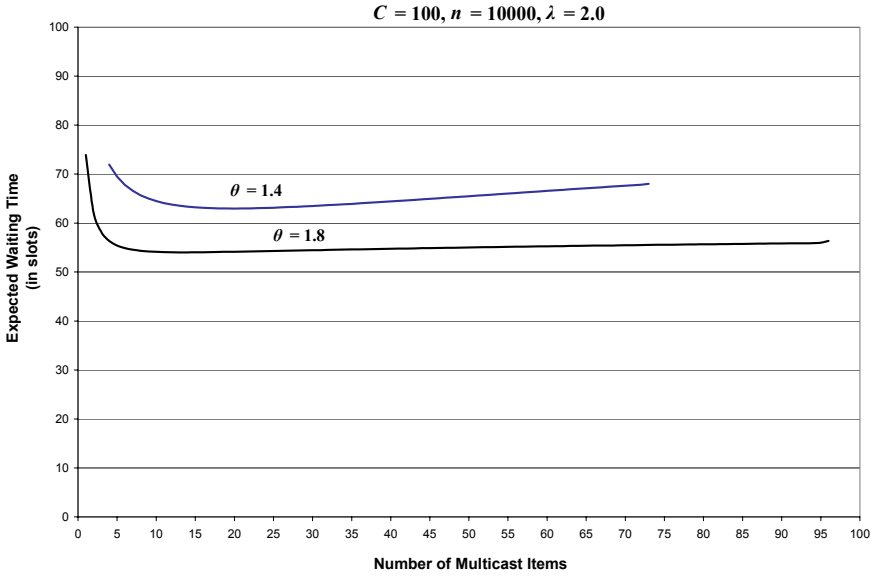

Figure 4.9(d) Expected waiting time as a function of number of multicast items per cycle with λ=2.0 arrivals/slot for various Zipf distributions with parameter θ.

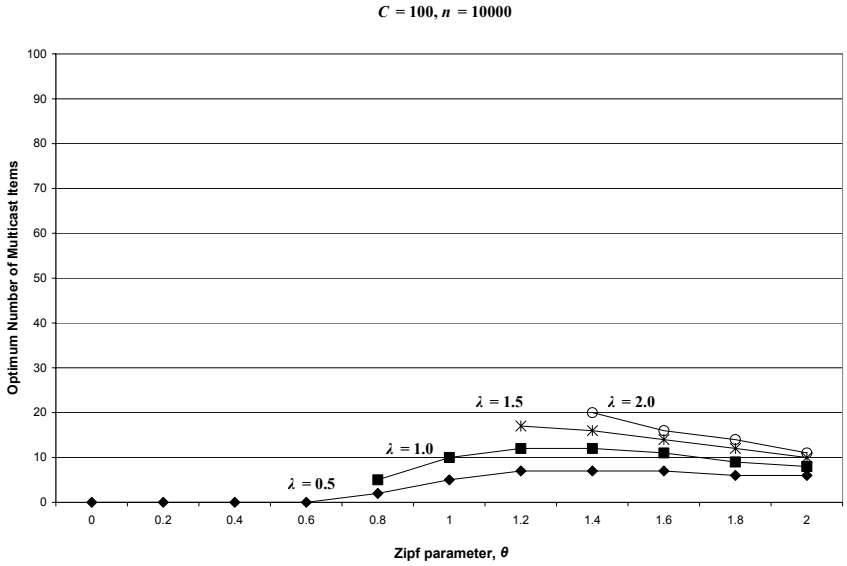

Figure 4.9(e) Optimal number of multicast items per cycle as a function of the Zipf request distribution with parameter θ for various arrival rates λ (in arrivals/slot).

4.5.2 Expected Waiting Time as a function of arrival rate

In this section, we will study the expected waiting time for a terminal as a function of the arrival rate for various request distributions based on the Zipf distribution. As in the previous subsection, we vary the size of the repository, starting with a small repository (that can be completely transmitted in one cycle using multicast) and going up to a very large repository (that requires many cycles to transmit each item of the repository once).

4.5.2.1 Number of repository items and number of cycle slots are equal (C=100 and n = 100)

Figure 4.10(a) shows the expected waiting time as a function of M and λ for a Zipf request distribution with parameter $\theta = 0.0$ and repository size $n = 100$ items. Recall that this request distribution is equivalent to a uniform distribution. In this scenario, if the arrival rate is greater than or equal to 1, the expected waiting time will be unbounded for $M < C$ since $\rho \geq 1$ (see equations 4.18 and 4.19). This is true if the schedule is not pure multicast, so the curves for $\lambda \geq 1$ are undefined when $\theta = 0.0$ and $M < 100$.

Figure 4.10(b) shows the expected waiting time as a function of M and λ for a Zipf request distribution with parameter θ=1.0 and repository size $n = 100$ items. As the request distribution begins to skew toward a small number of "popular" items, we can see clearly that there exists an optimal M such that the expected waiting time for terminals is minimized.

In figure 4.10(b), note that if we multicast the entire repository ($M = C = 100$), the expected waiting time is no longer a function of the arrival rate since the arrival rate only affects the waiting time for unicast items. Therefore, all curves converge to the same expected waiting time as M approaches C. Also, we can observe that as λ exceeds 1.0, these curves do not start at $M = 0$ for the same reason that was discussed in section 4.5.1. In these instances, the queue that holds the unicast requests becomes unstable since $\rho \geq 1$.

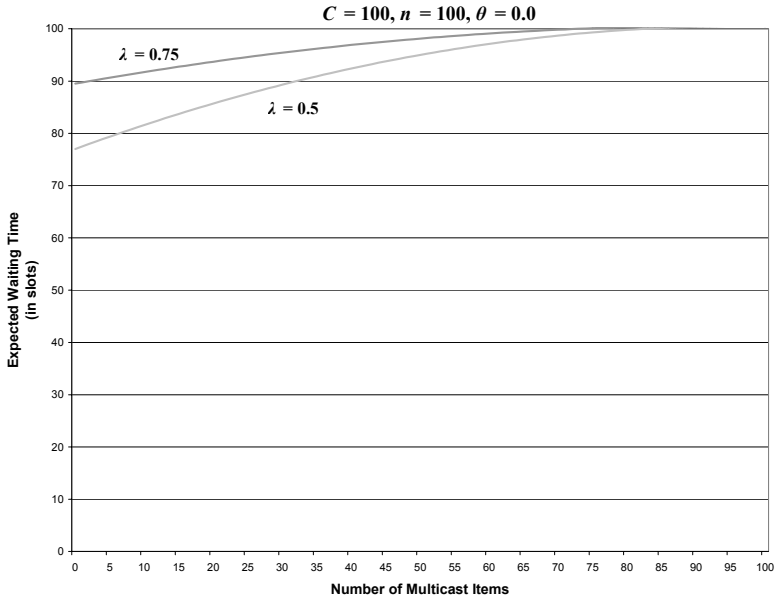

Figure 4.10(a) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 0.0$ for various λ (arrivals/slot).

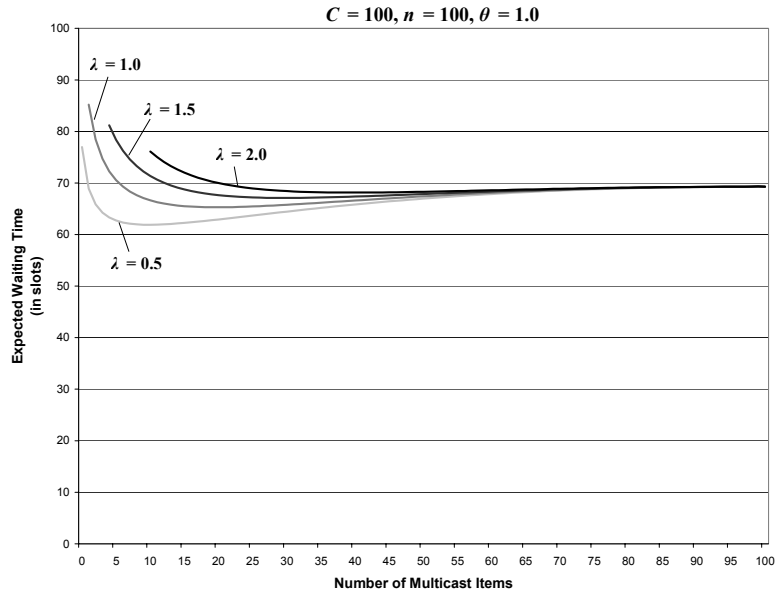

Figure 4.10(b) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 1.0$ for various λ (arrivals/slot).

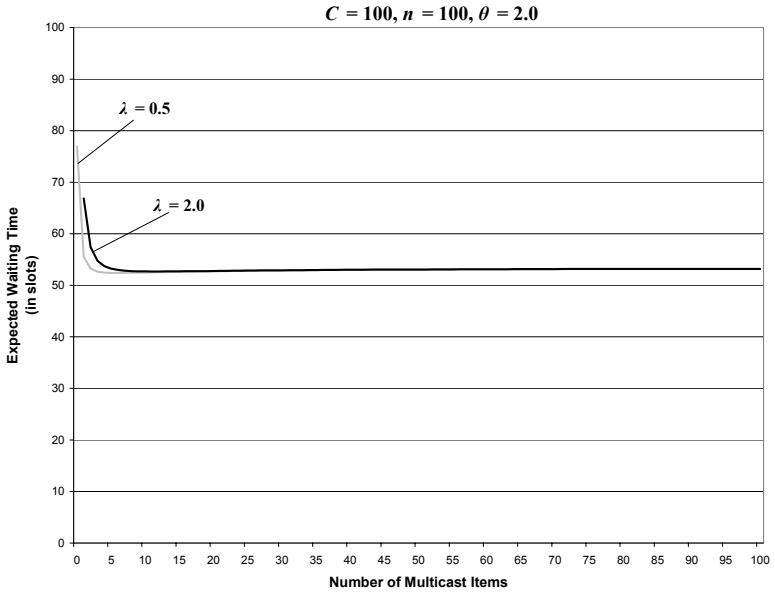

Figure 4.10(c) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 2.0$ for various λ (arrivals/slot).

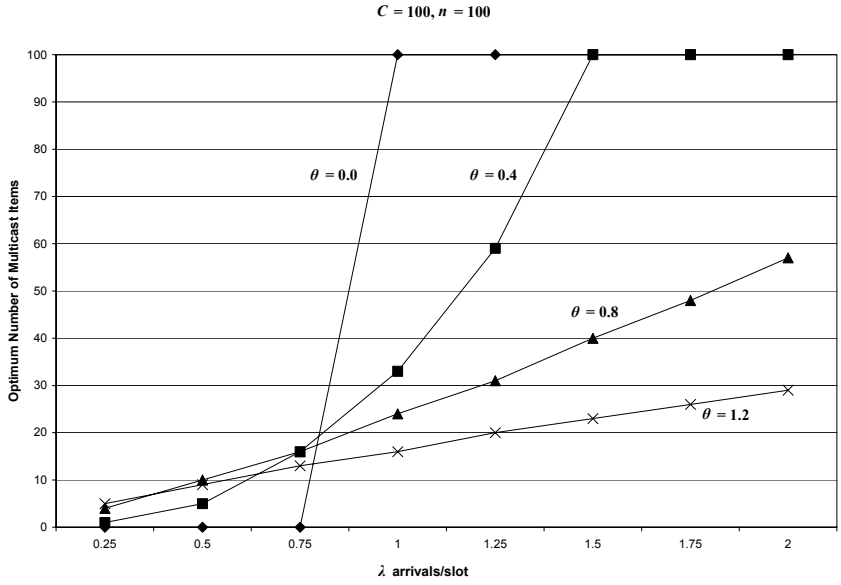

Figure 4.10(d) Optimal number of multicast items per cycle as a function of the arrival rate λ (in terminals/slot) for various Zipf request distributions with parameter θ.

In figure 4.10(c), we can see that for higher θ values, where most terminals want item #1, multicasting more items from the repository does not change the expected waiting time very much.

Figure 4.10(d) shows the values for M*, the optimal number of multicast items per cycle, as a function of the arrival rate for various Zipf request distributions. For $\theta = 0.0$, a uniform request distribution, it does not pay to multicast the repository until the arrival rate starts to saturate the unicast request queue. At this point, we immediately switch to multicasting the entire repository ($M = n = 100$), so that we can satisfy all incoming terminals. As soon as the request distribution begins to skew toward item #1, even with $\theta = 0.4$, we see that it pays to begin multicasting the most popular items even at low arrival rates in order to reduce the expected waiting time overall. As θ increases and the requests skew more toward item #1, we do not have to multicast the entire repository, even at high arrival rates, since most terminals want the first several popular items only, so multicasting the entire repository will not reduce the effective waiting time. Instead, we should multicast only what we need to serve the majority of the terminals, and then immediately unicast requests for higher-numbered (least requested) items rather than multicast them and have the terminal wait until the end of the cycle to receive them.

4.5.2.2 Number of repository items is larger than the number of cycle slots (C=100 and n = 1000)

Figure 4.11(a) shows the expected waiting time as a function of M and λ for a Zipf request distribution with parameter θ=0.0 and repository size $n = 1000$ items. Just as in figure 4.10(a), only curves for $\lambda < 1$ are shown since the expected waiting time is unbounded otherwise. The figure clearly shows that for a uniform request distribution, as long as the arrival rate does not overwhelm the infostation, we should unicast all requests to minimize the expected waiting time.

Figure 4.11(b) shows the expected waiting time as a function of M and λ for a Zipf request distribution with parameter θ=1.0 and repository size $n = 1000$ items. In this case, clear minimums are visible for each arrival rate. Recall that since the entire database cannot

be transmitted in one cycle anymore, we would not expect to see the curves all converge to one point when $M = 0$ as they did when $n = 100$. There is a minimum number of items that must be multicast from equation 4.20. For example, for $\theta = 1.0$ and $\lambda = 1.5$, the minimum number of items we must multicast is M' such that $w_1 + ... w_{M'} > 1 - 1/1.5$, which leads to $M' = 7$. Likewise, if we multicast too many items, the amount of bandwidth available for the unicast requests is far too small to handle requests for most items in the repository. For example, for θ=1.0 and $\lambda = 1.5$, if we multicast the first 31 items of the repository (i.e. M=64), we have $\rho = \lambda_u/\mu_u = 0.692988/0.69 > 1$ (from equation 4.21).

Figure 4.11(c) shows the expected waiting time as a function of M and λ for a Zipf request distribution with parameter θ=2.0 and repository size $n = 1000$ items. Now, the requests are skewed even more toward the first several items of the repository. We can see that unicasting all items (i.e. $M = 0$) will cause an unbounded expected waiting time since all requests for popular items will have to be served individually, and multicasting the maximum M=100 items will also cause an unbounded expected waiting time since there will be no time in the cycle to transmit any of the other remaining 900 items, even if they are requested rarely. An optimal number of multicast items exists but adding additional multicast items does not increase the expected waiting time very much until we reach $M = 100$.

Figure 4.11(d) shows the values for M^*, the optimal number of multicast items per cycle, as a function of the arrival rate for various Zipf request distributions. As θ increases, the graph shows that it pays to multicast more items per cycle until θ is larger than 1. At this point, the distribution becomes skewed enough toward item #1, that it pays to reduce the number of multicast items transmitted per cycle in order to provide the maximum amount of bandwidth for the unicast requests.

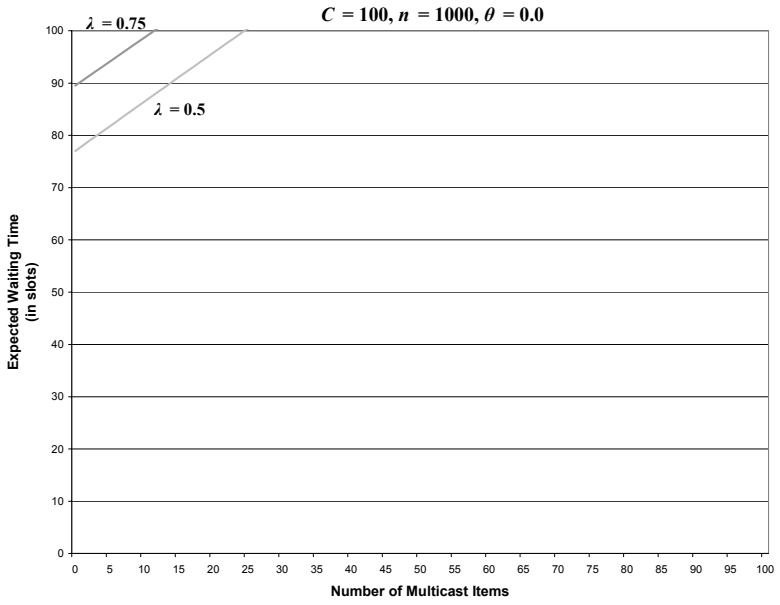

Figure 4.11(a) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 0.0$ for various λ (arrivals/slot).

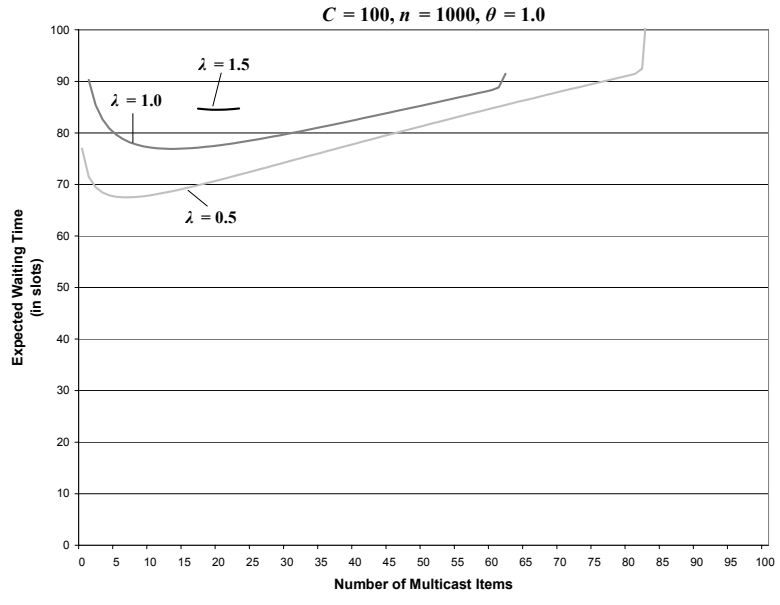

Figure 4.11(b) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 1.0$ for various λ (arrivals/slot).

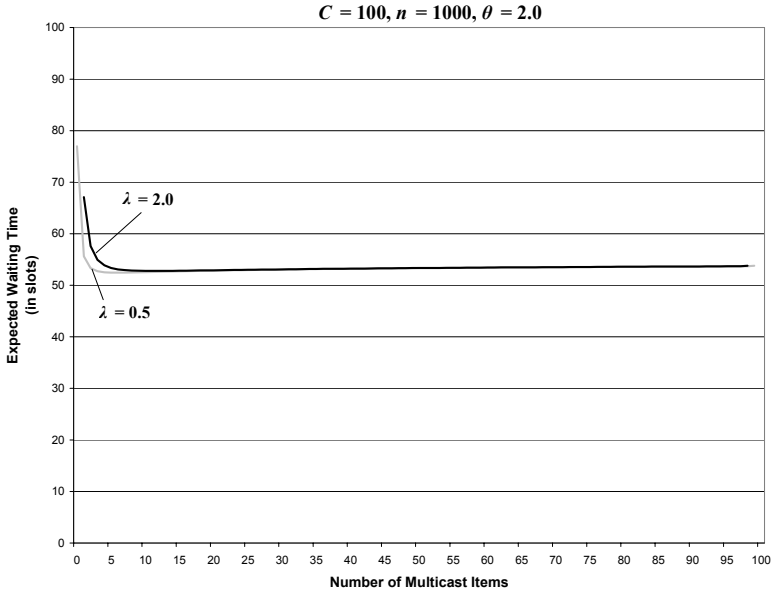

Figure 4.11(c) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 2.0$ for various λ (arrivals/slot).

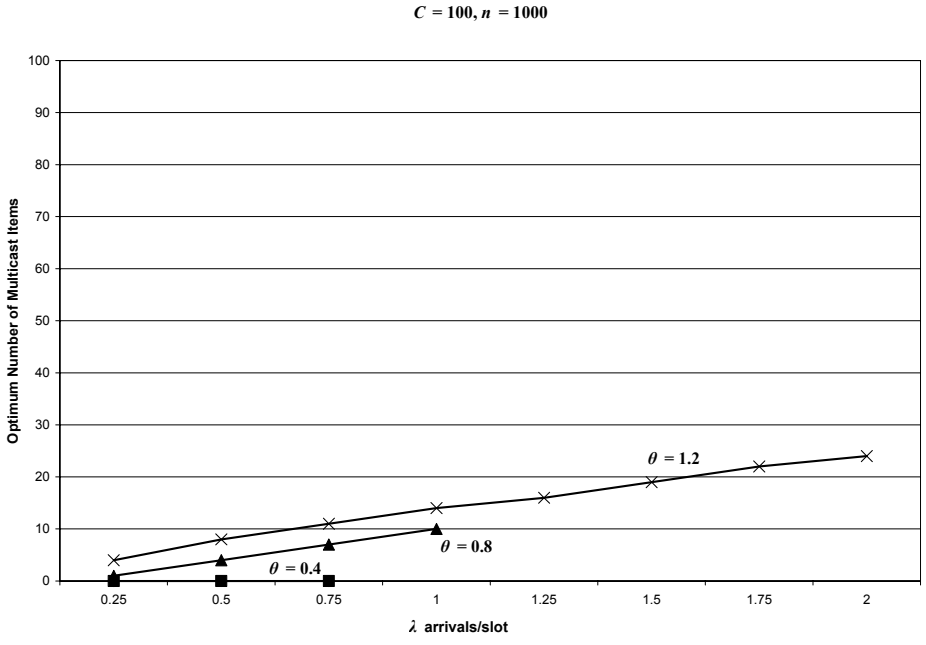

Figure 4.11(d) Optimal number of multicast items per cycle as a function of the arrival rate λ (in terminals/slot) for various Zipf request distributions with parameter θ.

4.5.2.3 Number of repository items is much larger than the number of cycle slots (C=100 and n = 10000)

Figure 4.12(a) shows the expected waiting time as a function of M and λ for a Zipf request distribution with parameter $\theta = 0.0$ and repository size $n = 10000$ items. This figure is very similar to figure 4.11(a), in which the infostation should simply unicast all items if the arrival rate is low. Once the arrival rate increases to the saturation point, the expected waiting time becomes unbounded.

Figures 4.12(b,c) show the expected waiting time as a function of M and λ for a Zipf request distribution with parameter $\theta = 1.0$ and $\theta = 2.0$, respectively, and repository size $n = 10000$ items. These graphs demonstrate similar properties to those shown in the previous subsection. Increasing the size of the repository to a much larger size does not change the design philosophy for the infostation in any significant way.

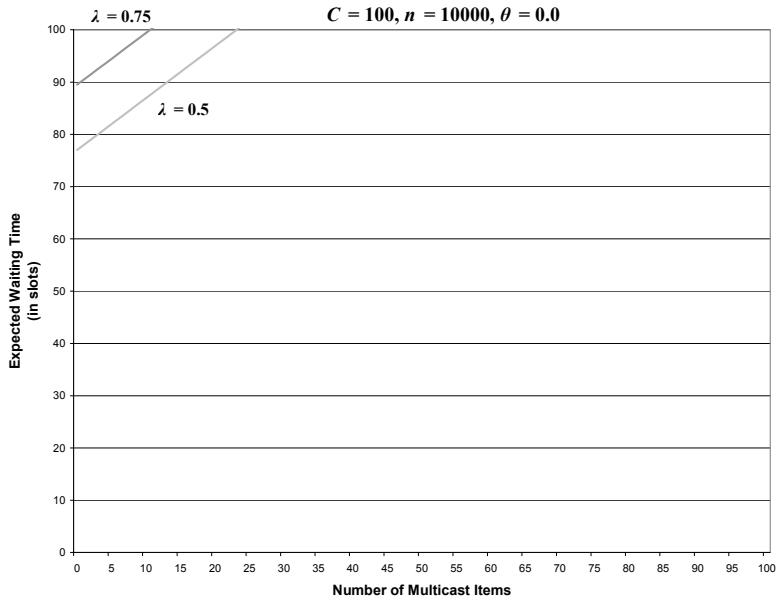

Figure 4.12(a) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 0.0$ for various λ (arrivals/slot).

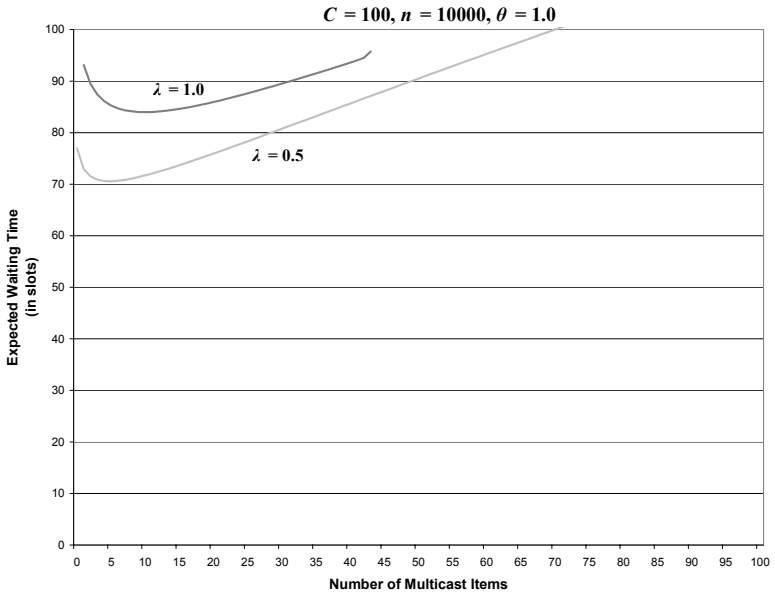

Figure 4.12(b) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 1.0$ for various λ (arrivals/slot).

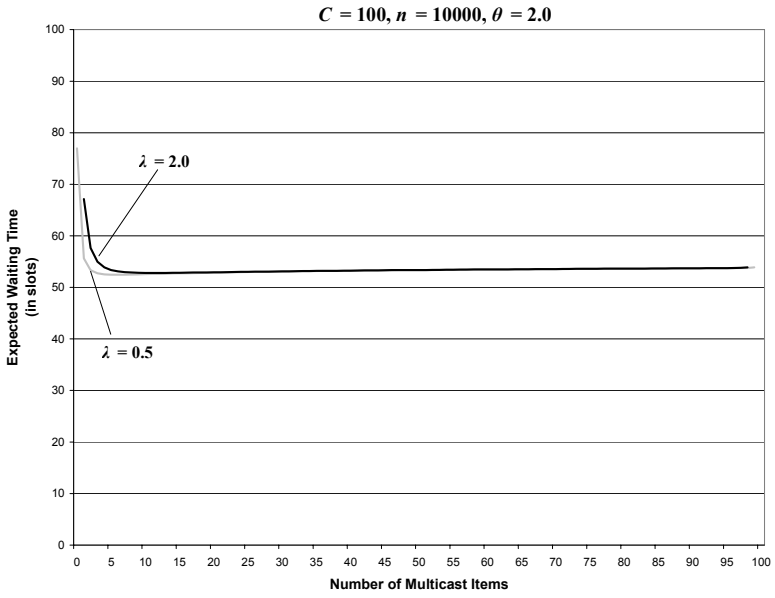

Figure 4.12(c) Expected waiting time as a function of number of multicast items per cycle using a Zipf request distribution with $\theta = 2.0$ for various λ (arrivals/slot).

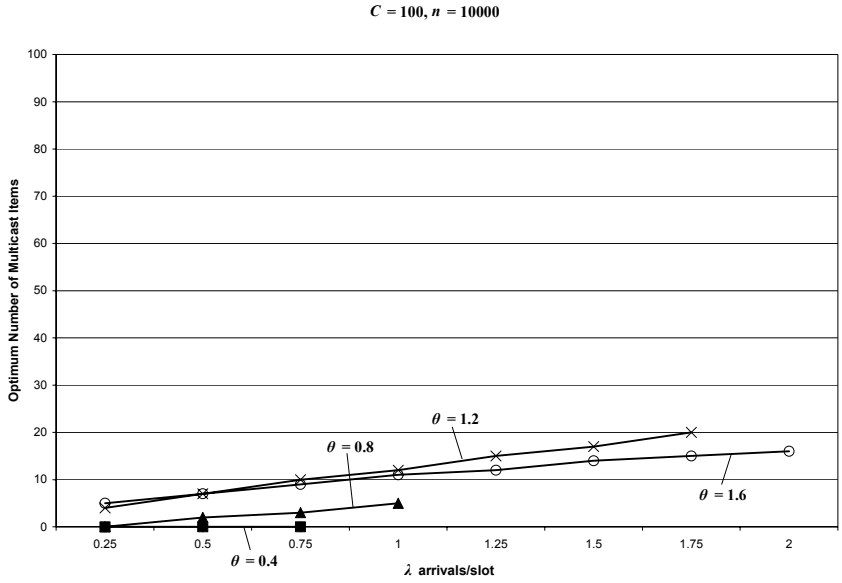

Figure 4.12(d) Optimal number of multicast items per cycle as a function of the arrival rate λ (in terminals/slot) for various Zipf request distributions with parameter θ.

Figure 4.12(d) shows the values for M^*, the optimal number of multicast items per cycle, as a function of the arrival rate for various Zipf request distributions. Just as in figure 4.11(d), if we increase the database size from $n = 1000$ items to $n = 10000$ items, the same design philosophy appears. As θ increases we should begin to multicast more items, but eventually we will begin to pull back and multicast less since most requests are skewed toward a small subset of items. This provides the maximum amount of bandwidth for requests for the large number of remaining items in the repository.

4.6 Summary

The infostation system can require each terminal to send a brief feedback message after an item is received. This information can be used in real time to approximate the arrival rate of terminals and the request distribution of the items in its repository. Based on this information, the items can be ordered based on popularity (probability of request) and the scheduler can use the formulas derived in this chapter to determine the optimal number of multicast items to transmit per cycle. The optimal number of multicast items to transmit

can be stored in a lookup table and used after a given number of cycles are transmitted in order to easily adapt to traffic changes (i.e. a rush hour with many arrivals) or changing data requests (i.e. special news events).

In conclusion, it is clear that as the arrival rate increases or the request distribution skews toward a few items of the repository, the optimal schedule for each cycle should include some multicast items in order to minimize the expected waiting time for the terminals using the infostation. This is especially important since the terminals using the infostation are mobile and will leave the coverage area in a short period of time.

Chapter 5

SIMULATION OF THE INFOSTATION

The infostation system described in chapter 2 is being constructed and tested at Polytechnic University. Part of the research work includes a simulator to study the performance of the infostation system under various real-world conditions. The simulator and results derived from its use are described in this chapter.

We begin with the differences between the mathematical description of the infostation system from chapter 4 and the actual system that is simulated. A description of the simulator design follows along with various simulation runs. Comparisons are made between the simulated results and the infostation model presented in chapter 4 which is based on the actual infostation system being designed. The simulator can also be used to study other infostation designs and criteria that would normally be difficult to analyze mathematically. The chapter concludes with two such models, one in which all items are multicast and another in which packet errors are introduced to the model.

5.1 Comparison of theoretical and simulated infostation systems

In the last chapter, a model consisting of a single broadcast channel was presented for the infostation. In this model, data items can be transmitted to waiting terminals using either multicast or unicast techniques. The goal was to study the system to determine how many items the system should multicast as a function of the arrival rate of terminals and the expected request distribution, in order to minimize the expected waiting time for a terminal between the time it needs a data item and the time that it is transmitted completely to the waiting terminal.

In the model described in chapter 4, the infostation dedicated the entire transmission bandwidth to transmission of single data items, where each item is transmitted using multicast or unicast. The infostation transmits in cycles, where each cycle begins with the transmission of all designated multicast items, followed by the transmission of unicast items which are scheduled based on prior requests from waiting terminals. The items that are chosen to be multicast are based on the arrival rate and expected request distribution in such a way that the expected waiting time for any item is minimized for all terminals.

When simulating the infostation, we need to make some assumptions in order to model the system. When terminals make requests for data items, these requests are made during the request phase of the cycle. An assumption made in the simulator is that these requests will not collide with one another at the infostation, although in a real system, there would need to some mechanism to handle the collisions such as some form of a back-off algorithm like that used in 802.11b. Another assumption that is made is that distribution of requests is already known, so the simulator can be set up to multicast a specific number of data items from the database. The number of multicast items in one simulation run does not change throughout the duration of the simulation. In a real system, terminals would use the feedback phase of each cycle to send information about data items that were received (either through multicast or unicast) so that the server can adjust the broadcast schedule to reflect the needs of the current terminals in the area. This could lead to a change in the number of multicast items and/or the order in which multicast items are transmitted in a cycle.

5.2 Simulator design

The simulator for this infostation system was written in C using the CSIM simulation libraries [19]. CSIM provides a number of components that are useful in system simulation including processes, events and mailboxes.

Each item (and index) that is transmitted is modeled in CSIM as an event. Processes in CSIM can set and clear events or can wait (i.e. suspend) for events to be set. Requests are stored in a mailbox which can be read by any process. (Specifically, only our server will be able to read these messages since we are modeling a real system where the requests would be directed toward the server and not other terminals.)

The simulator is made up of the following processes, as shown in Figure 5.1:

- Server

  At the start of each cycle, the server sets an event to indicate to all waiting terminals that the index is ready for download. Once the index time has passed, the server clears this event. The server "sends" each item out to the terminals by setting an event associated with the specific item. Once all terminals have had a chance to "download" the item if necessary, the event is cleared and the server sets the next event for the next item's broadcast. After all scheduled items for the cycle are "broadcast", the server reads all requests made from terminals that are stored in the mailbox and passes this information to the scheduler in order to derive the broadcast schedule for the next cycle.

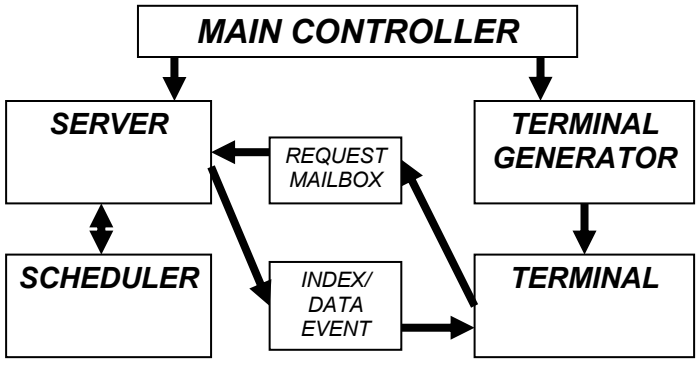

Figure 5.1 Principle components of the infostation simulator.

- Terminal Generator

  The terminal generator uses the expected arrival rate to create each new terminal at the required time. The terminal generator also monitors the average waiting time of terminals that have completed; once this average waiting time stabilizes, the terminal generator will stop generating new terminals and will monitor the remaining terminals until completion.

- Terminal

  Each terminal is a process that is created by the terminal generator at the appropriate time in the simulation based on the expected arrival rate λ according to a Poisson process. The terminal randomly generates one item that it wants based on the Zipf distribution with parameter θ. The terminal then suspends until an index event occurs, at which time the terminal continues execution and processes the index that is stored in a shared variable initialized by the server. If the terminal finds its item scheduled for this cycle, the terminal suspends again until the event associated with that item is set by the server. If the terminal does not find the item scheduled in the index, the terminal will suspend until the request period of the cycle, at which time the terminal will create a message with the item identifier it desires and its terminal ID and store this message in a global mailbox, simulating the uplink channel to the server.

- Scheduler

  The scheduler is tightly coupled with the server, and its main function is to build an index for each cycle specifying what will be broadcast (multicast and unicast). Requests from the mailbox are sent, once per cycle, to the scheduler for processing. The scheduler will determine the classification of the item desired (multicast or

unicast) and will queue up requests for unicast items for inclusion in a subsequent index. Unicast requests are served in a FIFO manner. Multicast requests are discarded in this simulator since these items are already scheduled for transmission in every cycle. Generally, terminals will not make requests for multicast items since there will be a flag in the index indicating this property for each multicast item. However, it is possible that a terminal may miss an index due to packet errors or multicast items may be split into groups to be transmitted in every other cycle, in which case the terminal may not know of the multicast status of its desired item when it receives its first index. In these cases, terminals may request an item designated as multicast which is ignored by the scheduler.

- Main controller

  The controller sets up the server and terminal generator processes and creates data structures to hold the results for each terminal as each terminal completes.

Once the simulation is complete, the simulator outputs the average waiting time for all terminals and the percentage of terminals that received their items before leaving the infostation coverage area. The simulator also outputs additional information including the number of uplink requests made by terminals per cycle, but these results are not presented in this dissertation.

5.3 Simulation Parameters

The first set of simulations analyze the infostation model as described in Chapter 4 with the following assumptions:

- Cycle lengths are fixed at $C = 100$ seconds.

- Each item takes exactly 1 second to transmit wirelessly. (i.e. $D = 1$)

- The index is broadcast at the start of each cycle only, and the time it takes to broadcast the index is set to 0 (since the size of the index with respect to the data items is negligible).

- The request time and feedback time are also set to 0 in order to compare our simulated results with those derived in Chapter 4. In the actual infostation system, these times would not be 0, but would be set depending on the expected arrival rate of terminals into the system area (for requests) and the expected completion rate of terminal within the area (for feedback).

- Terminals that arrive will make a request for their item immediately upon arrival instead of waiting for the next request period. This scenario models the system analyzed in Chapter 4 well, while the actual system will vary since terminals will not know if the infostation is within range until they receive an index or other beacon.

- We assume that there are no collisions on the uplink channel by terminals requesting items at the same time.

Each simulation was executed for at least 500 terminal arrivals. After the terminal generator starts 500 terminals, it checks the average waiting time of completed terminals once after an additional 50 terminals are generated. When the change in average waiting time is less than 0.1%, the terminal generator stops generating new terminals and waits for all remaining terminals to complete before calculating the final expected waiting time for the entire sequence of generated terminals.

5.4 General Infostation Simulation Results

The simulator was tested on the same set of parameters that were analyzed in Chapter 4. We compare the results of the mathematical analysis with the simulated system in this section.

5.4.1 Expected waiting time

Figures 5.2(a-d) show the results of simulations for C = 100 seconds, and a database size of N = 100 items. By comparing these figures to those in chapter 4 (figures 4.6a-d), we can see that the simulated infostation system behaves in a very similar fashion to the mathematical infostation model from Chapter 4, with respect to expected waiting time. In each of these cases, the optimal points appear at nearly the same locations in the corresponding graphs, which indicates that the mathematical model closely predicts the behavior of the real system under the same constraints. The jagged nature of the graphs in figures 5.2(b-d) is a result of the random generation of terminals using a predefined function in CSIM for a Poisson process.[4]

Figures 5.3(a-d) show the results of the simulator on a system with cycle length C = 100 seconds and a database size of N = 1000 items. The behavior of the simulated system again tracks the mathematical model derived in Chapter 4 very well (figures 4.8a-d). However, in this case, the simulator does provide results for expected waiting time in cases where the mathematical model could not (i.e. in cases where ρ is close to 1). We can clearly see that for low arrival rates, it does not pay to multicast any items when the request distribution is nearly uniform. However, as the distribution skews, it is advantageous to multicast approximately the first 5-10 most popular items. The results show that for highly-skewed distributions, even fewer items can be transmitted using multicast to minimize the expected waiting time. When the arrival rate increases, the expected waiting time is minimized by transmitting approximately the first 15 items using multicast (i.e. 1.5% of the database) in those cases where the request distribution is skewed. Finally, when the database size is increased dramatically with respect to the cycle length (C=100, N=10000), we show simulation results in figures 5.4(a-d). Again, these results agree with the mathematically-predicted results in Chapter 4 (figures 4.9a-d), showing clear minima as θ increases for the skewed distributions ($\theta \geq 1.0$).

[4] λ represents the expected arrival rate, and in most of the simulations, the actual average arrival rate was very close to these values, but there were small variations from run to run depending on the actual sequence of random numbers that were generated.

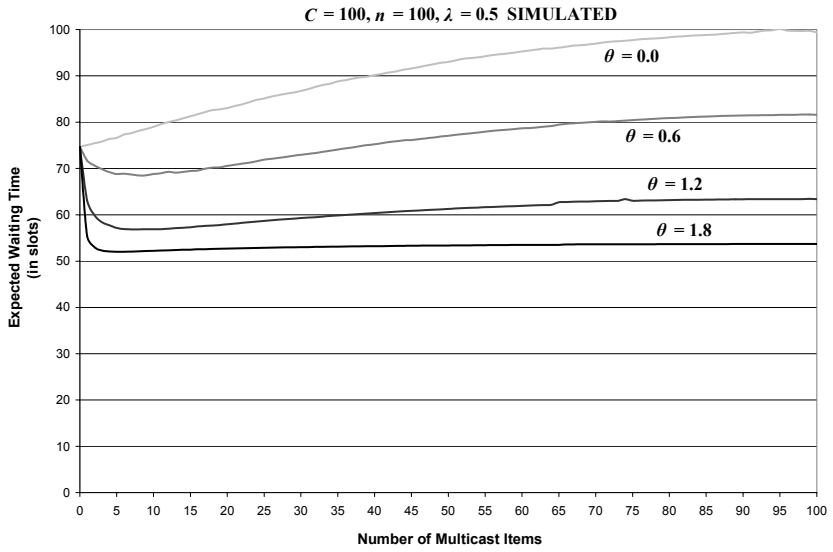

Figure 5.2(a) Simulated expected waiting time as a function of number of multicast items per cycle with λ=0.5 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

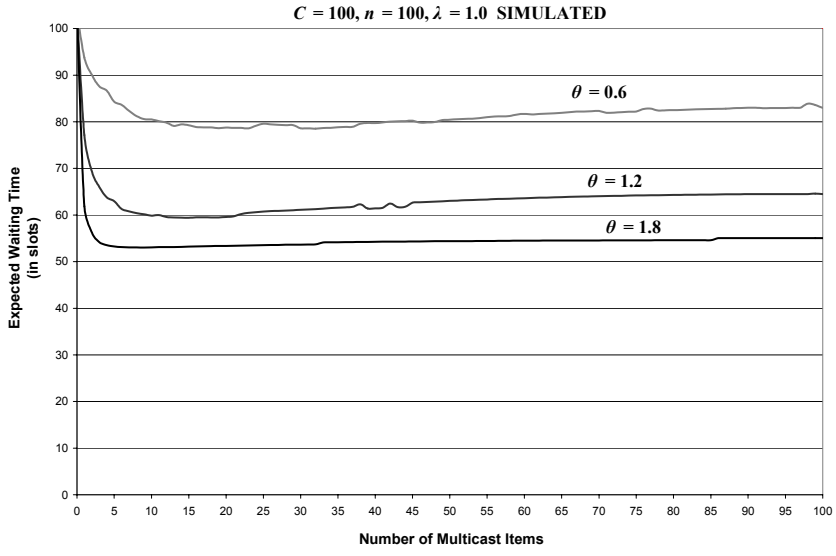

Figure 5.2(b) Simulated expected waiting time as a function of number of multicast items per cycle with λ=1.0 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

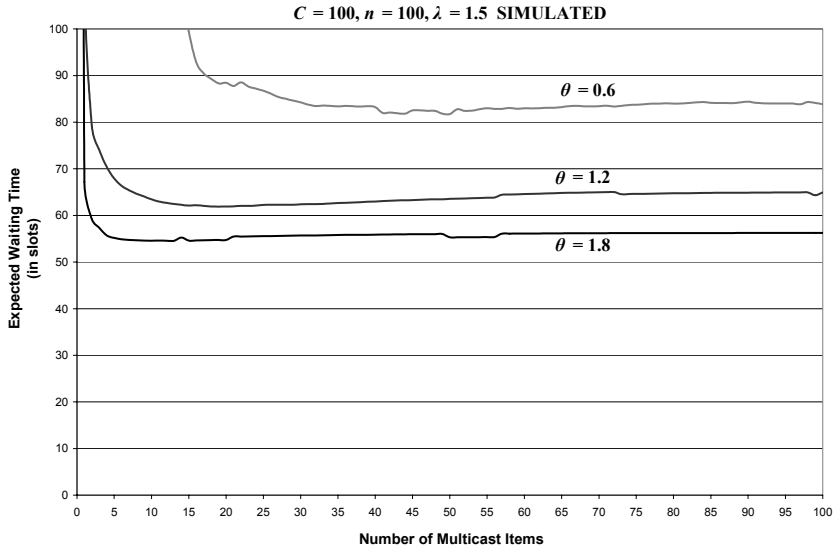

Figure 5.2(c) Simulated expected waiting time as a function of number of multicast items per cycle with λ=1.5 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

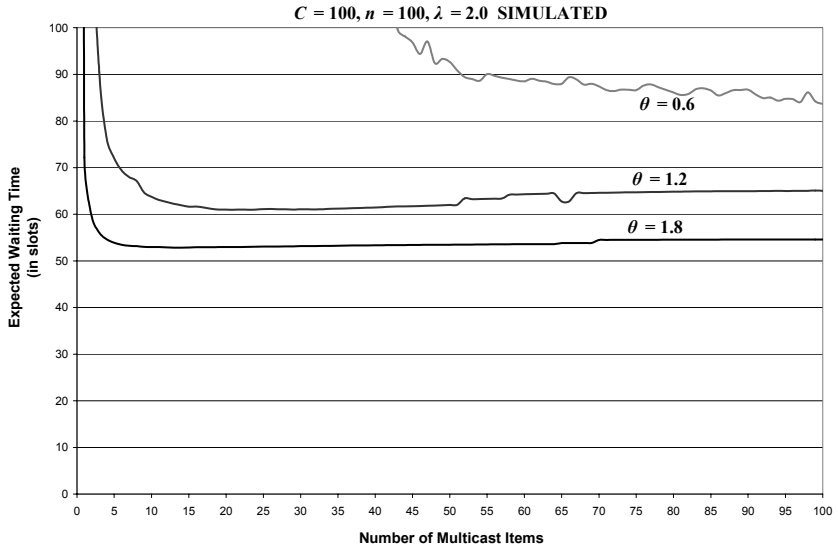

Figure 5.2(d) Simulated expected waiting time as a function of number of multicast items per cycle with λ=2.0 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

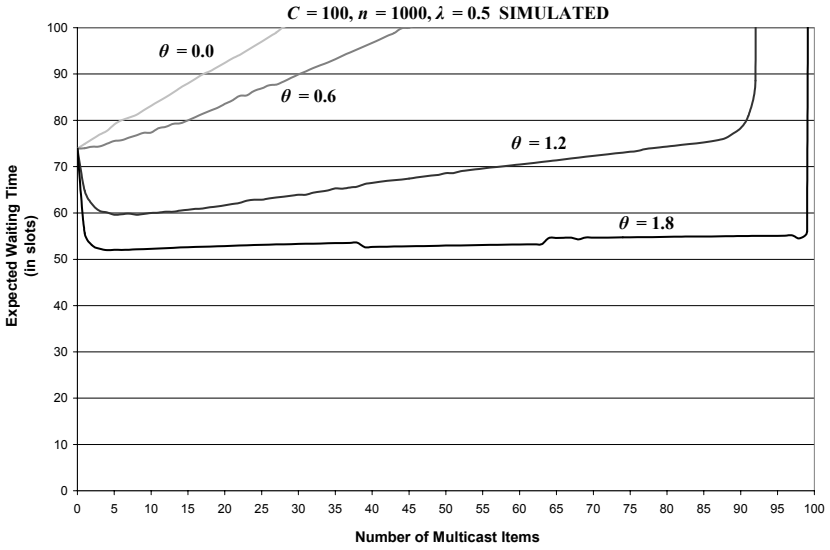

Figure 5.3(a) Simulated expected waiting time as a function of number of multicast items per cycle with λ=0.5 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

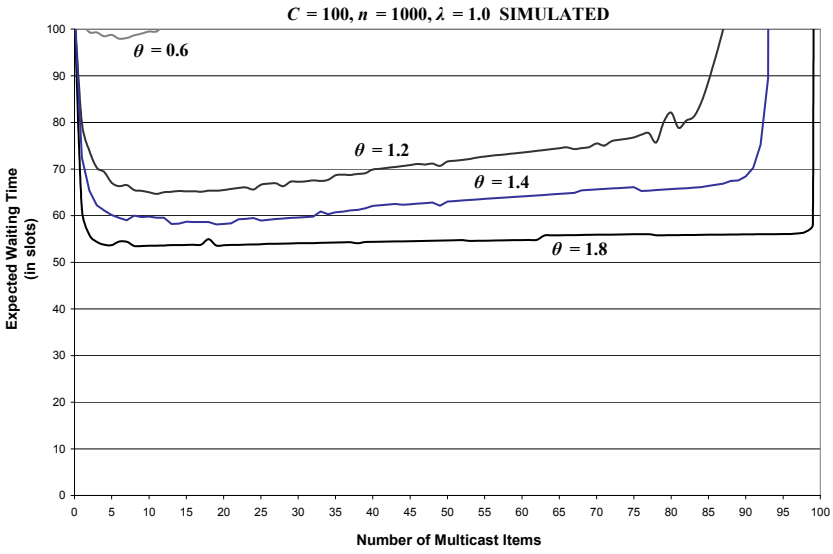

Figure 5.3(b) Simulated expected waiting time as a function of number of multicast items per cycle with λ=1.0 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

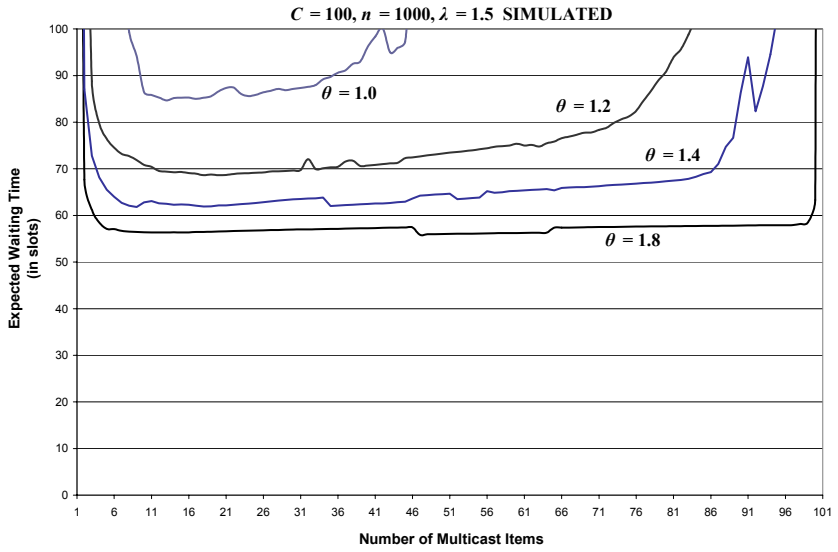

Figure 5.3(c) Simulated expected waiting time as a function of number of multicast items per cycle with λ=1.5 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

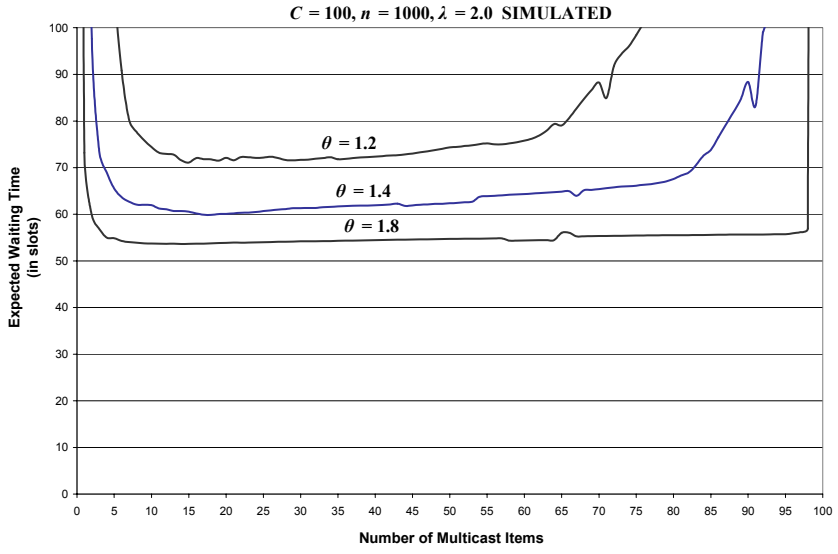

Figure 5.3(d) Simulated expected waiting time as a function of number of multicast items per cycle with λ=2.0 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

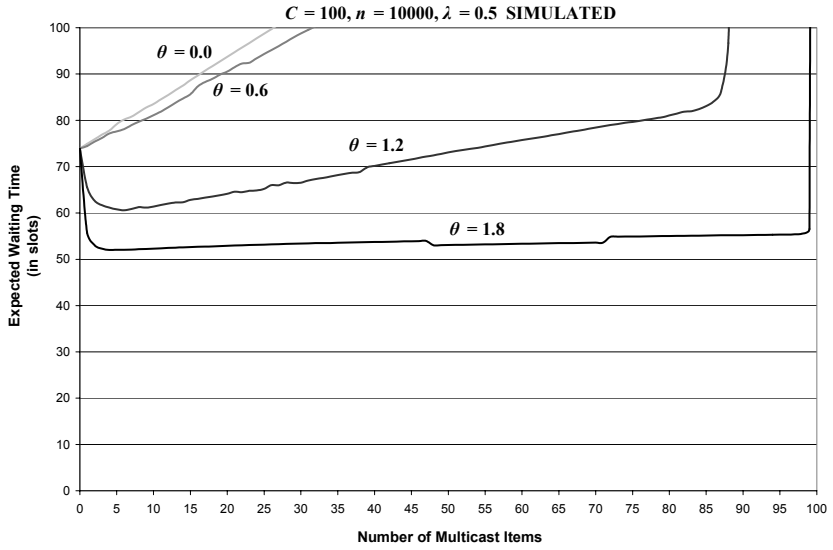

Figure 5.4(a) Simulated expected waiting time as a function of number of multicast items per cycle with λ=0.5 arrivals/slot and database size n = 10000 items for various Zipf distributions with parameter θ.

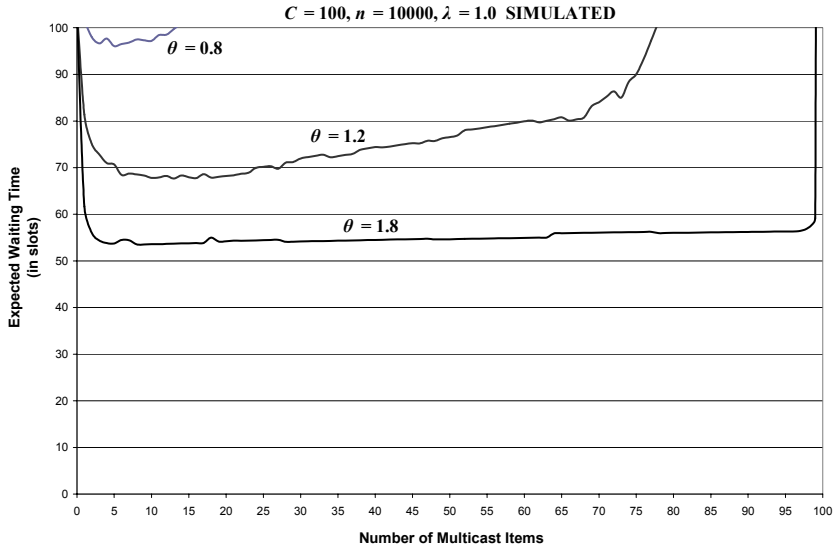

Figure 5.4(b) Simulated expected waiting time as a function of number of multicast items per cycle with λ=1.0 arrivals/slot and database size n = 10000 items for various Zipf distributions with parameter θ.

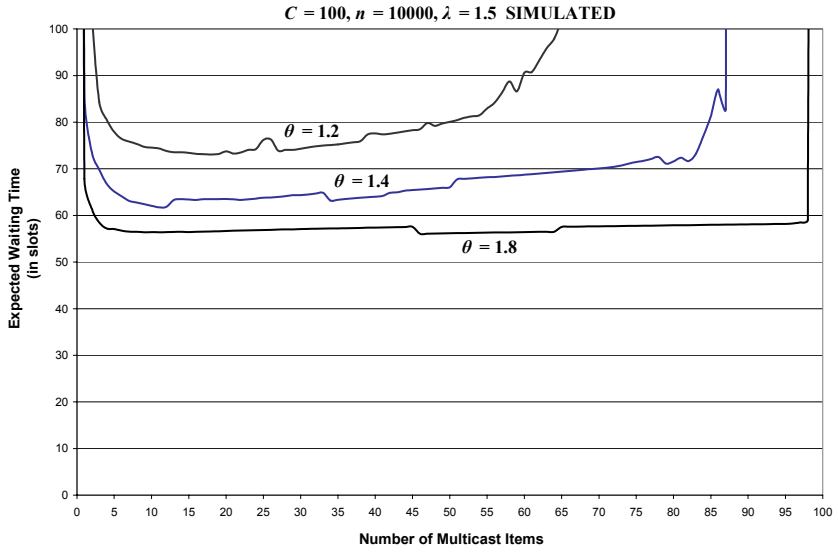

Figure 5.4(c) Simulated expected waiting time as a function of number of multicast items per cycle with λ=1.5 arrivals/slot and database size n = 10000 items for various Zipf distributions with parameter θ.

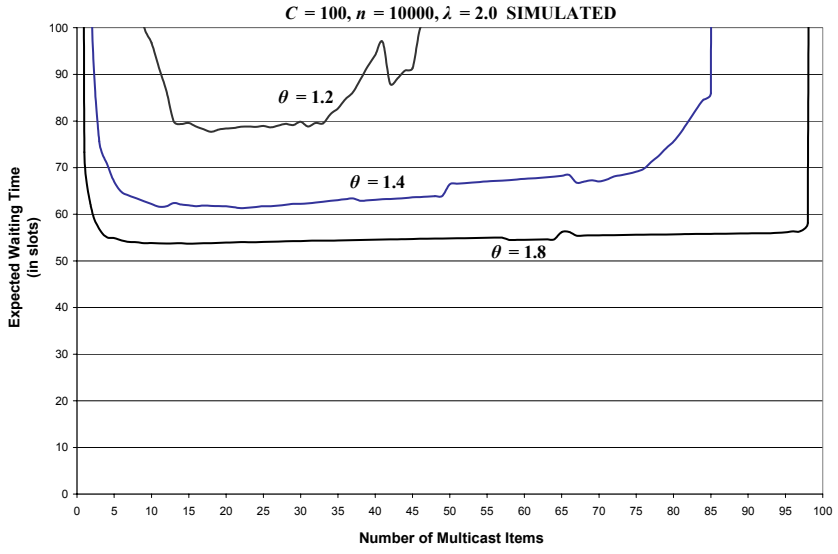

Figure 5.4(d) Simulated expected waiting time as a function of number of multicast items per cycle with λ=2.0 arrivals/slot and database size n = 10000 items for various Zipf distributions with parameter θ.

5.4.2 Probability of Loss

Recall that the probability of loss is the defined as the probability that a terminal will not receive its item within its time in the infostation range. In the previous simulations, there was no time limit set for reception for each terminal (i.e. each terminal was assumed to enter the infostation area and never leave). In this set of simulations, we set the time in the infostation area to 100 seconds to simulate a terminal traveling at a rate of 1 m/sec along the diameter of the circular infostation range. This traveling rate approximates a pedestrian walking at a brisk pace. Note that at this pace, a terminal will be in the coverage area for exactly one cycle, but this will force some terminals to leave without their items since a terminal must wait for an index first before receiving its item.

In Figures 5.5(a-d), simulations were performed for a system with a cycle length of C=100 seconds and a database size of N=100 items. Note that the cycle length is identical to the time in the infostation area. In figure 5.5(a), we can see that since the arrival rate is low, the probability of loss when M=0 is near 0. The reason for this is because the expected waiting time per terminal is around 75 seconds (from figure 5.2a). Therefore, we would expect roughly half of the terminals to get their items in less than 75 seconds, and the other half in more than 75 seconds. But since the arrival rate is relatively low, the number of terminals that wait more than 75 seconds is not large, and all terminals can receive their item in 100 seconds.

On the other hand, as the number of multicast items increases, terminals must wait for their multicast item in the designated slot, and this affects the probability of loss. When θ is low, the request distribution is more uniform, so terminals that want items later in the cycle will end up leaving the infostation area before they receive their items. On average, the expected waiting time for terminals for the index is 50 seconds, half of their travel time through the infostation area. Since the distribution is uniform, half of the terminals will want items 51-100 on average, which will not be transmitted until after the terminals leave the infostation area. This shows as a probability of loss near 50% (45% actually) for $\theta = 0$. However, as θ increases, most terminals will want items skewed toward item #1, which

makes it much more likely that these terminals will receive the item before departure, since on average the terminals will receive the index in 50 seconds and the desired item shortly thereafter. In this figure, this shows as a very low probability of loss, around 2%.

Figures 5.5(b-d) show that as the arrival rate increases, the probability of loss continues to behave in the same way for higher M. This is a result of the fact that the database can be transmitted in one cycle ($N = C$). When $M = C = N$, the system behavior is not dependent on the arrival rate so we get the same behavior in all four simulation runs.

As the number of multicast items approaches 0, the probability of loss is very dependent on the arrival rate. For $\lambda = 1$ and $M = 0$, our system is just becoming unstable. We would expect that in each cycle, approximately 50 terminals arrive before a new arrival (assuming the new terminal arrives in the middle of the cycle). Since the expected time until the next index is 50 seconds, and there are on average 50 terminals before a new arrival, a new arrival has a 50/50 chance to get its desired item in the allotted 100 seconds of travel time. In fact, this does not depend on the request distribution (governed by θ) since all requests are handled the same way. This is shown in the figure 5.5(b) as a probability of loss of approximately 55% for all θ.

As M increases, the request distribution is a deciding factor for the probability of loss. For lower θ, as M increases, the probability of loss increases since the request distribution is relatively uniform, but we are forcing terminals that want unicast items to wait until the multicast items are transmitted first. However, eventually M increases enough so that more of the terminals are being served by the multicast part of the transmission (which occurs first in the cycle) and the probability of loss decreases a small amount.

For higher θ, as M increases, the probability of loss decreases since those terminals that wanted items with an index near #1 are taken off the unicast service queue and served by the multicast transmission early in the cycle. This causes more terminals to complete before departure (even though the overall arrival rate is high) since many of these terminals want item #1. As M increases further toward C, the behavior of the system mimics that of a

lower arrival rate, since few terminals will want items multicast near the end of a cycle when θ is near 2.0.

Figures 5.6(a-d) show the probability of loss from simulations where the cycle length $C = 100$ seconds and the database size $N = 1000$ items. As in the previous simulation results, when $M = C$, we find that the probability of loss does not depend on the arrival rate of terminals. If $M = C$, the cycle is made of only the first M items of the database, and only those terminals that want those items have a chance of completion. The other terminals will always leave without their item since their item will never be transmitted by the infostation. For those terminals that want items #1-M, we have a similar scenario to the previous simulation runs. The probability of loss is higher here overall since the terminals that are never served are included in the overall probability of loss.

For low M, the arrival rate is a factor in the probability of loss. For $M = 0$, at low arrival rates, the system behaves like the previous simulated system. The infostation can handle all incoming requests since the arrival rate is low, and most terminals receive their items before departure. As M increases, the request distribution becomes a factor as well. An average terminal has 50 seconds to receive its item (since it waits 50 seconds for the index). When M reaches 50, the system is transmitting items 1-50 using multicast and all other requests are served afterwards in the cycle. But on average, any terminals that want items above item 50 will have left the terminal area when their item is transmitted, counting as a loss. Since on average the number of terminals that want items #1-50 in a 1000 item database is 5% when $\theta = 0$, the probability of loss reaches 95% when M reaches 50. After this point, addition of more multicast items does not help any terminals since these extra multicast items are transmitted after the average terminal has left the infostation area.

As θ increases, this "saturation point" occurs later (for $M > 50$) since more terminals want items closer to item #1. Therefore, of the terminals that want items above #50, there are fewer of these terminals when $M = 50$, so the probability of loss for all terminals is lower. Eventually, we reach the saturation point, where an increase of multicast

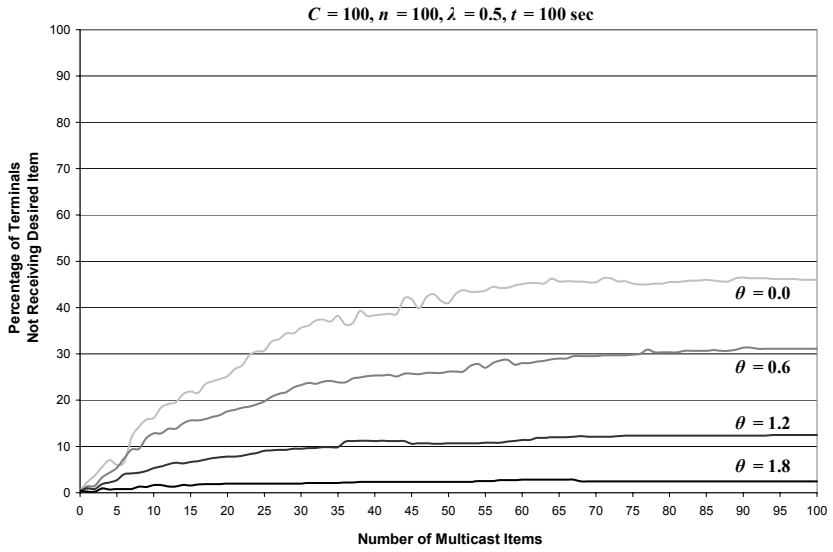

Figure 5.5(a): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 0.5 terminals/sec, D = 1 sec (i.e. ρ = 0.5), n = 100 items, t = 100 sec.

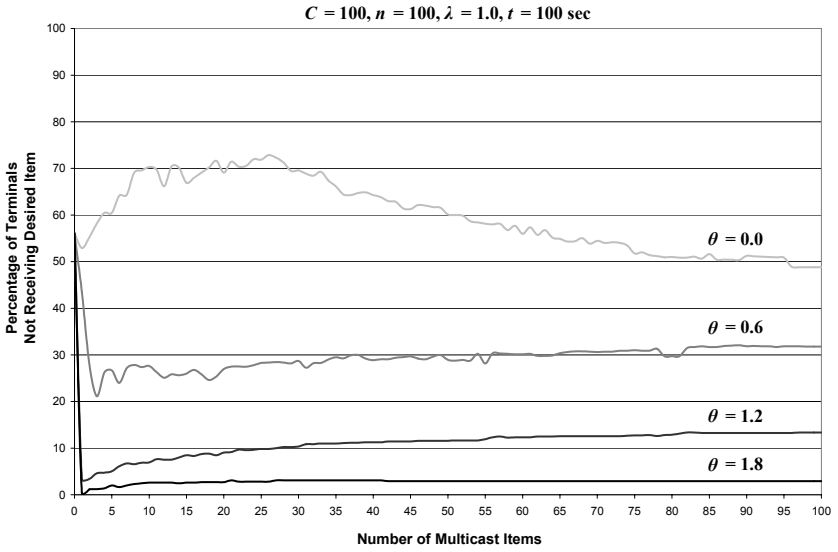

Figure 5.5(b): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 1.0 terminals/sec, D = 1 sec (i.e. ρ = 1.0), n = 100 items, t = 100 sec.

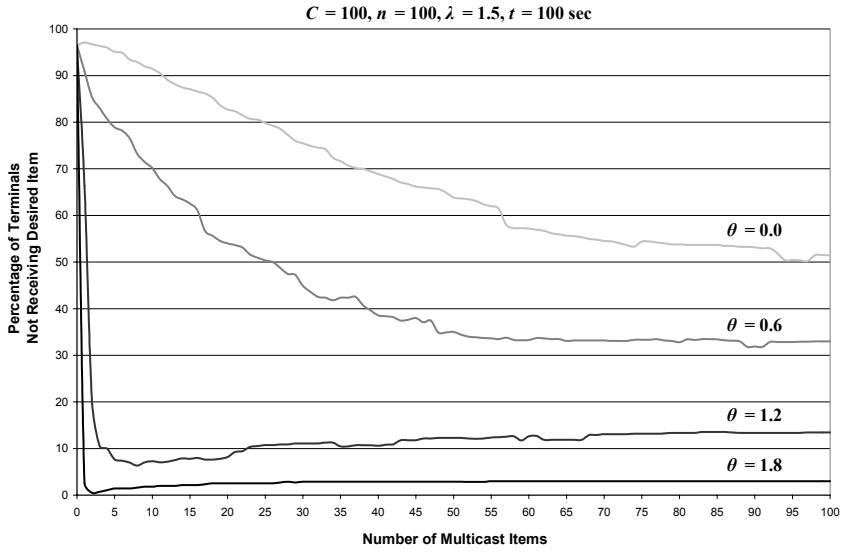

Figure 5.5(c): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 1.5 terminals/sec, D = 1 sec (i.e. ρ = 1.5), n = 100 items, t = 100 sec.

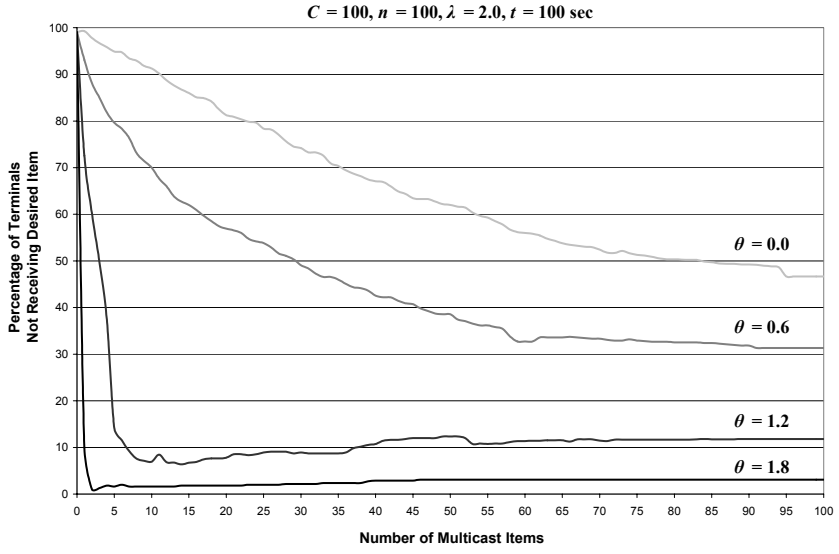

Figure 5.5(d): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 2.0 terminals/sec, D = 1 sec (i.e. ρ = 2.0), n = 100 items, t = 100 sec.

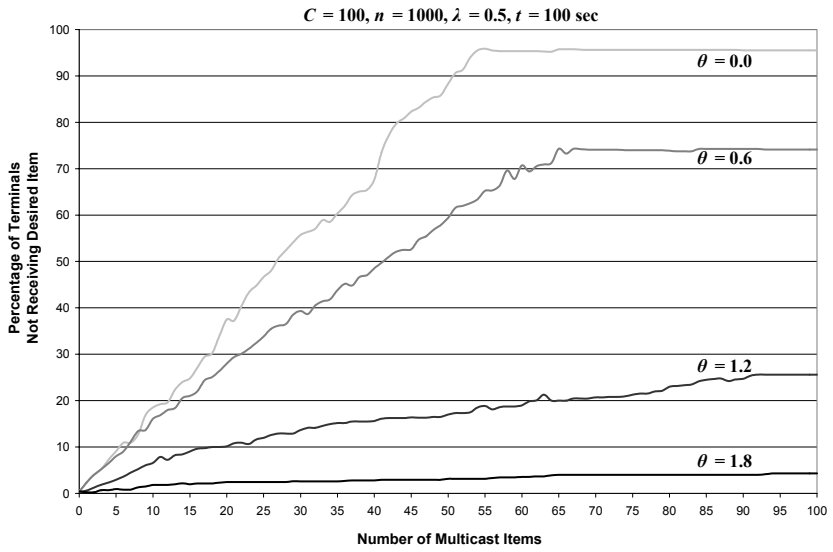

Figure 5.6(a): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 0.5 terminals/sec, D = 1 sec (i.e. ρ = 0.5), n = 1000 items, t = 100 sec.

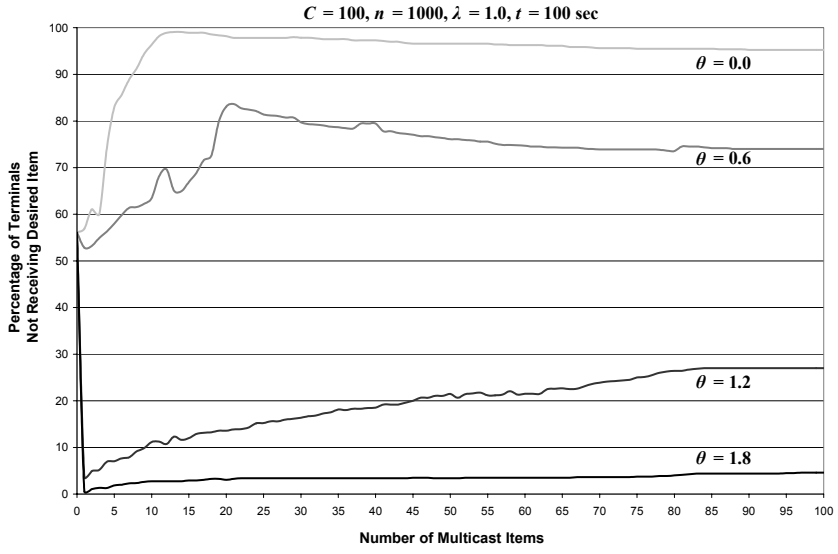

Figure 5.6(b): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 1.0 terminals/sec, D = 1 sec (i.e. ρ = 1.0), n = 1000 items, t = 100 sec.

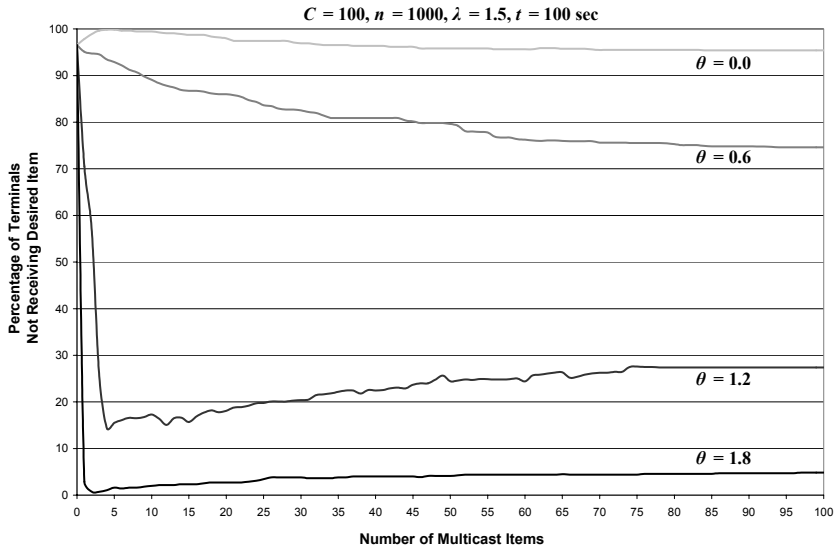

Figure 5.6(c): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 1.5 terminals/sec, D = 1 sec (i.e. ρ = 1.5), n = 1000 items, t = 100 sec.

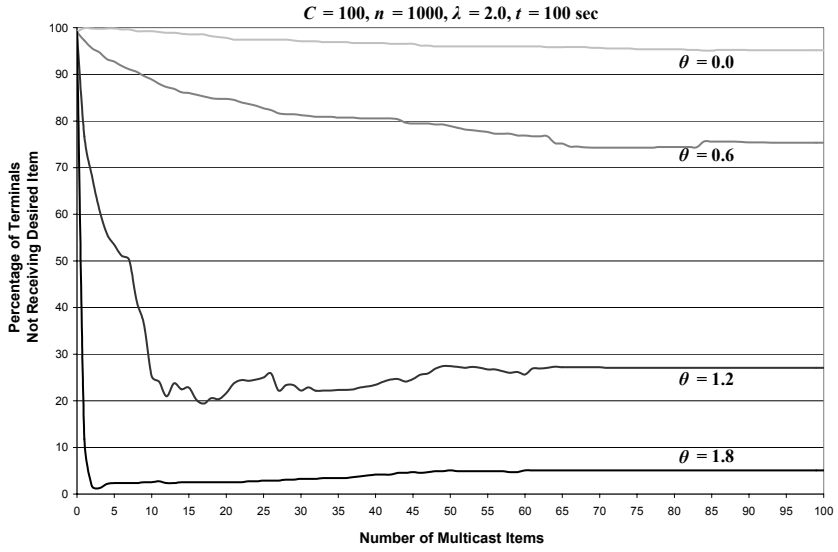

Figure 5.6(d): Simulated percentage of terminals that do not receive their desired items (i.e. probability of loss) as a function of number of multicast items for λ = 2.0 terminals/sec, D = 1 sec (i.e. ρ = 2.0), n = 1000 items, t = 100 sec.

items does not yield any change in the probability of loss, as we did with $\theta = 0$, but this point occurs for higher M since fewer terminals are leaving without their items after 100 seconds.

5.5 Modified Infostation Model

The main disadvantage of the original infostation model simulated in the previous section is that several terminals may want the same item, but if that item is unicast, each terminal must be served separately. It would make sense to serve these terminals together with a multicast transmission to reduce the expected waiting time.

The infostation model simulated in this section uses multicast to transmit all items. The items are split into two groups: M_p popular items that are scheduled for transmission once every cycle and M_o on-demand items that are scheduled for transmission only upon request by at least one terminal. The popular items correspond to the original multicast items from the original system. The on-demand items correspond to the original unicast items that are requested by terminals, with the exception that these items are now multicast as well. Note that $M_p + M_o = N$ and $M_p \leq C$.

Another advantage of this new model is that an arriving terminal that requires an on-demand item may find it already scheduled in the next cycle since another terminal may have made a request for this item. Therefore, the number of requests to the server can be reduced by eliminating redundant requests for the same item.

Figures 5.7(a-d) show the results of a simulation of the modified infostation model where all items are transmitted using multicast, as described above. In this set of simulations, the database size is equal to the cycle length, so each item can be transmitted once per cycle if necessary. Comparing figure 5.7(a) with the corresponding simulation of the original system in figure 5.5(a), we see that the request distribution does make a

difference in the expected waiting time when all items are requested ($M = 0$ in the original model, and $M_p = 0$ in the modified model). When $\theta = 0.0$, both models are equivalent, but when θ increases, this new model can handle the multiple requests for the same on-demand item using multicast, reducing the expected waiting time. When $\theta = 2.0$, the expected waiting time is only slightly higher than the expected time each terminal waits for the next index.

Examing figure 5.7(a) further, we can see that although there are minimum points on these curves for θ near 1.0, this minimum is not much lower than other points on the same curve. These curves suggest that for the modified model, we should mark as few items as possible as popular, since transmitting these once every cycle by default does not improve the expected waiting time. In fact, for low θ, designating many popular items makes the situation worse, since at this low arrival rate of 0.5 terminal/slot, a terminal that wants a high-numbered item which is designated as popular must wait until the appropriate slot later in the cycle to receive the item, causing it to wait longer than necessary.

Figures 5.7(b-d) show the modified system as the arrival rate increases. It is interesting to note that in all of these cases, the system behaves in a similar manner to a system with a low arrival rate. The curves show that there are optimal points where we can minimize the expected waiting time for a system with high arrival rates (greater than 1 arriving terminal per slot), but the improvement in the expected waiting time by multicasting a number of popular items every cycle is very small compared to a system where items are only transmitted if a request is received. Of course, if we consider the number of requests arriving at the server for these items, then for higher θ we can reduce the number of requests to the server and achieve a slightly lower expected waiting time by transmitting the most desired items in the database once per cycle (as popular items rather than on-demand items), eliminating all of the potential requests for the same items.

Figures 5.8(a-d) show the modified system when the database size becomes significantly larger than the cycle length (C=100, N=1000). In these cases, it is clear that

setting the number of popular items in the cycle to its maximum (100) will cause the expected waiting time to go to infinity just as in the original model when the first 100 items of the database were multicast. The simulations show that again there are optimal points where we can improve the expected waiting time overall, mainly when θ is close to 1.0, but these improvements are small as in the previous simulations.

Based on these simulations, multicasting the first 10-15 items of the database regularly every cycle and multicasting all others by request only yields a minimal expected waiting time for the terminals overall, irregardless of the arrival rate of the terminals. However, as θ approaches 0.0 (yielding a more uniform request distribution), we find that the expected waiting time becomes quite large as the arrival rate increases, no matter how many items are denoted as popular. This is true because as more terminals arrive, requesting different items uniformly, the size of the request queue will eventually exceed the cycle length as the arrival rate increases. For example, for an arrival rate of 2.0 terminals/slot and $\theta = 0.0$, the number of new requests on the request queue during one cycle can approach 200. Note that the size of the request queue is bounded by $N = 1000$, since all requests for items that are already on the request queue are discarded. Therefore, the maximum delay can eventually reach 10 cycles = 1000 seconds.

In summary, this modified infostation model is beneficial when the request distribution is skewed. It reduces the expected waiting time and the number of requests that the server receives from terminals. One potential problem with this new model is that all terminals must receive their items using multicast. This requires each terminal to subscribe to an appropriate multicast address which may take a significant amount of time. Additionally, multicast transmission in our system will utilize UDP rather than TCP and is subject to a much higher rate of data transmission errors. An analysis of the original infostation model with packet errors during transmission is a topic examined in the next section.

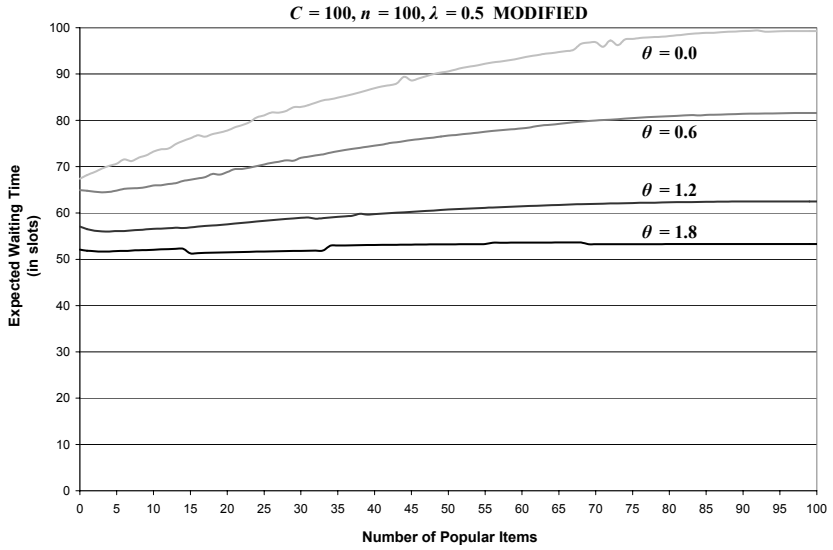

Figure 5.7(a) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=0.5 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

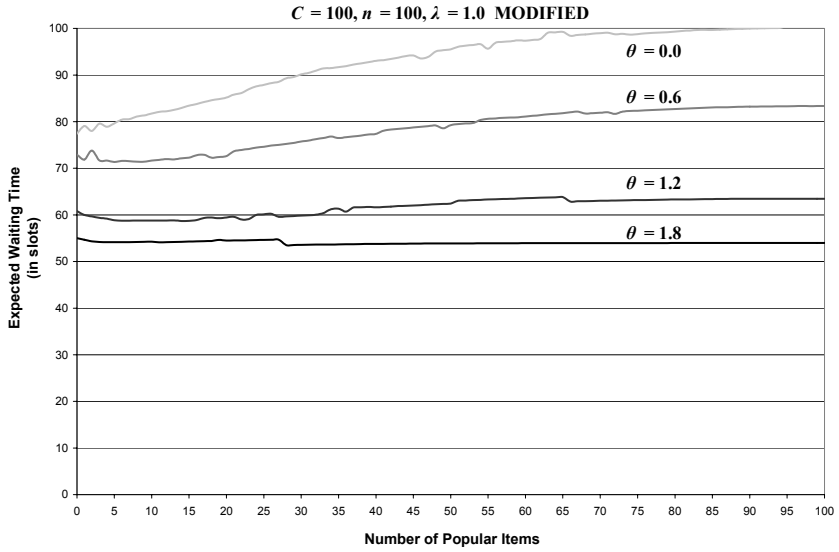

Figure 5.7(b) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=1.0 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

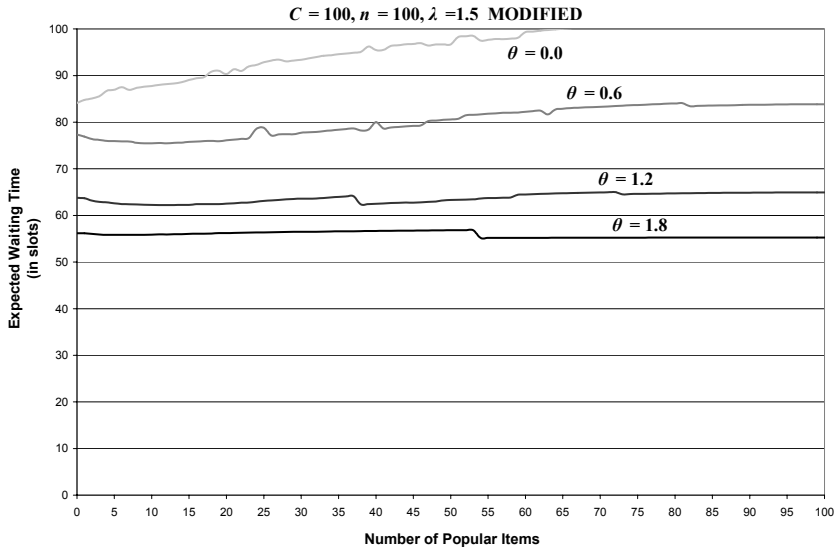

Figure 5.7(c) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=1.5 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

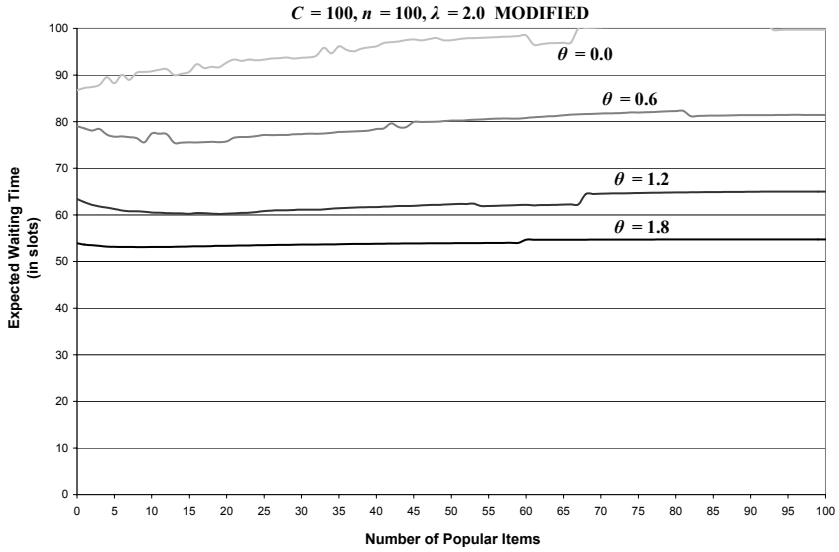

Figure 5.7(d) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=2.0 arrivals/slot and database size n = 100 items for various Zipf distributions with parameter θ.

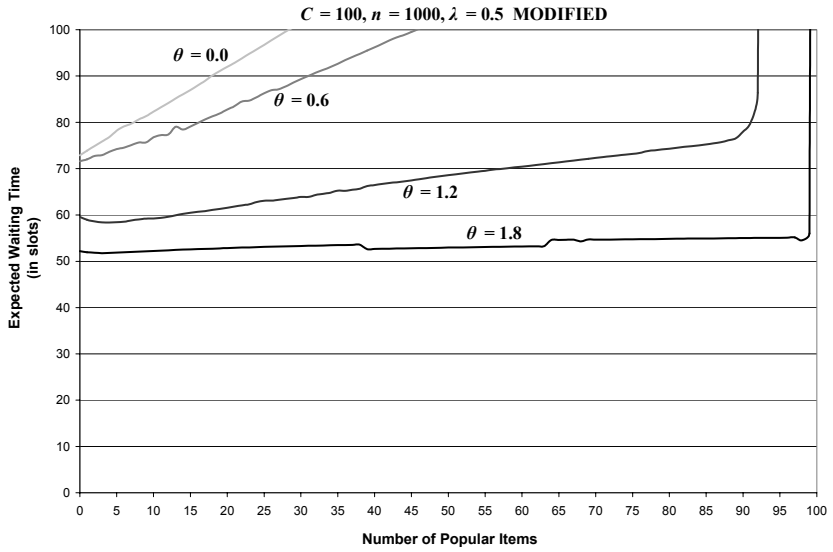

Figure 5.8(a) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=0.5 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

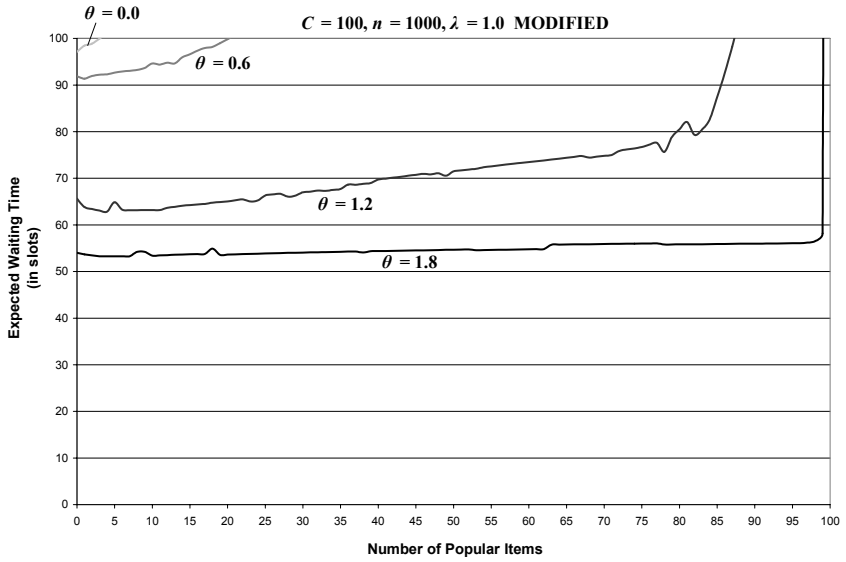

Figure 5.8(b) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=1.0 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

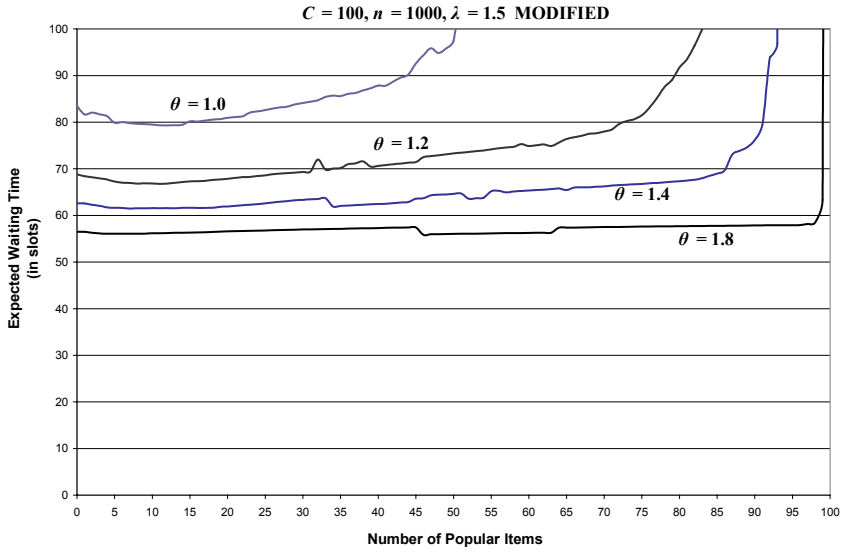

Figure 5.8(c) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=1.5 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

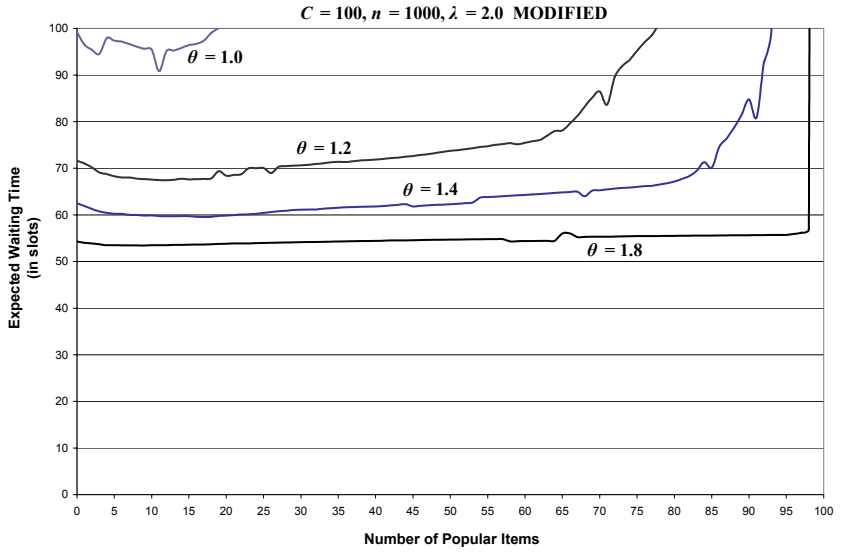

Figure 5.8(d) Simulated expected waiting time as a function of number of popular items per cycle for the modified infostation model with λ=2.0 arrivals/slot and database size n = 1000 items for various Zipf distributions with parameter θ.

5.6 Expected Waiting Time with Packet Errors

In reality, a terminal using an infostation system will not receive its data item in its entirety the first time the data item is transmitted from the infostation. In a wireless environment, packet errors are common and will affect the expected waiting time for a terminal since the terminal will have to wait through additional cycles to receive all of the packets that the terminal did not receive during the first transmission. In this section, we assume that the probability of packet error is primarily a function of the distance of the terminal from the infostation at the time of the packet transmission. For this analysis, we do not consider the additional effects of small-scale fading, shadowing and noise on the packet.

In this section, several mobility models for the terminals are defined. For each of these models, we determine the distance of the terminal from the infostation at any given time. From this information, we can use a packet error model for IEEE 802.11b to determine the approximate probability of error for a packet transmitted at that time.

The infostation simulator implements three mobility models for the terminals, as shown in Figure 5.9. These models are described below.

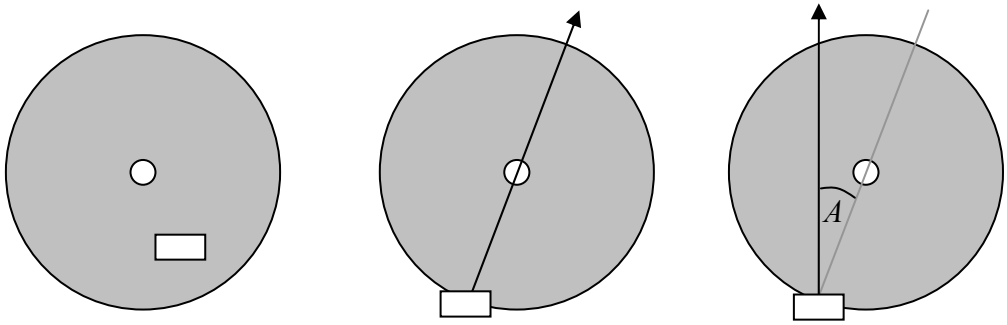

Figure 5.9 Mobility models for terminals in an infostation environment.

0D Model – Terminals are stationary. We assume in this model that a terminal "arrives" when the terminal requires an item from the infostation. It is at this point that the terminal begins to monitor the wireless channel for the next index. The terminal does not monitor the channel before its "arrival" to conserve power. Upon its "arrival", the terminal will be at some fixed distance from the infostation. The distance of the terminal from the infostation will determine the probability of packet error for the terminal.

1D Model – Terminals move at constant velocity along the diameter of the infostation coverage area. The path of travel for the terminal is a straight line. When the terminal first enters the infostation coverage area, the probability of packet error is quite high but this probability decreases until the terminal is directly next to the infostation, where the probability of packet error is negligible.

The probability of packet error for the terminal at any given time in the coverage area depends on the distance of the terminal from the infostation at the center of the coverage area. Since the terminal travels along the diameter of the infostation coverage area, the distance d from the infostation is simply

$$d = |r - vt| \tag{5.1}$$

where

r = radius of the infostation coverage area (in meters)
v = velocity of the terminal (in meters/sec)
t = time within the infostation from the time of arrival (sec)

2D Model – In this model, the terminal still travels in a straight line at a constant velocity, but the direction of travel does not necessarily proceed through the center of the infostation coverage area. A terminal may enter at any point on the circumference of the coverage circle and create a chord across the interior to another point on the circumference. Without loss of generality, we pick one point along the circumference as the entry point for all

terminals and assign a direction of travel defined by an angle A ($0 \le A \le 90$) with a path that traverses the diameter of the circle. The angle of deflection is chosen randomly using a uniform distribution.

The probability of packet error for the terminal at any given time in the coverage area depends on the distance of the terminal from the infostation at the center of the coverage area. This distance d (in meters) is derived from the Law of Cosines and is given by the equation:

$$d = \sqrt{r^2 - 2vtr\cos(A) + (vt)^2} \tag{5.2}$$

where

r = radius of the infostation coverage area (in meters)

v = velocity of the terminal (in meters/sec)

t = time within the infostation from the time of arrival (sec)

Note that if angle A is 0 degrees, the terminal behaves as in the 1D model, since

$$d = \sqrt{r^2 + (vt)^2 - 2vtr\cos 0} = \sqrt{r^2 - 2vtr + (vt)^2} = \sqrt{(r - vt)^2} = |r - vt|.$$

For each of these mobility models, we use a derivation from [8] to calculate the probability of packet error.[5] For a wireless transmitter using 802.11b in a building environment operating at 11Mbps, we can express the probability of packet error as

$$p_e = 1 - (1 - 10^{-0.1*(-20-35\log d)-10.8})^b \tag{5.3}$$

where

d = the distance of the terminal from the infostation (in meters)

b = the number of bits per packet.

[5] The complete details of this derivation are provided in Appendix A.

Figure 5.10 illustrates the probability of packet error p_e as a function of d for various typical packet sizes (in bits). As the packet size drops, the probability of packet error decreases at larger distances from the infostation. This would imply that to minimize the effects of packet error on the system, the designer should reduce the size of each packet, effectively to its minimum allowable size. However, decreasing the number of bits per packet will increase the number of packets per data item. For a fixed number of bits that are transmitted wirelessly, random losses will cause more requests if the packet sizes are small, since more packets will be affected by these losses. This can generate more work for the server if terminals begin requesting packets for retransmission, since there will be more requests per item. Additionally, the system will experience a decrease in overall throughput since the transmission of each packet has some overhead due to packet headers. As the packet size decreases, the amount of data bits transmitted with respect to the overall transmission (data and headers) will also decrease.

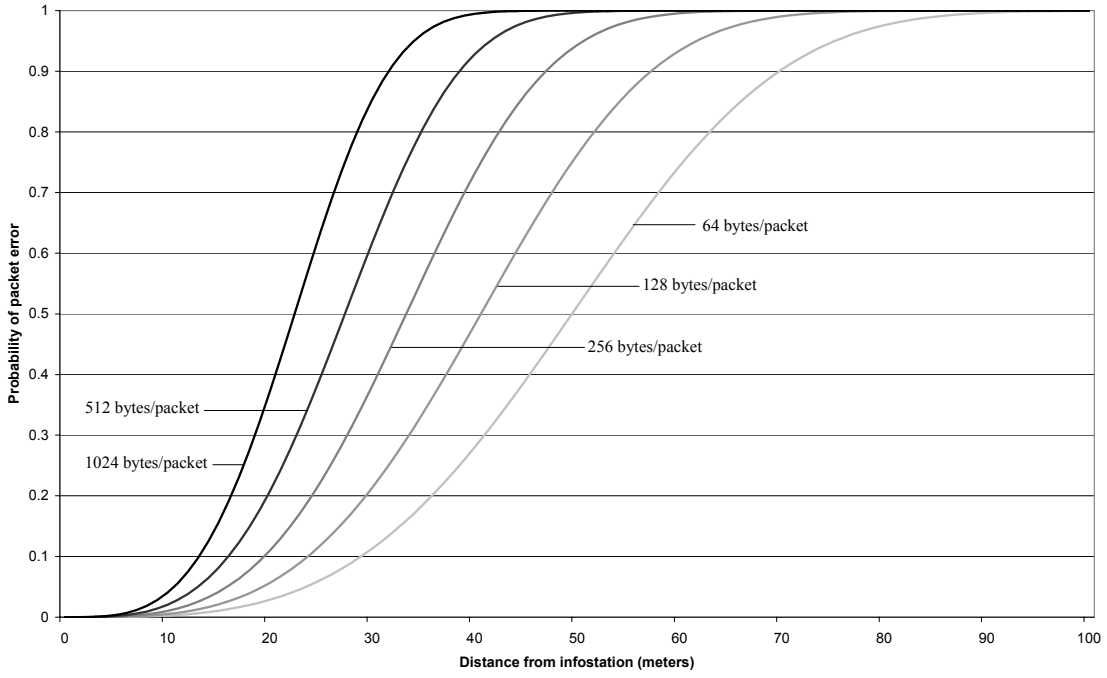

Figure 5.10 Probability of packet error as a function of distance for IEEE 802.11b in an office environment.

The simulator was used to analyze an infostation system under all three mobility models with the following parameters:

b = 4096 bits/packet (= 512 bytes/packet)
r = 50 meters
v = 1 meter/sec
θ = 1.0
C = 100 items/cycle
N = 100 items
D = 1/11 sec (= 256 packets/item)

The smaller item size is due to the fact the original cycle with C=100, N=100, and D=1 would give terminals a maximum of one cycle in the infostation area if they are mobile. Since there will likely be packet errors, this would make it nearly impossible for a terminal to receive its entire item successfully except in special cases (e.g. a mobile terminal receives its item at its closest distance to the infostation). By reducing the size of each item, we allow a terminal to have around 11 cycles maximum to receive its item. For higher packet error rates, this still gives a terminal additional chances to request and receive packets dropped in the initial cycles after its arrival.

In addition, we assume that index packets and request packets are not dropped in this simulation. Since these packets are usually quite small, we can envision a system where these packets are transmitted multiple times in order to compensate for dropped packets or forward error correction techniques can be used to add redundancy to the packets as they are transferred.

First, we consider a scenario where the arrival rate is relatively low to examine the effect of packet errors on the expected waiting time. By setting λ = 6 arrivals/sec, we have a system with $\rho = \lambda/\mu = \lambda D = 0.545$. Figures 5.11(a-b) show the results of the simulation runs for this arrival rate and the other parameters described above. Figure 5.11(a) shows the expected waiting time for each terminal in seconds (rather than slots) for those terminals

that successfully receive their items. Figure 5.11(b) shows what percentage of the terminals successfully receive their items before departure. (In figure 5.11(b), the 0D model is not shown since these terminals never depart and always successfully receive their items, although the waiting time may be extremely high.)

For the 0D model, we can see that as the distance increases from the infostation when the terminal wants a data item, the expected waiting time increases as expected. An interesting note, however, is the behavior of the system when there are few multicast items in the broadcast. In this case, the expected waiting time increases to very large values. This is due to the fact that a terminal receives its item with packet errors, it must make a request for the item again, causing it to join the request queue again along with the additional arriving terminals. Therefore, the queue holding the terminal requests begins to grow larger as each cycle passes, and the resulting expected waiting time will increase. As the number of multicast items increases, more of these terminals will not be placed in the queue since they are requesting multicast items now. These terminals will see their items once per cycle and will have more opportunities to receive the missing packets they need, reducing the expected waiting time overall.

For the 1D and 2D models, the probability of packet error changes as the terminals move through the infostation range. In the 1D model, the packet error rate lowers to nearly 0 as the terminal passes under the infostation itself. The expected waiting time will not be as sensitive to the number of multicast items in the cycle as it was in the 0D model, because a terminal will eventually have a very low packet error rate and will be able to receive most if not all of its missing packets at this point and will not make additional requests However, although the expected waiting time in these models is reasonable when few multicast items are broadcast, we can see in figure 5.11(b) that the number of terminals that are successful is also very low in this case. This suggests that even though the packet error rate is low at some point in the terminal's path, the queue of terminal requests grows in an unbounded manner as it did with the 0D model. For low arrival rates, we see that by transmitting more

items using multicast, we can reduce the expected waiting time and increase the percentage of terminals that successfully receive their entire items.

Next, we consider a scenario with a higher arrival rate where more than one terminal arrives during the broadcast of one item. By setting $\lambda = 12$ arrivals/sec, we have a system with $\rho = \lambda/\mu = \lambda D = 1.09$. Figures 5.12(a-b) show the results of the simulation runs for this arrival rate and the other parameters described above. Figure 5.12(a) show the expected waiting time for each terminal in seconds for those terminals that successfully receive their items. Figure 5.12(b) shows what percentage of the terminals successfully receive their items before departure. (Again, since terminals do not leave in the 0D model, figure 5.12(b) does not show this model.)

When $\rho > 1$ as in this simulation, we see that in the 0D model, terminals must be very close to the infostation in order to receive their items successfully. (The curve for a distance of 20m that appears in figure 5.11(a) does not appear in figure 5.12(a) since the expected waiting times are much higher.) When no items are multicast, the expected waiting time is quite large as the terminal is placed further away from the infostation. But as the number of multicast items per cycle increases, the expected waiting time decreases to a constant value since many terminals to do not have to wait on a request queue for their item.

Also, as the number of multicast items per cycle increases in the 1D and 2D models, the number of terminals that successfully receive their items also increases, but not as quickly as it did for a lower arrival rate. By multicasting more items per cycle, there will be a point in the terminal's path through the infostation when its packet error rate will be very low, and the missing packets of the item will be received.

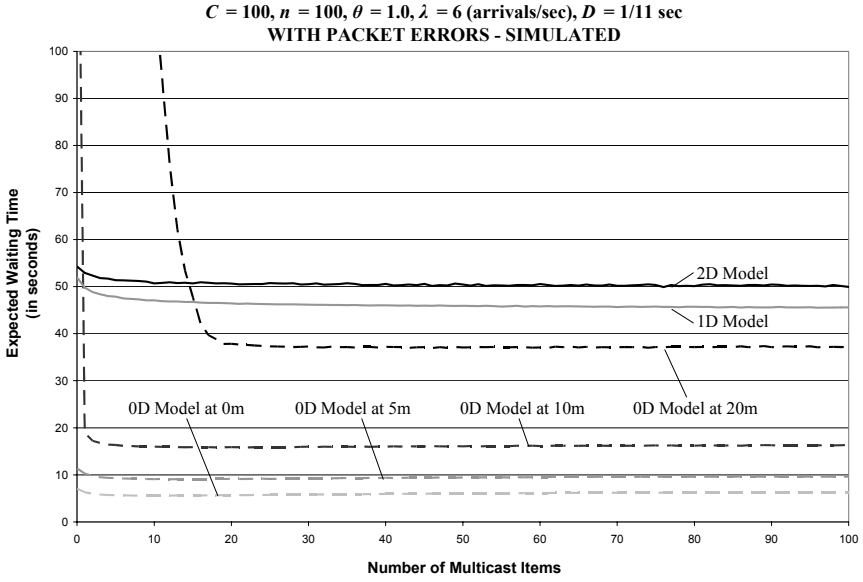

Figure 5.11(a) Simulated expected waiting time as a function of number of multicast items per cycle for the original infostation model of Chapter 4 with λ=6 arrivals/sec, database size n = 100 items, a Zipf request distribution with θ = 1.0, and a service time of D = 1/11 sec for the various mobility models.

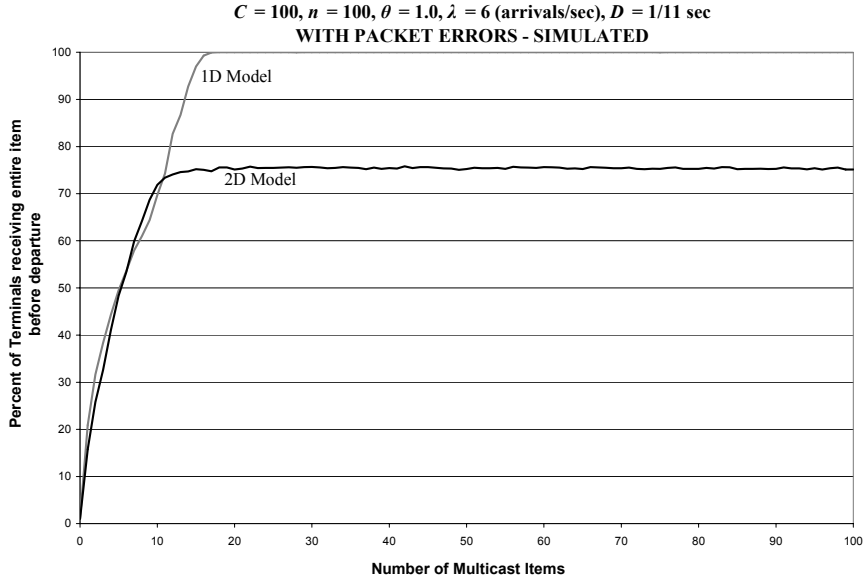

Figure 5.11(b) Percentage of terminals successfully receiving their item as a function of number of multicast items per cycle for the original infostation model with λ=6 arrivals/sec, database size n = 100 items, a Zipf request distribution with θ = 1.0, and a service time of D = 1/11 sec for various packet error rates.

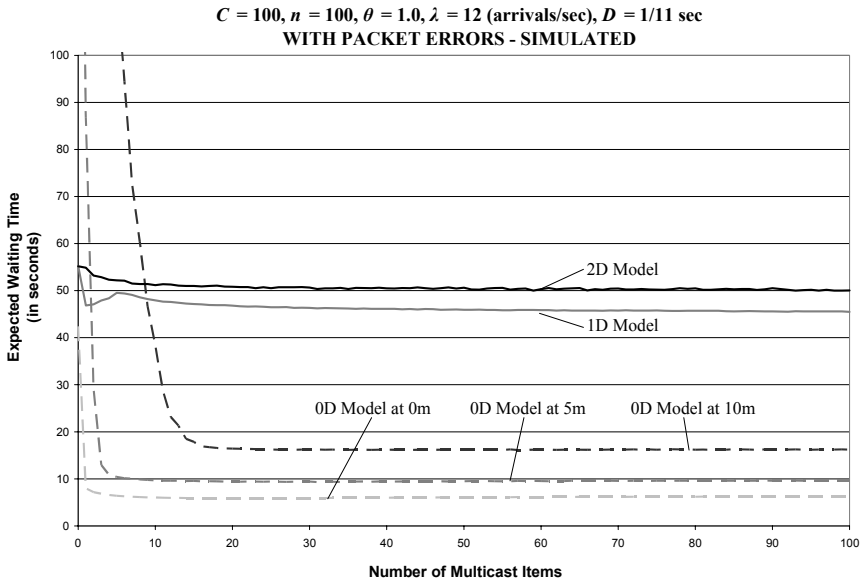

Figure 5.12(a) Simulated expected waiting time as a function of number of multicast items per cycle for the original infostation model of Chapter 4 with λ=12 arrivals/sec, database size n = 100 items, a Zipf request distribution with θ = 1.0, and a service time of D = 1/11 sec for the various mobility models.

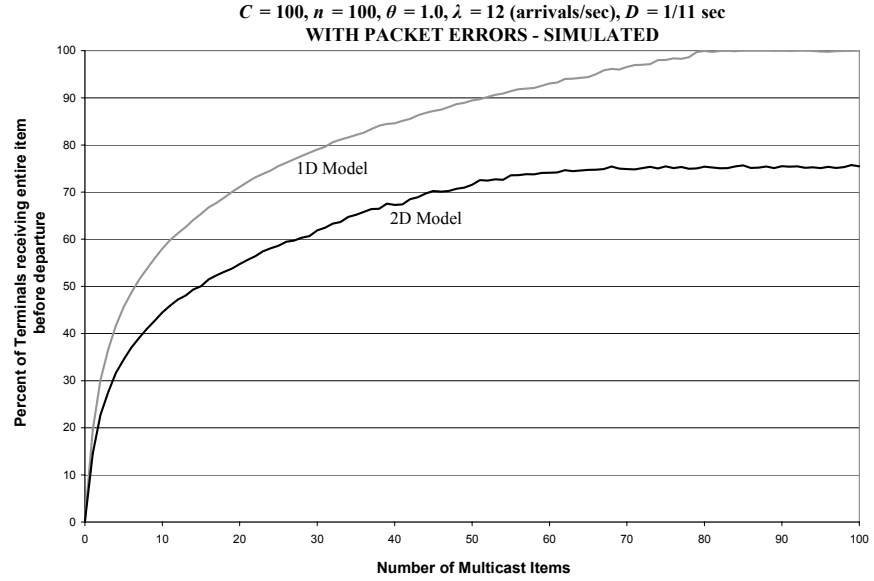

Figure 5.12(b) Percentage of terminals successfully receiving their item as a function of number of multicast items per cycle for the original infostation model with λ=12 arrivals/sec, database size n = 100 items, a Zipf request distribution with θ = 1.0, and a service time of D = 1/11 sec for various packet error rates.

5.7 Summary

In this chapter, we have presented the details of a simulator that was designed to measure the expected waiting time and probability of loss for an infostation environment modeled after the design described in Chapter 4. In addition to this model, the simulator can also handle other infostation models such as the pure-multicast model presented in section 5.5 and a model that includes packet errors as described in section 5.6. Simulation results are presented for all of these models in this chapter.

The primary results for the single-channel infostation system has verified the mathematical analysis presented in Chapter 5. Simulation results from the modified infostation system show that the expected waiting time can be reduced for skewed request distributions when all items are unicast. On the other hand, when each cycle contains multicast items, the performance of the system is very similar to the original single-channel infostation system described in chapter 5. A simple packet error model is presented in this chapter and the results show that packet errors increase the expected waiting time but this depends on the mobility model that the terminal follows.

The simulator used to present the results of this chapter can be adapted to handle more sophisticated error modeling or alternate infostation models easily due to its modular design.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Major Results

In this dissertation, a study of the use of multicast and unicast transmission in an infostation environment is presented. The infostation environment is a localized, high-speed, wireless data delivery system for mobile terminals. Unlike other research in broadcast data algorithms, this environment is unique in that deadlines are imposed on the delivery of data since terminals will leave the broadcast area, and the arrival of terminals in the broadcast area and the actual items that they want is unknown ahead of time. Multicast transmission is a useful way to transmit data to terminals to satisfy as many pending requests for a specific item in one transmission. However, we show that multicasting all items of the database is generally not beneficial for the mobile terminals.

We show that if we transmit data items using a data carousel approach with cycles of fixed length, the number of multicast items to transmit from the infostation is a function of the expected arrival rate of the terminals, the expected distribution of requests from the terminals, and the size of the database. We defined an infostation model that was analyzed mathematically using queuing theory in order to derive expressions for the expected waiting time for an arbitrary terminal in the infostation system. This model, which closely matches a real system being developed at Polytechnic University, was then simulated using the CSIM simulation library to compare its expected behavior with the mathematical results. These results matched very well, showing that when the database can be transmitted in one complete cycle, low arrival rates suggest that the infostation should unicast all data items no matter what the request distribution is, but as the arrival rate increases and overloads the infostation, multicast becomes effective for request distributions that are skewed toward a small subset of the items of the database. For much larger databases, a similar scenario exists. Transmitting all items using unicast causes the expected waiting time to increase without bound when the arrival rate is greater than 1

arrival per data item transmitted, independent of the distribution of the requests. Using multicast for an entire cycle also causes the expected waiting time to increase without bound since the cycle cannot accommodate all items from the database in this case. Our results show that multicasting the most popular 1-1.5% of the database results in a minimization of the expected waiting time.

The infostation simulator is designed in such a way that other infostation models may be studied that would be difficult to analyze mathematically. Two such models are presented in this work, one in which all items are sent using multicast, and another where packet errors are introduced into the data transmission using a model of IEEE 802.11b packet error loss derived in a separate work.

In such infostation environments, a designer could measure the arrival rate of terminals using the feedback segment of each cycle and tabulate all items that are successfully received by the terminals. Based on these results, the designer could organize the database in order of item popularity and determine which Zipf distribution fits these items the best. (Other distributions could be used as well, although many researchers have indicated that database requests such as those we assume in this model follow a Zipf distribution.) The infostation scheduler can then be set to broadcast using the current arrival rate, request distribution and database size. Automated tools can be used to constantly monitor the feedback from the terminals to adjust the order of the items in the database or the request distribution to find the optimal number of data items to multicast per cycle to keep the expected waiting time for mobile terminals at a minimum.

6.2 Areas for Future Study

There are a number of extensions to this research that will provide more insight for the infostation designer. The simulator described in this work can be used to analyze each of these modifications to the infostation model that would be difficult to analyze mathematically.

- Varying cycle lengths

  In the infostation model analyzed in this dissertation, the cycle length was fixed in order to facilitate the mathematical analysis of Chapter 4. However, as a result, there will be cycles that may possibly have empty data slots. Although this helps the terminals conserve power, since index packets are always transmitted at regular time intervals, the limited amount of time a terminal spends in the infostation area may require the system to condense all cycles that have empty slots so that the transmission channel is utilized to its fullest. An analysis of cycles that vary in length would yield useful information on how this scenario improves the expected waiting time and how it affects the power used by the terminals.

- Varying file sizes

  This work uses file sizes of equal size in order to make use of the M/D/1 queuing model to analyze the expected waiting time and probability of loss for the infostation. In many cases, it will be unlikely that all files are of equal size. Therefore, future research will look at varying the file sizes based on common size distributions found in databases for internet or commerce applications. The simulator described in this dissertation can easily derive the expected waiting time for this scenario, but care must be exercised to study the relationship between the item's size and the item's popularity in order to determine the most effective use of multicast for this environment.

- Indexing schemes

  Chapter 3 contains a brief summary of several clever techniques that are used to get an index to the terminal as quickly as possible. In our infostation study, we have used a very simple indexing scheme, transmitting one index per cycle. Terminals that miss the index must wait for the next index, even if their desired

items are being transmitted during the current cycle. A simulation study of alternate indexing schemes can lead to additional insight on the use of multicast for the infostation environment.

- Forward error correction

  Our model assumes that in order for a terminal to leave the infostation area with its item successfully, it must receive all packets of the item. Forward error correction techniques add redundancy to a data item so that a terminal would only need to receive a fraction of the total number of packets in order to reconstruct the original data item. (Note that forward error correction would also increase the overall size of the data item since it adds redundancy to the data item.) The simulator of this study can be easily adapted to handle forward error correction schemes by increasing simulated file sizes and requiring that a terminal only receive a certain percentage of the item's packets to be considered successful. This future research can also yield insight into the effectiveness of multicast transmission in such a wireless environment.

- Additional Packet Error Models

  Almost all research presented to this date on broadcast scheduling for wireless systems does not include error modeling. This is a very important area of future research for infostations since packet errors are a significant problem in wireless networks. Both mathematical analysis and simulation needs to be completed for current and developing wireless protocols to examine the effect of packet errors on the use of multicast in broadcast scheduling.

- Power usage studies

  It is clear that wireless devices will continue to suffer from limited power sources. Additional research needs to be completed on scheduling using

multicast that will minimize the power usage for terminals with limited battery power. Simulation can be done to measure the number of uplink requests and downlink data transfers which can be mapped to power usage based on currently available wireless expansion cards and batteries. This information can give us some insight on the best way to use multicast or broadcast transmission to service terminals using the minimum of power possible.

All of the research areas presented in the section are vital in improving our understanding of the use of multicast and unicast transmission in a high-speed wireless environment such as an infostation in order to provide the best service to mobile terminals. The research presented in this dissertation forms the groundwork for additional simulations and mathematical studies addressing these issues for the future of infostation development.

APPENDIX A

PROBABILITY OF PACKET ERRORS IN AN IEEE 802.11b ENVIRONMENT

In chapter 5, a simulation was described that included packet errors in the transmission of data items in order to analyze how packet errors degrade the infostation performance. In [8], Fainberg presents a study that includes a model for packet errors in an IEEE 802.11b office environment. This appendix describes the fundamental points presented in the thesis that lead to the packet error model used in Chapter 5.

Path loss is a measure of the loss of power in a signal as a function of distance from the transmitter. In his thesis, Fainberg assumes that the power of the transmitter is 100mW or 20dBm. The path loss *PL* (in dBm) as a function of the distance *d* (in meters) from the transmitter is then defined by

$$PL = 40 + 10(3.5)\log_{10}(d) \tag{A.1}$$

40dB is the power lost in the first meter from the transmitter, and the remainder of the formula represents the power loss beyond 1 meter where 3.5 is the path loss coefficient for an indoor environment. (This coefficient can vary higher or lower depending on the environmental setting.)

The power received *PR* (in dBm) at the terminal ignoring other effects like shadowing, small scale fading and noise, is given by

$$PR = 20 - PL \tag{A.2}$$

A plot of the power received at the terminal as a function of distance is shown in Figure A.1.

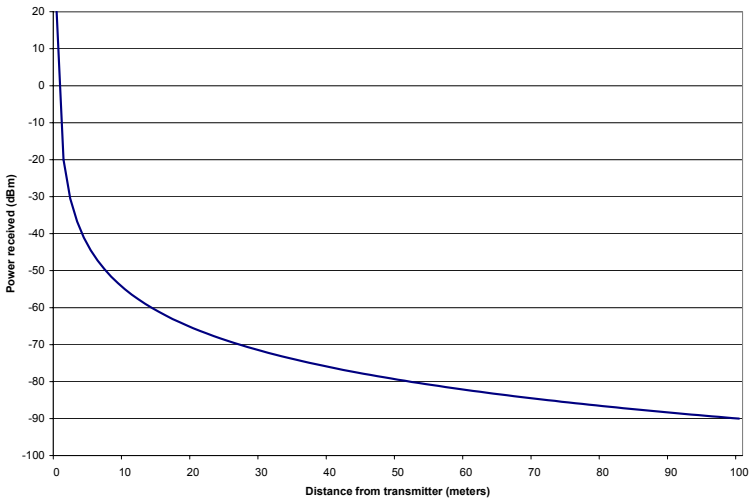

Figure A.1 Power received at the terminal as a function of distance from the transmitter.

Fainberg continues by deriving the probability of bit error in the Rayleigh multipath-fading channel for the Complementary Code Key (CCK) modulation at 11Mbps. Figure A.2 shows a graph of the probability of bit error as a function of the power received by the terminal.

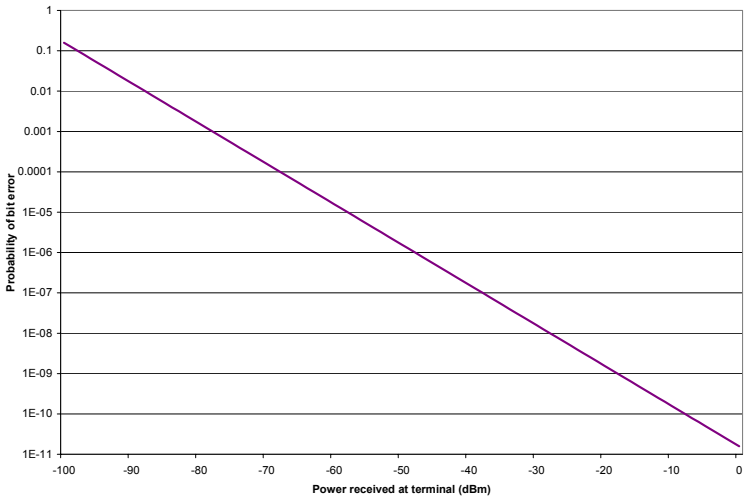

Figure A.2 Probability of bit error as a function of the power received at the terminal.

An equation that approximates this function is:

$$BER = 10^{-0.1(PR)-10.8} = 10^{-0.1(-20-35\log_{10} d)-10.8} \quad \text{(A.3)}$$

Using the assumption that each bit in a transmitted packet is independent of every other bit in the packet, we can derive the packet error rate *PER* as a function of the bit error rate and the size of the packet *b* in bits. The probability that a single bit is transmitted correctly is (1 – *BER*). The probability of all *b* bits transmitted correctly (under our assumption above) is $(1 - BER)^b$. Therefore, the probability that the packet is in error is the probability that at least one bit is in error:

$$PER = 1 - (1 - BER)^b = 1 - (1 - 10^{-0.1(-20-35\log_{10} d)-10.8})^b \quad \text{(A.4)}$$

This equation is used in Chapter 5 as equation 5.3 for the basic error model used in the simulation.

REFERENCES

1. Acharya, S., Alonso, R., Franklin, M. and Zdonik, S., Broadcast disks - data management for asymmetric communications environment. in *ACM SIGMOD International Conference on Management of Data*, (San Jose, CA, 1995), 199-210.

2. Acharya, S., Franklin, M. and Zdonik, S., Balancing push and pull for data broadcast. in *ACM SIGMOD International Conference on Management of Data*, (Tucson, AZ, 1997), 183-194.

3. Acharya, S. and Muthukrishnan, S., Scheduling on-demand broadcasts: new metrics and algorithms. in *International Conference on Mobile Computing and Networking (MOBICOM)*, (Dallas, TX, 1998), 43-54.

4. Aksoy, D. and Franklin, M., Scheduling for large-scale on-demand data broadcasting. in *IEEE Conference on Computer Communications (INFOCOM)*, (San Francisco, CA, 1998), 651-659.

5. Buchholz, S., Schill, A. and Ziegert, T., A simulation study of update techniques for cyclic data broadcast. in *MSWiM 2001*, (Rome, Italy, 2001), 115-122.

6. Celik, A. and Datta, A., A scalable approach for broadcasting data in a wireless network. in *MSWiM 2001*, (Rome, Italy, 2001), 131-138.

7. Duggirala, D. Broadcast mechanism for data transmission in mobile computing environments *Electrical and Computer Engineering*, Rutgers, State University of New Jersey, New Brunswick, NJ, 2000.

8. Fainberg, M. A performance analysis of the IEEE 802.11b local area network in the presence of Bluetooth personal area network *Electrical and Computer Engineering*, Polytechnic University, Brooklyn, NY, 2001.

9. Goodman, D.J., Borras, J., Mandayam, N.B. and Yates, R.D., Infostations: a new system model for data and messaging services. in *IEEE Vehicular Technology Conference*, (1997), 969-973.

10. Guo, Y., Das, S.K. and Pinotti, C.M., A new hybrid broadcast scheduling algorithm for asymmetric communication systems: push and pull data based on optimal cut-off point. in *MSWiM 2001*, (Rome, Italy, 2001), 123-130.

11. Hameed, S. and Vaidya, N. Efficient algorithms for scheduling data broadcast. *Wireless Networks*, *5*. 183-193.

12. Hu, Q., Lee, D.L. and Lee, W.C., Performance evaluation of a wireless hierarchical data dissemination system. in *International Conference on Mobile Computing and Networking (MOBICOM)*, (Seattle, WA, 1999), 163-173.

13. Iacono, A.L. and Rose, C., Bounds on file delivery delay in an infostations system. in *Proc. of IEEE Vehicular Technology*, (2000), 2295-2299.

14. Imielinski, T., Viswanathan, S. and Badrinath, B.R. Data on air - organization and access. *IEEE Transactions of Knowledge and Data Engineering*, *9* (3). 353-372.

15. Imielinski, T., Viswanathan, S. and Badrinath, B.R., Power efficient filtering of data on air. in *4th International Conference on Extending Database Technology*, (Cambridge, United Kingdom, 1994), 245-258.

16. Lee, W.C., Hu, Q. and Lee, D.L. A study on channel allocation for data dissemination in mobile computing environments. *Mobile Networks and Applications*, *4*. 117-129.

17. Liberatore, V., Multicast scheduling for list requests. in *IEEE Conference on Computer Communications (INFOCOM)*, (New York, NY, 2002), 1129-1137.

18. Natkaniec, M. and Pach, A.R., An analysis of the back-off mechanism used in IEEE 802.11 networks. in *5th IEEE Symposium on Computers and Communications (ISCC)*, (Antibes, France, 2000), 444-449.

19. Schwetman, H. CSIM Reference Manual (Version 18), Mesquite Software, Inc., Austin, TX.

20. Stathatos, K., Roussopoulos, N. and Baras, J., Adaptive data broadcast in hybrid networks. in *23rd Very Large Data Base (VLDB) Conference*, (1997), 326-335.

21. Su, C.J. Information dissemination through broadcast delivery *Electrical Engineering*, Polytechnic University, Brooklyn, NY, 1998.

22. Su, C.J., Tassiulas, L. and Tsotras, V. Broadcast scheduling for information distribution. *Wireless Networks*, *5*. 137-147.

23. Sun, W., Shi, W., Shi, B., Ji, W. and Yu, Y., A self-adaptive scheduling algorithm of on-demand broadcasts. in *MSWiM 2001*, (Rome, Italy, 2001), 139-146.

24. Vaidya, N. and Hameed, S. Scheduling data broadcast in asymmetric communication environments. *Wireless Networks*, *5*. 171-182.

25. Varshney, U. and Vetter, R. Emerging mobile and wireless networks *Communications of the ACM*, 2000, 73-81.

26. Wong, J.W. Broadcast delivery. *Proceedings of the IEEE*, *76* (12). 1566-1577.

27. Wu, G., Chu, C.W., Wine, K., Evans, J. and Frenkiel, R., WINMAC: A novel transmission protocol for infostations. in *IEEE Vehicular Technology Conference*, (1999), 1340-1344.

28. Ye, T., Jacobsen, H.A. and Katz, R., Mobile awareness in a wide area wireless network of info-stations. in *International Conference on Mobile Computing and Networking (MOBICOM)*, (Dallas, TX, 1998), 109-120.

29. Yuen, W.H., Yates, R.D. and Mau, S., Exploiting data diversity and multiuser diversity in non-cooperative mobile infostation networks. in *IEEE Conference on Computer Communications (INFOCOM)*, (San Francisco, CA, 2003).