PERFORMANCE ANALYSIS OF WIRELESS NETWORKS WITH QoS ADAPTATIONS

Trivikram Dash, B.S., B.S.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2003

APPROVED:

Azzedine Boukerche, Co-Major Professor
Roy T. Jacob, Co-Major Professor
Rada Mihalcea, Committee Member
Armin Mikler, Committee Member
Paul Tarau, Committee Member
Krishna Kavi, Chair of the Department of Computer Science and Engineering
Oscar Garcia, Dean of the College of Engineering
C. Neal Tate, Dean of the Robert B. Toulouse School of Graduate Studies

Dash, Trivikram, *Performance Analysis of Wireless Networks with QoS Adaptations*, Master of Science (Computer Science), August 2003, 99 pp., 2 tables, 16 figures, references, 27 titles.

The explosive demand for multimedia and fast transmission of continuous media on wireless networks means the simultaneous existence of traffic requiring different qualities of service (QoS). In this thesis, several efficient algorithms have been developed which offer several QoS to the end-user. We first look at a request TDMA/CDMA protocol for supporting wireless multimedia traffic, where CDMA is laid over TDMA. Then we look at a hybrid push-pull algorithm for wireless networks, and present a generalized performance analysis of the proposed protocol. Some of the QoS factors considered include customer retrial rates due to user impatience and system timeouts and different levels of priority and weights for mobile hosts. We have also looked at how customer impatience and system timeouts affect the QoS provided by several queuing and scheduling schemes such as FIFO, priority, weighted fair queuing, and the application of the stretch-optimal algorithm to scheduling.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation, thanks, and gratitude to my co-advisors Professor Azzedine Boukerche and Professor Roy T. Jacob. This thesis has been made possible by their expert guidance and valuable advice. Their support, leadership, patience, attention to detail, hard work, and scholarship have set an example I hope to match some day. I would also like to thank my committee members Professor Paul Tarau, Professor Rada Mihalcea, and Professor Armin Mikler for their valuable suggestions. Many thanks also to my parents Professor Rangadhar Dash and Mrs. Kalyani Dash for their love, moral support, and encouragement during my days at UNT as a Masters student. A final thank you goes out to all those students in UNT who have made my life as a Masters student enjoyable by providing friendship and humor.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The desire for being able to communicate and receive information anytime and anyplace has fueled the explosive growth of wireless networks and in turn has changed the way people work and live. Cell phones, notebook computers, video conferencing, smart appliances, distance learning, satellite television, short message services, paging, and many other applications of wireless networks have become part of the daily lives of many people both at home and in the business world. People do not want to be tethered to a desk or wall outlet to get in touch with others and also want to be able to have multimedia information on demand quickly. Mobile and wireless technologies have made all of these possible.

Wireless networks are made up of hosts and nodes which are not physically connected, but instead linked together via means of radio frequency channels. A wireless network is divided up into many approximately hexagonal regions called cells. Each cell is controlled by a base station which serves as the terminal point of the wired network. A mobile host which wants to communicate with other hosts sends its message through the base station controlling the cell that the sending mobile host is presently in. This base station then sends this message to a mobile switching center which checks its home location register database to determine the location of the

destination host. Each mobile switching center is in charge of controlling several base stations, all of them controlling contiguous cells. The mobile switching center then sends the message to a gateway mobile switching center which then directs it to a wired network (such as the PSTN in case of a cell phone call) if the destination host is directly connected to a wired network. If on the other hand, the destination host is mobile then the message is instead directed to an appropriate mobile switching center. If the message has been directed to a mobile switching center, the mobile switching center then determines the base station which has control over the destination mobile host. This base station delivers the message to the destination mobile host [13].

Although wireless networks are now a part of our everyday lives, the work needed to be done in wireless research is far from over. Location management, security and encryption techniques, minimizing power consumption, novel microwave circuit and DSP architectures, data and image compression techniques, and quality of service obtained by users of wireless networks are but only a few of the many issues that are being investigated by researchers.

Quality of service (QoS) is an important issue and consideration which needs to be addressed in the design of future generation of wireless networks. Wireless services are evolving from equal access to all users with no guarantees of delivery to preferential services whose guarantees can be calculated. The QoS parameters examined in this thesis have to do with customer retrial rates affected by user impatience and system

timeouts. Also looked at, are the impact of different packet queuing schemes on the QoS received by users. We also look at both overall and individual data access and response times in wireless networks.

The thesis is divided up into two parts. In the first part, an existing hybrid TDMA/CDMA protocol for wireless networks has been generalized by introducing several QoS factors to support multimedia and fast transmission of live audio and video. An analytical model is developed in order to be able to compute parameters affecting the QoS received by customers. The QoS is analyzed at every stage from message arrival to reservation to departure to see if end-to-end QoS can be guaranteed or not.

The customer arrival and reservation rates are generalized to take into account customer impatience and system generated timeouts. Customer impatience can refer to a mobile host giving up while waiting to obtain a connection to the wireless network or it can refer to a mobile host giving up while waiting to obtain a reservation after it has obtained a connection. Then three queuing schemes for scheduling are looked at: first-in-first-out (FIFO), priority queuing (PQ), and weighted fair queuing (WFQ). Then the impact of these scheduling schemes on QoS related factors are examined. After that, further generalizations are made and then shown are the effects of customer impatience and system timeouts on QoS relevant parameters in the PQ, WFQ, and a hybrid PQ-WFQ scheme. Several efficient design schemes which take advantage of

the preceding three scheduling techniques are suggested. Finally, the stretch-optimal scheduling algorithm, normally used only for broadcasting of data, is applied in an unique way to the Request TDMA/CDMA protocol.

The second part of the thesis analyzes the performance of a hybrid push-pull algorithm for wireless networks and introduces an enhanced hybrid push-pull algorithm which incorporates QoS factors. We focused upon a hybrid push-pull algorithm in an asymmetric communication environment in which there is one server and many clients. Those items which are popular are sent by the server without the need for explicit requests from the clients. This is known as broadcasting and the data items which are broadcasted are known as push data. Those items which are infrequently used have to be explicitly requested and are known as pull data. The pull queue contains the message requests made for the pull data. The hybrid push-pull algorithm tries to determine a cut-off point between the push and pull data so that the overall expected time is minimized. Analytical expressions determining the overall response time for the pull queue to be flushed out and the overall response time for individual pull items to be disseminated using this algorithm are determined.

An enhanced hybrid push-pull algorithm is developed next, which is able to handle systems which have data of non-uniform sizes and multiple client priority levels. Another aspect of this algorithm is that no a priori knowledge of data access probabilities is needed and spurious or aberrant cases are handled properly. This algorithm is the

only push-pull algorithm which handles all four factors mentioned in this paragraph simultaneously. Again, analytical expressions for both the overall and individual response times for the pull items are determined.

The rest of this thesis is organized as follows. Chapter 2 presents a performance analysis of a request TDMA/CDMA protocol by introducing several QoS factors. Chapter 3 analyzes the performance of a hybrid push-pull algorithm by introducing several QoS factors. Finally, Chapter 4 is a discussion of conclusions.

CHAPTER 2

PERFORMANCE ANALYSIS OF A REFINED HYBRID TDMA/CDMA PROTOCOL WITH QoS ADAPTATIONS IN WIRELESS NETWORKS

As multimedia and high transmission of continuous media, such as live audio and video, with high quality become more popular on wireless networks, the various traffic requiring different qualities of service (QoS) will co-exist. To this end, a request TDMA/CDMA protocol for supporting multimedia traffic in wireless networks, where CDMA is laid over TDMA has been proposed recently.

In this chapter, we wish to extend this scheme by introducing several qualities of service to the end-user, and present a generalized performance analysis of the request TDMA/CDMA QoS-based protocol [10]. Our analytical model allows us to understand how the end-to-end QoS guarantee can be obtained by analyzing the QoS requirements at every stage of the message delivery, from arrival to contention and transmission.

Quality of service factors we will look at include customer retrial rates due to user impatience and system timeouts. Also looked at are various queuing schemes: FIFO, priority, and weighted fair queueing schemes. We will look at how customer impatience and system timeouts affect the QoS provided by the scheduling schemes.

Finally, we will apply the stretch-optimal algorithm to our protocol and see what the resulting QoS our users obtain.

2.1 Introduction

In recent years, wireless multimedia and Internet services have rapidly evolved to meet a variety of user demands. As a result, looking at the network layer in isolation may not always be sufficient to guarantee QoS requirements. Researchers must look at the QoS requirements at all the different medium access control (MAC) protocol levels, since various traffic requires different methods of QoS.

Several MAC protocols have been proposed for supporting multimedia traffic in wireless networks. In this chapter, we focus upon a hybrid protocol, where we layer code division multiple access (CDMA) over time division multiple access (TDMA). In the protocol, a time frame has two types of slots: data slots and control slots. The idea behind this scheme is to take advantage of both TDMA and CDMA features. Some advantages of TDMA include the need for fewer number of radio channels, no variable bandwidth needed, no duplexer needed, and lower power consumption [24]. CDMA was selected mainly because many users can be accommodated on a single channel at the same time. CDMA provides higher security compared to other transmission schemes. To realize this higher security level, CDMA transmits the message signal over a wider bandwidth than that of the original signal. Furthermore, CDMA can

accommodate eight to ten times more users than the AMPS system and four to five times more users than a GSM system [24].

In [11], the authors present a performance analysis of their hybrid TDMA/CDMA protocol for supporting multimedia traffic in wireless networks. The reported results are quite encouraging even though they did not incorporate many factors that are crucial to guarantee the QoS requirements needed to support a variety of user demands as well as the multimedia traffic. Some QoS parameters that are of interest include: bandwidth, delay, jitter bound, bit rate, temporal and spatial synchronization, and multiple priority and weight levels. We can classify the QoS requirements in many different ways. One way is to categorize it to be either of preferable quality or acceptable quality [23]. One may also use three guarantee levels: hard or deterministic in which the QoS is fully satisfied, soft or statistic in which the QoS is satisfied within a certain probability, p, and finally a best effort, which is not really a guarantee and might be hard to achieve [23]. Another way to categorize QoS is based on bit rates allocated to an user, i.e., constant, variable, and available bit rates [23]. In this chapter, we focus on the QoS requirements that deal with the following parameters: customer retrials, user-initiated timeouts, and system generated timeouts. With this chapter, we intend to extend the model, developed in [11], to study the implementation of QoS requirements at different MAC protocol levels. We wish to generalize the TDMA/CDMA analytical model by taking into account several QoS factors, such as

call blocking (i.e., whether it be initiated by the customer or the system), customer retrials, timeout during the contention process, and the following packet class-based priority policies: First-in First-out (FIFO), Weighted Fair Queuing (WFQ), Priority Queueing (PQ). Although priority queues with respect to wireless multimedia have been looked at in [21] and [22], none have been looked at for a hybrid CDMA-TDMA protocol. We will show how customer impatience and system timeouts affect the scheduling schemes and how that in turn affects the QoS received by the users. Finally, we will show how the stretch-optimal algorithm can be applied to our protocol and see what the resulting QoS we obtain.

The remainder of this chapter is organized as follows: Section 2.2 gives the notation we will be using throughout our chapter. Section 2.3 reviews the hybrid TDMA/CDMA hybrid protocol in [11]. Section 2.4 presents our generalized analytical model of the request TDMA/CDMA protocol. Section 2.5 discusses various scheduling disciplines to meet the QoS requirements. Section 2.6 discusses the impact of customer impatience and system timeouts on the scheduling schemes and the resulting QoS the user receives. Section 2.7 applies the stretch-optimal scheduling algorithm to our protocol and sees how the QoS is affected. Section 2.8 presents our conclusions.

2.2 Nomenclature

In Table 2.1 we show the notation most frequently used throughout our chapter.

2.3 Hybrid TDMA/CDMA Protocol

In this section, we outline the basic elements of the hybrid TDMA/CDMA protocol that we use in our analysis. Interested readers may consult [11] for a more detailed discussion of the TDMA/CDMA protocol. The key to successful utilization of the TDMA/CDMA scheme is taking advantage of the benefits of both TDMA and CDMA features. In this hybrid protocol, the time is divided up into equal sized frames each of duration T, and repeats itself. Each frame consists of an RTS (request to send) slot, S data slots of equal length, and a CTS (clear to send) slot. Each data slot can accommodate up to U users simultaneously. Each user is assigned a code that has a minimal correlation with the other codes so that he can transmit the message with minimal interference.

The base station will scan the RTS slot in order to determine how to schedule all incoming requests for a call establishment, as well as all other events that are occurring in the system. The CTS slot is divided into a two dimensional grid of S columns and U rows. Each column represents a data slot and each row represents an unique code assigned to a mobile host (MH) for a time frame. This means up to U

users can simultaneously use a data slot. The codes are unique and orthogonal and there is little cross-correlation between any two codes, if any. The frame structure is shown in Figure 2.1.

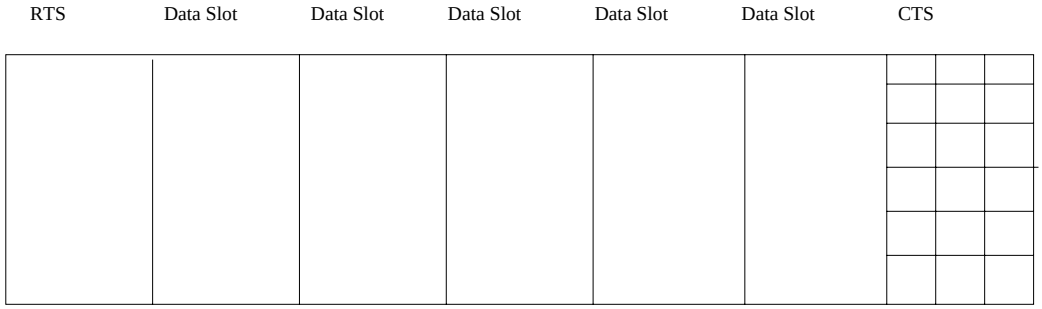


Figure 2.1: Frame Structure

Once an user gets a reservation, he will keep that reservation until the complete message is transmitted. Note that the number of data slots a MH receives per frame is linearly proportional to the bit rate requirement of that MH. In the hybrid CDMA-TDMA protocol, it is possible to have a situation where there are no reserved slots, or only a limited number of slots that may be reserved for certain classes of multimedia traffic. A MH cannot generate a new message if that MH is in contention mode. Also, a MH that generates a message in the present frame can only access the data slots in subsequent frames.

The state of the system, during the next frame, can be determined by the information contained in the current frame's CTS slot. The hybrid protocol assumes that no MHs enter or leave the cell and the number of MHs is constant. We represent the

system state by the following tuple (n_c, n_r), where n_c is the number of MHs in the contention mode and n_r is the number of MHs in the reservation mode. If n_t is the number of MHs in a thinking mode, then $n = n_c + n_r + n_t$ indicates the total number of MHs in the whole system.

In order to capture the state of the system, the following probabilities were derived in the original model [11]

$$\begin{aligned}
A(n_t, j, \lambda, T) &= \binom{n_t}{j} e^{-\lambda T(n_t - j)}(1 - e^{-\lambda T})^j \\
S(n_c, j, q) &= \binom{n_c}{j} q^j (1-q)^{n_c - j} \\
D(n_r, j, L_m) &= \prod_{i=1}^{m} Pr[j_{i_{dep}}] \\
Pr(j_{i_{dep}}) &= \binom{n_i}{j_i} (\tfrac{i}{L_m})^{j_i} (1 - (\tfrac{i}{L_m}))^{n_i - j_i} \\
E(w) &= (1-q)^{-U} - (\tfrac{T}{2})
\end{aligned}$$

where A represents the probability of j request arrivals, j is the number of MHs, λ represents the message generation rate and T is the time duration of a frame. S is the probability of obtaining j reservations and q, a function of the number of requested slots, is the probability of getting a reservation. $E(w)$ is the expected waiting time, D is the probability of obtaining j reservations from all the classes combined, where L_m is the average message length. $j_{m_{dep}}$ is the number of departures of MHs of traffic class m. A MH of traffic class m is a MH which requires m slots per frame. The transition probabilities can be thought of as shown in Figure 2.2.

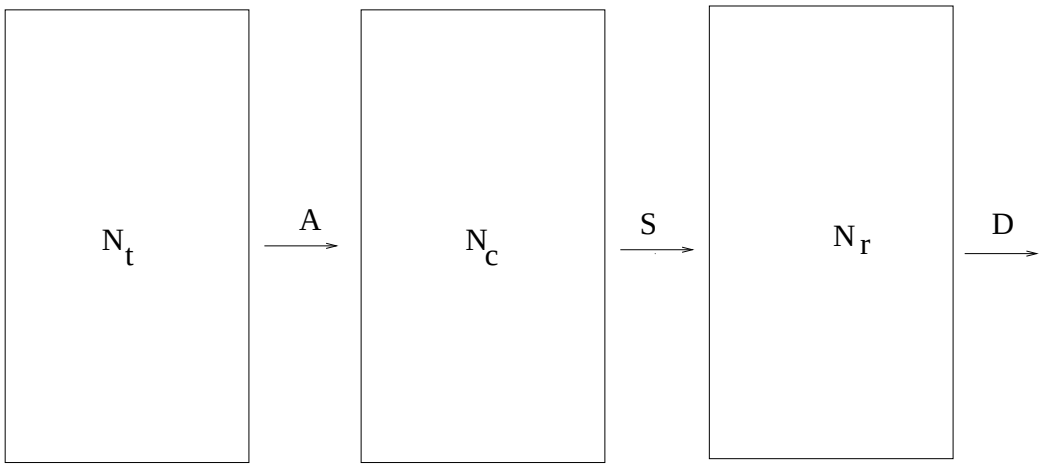


Figure 2.2: Transition Probabilities from Thinking Mode to Contention Mode to Reserved Mode

Now, let us assume that A_n represents the number of MHs looking for a reservation between the n^{th} frame and $(n+1)^{th}$ frame. S_n represents the number of MHs which are able to get a reservation between the n^{th} and $(n+1)^{th}$ frame. D_n is the number of MHs leaving after they have finished sending their message.

Note that the state of the system is identified by (n_c, n_r), after the n^{th} frame and by (l_c, l_r), after the $(n+1)^{th}$ frame. Let us say that l_c - $n_c = k$ and l_r - $n_r = l$, where k and l are integer values. Depending upon the values of k and l, four cases may arise for the probability in which the state (n_c, n_r) can go to (l_c, l_r).

Case 1: $k \geq 0; l \geq 0$

$$Pr[(n_c, n_r) -> (n_c + k, n_r + l)] = \sum_{j=l}^{Z} A(n_t, k+j, \lambda, T) D(n_r, j-l, L_m) S(n_c, j, q)$$

where $Z = min(n_c, n_t - k, n_r + l)$

Case 2: $k \geq 0, l < 0$

$$Pr[(n_c, n_r) -> (n_c + k, n_r - l)] = \sum_{j=0}^{Z} A(n_t, k + j, \lambda, T) D(n_r, j + l, L_m) S(n_c, j, q)$$

where $Z = min(n_c, n_t - k, n_r - l)$

Case 3: $k < 0, l \geq 0$

$$Pr[(n_c, n_r) -> (n_c - k, n_r - l)] = \sum_{j=X}^{Z} A(n_t, j - k, \lambda, T) D(n_r, j - l, L_m) S(n_c, j, q)$$

where $X = max(k, l)$ and $Z = min(n_c, n_t + k, n_r + l)$

Case 4: $k < 0, l < 0$

$$Pr[(n_c, n_r) -> (n_c - k, n_r - l)] = \sum_{j=k}^{Z} A(n_t, j - k, \lambda, T) D(n_r, j + l, L_m) S(n_c, j, q)$$

where $Z = min(n_c, n_t + k, n_r + l)$

2.4 Generalized Analytical Model for The Hybrid TDMA/CDMA Protocol

In this section, we present our generalized analytical model for the TDMA/CDMA protocol and discuss how this generalized model incorporates the QoS requirements, to support a variety of multimedia traffic in wireless networks. More specifically, we discuss how we can extend the original performance model by generalizing the probability of customer arrival and the probability of obtaining a reservation.

2.4.1 Analysis of Customer Arrival Rates

Customer impatience or retrials and system timeouts can significantly influence the arrival rate of messages generated by MHs. Waiting too long for a call to connect and then hanging up and trying again is an example of customer impatience. On the other hand, a system timeout occurs when a message request is removed from the base station's scheduler request queue. Waiting too long for a call to connect or getting disconnected by the system can certainly impact the arrival rate of a message by a MH. Assuming that the arrival rate of a message by a MH never changes is not always a safe assumption to make.

The original analytical model developed in [11] must be refined in order to account for QoS requirements. We will see how customer impatience and system timeouts can change the values of A (probability of arrivals) and S (probability of reservations) from that of the original model. Being able to determine these probabilities will help

us to analyze the QoS parameters that are involved. Let us first look at how A can be generalized to consider these factors.

In the original analytical model [11], each MH generates messages at a rate λ. In our generalized approach, we say that the rate at which a MH generates a message can be affected by the following three parameters: the number of customer retrials, the number of system generated timeouts experienced and the class of multimedia traffic, i, indicates the the number of bits required per frame.

If we denote by λ_x, the rate at which a MH x generates a message, then we have the following:

$$\lambda_x = \frac{\lambda e^{-r*\lambda_A} e^{-s*\lambda_B}}{f(i)^{sgn(r+s)}}$$

$$\text{where } sgn(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases},$$

r = number of retrials, s = number of system timeouts, λ_A = a constant parameter related to the customer arrival rate, λ_B = a constant parameter related to the system timeout rate, $f(i)$ = a monotonically strictly increasing function to be determined by the cellular network operator, and i = the required number of bits per frame for our message request.

The above equation reduces to $\lambda_x = \lambda$, when there are neither retrials nor system timeouts. Note, as the number of retrials and/or the number of system timeouts increase, the lower the probability of a message being sent again. Furthermore, the probability of retrying a call for the tenth time is less likely than one-tenth of the probability of retrying it for the first time. We identify this type of behavior using the exponential distribution in the above equation. A customer is more likely to retry a call more frequently if the retry is as a result of the customer's impatience, rather than the system timing him out. It takes longer for a system to timeout a call than it does for an user to end the call on his own. We can represent this factor by saying that $\lambda_A << \lambda_B$ (these are both constants).

If retrials are involved, the rate at which the MH tries to regenerate the message will be affected by the class of traffic involved. It is not likely that we will retry sending a video message as many times as we would retry sending a voice message, since waiting for a video message to get a reservation normally takes a significant amount of time, compared to a voice message. Hence, we leave $f(i)$, where i is the number of bits required per frame, in the denominator of our equation. $f(i)$ is a monotonically, strictly increasing function. Determining precisely what $f(i)$ is and what the constant lambda factors are for a particular cell necessitates statistical data collection and is beyond the scope of this thesis.

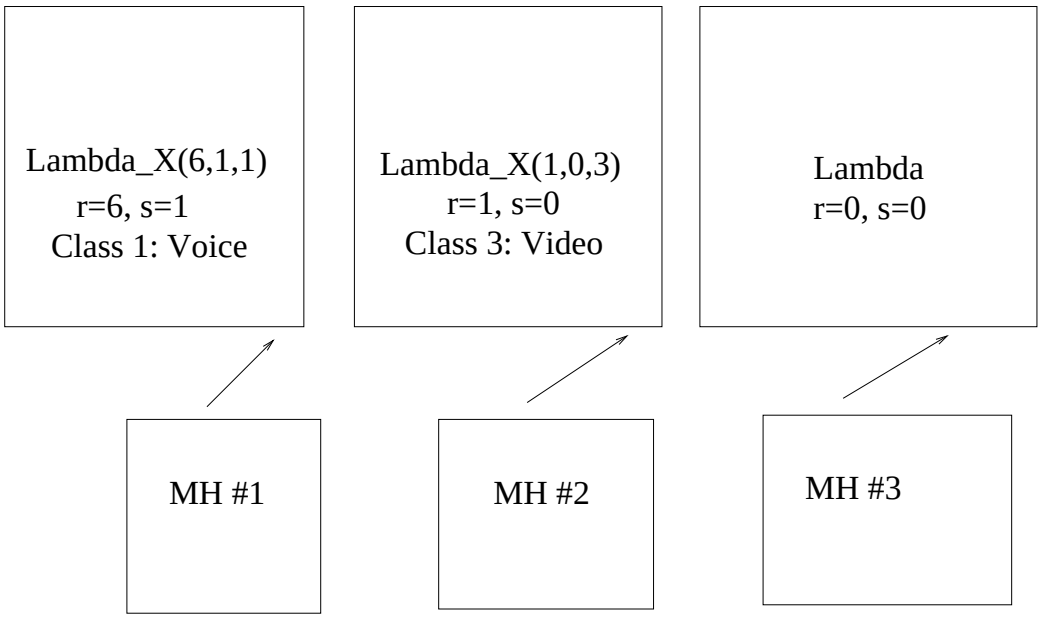


Figure 2.3: Example of Generalized Message Arrival Rate as a Function of the Number of Retrials (r) and System Timeouts (s) for 3 MHs

Similar to the original model [11], we assume that the probability of a MH generating a message is the same for all MHs if no retrials are involved. Therefore, the probability that a MH x generates a message, PG_x, within a time interval T can be computed as follows

$$PG_x = 1 - \prod_{r=1}^{\infty} \prod_{s=1}^{\infty} (1 - e^{-\lambda_x T})$$

where r is the number of retrials, s is the number of system timeouts, and λ_x (a function of r and s) is the generalized message arrival rate.

Thus, the probability of exactly j MHs generating a message when taking QoS parameters into consideration can be expressed as follows:

$$A(n_t, j, \lambda, T) = \sum_{k=1}^{\binom{n_t}{j}} \prod_{x \in S_k} PG_x \prod_{x \in (S - S_k)} (1 - PG_x)$$

where $S = \{1, 2, 3, ..., n_t\}$ and $S_k = k^{th}$ distinct non-empty subset of S containing j elements, and λ as mentioned before is the message generation rate of a MH in the absence of retrials and system timeouts.

In Figure 2.4, we portray the rate at which a MH generates a message, based on customer retrials and system timeouts. The graph indicates that the rate at which a MH retries the message drops sharply after approximately four customer initiated retrials or two system timeouts. Figure 2.4 also shows the rate at which a MH retries the message approaches zero after approximately six timeouts.

2.4.2 Analysis of Reservation Rates

Just as the arrival rate of messages generated by MHs can be affected by user impatience or system timeouts, so too can the chances of a MH getting a reservation. In a highly dynamic environment, we need to take this into consideration in order to derive realistic conclusions.

We generalize the probability of obtaining a reservation, from the original model

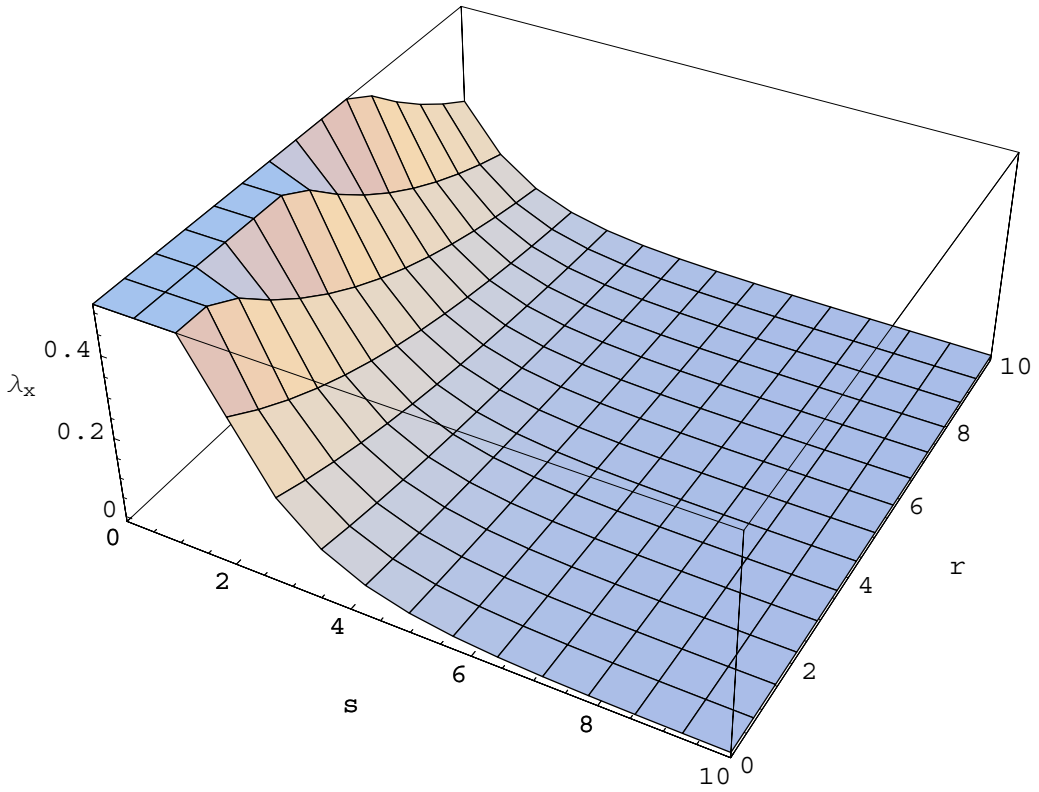


Figure 2.4: Generalized Message Arrival Rate (λ_x) as a Function of the Number of Retrials (r) and System Timeouts (s) where $\lambda_A = 0.2$ and $\lambda_B = 0.8$

[11], by introducing a parameter $\lambda_u(i)$, the rate of user block (impatience) for class i traffic. As mentioned before, class i traffic refers to a message transmission that requires a reservation of i data slots per frame.

The parameter mentioned above is a function of i. As i increases, the user is willing to wait longer than usual. For example, people are accustomed to waiting longer to send a video message than to send a voice message. Therefore, these rates

are inversely proportional to i. That means that λ_u is inversely proportional to i and $\frac{d\lambda_u}{di} < 0$.

Note that an increase in the value of i increases the chances of the system timing out, particularly, if we do not have reserved slots in the system. On the other hand, if we do have reserved slots, then we can reserve these slots so that $\frac{d\lambda_u}{di} < 0$. Therefore, the probability of an user timing out a call, due to caller impatience, is computed as $e^{-\lambda_u(i)T}$.

Let us now assume that the system does time out a call if it cannot go from contention mode to reservation mode within H frames. Let $q_k(i)$ denote the probability of getting a reservation for i slots in the k^{th} frame, if a reservation was not able to be made in the previous k frames, where $1 \leq k \leq H$. Then, the probability of a reservation succeeding in the first frame can be determined as $q_1(1 - e^{-\lambda_u(i)T})$. Note that the first factor identifies the probability of getting a reservation in the first frame and the second factor is the probability of not having an user initiated time out for a message of traffic class i in the first frame.

If we let $PH_x(i)$ be the probability of MH x succeeding to get a reservation for i slots, without giving up due to system timeout or user impatience, we can then say,

$$PH_x(i) = [1 - \prod_{k=1}^{H}(1 - q_k(i))]e^{-\lambda_u HT}$$

where H = number of frames needed for timing out, $q_k(i)$ is the probability of getting a reservation of i slots during the k^{th} trial, and λ_u is the user block rate.

Furthermore, the probability of j_i MHs getting a reservation if the system times out after H frames can be determined as follows:

$$S(n_{c_i}, j_i, q, H) = \binom{n_{c_i}}{j_i} PH_x(i)^{j_i}(1 - PH_x(i))^{n_{c_i} - j_i}$$

where n_{c_i} represents the number of contention MHs that are trying to get a reservation of i slots per frame, j_i represents the number of contention MHs which have a requirement of i slots per frame, and q represents the initial probability of obtaining a reservation requiring one slot per frame.

The results of the probability of obtaining a reservation are summarized in Figure 2.5, which indicates that as we decrease the user block rate and increase the number of frames, the probability of obtaining a reservation increases.

2.5 Scheduling Schemes in the Request TDMA/CDMA Protocol

Scheduling schemes are a way of determining which MH's request gets honored. The scheduler in the base station contains a queue which contains the requests for establishing a connection made by the MHs. It also contains all associated information such as the length of a message and bandwidth requirements for the message the MH wants to transmit. The scheduling scheme chosen can greatly impact the QoS

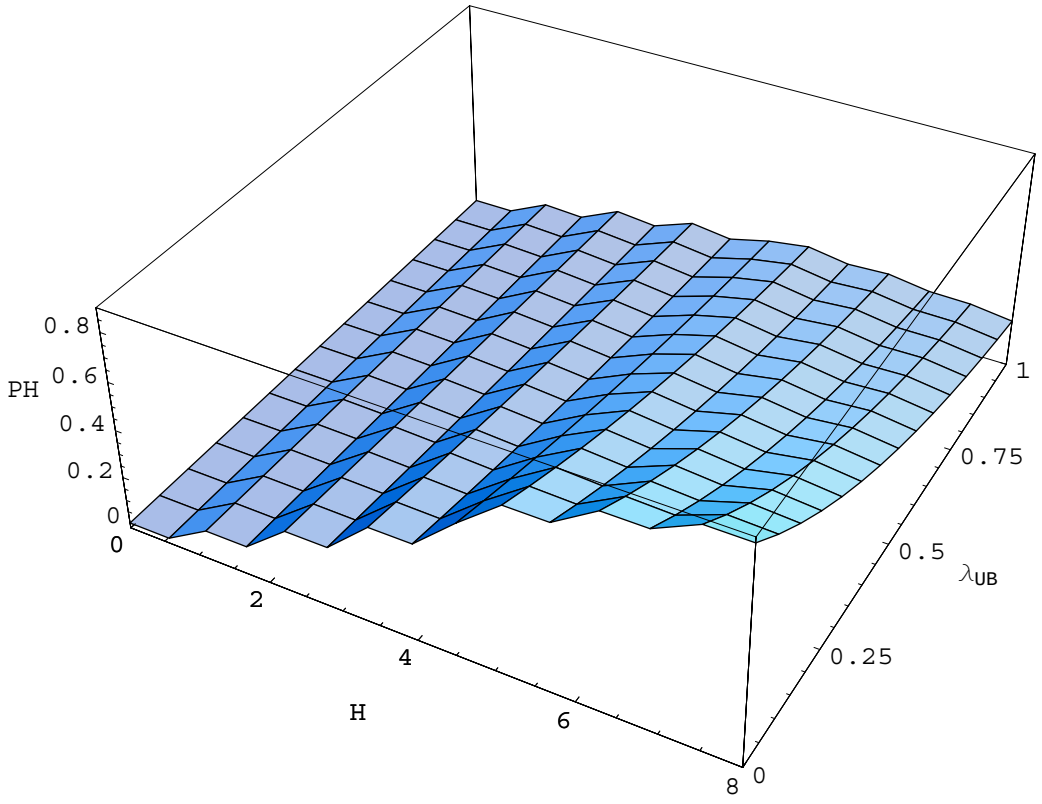


Figure 2.5: Reservation Probability (PH) as a Function of the Number of Frames Required to Timeout (H) and the User Block Rate (λ_{UB})

received by the MHs. We have to pick the right scheduling scheme for our needs. The arrival and reservation probabilities are affected by the scheduling scheme we choose. Just how they are affected will be explained in this section. The three scheduling schemes we will look at are: First In First Out (FIFO), Priority Queues (PQ), and Weighted Fair (WF) Scheduling.

2.5.1 FIFO Queues

First In First Out (FIFO) is the most common scheduling technique used by applications. FIFO is easy to implement, but is certainly not the best scheme to incorporate for providing good quality of service to MHs. In a FIFO scheme, the message requests in the queue are honored in the time order of their arrival. The earlier message requests get honored before the later ones. In other words, those message requests which come earlier are more likely to obtain a reservation than those coming later. The reservation probability for a message transmission request being honored in the next frame increases the earlier that message transmission request arrives. A FIFO scheme does not give preferential treatment for different traffic classes. A FIFO scheme is well suited if all MHs are of equal importance and transmit messages of only one traffic class.

We know that the probability that a request with a requirement of b_1 data slots per frame gets a reservation is

$$q_1 = 1 - [\sum_{i=0}^{b_1-1} \binom{S}{i} p^i (1-p)^{S-i}]^U$$

where S = the number of data slots in a frame, and U = the maximum number of MHs which can transmit a message simultaneously.

Therefore, we can say that the probability of a second MH, with a bit rate requirement of b_2 data slots per frame, getting a reservation in that same time frame, given that the first MH was successful in obtaining one is as follows

$$q_2 = 1 - \sum_{i=0}^{b_2-1} [\binom{S}{i} (\frac{p}{1-pb_1})^i (\frac{1-pb_1-p}{1-pb_1})^{S-i}]^U$$

where once again S = the number of data slots in a frame, and U = the maximum number of MHs which can transmit a message simultaneously.

Thus, the probability of a k^{th} MH, with a bit rate requirement of b_k data slots per frame, getting a reservation in that same time frame, given that the first $(k-1)$ MHs have obtained their reservations is as shown below.

$$q_k = 1 - \sum_{i=0}^{b_k-1} [\binom{S}{i} (\frac{p}{1-p\sum_{h=1}^{k-1} b_h})^i (\frac{1-p+p\sum_{h=1}^{k-1} b_h}{1-p\sum_{h=1}^{k-1} b_h})^{S-i}]^U$$

Thus, we can say that the probability of j MHs obtaining a reservation under a FIFO queue discipline is as follows

$$S(n_c, j) = 1 - \prod_{k=1}^{j} (1 - S(n_c, k, q_k))$$

where q_k is the probability of getting a reservation in the k^{th} frame, and n_c is the number of MHs in contention mode.

Figure 2.6 portrays this probability and clearly shows that the probability of a MH obtaining a reservation in the next frame decreases as the arrival order of the MH in contention increases.

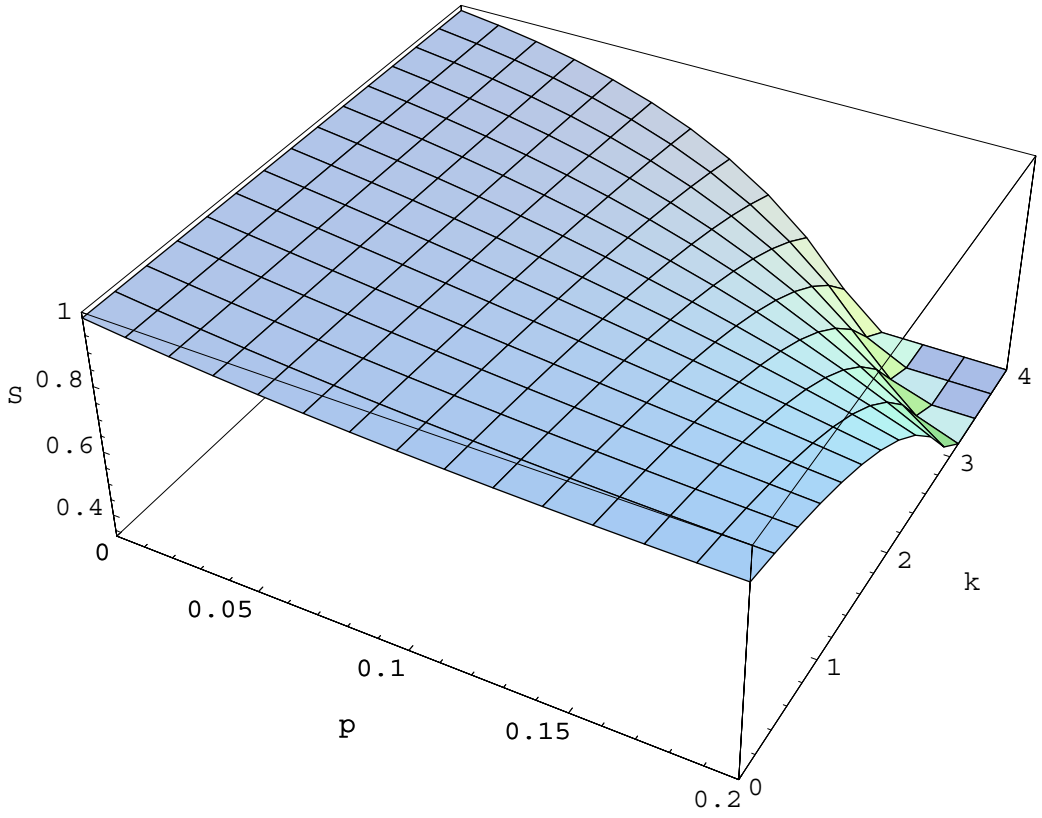


Figure 2.6: Reservation Probability (S) as a Function of the Probability of an Unreserved Slot (p) and the Order of Arrival (k)

2.5.2 Priority Queuing (PQ)

Priority queues can have a significant impact on the quality of service received by MHs. For example, in a cellular network, customers can be given the option of subscribing to various subscription packages. Those who pay for premium packages can be given preferential service. In other words, the message requests generated by those who pay for premium packages can be given preference or priority over those message requests generated by those who subscribe to the basic plan.

The scheduler in the base station can implement multiple request queues. We can number the request queues from 1 to k, where k is the number of request queues. We will adopt the standard convention that the lower the request queue is numbered the higher its priority. The set of MHs are partitioned into k subsets with each subset corresponding to a distinct priority level.

Let us now analyze how the QoS is affected by this scheduling scheme. More specifically, let us see how the arrival and reservation probabilities are affected. In the original analytical model [11], each MH generates messages at a certain rate. Since our model incorporates the priority queue discipline, we can decompose the overall arrival rate to be the sum of the arrival rate of all priority levels. In other words, if k is the number of queuing levels, then we have $\lambda = \sum_{i=1}^{k} \lambda_i$.

We can also look at the probability of a certain number of arrivals as we did earlier. The probability of j_i MH arrivals with priority level i can be determined by

the following equation.

$$A(n_t, j_i, \lambda_i, T) = \binom{n_t}{j_i} e^{\lambda_i T(n_t - j_i)} (1 - e^{-\lambda_i T})^{j_i}$$

where λ_i = the arrival rate of MHs with priority level i.

The probability of j MH arrivals can be formulated as follows:

$$A(n_t, j, \lambda, T) = \sum_{i=1}^{k} A(n_t, j_i, \lambda_i, T)$$

where $j = \sum_{i=1}^{k} j_i, \lambda = \sum_{i=1}^{k} \lambda_i$, λ_i = the arrival rate of MHs with priority level i, and j_i is the number of MHs of priority level i we are examining.

Let us now look at defining S, the probability that a certain number of MHs will obtain a reservation. It is easy to see that the probability that j_1 priority level 1 MHs get a reservation can be computed as follows:

$$S(n_c, j_1, q_1) = \binom{n_c}{j_1} q_1^j (1 - q)^{n_c - j_1}$$

Based upon the above result, it is easy to compute the probability that j_2 priority level 2 MHs get a reservation, given that all the priority level 1 MHs have obtained a reservation. This is computed as follows:

$$\frac{S(n_c, j_{1_{total}} + j_2, q)}{S(n_c, j_{1_{total}}, q)}$$

where $j_{i_{total}}$ is the number of all contention MHs with priority level i that are in the network. Note that the denominator represents the probability that all of the priority level 1 MHs in the queue will obtain a reservation. We have a conditional probability because all of the priority level 1 MHs in contention have to obtain a reservation before we can grant a reservation to lower level MHs.

Recursively, based upon the above results, we conclude that the probability of j_i MHs, in the i^{th} priority queue, for obtaining a reservation can be formulated as follows:

$$S = \frac{AB}{C}$$

$$A = S(n_c, j_i + \textstyle\sum_{k=1}^{(i-1)} j_{k_{total}}, q)$$

$$B = S(n_c, j_{i-2} + \textstyle\sum_{k=1}^{(i-1)} j_{k_{total}}, q)$$

$$C = S(n_c, \textstyle\sum_{k=1}^{(i-1)} j_{k_{total}}, q)$$

If μ represents the service time, then the expected waiting time, w_i, in the contention mode for a i^{th} priority MH is computed as follows:

$$E(w_i) = \frac{1}{(AB_{i-1}B_i)} + \frac{1}{\mu}$$

$$A = \frac{\mu - \lambda}{\rho} + \mu$$

$$\rho = \frac{\lambda}{\mu}$$

$$B_0 = 1$$

$$B_i = 1 - \frac{\sum_{k=1}^{i} \lambda_k}{\mu}$$

Let us now extend our priority queue model to incorporate several other QoS factors in the new model. The main extension is that we now have additional priority parameters to consider, namely, the number of retrials and number of system timeouts.

Thus, the message generation rate of MH y, λ_y, can be computed as follows:

$$\lambda_y = \frac{\lambda e^{-r*\lambda_A} e^{-s*\lambda_B} (1 - e^{(-priority*\lambda_C)})}{f(i)^{sgn(r+s)}}$$

where $sgn(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$,

r = number of retrials, s = number of system timeouts, and λ is the message arrival rate in the absence of retrials and system timeouts, λ_A , λ_B, and λ_C are constant

parameters related to message retrials, system timeouts, and priority levels in respective order, and *priority* is the number of priority levels which exist in our network. As before, $f(i)$ is a strictly monotonically increasing function which is determined by statistical analysis of data collected by the cellular network operator, and is the number of data slots per frame required.

Therefore, the arrival probability of j_i priority level i MHs can be formulated as follows:

$$A(n_t, j_i, \lambda_i, T) = 1 - (1 - e^{-\lambda_i T}) \prod_{r=1}^{\infty} \prod_{s=1}^{\infty} (1 - e^{-\lambda_y T})$$

and the probability of j MHs arriving can be calculated as follows:

$$A(n_t, j, \lambda, T) = npl - \sum_{i=1}^{npl} (1 - e^{-\lambda_i T}) \prod_{r=1}^{\infty} \prod_{s=1}^{\infty} (1 - e^{-\lambda_y T})$$

where npl is the number of priority levels.

The reservation probability equations, when we have a priority queue, can also take into account the QoS factors. In this case, the reservation probability of j_1 priority level 1 MHs obtaining a reservation, where i is the user class traffic, can be defined as follows:

$$S(n_{c_1}, j_1, q, H) = \binom{n_{c_1}}{j_1} PH_x(i)^{j_1} (1 - PH_x(i))^{n_c - j_1}$$

where n_{c_1} is the number of priority level 1 contention MHs. Then the reservation probability of j_i MHs with i priority levels getting a reservation, when accounting for QoS factors, can be defined as:

$$
\begin{aligned}
&S(n_c, j_i, q, H) = \tfrac{X}{Y} \\
&X = \left(\begin{smallmatrix} n_c \\ j_i + \sum_{k=1}^{i-1} j_{k_{total}} \end{smallmatrix}\right) AB^{n_c - (j_i + \sum_{k=1}^{i-1} j_{k_{total}})} \\
&Y = \left(\begin{smallmatrix} n_c \\ \sum_{k=1}^{i-1} j_{k_{total}} \end{smallmatrix}\right) CB^{n_c - \sum_{k=1}^{i-1} j_{k_{total}}} \\
&A = PH_x(i^{'})^{j_i + \sum_{k=1}^{i-1} j_{k_{total}}} \\
&C = PH_x(i^{'})^{\sum_{k=1}^{i-1} j_{k_{total}}} \\
&B = (1 - PH_X(i^{'}))
\end{aligned}
$$

In Figure 2.7, we see that the reservation probability of one, two and three MHs, of priority level 3, being able to access a data slot in the next frame versus PH, when three priority level 1 MHs and four priority level 2 MHs already have obtained a reservation. When PH is greater than 0.93, we are almost certain of obtaining a reservation for a MH of level 3, and if PH is less than 0.7 then no priority level 3 MHs are likely to obtain a reservation.

2.5.3 Weighted Fair Scheduling Schemes

Weighted fair (WF) scheduling schemes can also have a significant impact on the quality of service received by customers. The weight of each MH keeps on increasing

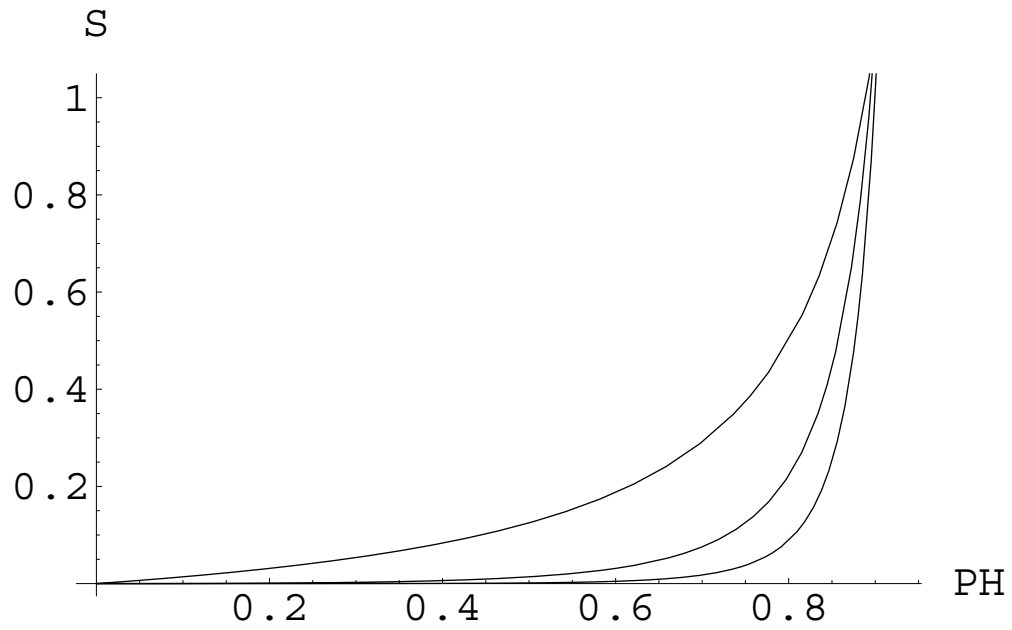


Figure 2.7: Reservation Probability (S) as a Function of PH. Top Curve j=1, Middle Curve j=2, and Bottom Curve j=3

for every extra frame its message request stays in contention mode. This scheme can help us to reduce the chances of getting into a starvation mode. Overall service time is reduced in this scheme.

In this section, we extend the original model [11] to include a weighted fair queueing scheme [12, 25]. We let $A = \{\lambda_1, \lambda_2, ..., \lambda_{n_c}\}$, where each λ_i represents the number of frames for which a MH has been in contention. We let A be a multiset (a set which allows for repeating elements). Let I be a subset of A with j elements. We can assign a nonnegative number, symbolizing a normalized weight (from 0 to 1), for each MH, where w is a strictly monotonically increasing function of λ_i.

We can then say that the probability of j MHs getting a reservation in a frame is as follows:

$$S(n_c, j) = 1 - \sum_{I \subset A} E * F$$

$$E = \prod_{i_1 \in I} (w(i_1))^{b_{i_1}}$$

$$F = \prod_{i_2 \in A-I} (1 - w(i_2))^{b_{i_2}}$$

where b_{i_1} refers to the bit rate requirement for the contending MHs coming from the set I and b_{i_2} refers to those coming from the set $A - I$.

In Figure 2.8, we portray the results of the reservation probability versus the weighting factor w, when we have three MHs in our system and we assign a weight of 0.1 to the first MH, a weight of w to the second MH and a weight of 0.9 to the third MH. We can observe that as the weighting factor of the second MH increases, the probability that we will have two MHs obtaining a reservation in the next frame also increases. On the other hand, as the weighting factor decreases, the chance of having only one MH obtaining a reservation in the next frame increases.

2.6 QoS Factors in Scheduling Schemes

In Section 2.5 we saw how to calculate the arrival and reservation probabilities for the different scheduling schemes. The analysis presented in that section gave us good first-order answers. We can generalize the analysis even further by introducing

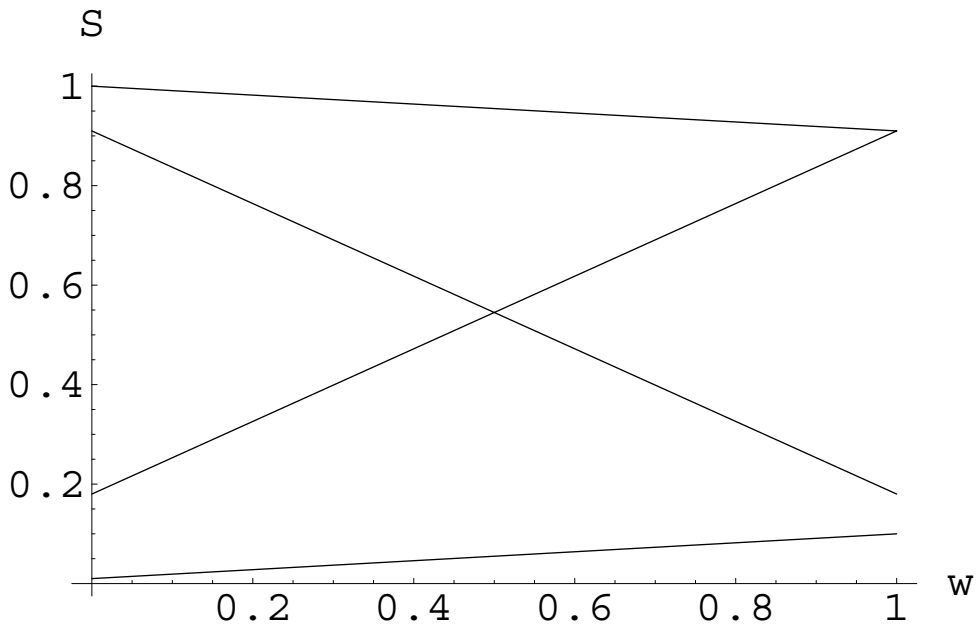


Figure 2.8: Reservation Probability (S) as a Function of the Weight (w). Top Curve j=0, Upper Middle Curve j=1, Lower Middle Curve j=2, and Bottom Curve j=3, where $w1$=0.1, $w2$=w, and $w3$=0.9

QoS factors such as non-constant user block rates and non-constant system timeout lengths. This will allow us to more accurately obtain the reservation probabilities. This will, in turn, provide us a guideline to meeting QoS requirements.

First, we will look at the effects of customer impatience and system timeouts in the priority queueing scheme. Then we will also do the same for the weighted fair scheduling schemes. After that, we will look at the effects of customer impatience and system timeouts in a hybrid priority and weighted fair scheduling scheme. Throughout this section, we will also look at efficient design techniques that can be incorporated

in our scheduling schemes which can improve the overall QoS offered to the users.

2.6.1 QoS Factors in Priority Queues

If we have a system where MHs can have differing priority levels or differing levels of importance, we can be confident that the user block rates of the MHs would also be different from one priority level to another. The bit rate requirement of the message the MH wishes to transmit would also affect the user block rate.

It is intuitive that higher priority MHs are more likely to lose patience while waiting for a call to connect since they are accustomed to receiving quicker responses and better service. We can say that the time a higher priority MH waits before losing patience is going to be shorter than it is for a lower priority one.

In calculating the user block rate, we also have to consider the bit rate requirement of the message a MH wants to transmit too. Based on everyday experience, we know that people are more willing to wait longer for video to be transmitted than for voice or text messages. In other words, people are more willing to wait longer for high bit rate messages to start transmitting than for lower bit rate ones.

Let us introduce the generalized user block rate λ_{UBG} below:

$$\lambda_{UBG}(p, b) = \frac{\lambda_{UB} f(p)}{g(b)}$$

where p is the priority level of our MH, b is the bit rate requirement of the message that our MH wishes to transmit, and f and g are strictly monotonically increasing functions of the priority and bit rate respectively and can be determined by statistical data collection. We follow the convention that the higher the numerical value of p corresponding to a MH, the lower its priority is. The above equation tells us that the generalized user block rate is directly proportional to the value of p and inversely proportional to the bit rate requirement, b.

We will assume as before, that the time for user block or customer impatience is exponentially distributed. We can then say that the probability that a MH of priority level p wanting to transmit a message with a bit rate requirement of b will not give up in time duration T is computed as follows:

$$e^{\frac{-\lambda_{UB} f(p) T}{g(b)}}$$

where λ_{UB} is the user block rate if we do not take into account the MH's priority level nor bit rate requirement, T is the time duration of a frame, p is the priority level of the MH, b is its bit rate requirement, and f and g are strictly monotonically increasing functions of the priority and bit rate respectively and can be determined by statistical data collection.

There is still another important consideration to keep in mind. In a well-designed

system which has different priority levels for MHs, the number of frames before the system times out a MH should depend upon its priority level. This will allow us to further meet QoS requirements. We would like to give higher priority MHs more chances to establish a connection than to lower priority ones. We can define a function, $H(p)$, which is a strictly monotonically decreasing function of the priority level, p. $H(p)$ is the number of frames that a MH can be in contention mode for before it gets timed out by the system.

So, we can define the probability of a MH of priority level p wanting to transmit a message whose bit rate requirement is b and not losing patience if it has not established a connection within $H(p)$ time frames to be the following:

$$e^{\frac{-\lambda_{UB} f(p) TH(p)}{g(b)}}$$

where again f and g are respectively strictly monotonically increasing functions of the priority level and bit rate requirements of the MH, and λ_{UB} is the user block rate if we do not take into consideration the priority nor bit rate requirement of a MH.

If we let $PH_x(p, b)$ to be the probability of MH x, whose priority level is p and bit rate requirement is b, to be able to succeed in getting a reservation for b slots per frame without timing out, then we can say $PH_x(p, b)$ to be the following:

$$PH_x(p,b) = [1 - \prod_{k=1}^{H(p)}(1 - q_k(b))]$$

where again $H(p)$ is the number of frames that a MH of priority level p will be in contention mode for before it gets timed out, and $q_k(b)$ is the probability of getting a reservation for b slots per frame during the k^{th} trial.

We can thus say that the probability of a MH of priority level p and bit rate requirement of b succeeding to get a reservation without giving up due to system timeout or customer impatience to be the following:

$$(PH_x(p,b))e^{\frac{-\lambda_{UB}f(p)TH(p)}{g(b)}}$$

where the first product term represents the probability of the MH of priority level p and bit rate requirement of b succeeding in getting a reservation within $H(p)$ time frames and the second product term represents the probability that the customer will not lose patience within $H(p)$ time frames.

Based on the above results, we can say that the reservation probability for j_1 MHs of priority level 1 to be the following:

$$S(n_{c_1}, j_1, q_{b_1}) = XY$$
$$X = \binom{n_{c_1}}{j_1}(PH_x(p, b_1))^{j_1}(1 - PH_x(p, b_1))^{(n_{c_1} - j_1)}$$
$$Y = e^{\frac{-\lambda_{UB} f(p) TH(p)}{g(b_1)}}$$

where n_{c_1} is the number of contention MHs of priority level 1, and q_{b_1} is the probability of a MH with a requirement of b_1 data slots per frame gets a reservation, and f and g again are strictly monotonically increasing functions of the priority and bit rate respectively.

More generally, we can say that the probability of j MHs obtaining a reservation where there are n_c contention MHs and q is the probability of obtaining a data slot in the next frame is computed as follows:

$$S(n_c, j, q) = AB \prod_{i=1}^{z} CD$$
$$A = \binom{n_{c_{z+1}}}{d}(PH_x(z, b_z))^{d}(1 - PH_x(z, b_z))^{(n_{cz} - d)}$$
$$B = (e^{\frac{-\lambda_{UB} f(z) TH(z)}{g(b_z)}})^{d}$$
$$C = (PH_x(i, b_i))^{j_i}$$
$$D = (e^{\frac{-\lambda_{UB} f(i) TH(i)}{g(b_i)}})^{j_i}$$

where $d = j - n_{c_i}$, z is the largest integer for which $\sum_{i=1}^{z} n_{c_i} \leq j$ holds true, and b_i is the average bit rate requirement for a MH of priority level i. Also, n_{c_i} is the number

of contention MHs at priority level i, $PH_x(p, b_i)$ is the probability of a MH of priority level i obtaining a reservation of b_i slots per frame without the MH losing patience nor being timed out by the system, $H(p)$ is the number of frames required for a MH of priority level p to time out, λ_{UB} is the user block rate when priority and bit rate requirements are not considered, and f and g are strictly monotonically increasing functions of priority and bit rate requirements respectively.

The graph in Figure 2.9 shows the probability of getting a reservation, S, for j number of MHs at priority level 3, versus the time elapsed since the beginning of contention. We have assumed that there are only three priority levels. We have also assumed there are 5 MHs at priority level 1, 3 MHs at priority level 2, and 4 MHs at priority level 3. The top curve is when j=3, the upper middle curve is for j=2, the lower middle curve for j=4, and finally the bottom curve is for when j=1.

2.6.2 QoS Factors in Weighted Fair Scheduling Schemes

In Section 2.5, we looked at the effects of weighted fair scheduling schemes on reservation probabilities, which in turn impacts the QoS received by MHs. What we did works well and is fine for a first-order analysis involving a system with only a few MHs. However, for a system with many MHs, we need to refine our analysis and generalize even further.

In a well-designed system, the number of frames to timeout should vary depending

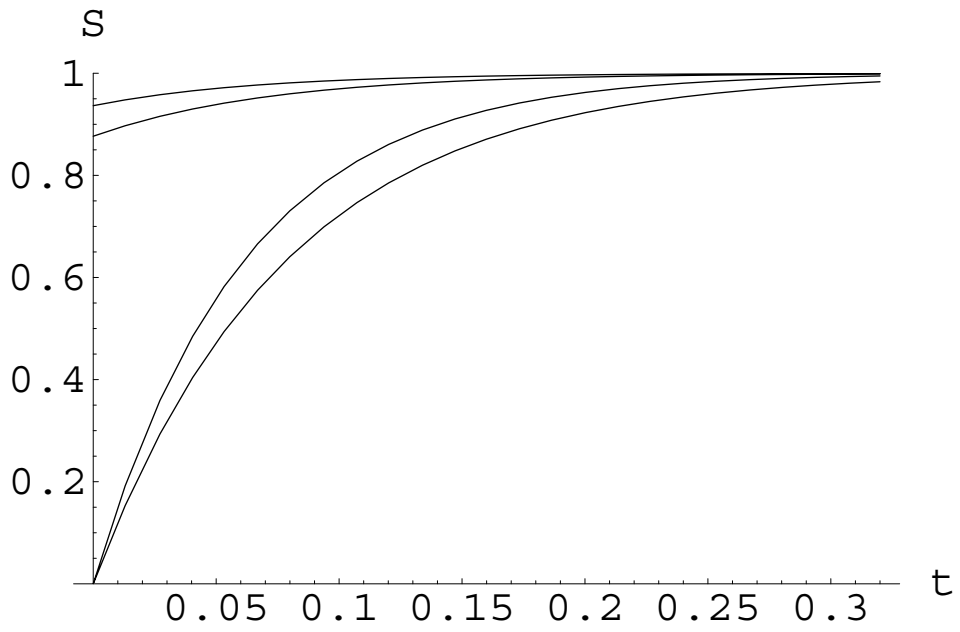


Figure 2.9: Reservation Probability (S) as a Function of the Time (t) for j Number of MHs at Priority Level 3, where there are 5 MHs at Priority Level 1, 3 MHs at Priority Level 2, and 4 MHs at Priority Level 3. $f(p)$=p, $g(b)$=2 where b is Constant, and $H(p)$=$20 - 4p$. p = Priority Level, b = Bit Rate (constant in our case), H = Number of Frames Required to Timeout. Top Curve j=3, Upper Middle Curve j=2, Lower Middle Curve j=4, Bottom Curve j=1

upon the initial weight assigned to the MH. Let us introduce a function $H(w_i)$, which is the number of frames a MH can be in contention for before it times out and w_i is the initial weight assigned to it. $H(w_i)$ must be a strictly monotonically increasing function of the MH's initial weight, and also $\lim_{w_i \to 1} H(w_i) = Z$, where Z is some constant value. In other words, the number of frames before a MH times out should increase as the initial weight assigned to it increases. As mentioned in the previous section, the weights vary from 0 to 1. The limit, Z, for the number of frames, is

imposed so that the number of frames for timing out is always finite.

The probability that a MH of initial weight w_i will not give up due to timing out by the system nor by the user losing patience can be calculated as follows:

$$e^{\frac{-\lambda_{UB} H(w_i) T}{g(b)}}$$

where b is the bit rate requested by the MH, g is a monotonically decreasing function of the bit rate, λ_{UB} is the user block rate without considering the weight of the MH and its bit rate requirement, and H is a monotonically increasing function of the initial weight.

We can then say that the reservation probability of j MHs, where q is the probability of getting a reservation for one data slot per frame, and n_c is the number of MHs in contention can be computed as follows:

$$S(n_c, j, q) = 1 - \binom{n_c}{j} \sum_{k=1}^{n_c} \sum_{a \in C_k, r \in D_k} AB$$

$$A = w_a q^{b_a} e^{\frac{-\lambda_{UB} H(w_{a_i}) T}{g(b_a)}}$$

$$B = L - M + LM$$

$$L = [w_r (1-q)^{b_r}]$$

$$M = [(1 - e^{\frac{-\lambda_{UB} H(w_{r_i}) T}{g(b_r)}})]$$

where $F = \{1, 2, ..., n_c\}$, C_k is the k^{th} distinct subset of F containing j elements,

$D_k = F - C_k$, and w_a is the weight of MH a, and $H(w_{a_i})$ is the number of frames needed for a MH of initial weight w_a to be timed out.

The first term inside the product notation represents the probability that a particular combination of j MHs will be able to succeed in obtaining a reservation. The second term inside the product notation represents the probability that the remaining $(n_c - j)$ MHs will not be able to succeed in making a reservation.

The graph in Figure 2.10 shows the probability of 3 MHs obtaining a reservation, S, if the MHs have initial weights of 0.1, 0.5, and 0.6, as a function of the time since the MHs have been in contention mode where the number of frames required to time out as a function of the initial weight is $H(w) = 16^w$.

2.6.3 A Hybrid Priority and Weighted Fair Scheduling Scheme

A hybrid scheduling scheme could be chosen which combines the ideas of both priority scheduling and weighted fair scheduling. It can also incorporate the design suggestions outlined in the two subsections above. The request corresponding to the MH with the largest priority times weight product value is chosen to be honored or transmitted.

Hybrid scheduling schemes, such as RxW scheduling [4], and RPW scheduling scheme for pull channel [18] have been looked at in a broadcasting context and have been mathematically analyzed. We can however use the ideas inspired from these schemes in our wireless multimedia application and do some mathematical analysis

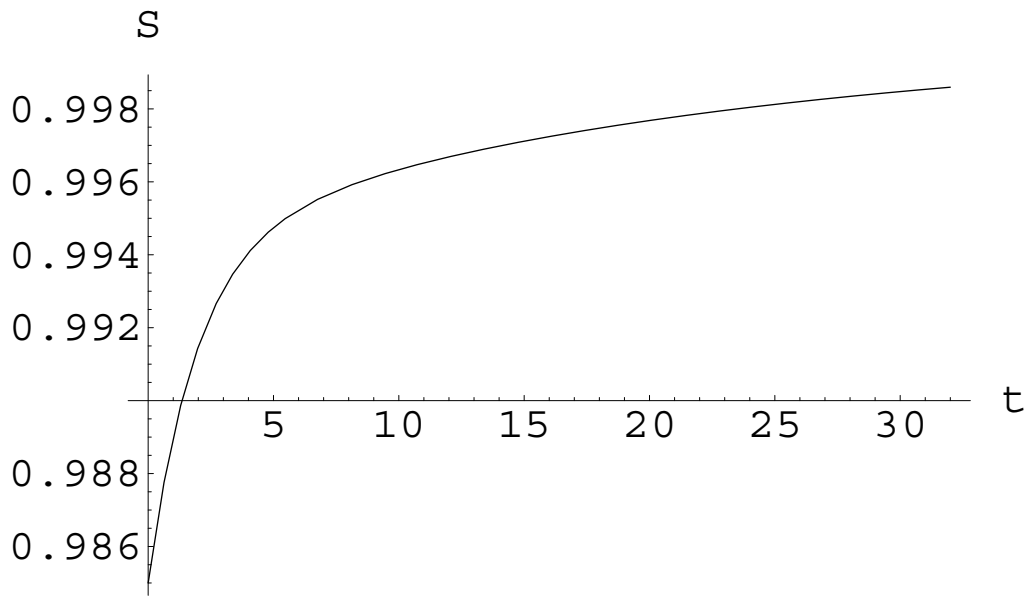


Figure 2.10: Reservation Probability of 3 MHs (S) as a Function of the Time (t), where Initial Weights are 0.1, 0.5, and 0.6 and the Number of Frames Required to Timeout as a Function of the Initial Weight is $H(w)$=16^w

to obtain results which will help us to determine if we are meeting our QoS goals.

In our hybrid scheme the probability of an user not losing patience nor getting timed out is given below:

$$e^{\frac{-\lambda_{UB} Pmin(H_1(p),H_2(w_i))T}{g(b)}}$$

where P is the priority level of the MH, T is the time duration of a frame, b is the bit rate requirement, and g is a strictly monotonically increasing function of the bit rate.

H_1 and H_2 are functions of the MH's priority and initial weight respectively which give the number of frames the MH can be in contention before the system times it out. H_1 is a strictly monotonically decreasing function of the priority value, and H_2 is a strictly monotonically increasing function of the initial weight. The number of time frames for a MH to time out as a function of the MH's priority level can be different from the number of time frames for a MH to time out as a function of the MH's initial weight. Therefore, the number of frames for the MH to time out is the minimum of the two possible values and hence the presence of the min function in the above mathematical expression.

2.7 Stretch-Optimal Scheduling in the Request TDMA/CDMA Protocol

If we have mobile hosts with different bit rates we can use the stretch-optimal algorithm as a possible additional scheduling scheme in deciding which order to have the MHs transmit their data.

The stretch-optimal algorithm [27] can be applied as a scheduling scheme in our Request TDMA/CDMA protocol. Previously, this algorithm has been applied only to broadcasting of items. The concept of stretch was introduced as a way of ensuring fairness for systems with variable data sizes. The stretch is defined to be the quotient of the response time and transmission time. In the stretch-optimal algorithm, the item which has the largest value for the quotient of the number of requests and the

square of the data item size is taken as the one to be transmitted first.

The way that we will apply stretch to our needs is to give the first chance for reservation of data slots to the MH which has the largest $\frac{R}{L^2}$ value, where R is the number of frames our MH has been in contention so far and L is the the length of the message our MH wishes to transmit. The objective is to try to maximize the number of messages we can transmit per time unit.

Let us first introduce $N_{L,R}$ as the probability that a MH in contention mode which wants to transmit a message has a length of at least L and has been in contention mode for at least R frames:

$$N_{L,R} = e^{-\frac{L}{L_m}} e^{\frac{-\lambda_{UB} TR}{g(L)}}$$

where $g(L)$ is a monotonically strictly increasing function of the length (we have assumed as in the original model [11] that the message lengths are exponentially distributed with a mean length of L_m) that is determined from statistical data collection, λ_{UB} is the user block rate, L is the message length, and T is the time duration of a frame.

The first product term represents the probability that the length of the message a MH wishes to transmit is at least L. The second term represents the probability that a MH will be in contention for at least R frames.

Figure 2.11 shows the probability of a MH existing in contention mode wanting to transmit a message of length of at least 10, $N_{10,R}$, versus the mean length of messages, L_m, and the time passed since contention mode began, T, where each frame is of length 32, and where the user block rate is 0.2.

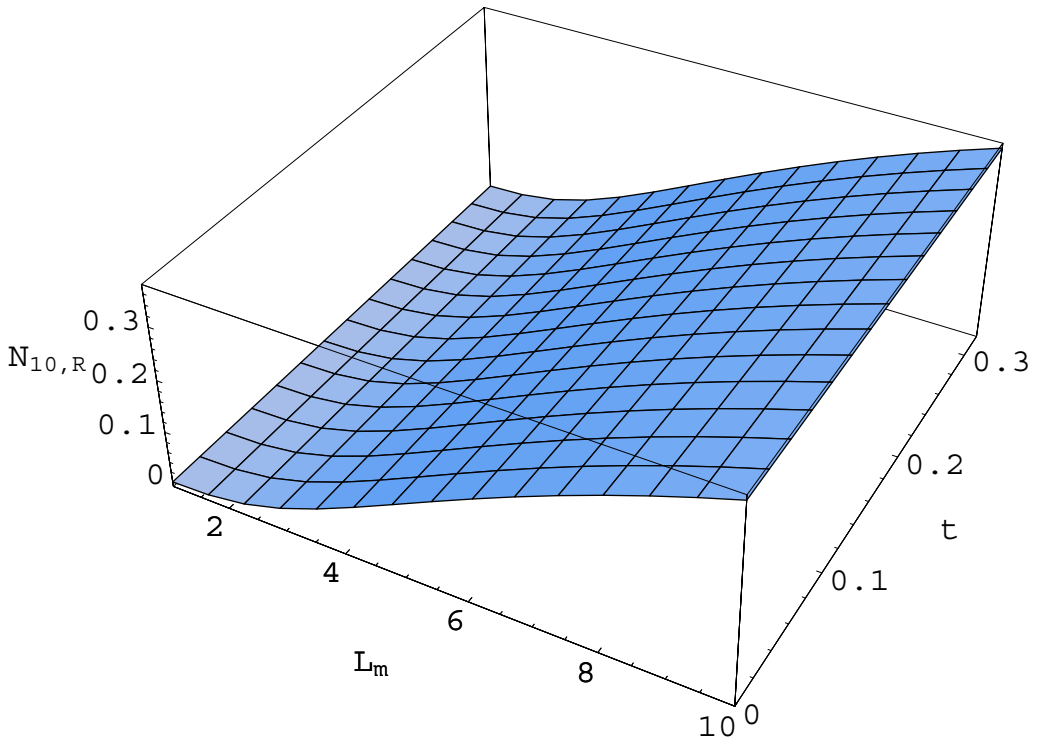


Figure 2.11: Probability of a MH Existing in Contention Mode and Wanting to Transmit a Message of Length at least 10 if in Contention Mode for R frames ($N_{10,R}$) as a Function of the Mean Length of Messages (L_m) and Time since Contention Mode Began (T) where $\lambda_{UB} = 0.2$ and Frame Length = 32

Let us define the tuple

$$LR = ((L_1, R_1), (L_2, R_2), ..., (L_{n_c}, R_{n_c}))$$

where for each i, $\frac{R_i}{L_i^2} \geq \frac{R_{i+1}}{L_{i+1}^2}$. Also, let us associate an integer i for every MH such that the message length the MH wants to transmit and the number of frames it has been in contention so far are L_i and R_i respectively.

We can then say that the probability of j MHs going from contention mode to reserved mode when there are n_c MHs in contention mode and q is the probability of obtaining one data slot to be:

$$\begin{aligned} S(n_c, j, q) &= AB \\ A &= [1 - \textstyle\prod_{i=1}^{j}(1 - e^{\frac{-\lambda_{UB} R_A T}{g(L_i) q^{b_i}}})] \\ B &= [\textstyle\prod_{i=j+1}^{n_c}(1 - e^{\frac{-\lambda_{UB} R_B T}{g(L_i) q^{b_i}}})] \end{aligned}$$

where $R_A = min(R_1, R_2, ..., R_j)$ and $R_B = min(R_{j+1}, R_{j+2}, ..., R_{n_c})$, λ_{UB} is the user block rate, T is the time duration of a frame, b is the bit rate required by the MH, and $g(L_i)$ is a monotonically strictly increasing function of the length which is determined by statistical analysis of data collection.

One measure of how good the performance of our stretch-optimal algorithm is to see how many MHs are able to transmit their message in the subsequent frame and

thus able to be serviced. The number of MHs expected to transmit a message during a time interval, T, the duration of a time frame is computed as shown below:

$$\sum_{j=1}^{n_c} jA_jB_j$$
$$A_j = [1 - \prod_{i=1}^{j}(1 - e^{\frac{-\lambda_{UB}R_AT}{g(L_i)q^{b_i}}})]$$
$$B_j = [\prod_{i=j+1}^{n_c}(1 - e^{\frac{-\lambda_{UB}R_BT}{g(L_i)q^{b_i}}})]$$

where n_c is the number of contention modes, and λ_{UB} is the user block rate.

Finally, we can determine what is the expected time for transmitting all the messages by taking the maximum of all the values given below

$$max(\frac{L_1}{b_1}, \frac{L_2}{b_2}, ..., \frac{L_u}{b_u})$$

where $u = \lfloor \sum_{j=1}^{n_c} jA_jB_j \rfloor$.

2.8 Conclusions

With the advent of wireless and mobile networks, there is a growing interest in designing multiple access schemes to support a variety of multimedia traffic with a high QoS.

The existing analytical model for a request TDMA/CDMA protocol has been generalized in this chapter to take into account QoS factors. We have looked at QoS

factors such as retrials and call blocking due to customer impatience and system timeouts. We have also looked at how the three scheduling schemes can affect the QoS to the MH. We have looked at FIFO scheduling, priority queue scheduling, and weighted fair scheduling. We have also looked at how customer impatience and system timeouts impact the scheduling schemes in terms of QoS. Finally, we looked at applying the stretch-optimal algorithm to our protocol to see what QoS is obtained by doing this. The mathematical results derived will allow us to better meet users' QoS requirements and produce end-to-end guarantees as it allows us to determine the quality of service parameters at every stage of the transmission process.

Although an analytical model has been developed for this protocol, to the best of our knowledge, no one has implemented or researched this model comparing several QoS factors simultaneously with several packet scheduling queuing schemes. In the future, we plan to experimentally study the refined TDMA/CDMA model with QoS adaptations.

Table 2.1: Nomenclature

Symbol	Meaning
MH	mobile host
n_c	number of MHs in contention mode
n_{c_i}	number of MHs in contention mode at priority level i
n_t	number of MHs in thinking mode
j	number of MHs we are dealing with
j_i	number of MHs of priority level i we are dealing with or number of MHs of traffic class i depending on context
$j_{i_{total}}$	total number of MHs of priority level i
λ	message arrival rate
T	time duration of a frame
r	number of retrials
s	number of system timeouts
q	probability of getting a reservation
p	probability of not getting a reservation
q_k	probability of getting a reservation in the k^{th} frame
b_i	bit rate per frame required for the i^{th} MH to go from contention mode to thinking mode
PG_x	probability of MH x generating a message
$PH_x(i)$	probability of MH x succeeding to get a reservation for i slots without giving up or being timed out
A	arrival probability of MHs
S	reservation probability of MHs

CHAPTER 3

PERFORMANCE ANALYSIS OF A HYBRID PUSH-PULL ALGORITHM WITH QoS ADAPTATIONS IN WIRELESS NETWORKS

In this chapter we will first look at a hybrid push-pull algorithm. This hybrid algorithm combines broadcasting of data items which are popular (push data) with dissemination upon request of less popular items (pull data) in asymmetric communication environments. These environments are made up only one database server and many clients. Requests made by the clients are queued up for the pull items. The pull item for which the number of pending requests is the most is the one selected to be pulled. We determine the value of the average expected waiting time for a client analytically and pick a cut-off point between the push and pull items such that the waiting time is minimized. Then we will do a performance analysis of this algorithm and determine the individual response time for each item disseminated and the overall time for the pull queue to be flushed. After that, we will enhance this algorithm by considering multiple data sizes and QoS factors such as multiple priority levels. We will then do some performance analysis on our enhanced algorithm. Knowing what items to broadcast and what not to is of particular importance in wireless networks, due to limited bandwidth, power consumption, and questions of connection reliability.

3.1 Introduction

In this world of information on demand at your fingertips whether it be traffic updates, stock quotes, horoscopes, sports results, or weather information where the clients far outnumber the servers, the traditional client-server approach by itself is no longer efficient. Popular data must be broadcasted while those infrequently needed can be pulled. Developing efficient schemes to do this is a challenge which has to be met. It is especially important in wireless networks, where we have to deal with such issues as limited bandwidth, power consumption, and reliable connections.

An asymmetric communication environment is made up of one server and many clients. There are two ways for a client to receive data. The data could either be given without the client asking for it or it could be explicitly requested by the client. When the data is sent to all the clients at the same time regardless of whether any of the clients had requested that data item or not, this is known as broadcasting. When the data is given to a particular client only if it is requested then that data item is referred to as being pulled.

Our communication environment can either be push-based or pull-based. A push-based environment is one in which the server sends out items to the client without them requesting it. A pull-based environment is one in which all items sent are those explicitly requested by the client. A hybrid system combines these two approaches.

There have been many algorithms in the literature which combine the two approaches. The goal is to minimize both the individual access and response times and overall access and response times or strike some kind of a balance between the two. Access time is how long a client has to wait for a data item to arrive after it starts listening to the broadcast schedule. Response time is how long a client has to wait for a data item to arrive after it explicitly requests it. A broadcast schedule is a specified sequence or order for data delivery of the most popular items.

A hybrid scheme is the best way to approach the problem. A pure broadcast system will broadcast not only frequently requested items, but also those ones that are rarely needed. This will unnecessarily increase the average access time of a data item. A pure pull system on the other hand is also not efficient as well.

The rest of this chapter is organized as follows. Section 3.2 will talk about past related work. Section 3.3 will discuss the hybrid push-pull algorithm and give its pseudocode. Section 3.4 will do a performance analysis on this algorithm. Section 3.5 will give an enhanced hybrid push-pull algorithm. Section 3.6 will give its pseudocode. Section 3.7 will analyze the performance of the enhanced hybrid push-pull algorithm. Finally, Section 3.8 will discuss our conclusions.

3.2 Past Related Work

There are many schemes [1-9, 15, 16, 18, 19, 27] which have been developed to partition the data items into either the push-based set or the pull-based set. However, none exist so far which have looked at all these four factors in combination: multiple data set sizes, quality of service, spurious cases leading to inaccurate portrayals of the system dynamics, and doing implementation without previous knowledge of data access probabilities. Although there have been papers which have looked at some of these factors in isolation, none so far has looked at all four of these factors together.

For example, the RxW algorithm [4] looks at a broadcast system, but looks only at data items of uniform sizes. It has also only been applied to a pure broadcasting system. The stretch-optimal scheduling algorithm [27] does look at non-uniform data item sizes, but it is only for a pure broadcast system. Other algorithms [1-3, 5, 8, 15, 16, 19] have been developed but most of these assumed a priori knowledge of the data item access probabilities. Yet other algorithms are based on static probabilities [6, 7, 9] due to the fact that there is only downward communication from the server to the clients.

Quality of service is another important component. Though looked at in [18], it is only applied to the pull-based case, not to a hybrid system. Even when it is applied to the pull-based case, it only takes into consideration two levels of priority and not multiple levels.

Spurious cases and anomalies have not been looked at in the literature. Suppose we have one client requesting a particular data item 2000 times in the last 30 minutes and we have no other client requesting that particular data item, previous algorithms would place that particular data item in the broadcast cycle due to that one client making the probability of access for that data item high. The outlier creates unfairness. Our algorithm detects the outliers and creates a more fair situation.

The enhanced algorithm (in Section 3.5) will broadcast the most popular data items during a cycle. After broadcasting a single item, the pull queue will be examined and the item which maximizes the product of the overall stretch and time of first request for that item will be selected for dissemination.

3.2.1 Stretch-Optimal Scheduling Algorithm

The stretch-optimal scheduling algorithm [27] based on the LTSF algorithm developed in [3] is used for broadcasting data items of variable size. The enhanced hybrid push-pull algorithm will however with some modifications use it in the pull set. The modification we make is that we will take into consideration the time that the earliest request for a particular data item was made and not just the number of requests that have been made for that item. Another modification we will make is that we will look at the effective number of requests (to be explained later) and not the actual number of requests.

Stretch is a metric which can be used to create fairness in systems which have items of non-uniform data sizes. The stretch is defined to be the response time for an item divided by the time it takes to transmit that item. The transmission time of an item is its size divided by the system bandwidth.

In the LTSF algorithm, we pick the data item for which the sum of all the stretches for unfulfilled requests for that item is the greatest. The stretch optimal algorithm is based on LTSF and is used instead of it because LTSF does not give us the overall stretch value which is optimal. The stretch optimal algorithm also requires less overhead and is simpler to implement. The stretch optimal algorithm picks the data item for which the value of the ratio of the number of unfulfilled requests for it and its size is the greatest. Suppose we have 3 data items whose sizes are $L_1 = 2$, $L_2 = 5$, $L_3 = 7$, and the number of requests made for them are in respective order, $R_1 = 5$, $R_2 = 4$, and $R_3 = 6$. We will then pick data item 1 since its $\frac{R}{L^2}$ value, 1.25 is the most. This will be picked instead of another item, in order to minimize the overall stretch.

3.3 Hybrid Push-Pull Algorithm

In this section we will describe the hybrid push-pull algorithm. The main objective of this algorithm is to pick a cut-off point K which divides the push data items from the pull data items in such a way as to minimize the sum of the average access

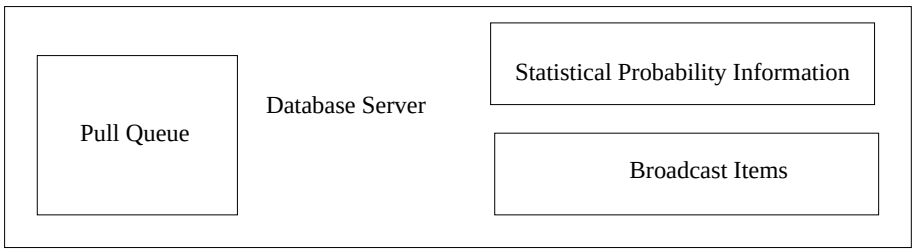
time and average response time. After describing the algorithm we will show its pseudocode. The pseudocode consists of three main parts: a function to determine the cut-off point, a procedure for the actual scheduling of the data items, and finally a procedure for requesting of data items which runs at the client end.

3.3.1 Description of Hybrid Push-Pull Algorithm

Our system is made up of 1 database server holding D data items and c clients. Figure 3.1 shows what our system looks like. K of the D items will be broadcasted and the remaining $(D - K)$ items will form part of the pull set. The foundation of our hybrid algorithm comes from [14, 16, 26]. We will generalize this algorithm even further to consider the four factors mentioned earlier. Before we do that, let's briefly explain what has been done so far in [14, 16, 26].

Access probabilities are assigned based on the popularity of the items. Each item is assumed to be of uniform data size. The access time T_{acc} is the amount of time a client has to spend in waiting for an item to be broadcasted after listening. The response time T_{res} is the amount of time a client spends waiting after it has requested an item from the server. The idea for the push based scheduling is to minimize the access time and the idea of the pull based scheduling is to minimize the response time.

In the hybrid system we try to minimize the combined time. Let $\overline{T_{acc,i}}$ be the



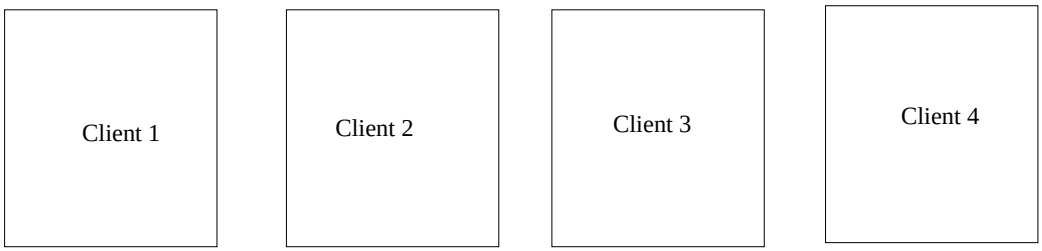


Figure 3.1: Asymmetric Client-Server Architecture

average access time for data item i and $\overline{T_{res,i}}$ be the average response time for data item i. Then the average expected hybrid wait time is defined to be the sum of the average expected access time and average expected response time. In other words, $T_{exp-hyb} = T_{exp-acc} + T_{exp-res} = \sum_{i=1}^{K} P_i * \overline{T_{acc,i}} + \sum_{i=K+1}^{D} P_i * \overline{T_{res,i}}$. The P_i values refer to the access probabilities of the data items. The K most popular items are chosen to be broadcasted and the remaining $(D - K)$ items have to be explicitly requested by the client. The idea is to choose the value of K which will enable us to do things in an efficient manner.

It turns out that $T_{exp-hyb} = \sum_{i=1}^{K} S_i P_i + \sum_{i=K+1}^{D} P_i * (D-K)$, where $S_i = \frac{\sum_{j=1}^{K} \sqrt{\widehat{P_j}}}{\sqrt{\widehat{P_i}}}$. S_i is defined to be the optimal instance space and $\widehat{P_i}$ is defined to be the normalized

probability where $\widehat{P}_i = \frac{P_i}{\sum_{j=1}^{K} P_j}$.

In the hybrid algorithm we try setting K to all values from 1 to D. When $T_{exp-hyb}(K) \geq T_{exp-hyb}(K+1)$ we stop and we have obtained the K value which will minimize the access/request time for the hybrid algorithm. The $T_{exp-hyb}$ values are initialized to D for $K = 0$ and $K = 1$.

The pull scheduling is done in such a way that on average we get no more than 1 request for an item numbered from $(K+1)$ to D during an unit time interval. If we do have more than one item in the pull queue, then we choose the item which has been requested the most. The pull queue is implemented by a max heap.

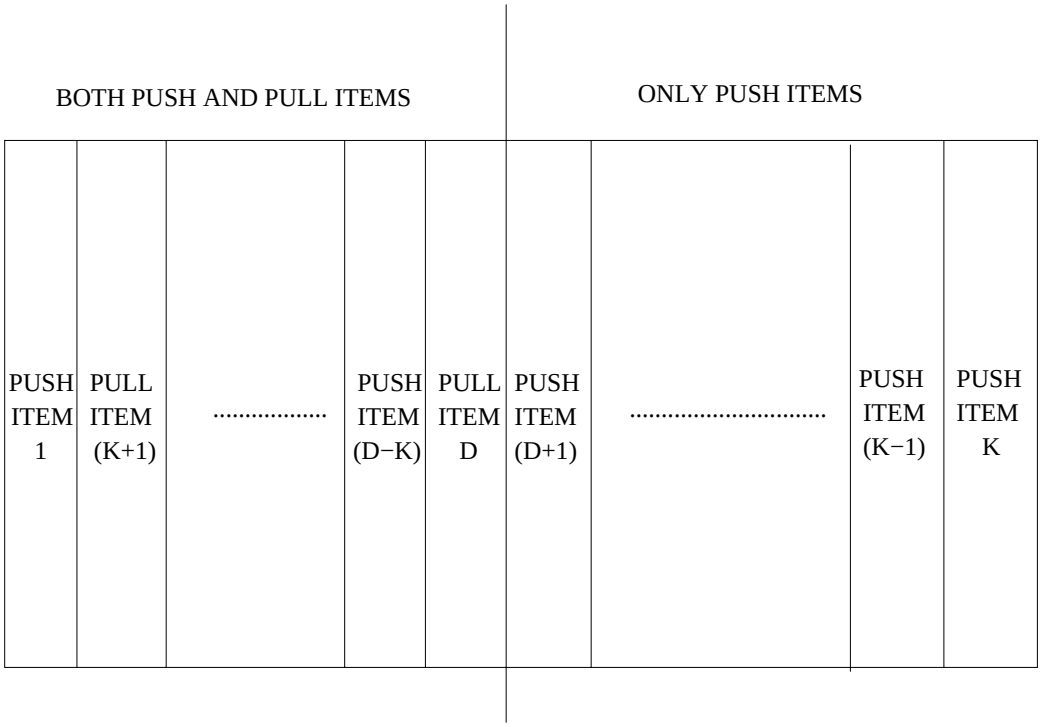


Figure 3.2: Frame Structure of a Broadcast Cycle (All Pull Set Items Requested)

3.3.2 Pseudocode of Hybrid Push-Pull Algorithm

The pseudocode of the hybrid push-pull algorithm is described below.

Integer Function Cut-Off Point (D, P_1, P_2, ... P_D):K

/* The algorithm to determine the cut-off point K between the push and pull items, */

/* is given as below and taken directly from [26]. */

D = Number of Database Items

K = Optimal Cut-Off Point

$K := 1$; $T_{exp-hyb}(0) := T_{exp-hyb}(1) := D$;

while $K \leq D$ and $T_{exp-hyb}(K-1) \geq T_{exp-hyb}(K)$ do

... begin

..... Set $S_i = \frac{\sum_{j=1}^{K} \sqrt{\widehat{P_j}}}{\sqrt{\widehat{P_i}}}$, where $\widehat{P_i} = \frac{P_i}{\sum_{j=1}^{K} P_j}$

..... $T_{exp-hyb}(K) = \sum_{i=1}^{K} S_i P_i + \sum_{i=K+1}^{D} P_i * (D-K)$; $K := K+1$;

... end

return $(K-1)$

Procedure Hybrid Scheduling

/* The algorithm at the server which gives us the hybrid scheduling is given below. */

while true do

... begin

..... compute an item from the push scheduling and broadcast it;

..... if the pull-queue is not empty then extract the most requested item from the pull-queue

..... clear the number of pending requests for that item and pull it

... end

Procedure Client Request (i):

/* Finally the algorithm which runs at the client's side is shown below. */

... begin

..... send sever request for item i

..... wait until listen for i on the channel

... end

3.4 Performance Analysis of Hybrid Push-Pull Algorithm

The performance analysis that has been done in the hybrid push-pull scheduling algorithm [26] has only looked at the expected overall response time, but not at expected response times for individual data items in the pull set. This analysis could be useful in systems where meeting deadlines is crucial or for better establishing QoS criteria. The expected response time for an individual data item is the time we expect to wait on average for that item to be disseminated. We will derive a theorem which gives us the expected response time for item b to be disseminated. Before we do that, let us introduce two lemmas.

Lemma 1 The probability of i requests for data item s during an unit time interval, $Q_{i,s}$, where c is the number of clients in our system, and P_s is the probability of requesting item s during an unit time interval is computed as follows:

$$Q_{i,s} = \binom{c}{i}(P_s)^i(1-P_s)^{c-i}$$

Sketch of Proof: The above is simply an application of Bernoulli trials where P_s is the probability that a client will make a request for data item s during an unit time interval and $(1 - P_s)$ is the probability that the client will not make a request for data item s during that time interval. Q.E.D. $\triangle$

Lemma 2 The probability of pull item b being the a^{th} item to be flushed out of the pull queue during a broadcast cycle, $F_{a,b}$, is determined as follows, where c is the number of clients in our system, is calculated as given below:

$$F_{a,b} = \sum_{m=1}^{(c-a+1)} Q_{m,b} \sum_{n \in T_{i,a,b,m}} R(n,b)$$

where $Q_{m,b}$ is the probability of m requests for data item b and can be calculated as shown in Lemma 1, P_b is the probability of pull item b being requested, $R(i,b) = \frac{Q_{i_1,K+1}Q_{i_2,K+2}...Qi_{D-K,D}}{Q_{i_{b-K,b}}}$, $i = (i_1, i_2, ..., i_{D-K})$ $T_{i,a,b,m} = \{(i_1, i_2, ..., i_{D-K}) - (i_{b-K}) | max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}] < m$, $Index(max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}]) \in Y_{a,b,m} \cup Z_{a,b,m}$, $max^v[(i_1, i_2, ..., i_{D-K}) - i_{b-K}] > m$, $1 \leq v \leq (a-1), 0 \leq i_j \leq c, 1 \leq j \leq (D-K)\}$, where $Y_{a,b,m} = \{u | i_u = max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}], u < (b-K), i_u < m\}$, $Z_{a,b,m} = \{u | i_u = max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}], u > (b-K), i_u \leq m\}$, $Index(i_n) = n$ and where our "-" operator for tuples removes one component from our tuple, $max^a I$ is the a^{th} largest element in I, D is the number of items in our database server, and K is the number of items in the push set.

Sketch of Proof: For those items in the pull set we can determine probability distributions for which order the items are released in. We will determine a general formula for $F_{a,b}$. Let us develop our analysis by first looking at $F_{1,k+1}$.

Before we proceed further, let us define some notations which will make the equations simpler to write.

Let $I = \{(i_1, i_2, ..., i_{D-K-1}) | 0 \leq i_j \leq c, 1 \leq j \leq (D-K-1)\}$. The set I represents all the possible combinations of the number of requests that can be made for each item in the pull set. If $i \in I$, we define $max^a i$ to be the a^{th} largest component of i. Do note that the way we refer to the a^{th} largest value maybe slightly different than the way some people use the term. For example, if we have a tuple (3, 3, 2), the first largest value will be 3 and the second largest value will also be 3 (instead of 2).

So we can say that the probability of item $(K+1)$ being the first one to be flushed out of the pull queue is as follows:

$$F_{1,K+1} = (Q_{1,K+1} \sum_{i \in I, (max^1 i) \leq 1} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D}) +$$

$$(Q_{2,K+1} \sum_{i \in I, (max^1 i) \leq 2} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D}) +$$

$$(Q_{3,K+1} \sum_{i \in I, (max^1 i) \leq 3} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D}) +$$

$$.....+$$

$$(Q_{c,K+1} \sum_{i \in I, (max^1 i) \leq c} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D})$$

The term on the left of the summation symbols represents the probability of having x requests for item $(K+1)$, where x varies from 1 to c, the number of clients. The term on the right of the summation symbols represents the product of the probabilities of having the requests for the other items to be less than or equal to x; this is necessary in order for us to have item $(K+1)$ be the first one to be flushed out.

The following series of computations can determine the probability for item $(K+1)$ to be the second item to be flushed out of the pull queue.

$$F_{2,K+1} = (Q_{1,K+1} \sum_{i \in I, (max^2 i) \leq 1, (max^1 i) > 1} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D}) +$$

$$(Q_{2,K+1} \sum_{i \in I, (max^2 i) \leq 2, (max^1 i) > 2} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D}) +$$

$$(Q_{3,K+1} \sum_{i \in I, (max^2 i) \leq 3, (max^1 i) > 3} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D}) +$$

$$.....+$$

$$(Q_{c-1,K+1} \sum_{i \in I, (max^2 i) \leq (c-1), (max^1 i) > (c-1)} Q_{i_1,K+2} Q_{i_2,K+3} ... Q_{i_{D-K-1},D})$$

The reasoning involved in determining the above equation is the same as before. In addition to having the same conditions as before we also make sure that there has to be one and only one item for which the number of requests must exceed x. Another

difference is that x varies from 1 to $(c-1)$, where again, c is the number of clients in our system. x cannot be equal to c because that would mean the item referred to by the term which is left of the summation symbol would be flushed out first.

If we introduce a few more terms as given below

$$R(i,b) = \frac{Q_{i_1,K+1}Q_{i_2,K+2}...Qi_{D-K,D}}{Q_{i_{b-K},b}}$$

$$T_{i,a,b,m} = \{(i_1, i_2, ..., i_{D-K}) - (i_{b-K}) | max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}] < m,$$

$$Index(max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}]) \in Y_{a,b,m} \cup Z_{a,b,m},$$

$$max^v[(i_1, i_2, ..., i_{D-K}) - i_{b-K}] > m, 1 \leq v \leq (a-1), 0 \leq i_j \leq c, 1 \leq j \leq (D-K)\}$$

where $Y_{a,b,m}$, $Z_{a,b,m}$, and $Index$ are defined as shown below

$$Y_{a,b,m} = \{u | i_u = max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}], u < (b-K), i_u < m\}$$

$$Z_{a,b,m} = \{u | i_u = max^a[(i_1, i_2, ..., i_{D-K}) - i_{b-K}], u > (b-K), i_u \leq m\}$$

$$Index(i_n) = n$$

and where our "-" operator for tuples removes one component from our tuple and also $i = (i_1, i_2, ..., i_{D-K})$, we can then using similar reasoning as above, define the

probability of item b being the a^{th} one to be flushed out as follows:

$$F_{a,b} = \sum_{m=1}^{(c-a+1)} Q_{m,b} \sum_{n \in T_{i,a,b,m}} R(n,b)$$

Q.E.D. $\triangle$

We will now determine a formula for the expected response time for item b to be disseminated.

Theorem 1: The expected response time for item b to be disseminated where $F_{a,b}$ is the probability of pull item b being the a^{th} item to be flushed out of the pull queue, P_b is the probability of item b being requested during an unit time interval, D is the number of data items in our system, and K is the number of items in the push set, is given as follows:

$$\sum_{g=0}^{D-K} [(1-P_b)^g P_b((2D-K-g+\frac{1}{2}) + \sum_{a=0}^{D-K} aF_{a,b})] +$$

$$\sum_{h=D-K+1}^{D} [(1-P_b)^{D-K+h+1} P_b((K-h) + \sum_{a=0}^{D-K} aF_{a,b})]$$

Sketch of Proof: The length of the broadcast cycle is at most $[2(D-K)+K]$ where K is the number of items in the push set and D is the total number of data items. The first term is the number of time units taken for $(D-K)$ items to be broadcasted and $(D-K)$ items to be pulled. The second term is the number of time

units it will take for us to broadcast the remaining K items. So the length of the broadcast cycle is at most $(2D - K)$.

If a request for item b to be disseminated comes just prior to the start of a new broadcast cycle, then the expected time for the item to be disseminated is given as below:

$$\sum_{a=0}^{D-K} aF_{a,b}$$

If the broadcast cycle has just commenced and P_b is the probability of item b being requested by the client, then we can say that the expected response time for item b to be disseminated is as follows:

$$P_b((2D - K - \frac{1}{2}) + \sum_{a=0}^{D-K} aF_{a,b})$$

If the request for item b comes in while the first item is being pulled, then the expected response time for item b to be disseminated is as follows:

$$(1 - P_b)P_b((2D - K - \frac{3}{2}) + \sum_{a=0}^{D-K} aF_{a,b})$$

If the request for item b comes in while the g^{th} item is being pulled then the expected response time for item b to be disseminated is as follows:

$$(1-P_b)^{2g+1}P_b((2D-K-\frac{(2g+1)}{2})+\sum_{a=0}^{D-K}aF_{a,b})$$

If the request for item b comes in when item h is broadcasted, where $h > D-K$, the expected response time for item b to be disseminated is as follows:

$$(1-P_b)^{2(D-K)+1}(1-P_b)^{h-D+K}P_b((2D-K)-(2(D-K)+h)+\sum_{a=0}^{D-K}aF_{a,b})$$

We can simplify this to:

$$(1-P_b)^{D-K+h+1}P_b((K-h)+\sum_{a=0}^{D-K}aF_{a,b})$$

Therefore, the expected response time for item b to be disseminated is given as follows, where $F_{a,b}$ can be computed using Lemma 2.

$$\sum_{g=0}^{D-K}[(1-P_b)^gP_b((2D-K-g+\frac{1}{2})+\sum_{a=0}^{D-K}aF_{a,b})]+$$

$$\sum_{h=D-K+1}^{D}[(1-P_b)^{D-K+h+1}P_b((K-h)+\sum_{a=0}^{D-K}aF_{a,b})]$$

Q.E.D. $\triangle$

In Figure 3.3, we see the response time for a pull item b as a function of the number of items in the push set (K) and the probability of an item b being requested (P_b). As the probability of a pull item being requested increases, the response time for the item increases slightly. This is because the more probable a request for a data item is, the more likely it will be requested towards the beginning of a broadcast cycle, thereby increasing the wait for the next broadcast cycle to start. The start of the next broadcast cycle is when requests made in the previous broadcast cycle are honored. The response time for an item b decreases as we increase the value of the cut-off point K, because there are less items in the pull set queue.

We can also look at the expected overall time for the pull queue to be flushed out. The analysis done previously [26] gives us only an upper bound. We can refine it further. Computing more accurately the expected overall time for the pull queue to be flushed out can help us to better meet QoS requirements.

Theorem 2: The expected overall time for the pull queue to be flushed out at the start of the broadcast cycle is computed as follows, where $H = \{K+1, K+2, ..., D\}$, and P_b is the probability of pull item b being requested during an unit time interval

$$\sum_{i=1}^{D-K} 2i \sum_{a_1 \in H, a_2 \in H-a_1, \ldots a_i \in H-a_1-\ldots-a_{i-1}} P_{a_1} P_{a_2} \ldots P_{a_i} \prod_{m \in H-a_1-\ldots-a_i} (1 - P_m)$$

Sketch of Proof: To compute the expected overall time for the pull queue more

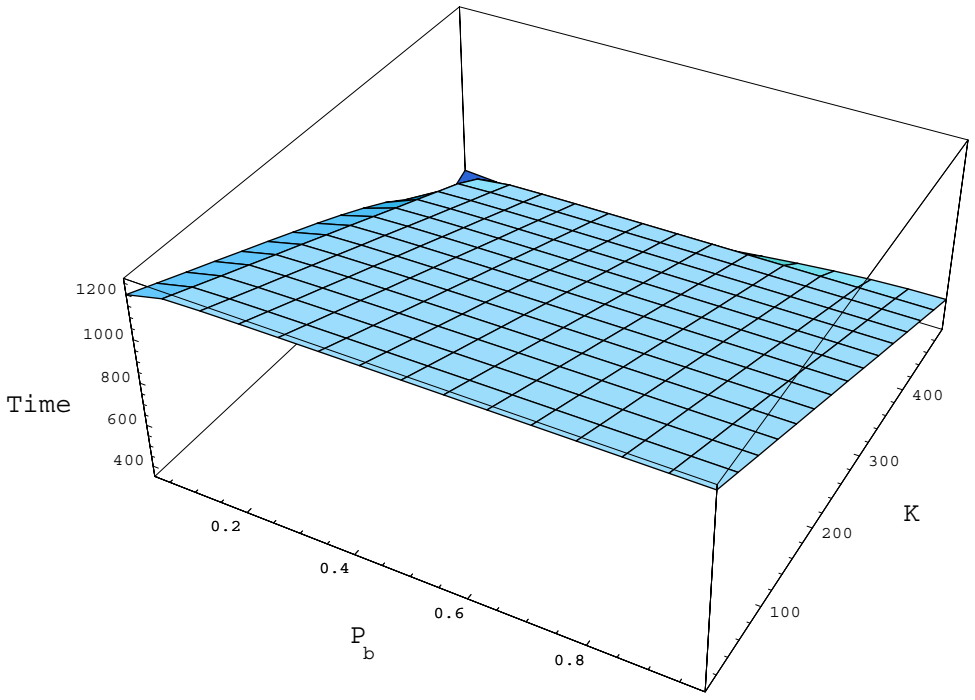


Figure 3.3: Response Time for Pull Item b as a Function of the Number of Items in the Push Set (K) and the Probability of Item b being Requested During an Unit Time Interval (P_b)

accurately, we need to be able to compute the probability of having g items in the pull queue.

The probability of having 0 items in the queue can be computed as follows

$$(1 - P_{K+1})(1 - P_{K+2})...(1 - P_D)$$

Then we can compute the probability of having only 1 item in the queue as such

$$\sum_{h \in H} P_h \prod_{m \in H-h} (1 - P_m)$$

The probability of 2 items in the queue can then be computed as

$$\sum_{h \in H, i \in H-h} P_h P_i \prod_{m \in H-h-i} (1 - P_m)$$

So the probability of having g items in the queue is determined as follows

$$\sum_{a_1 \in H, a_2 \in H-a_1, a_g \in H-a_1-...-a_{g-1}} P_{a_1} P_{a_2} ... P_{a_g} \prod_{m \in H-a_1-...-a_g} (1 - P_m)$$

Thus, the expected overall time for the pull queue to be flushed out at the start of the broadcast cycle is computed as follows:

$$\sum_{i=1}^{D-K} 2i \sum_{a_1 \in H, a_2 \in H-a_1, ... a_i \in H-a_1-...-a_{i-1}} P_{a_1} P_{a_2} ... P_{a_i} \prod_{m \in H-a_1-...-a_i} (1 - P_m)$$

Q.E.D. $\triangle$

3.5 Enhanced Hybrid Push-Pull Algorithm

We have come up with an enhanced hybrid push-pull algorithm which can be used in systems with non-uniform data sizes and in which clients can have multiple priority levels. Furthermore, the enhanced hybrid push-pull algorithm does not require any a priori knowledge of data access probabilities. The enhanced hybrid push-pull algorithm is also able to handle spurious cases as well. For example, if only one normal or low priority client always requests a particular data item, then that item will be placed in the pull set instead of the push set as would likely be done instead with other algorithms. The explanation of the algorithm's pseudocode (given in Section 3.5) is explained below. The reader will find it helpful to look at it while reading the description of the algorithm below.

In procedure client request, whenever a client wants an item, it sends a message to the server identifying itself and which item number it wants.

The server identifies whether the item requested is part of the broadcast cycle or something which needs to be pulled. This is determined by looking at the item number assigned to the item. If we have D items in our database server, they are numbered from 1 to D. We assign the variable K to be the number given to the highest numbered item which is broadcasted. In other words items 1 to K are broadcasted and items $(K + 1)$ to D are pulled. The K value is recomputed at the end of each broadcast cycle. Clients are also numbered from 1 to c, where c is the number of

clients in our system. The numbering of clients is purely arbitrary.

If client j requests an item numbered greater than K, then the request made by client j is added to the pull request queue. We keep a running total of the number of requests made for an item by a client. We also keep track of when the earliest request for an item has been made during a broadcast cycle. In our system, requests made during a broadcast cycle which has already commenced, do not get honored until the next one starts. We alternate between broadcasting an item and pulling an item if there is something that is waiting to be pulled. Another thing that we keep track of is which is the most important or highest priority client that has made a request for item i during a broadcast cycle.

When the system has reached steady-state we check to see whether the total number of requests for item i by client j is an outlier or too much greater than the average number of requests made by all the clients for that item i. If the number of requests made for item i by client j is too much greater than above the average we disregard it in our calculations for the effective number of requests (shown in the enhanced push-pull algorithm with the subscript "eff"). If things do change later on, and it turns out that what was considered to be too much greater than the average is no longer so, then we include it in our calculations. The $B_{i,j}$ Boolean values indicate to us whether or not to include it in our calculations.

What constitutes as being "too much greater than the average" depends on the

priority value assigned to the client. The higher the priority a client is, the more the number of requests it can make for a particular item before it is considered too far above the average. We define a strictly monotonically increasing function, α, which is a function of the client's priority value. This α value tells us how many multiples of the standard deviation for the average number of requests made for item i we can stray before being considered as going too far above the average. The rationale for this is that for example, if a data item is requested 300 times by a certain client and the other clients seldom request that same item and also if that certain client is important enough, we will include the 300 requests as part of the probability calculations and possibly include the corresponding item in the push set as well. However, if it turns out that certain client is not important enough, then we will not consider the 300 requests made as part of our probability calculations and very likely instead include that item in the pull set.

At the end of every broadcast cycle, the number of requests made for an item by a client is reinitialized to its current value minus the value it had T time units ago. The pre-defined time interval, T, that we choose should not be too small nor too large. If it is too large, then we are not able to properly capture the current state of what is going on. On the other hand, if T is too small, then we will not be able to weed out spurious cases properly.

The scheduler broadcasts each item in descending order of effective access probabilities. We say effective access probabilities instead of just access probabilities because a request made by a client is not always counted as one request. It could be counted as two or one-half requests for example, depending upon if the client is a higher than normal priority client or if it is a lower than normal priority client. The client priorities assigned, w, are positive values. A normal priority client is assigned a w value of 1. Important or high priority clients are assigned w values greater than 1. Low priority clients are assigned values greater than 0 but less than 1. In Figure 3.4 we see that six clients with different priority levels, indicated by their w values, requesting either data item 1, 2, or 3. From the figure we can calculate $R_{eff_1} = 0.5 + 1 + 1 = 2.5$, $R_{eff_2} = 1 + 2 = 3$, and $R_{eff_3} = 0.5$.

After we broadcast an item, we check to see if there are any items in the pull queue. We pull the item which has the largest modified optimal stretch value. The optimal stretch value is the number of effective requests made for an item divided by the square of its length. The modified optimal stretch value is the product of the optimal stretch value and the product of the time that the earliest request was made for that item. In the past, the stretch optimal algorithm has been applied only to the push set, but now we are applying it to the pull set and we are also looking at the earliest request made for that item (not just the number of effective requests which have been made). If there is a tie, then we break the tie by pulling the item requested

by the highest priority client. If there is still a tie, then we break the tie by pulling the item which was requested the earliest. If again we still have a tie, then we pull the item which has been numbered with the smaller value. There is no possibility of further ties as each item is distinctly numbered.

After a broadcast cycle has completed, we recalculate the item access probabilities, the normalized item access probabilities, and also the optimal instance space between the broadcasted data items. We then determine the cut-off point between the items to be broadcasted and the items which have to be pulled. The constraint imposed is that we have to make sure that the expected aggregate length of all the pull items which need to be disseminated during our broadcast cycle is less than or equal to the average length of a broadcasted item.

3.6 Pseudocode of the Enhanced Hybrid Push-Pull Algorithm

In Table 3.1 we show the notation used throughout the enhanced hybrid push-pull algorithm.

Given below is the general outline of the pseudocode of the enhanced hybrid push-pull algorithm.

PROCEDURE Initialize-Values [

...$K = \frac{D}{2}$

...$R_{eff_{TOTAL}} = 0$

Table 3.1: Nomenclature

Symbol	Meaning
N	Number of Clients in System
D	Number of Data Items in Server Database
K	cut-off point between push and pull data
T	a predefined time interval or quantum we select
P_i	Probability of Client Requesting Item i in 1 Time Unit
$R_{i,j}$	Number of Requests for Item i by Client j
$\overline{R_i}$	The average number of requests made for item i by a client in time period ($CurrentTime$ - T)
R_{eff_i}	The Effective Number of Requests for Data Item i
$R_{eff_{total}}$	The Effective Number of Total Requests for the Data Items
W_j	Priority or Weight Given to Client j
L_i	The Length of Data Item i
i	Refers to data item
j	Refers to client number
c	Refers to total number of clients in the system
α	A function of client priority
$B_{i,j}$	A boolean variable which tells us whether or not to include $R_{i,j}$ in the calculations for R_{eff_i}
$Time[i]$	Contains the time that earliest request was made for item i
$HighestPriority[i]$	Contains the number of the highest priority client which requested item i

...for $i = 1$ to D [

.....$Time[i]$ = "Negative-Infinity"

.....$HighestPriority[i]$ = "Negative-Infinity"

.....$P_i = \frac{1}{D}$

.....$R_{eff_i} = 0$

.....for $j = 1$ to c

.....$R_{i,j} = 0$

...]

]

PROCEDURE Client-Request [

...Request (i, j);

]

PROCEDURE Request (i, j) [

...if ($i > K$)

.....then add-pull-queue (j)

...$R_{i,j} = R_{i,j} + 1$

...if ($Time[i]$ == "Negative Infinity")

.....then $Time[i] = CurrentTime$

...if ($w[j] > HighestPriority[i]$)

.....then $HighestPriority[i] = j$

...if (not (steady-state))

.....then $B_{i,j} = 1$

.....else if (out-of-range (i, j, $R_{i,j}$))

.....then $B_{i,j} = 0$

.....else [$B_{i,j} = 1$

.....$R_{eff_i} = \sum_{j=1}^{c} R_{i,j} w_j B_{i,j}$

.....$R_{eff_{total}} = \sum_{i=1}^{D} R_{eff_i}$

.....]

]

PROCEDURE Scheduler [

...for $item = 1$ to $num.items.in.broadcast.cycle$

.....Broadcast (next broadcast item)

.....if (not (empty(pull-queue)))

.....then [

.....pull(item with largest $\frac{R_{eff_i} Time[i]}{{L_i}^2}$)

.....if TIE

.....then [

.....pull (item with largest $HighestPriority[i]$ value)

.....if TIE

.....then [

.....pull (item with smallest $Time[i]$)

.....if TIE

.....then pull (item with smallest index)

.....]

.....]

.....]

...for $i = 1$ to D [

.....$P_i = \frac{R_{eff_i}}{R_{eff_{total}}}$

.....$\widehat{P_i} = \frac{L_i Pi}{\sum_{j=1}^{K} L_j P_j}$

.....$S_i = \frac{\sum_{j=1}^{K} \sqrt{\widehat{P_j}}}{\sqrt{\widehat{P_i}}}$

...]

...find K which minimizes $\sum_{i=1}^{K} S_i L_i P_i + \sum_{i=K+1}^{D} L_i P_i$ subject to $\sum_{i=K+1}^{D} L_i P_i \leq \frac{1}{K} \sum_{i=1}^{K} L_i$

...Renumber i values in descending order of P_i

.....if TIE

.....then [

.....assign(i values in descending order of $HighestPriority$)

.....if TIE

.....then [

.....assign(i values in ascending order of $Time$)

.....if TIE

.....then randomly decide

.....]

.....]

.....]

.../* Reset at end of every broadcast cycle */

...for $i = 1$ to D [

.....$R_{i,j} = R_{i,j}(CurrentTime) - R_{i,j}(CurrentTime - T)$

.....$Time[i] =$ "$NegativeInfinity$"

.....$HighestPriority[i] =$ "$NegativeInfinity$"

...]

]

PROCEDURE out-of-range ($number$, i, j) [

...if $(number + w_j) \geq (\overline{R_i} + \alpha(w_j)\ \sigma_{R_i})$

.....then return TRUE

.....else return FALSE

]

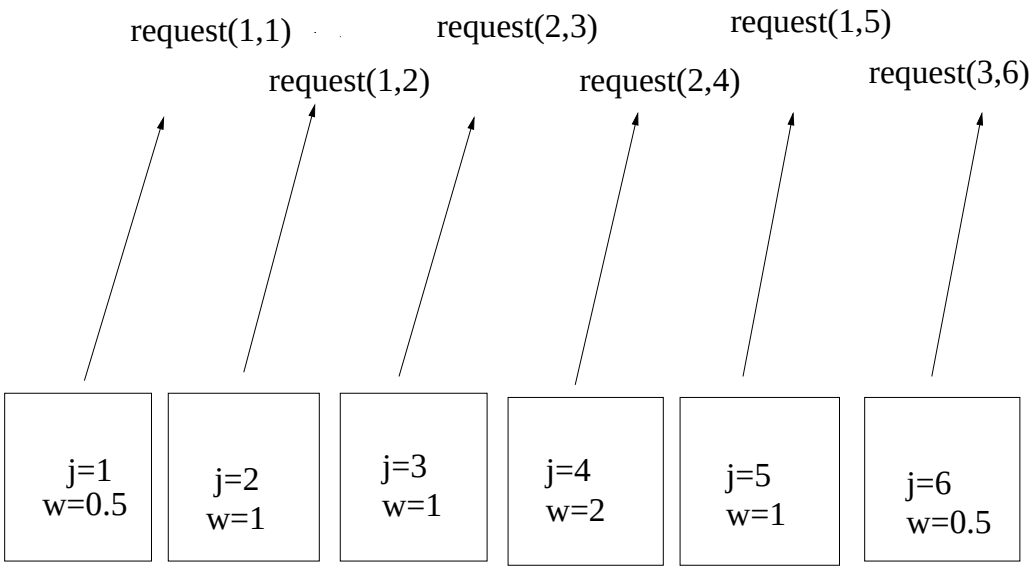


Figure 3.4: Multiple Priority Level Clients Requesting Data Items

3.7 Performance Analysis of the Enhanced Hybrid Push-Pull Algorithm

Let us now analyze the performance of the enhanced hybrid push-pull algorithm. In the enhanced algorithm we also consider differing priority values for clients and non-uniform data sizes. This certainly makes the mathematical analysis far more complex. However, we will make things much simpler by using the mathematical analysis done in Section 3.4 as a foundation from which to build upon. Finding out the response times for individual data items may help us to meet deadlines and also achieve QoS requirements. Once again, the expected response time for an individual data item is the time we expect to wait on average for that item to be disseminated.

Having a system where clients have different priority levels instead of having a system where each client is of the same importance will make achieving quality of

service requirements far easier. Clients who want higher priority levels can subscribe to premium services. Those clients who need to meet some deadlines can be assigned higher priority levels, so that each request for a pull item is counted as more than just one request. By the same token, for those clients where time is not so crucial, can be assigned lower priority levels. In most practical systems, the sizes of data items will vary considerably. Our mathematical analysis takes this into consideration.

Let us define the effective number of clients, $c_{eff} = \sum_{i=1}^{c} w_i$. For example, if there are three clients, with priorities 1 (normal), 2 (high), and 0.5 (low), then the effective number of clients is 3.5

We will soon introduce a theorem which allows us to compute the expected response time for dissemination of item b. Before we do that, we will introduce two lemmas.

Lemma 3 The probability of i effective requests for item j, $U_{i,j}$, where item j belongs to the pull set and c is the number of clients in our system is calculated as such:

$$U_{a,b} = \sum_{i=1}^{c} W_{i,a} Q_{i,b}$$

where $Q_{a,b}$ is the probability of i requests for item b and is calculated as shown in Lemma 1, $W_{i,a}$ is the probability that the sum of i distinct elements taken from W

is equal to a, where $W = \{w_j | w_j \geq 0, 1 \leq j \leq c\}$.

Sketch of Proof: We can start by saying that the probability of receiving 1 effective request for item number $(K+1)$, where c is the total number of clients in the system can be determined as follows:

$$U_{1,K+1} = W_{1,1}Q_{1,K+1} + W_{2,1}Q_{2,K+1} + ... + W_{c,1}Q_{c,K+1}$$

where $Q_{a,b}$ is calculated as shown in Lemma 1 and is the probability of a number of requests made for item b, $W = \{w_j | w_j \geq 0, 1 \leq j \leq c\}$, $W_{i,a}$ is the probability that the sum of i distinct elements taken from W is equal to a and w_j is the priority level of client j.

We can generalize and say,

$$U_{a,b} = \sum_{i=1}^{c} W_{i,a}Q_{i,b}$$

Q.E.D. $\triangle$

Lemma 4 The probability that item b will be the a^{th} one to be flushed out, $G_{a,b}$, in a system with different priority levels for clients is computed as follows:

$$G_{a,b} = \sum_{m=max^c A, m \in A}^{c_{eff}} U_{m,b} \sum_{n \in X_{i,a,b,m}} Y(n,b)$$

where $U_{m,b}$ is the probability of m effective requests for item b, and is calculated as shown in Lemma 3, $Y(i,b) = \frac{U_{i_1,K+1}U_{i_2,K+2}...Ui_{D-K,D}}{U_{i_{b-K},b}}$, $X_{i,a,b,m} = \{(i_1, i_2, ..., i_{D-K}) - (i_{b-K})|max^a[(m_{K+1}, m_{K+2}, ..., m_D) - m_b] < \frac{mTime[b]}{{L_b}^2}$, $max^v[(m_{K+1}, m_{K+2}, ..., m_D) - m_b] > \frac{mTime[b]}{{L_b}^2}, 1 \leq v \leq (a-1), 0 \leq i_j \leq c_{eff}, 1 \leq j \leq (D-K)\}$, $max^v M$ is the v^{th} largest element in the set M, c is the number of clients, $c_{eff} = \sum_{i=1}^{c} w_i$, $Time[b]$ is defined to be the earliest recorded time for which item b was requested, $i = (i_1, i_2, ..., i_{D-K})$, $m = (m_{K+1}, m_{K+2}, ..., m_{K+D})$, $m_n = \frac{i_{n+k}Time[n]}{{L_{n+k}}^2}$, $A = \{\sum_{j=1}^{c} a_j w_j | 1 \leq j \leq c, a_j \in Z\}$, Z is the set of nonnegative integers, L_b is the length of item b, D is the number of items in the database server, and K is the number of items in the push set.

Sketch of Proof: The reasoning is similar to the reasoning used to derive the proof for Lemma 2. Q.E.D. $\triangle$

Now that we have a general formula for $G_{a,b}$ let us use it to develop the expected response time for item b, where item b is an item in the pull set. The analysis is more complex than what we had for Theorem 1, but follows a similar train of thought.

Theorem 3: The expected response time for item b to be disseminated is given as follows:

$$\sum_{g=1}^{D-K} (P_b(1-P_b)^{L_p g}(2DL_p - \sum_{i=1}^{g} L_p - \frac{gL_a}{2} + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))+$$

$$\sum_{h=D-K+1}^{D} (P_b(1-P_b)^{L_p(h-D+K)}(DL_p - \sum_{i=1}^{h-1} L_p + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))$$

where D is the number of items in the database server, K is the number of items in the push set, P_b is the probability of pull item b being requested by a client during an unit time interval, L_p is the average length of an item in the pull set, L_a is the average length of all the items, L_b is the length of pull item b, and $G_{a,b}$ is the probability that pull item b will be the a^{th} one to be flushed out in a system with non-uniform sized data items

Sketch of Proof: Suppose a request for pull item b is given just after the start of a new broadcast cycle. We can then say that the expected response time for item b can be calculated as such

$$P_b(2DL_p + \sum_{i=0}^{D-K} iG_{i,b}(L_b + L_a))$$

where L_a is the average length of all the items in the database server, L_p is the average length of an item in the pull set, D is the number of items in the database server, K is the number of items in the push set, L_b is the length of pull item b, and $G_{a,b}$ is the probability that item b will be the a^{th} one to be flushed out of the pull set

Continuing down this road of analysis, we can say that the expected time taken for a request for item b to be satisfied soon after the first item is pulled to be as follows:

$$P_b(1-P_b)(2DL_p - L_1 - \frac{L_a}{2} + \sum_{i=0}^{D-K} iG_{i,b}(L_b + L_a))$$

Therefore, the expected time for a request for item b to be fulfilled when the request is made after the final possible item is pulled is given below as

$$\sum_{g=1}^{D-K} (P_b(1-P_b)^{L_p g}(2DL_p - \sum_{i=1}^{g} L_p - \frac{gL_a}{2} + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))$$

So the expected response time for item b to be disseminated after the w^{th} item in the push set has been broadcasted, where $w > K$, is given below as

$$\sum_{g=1}^{D-K} (P_b(1-P_b)^{L_p g}(2DL_p - \sum_{i=1}^{g} L_p - \frac{gL_a}{2} + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))+$$
$$\sum_{h=D-K+1}^{w} (P_b(1-P_b)^{L_p(h-D+K)}(DL_p - \sum_{i=1}^{h-1} L_p + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))$$

Therefore, we can conclude that the expected response time for item b to be disseminated is as follows:

$$\sum_{g=1}^{D-K} (P_b(1-P_b)^{L_p g}(2DL_p - \sum_{i=1}^{g} L_p - \frac{gL_a}{2} + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))+$$
$$\sum_{h=D-K+1}^{D} (P_b(1-P_b)^{L_p(h-D+K)}(DL_p - \sum_{i=1}^{h-1} L_p + \sum_{i=0}^{D-K} (i(L_a + L_b)G_{i,b})))$$

where $G_{i,b}$ is computed using Lemma 3. Q.E.D. $\triangle$

Figure 3.5 shows the response time for item 5 after the previous broadcast cycle

has ended as a function of the average data item size, L_A, and the number of items in the push set, K, when the probability of item 5 being requested during an unit time interval, P_b is 0.5, the total number of items in the database server both push and pull, D, is 500, and the average length of a data item in the pull set is L_P. We can see that as the average data item size increases the response time also increases. This is because it generally takes longer for the previous items to be sent out if the average data size of an item is larger. Similarly, we can see that, as the number of items in the push set, K, increases, there are less items in the pull set to be flushed out, so the response time will decrease.

Let us now determine the expected overall time for the pull queue to be flushed out for the general case. The analysis is quite similar to what we did in Theorem 2.

Theorem 4 The expected overall time for the pull queue to be flushed when we have non-uniform data sizes, where L_x is the length of item x and P_x is the probability of item x being requested, and $H = \{K+1, ..., D\}$ is computed as follows:

$$\sum_{i=1}^{D-K} i \sum_{a_1 \in H, a_2 \in H-a_1, ... a_i \in H-a_1-...-a_{i-1}} (L_{a_1} + ...L_{a_i}) P_{a_1} P_{a_2} ... P_{a_i} \prod_{m \in H-a_1-...-a_i} (1 - P_m)$$

Sketch of Proof: From Theorem 2 we know that the expected overall time for the pull queue with all items of uniform size, namely unity, to be flushed out is:

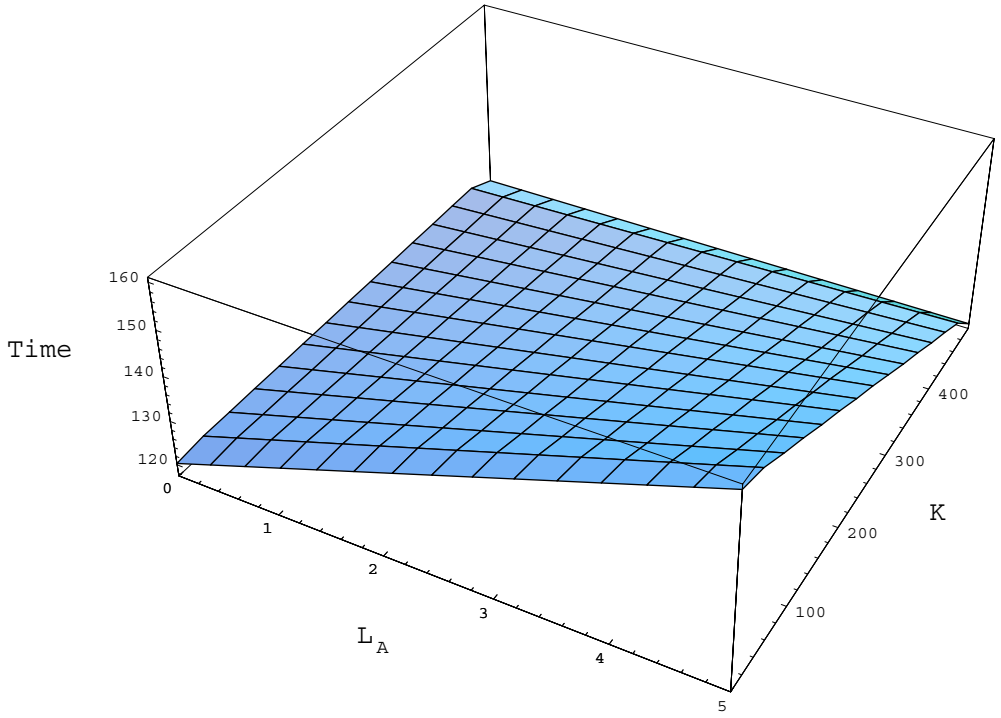


Figure 3.5: Response Time for Item 5 after Previous Broadcast Cycle Ends As a Function of Average Data Item Size (L_A) and Number of Items in Push Set (K) where $P_b = 0.5$ (probability of item 5 being requested during an unit time interval), $L_P = 5$ (the average length of an item in the pull set), and $D = 500$ (number of items in database server)

$$\sum_{i=1}^{D-K} i \sum_{a_1 \in H, a_2 \in H-a_1, \ldots a_K \in H-a_1-\ldots-a_{K-1}} P_{a_1} P_{a_2} \ldots P_{a_K} \prod_{m \in H-a_1-\ldots-a_K} (1 - P_m)$$

From this we can determine that the expected overall time for the pull queue to be flushed is given as below when we have non-uniform data sizes. The L_x values

after the second summation symbol below, refer to the different possible aggregate lengths of the items requested to be disseminated.

$$\sum_{i=1}^{D-K} i \sum_{a_1 \in H, a_2 \in H-a_1, \ldots a_i \in H-a_1-\ldots-a_{i-1}} (L_{a_1} + \ldots L_{a_i}) P_{a_1} P_{a_2} \ldots P_{a_i} \prod_{m \in H-a_1-\ldots-a_i} (1 - P_m)$$

Q.E.D. $\triangle$

3.8 Conclusions

A generalized hybrid scheduling algorithm has been developed for an asymmetric communication environment, one which consists of one server and multiple clients. The enhanced hybrid push-pull algorithm has taken into consideration multiple data sizes, QoS criteria such as priority levels, computing data access probabilities on the fly, and handling aberrant or stray situations which can lead to distorted probability calculations. Our scheme makes sure that these stray situations will be taken care of by the pull set instead of the push set. Ours is the first hybrid scheduling algorithm to effectively handle all four of these factors simultaneously. We extended the performance analysis on the existing algorithm by looking at the response time for individual data items in the pull set, as well as fine tuning the analysis for the expected overall time for the pull queue to be flushed out. Finally, we did the same

performance evaluation on the enhanced hybrid push-pull algorithm. The performance analysis will help us to better meet QoS requirements and also be of use in systems with deadlines.

CHAPTER 4

CONCLUSIONS

With the fast growth and pervasiveness of wireless networks combined with ever more increasingly sophisticated multimedia applications running on them, there is a growing need for being able to determine what kind of quality of service customers are receiving and introducing various qualities of service to them. The restrictions and constraints imposed in a wireless environment require that schemes and algorithms developed in order to meet quality of service requirements be highly efficient. The contributions of this thesis are outlined below.

- Developed an analytical model which allowed us to study the performance of a hybrid CDMA/TDMA protocol with QoS adaptation. In this model, we have also investigated several scheduling schemes to study how they impacted the quality of service received by the customers.

- Developed a refined hybrid push-pull algorithm for wireless networks in asymmetric communication environments. The hybrid push-pull algorithm was developed to be able to handle non-uniform data sizes and multiple levels of priority. We also proposed an analytical model to study its performance.

4.1 Future Directions

Although generalized analytical models have been developed, work in implementing and simulating what has been done in this thesis is still needed. The models developed in this thesis are quite general enough to be of relevance for a long time to come. Pricing and economic strategies need to be determined as well in deciding who to provide what kind of quality of service. With applications such as virtual reality looming in the horizon and the ever increasing demand for communication anywhere and anytime, the importance of analyzing quality of service in detail will be even more critical in the future.

BIBLIOGRAPHY

[1] Acharya, S. and Franklin, M. and Zdonik, S., "Dissemination-Based Data Delivery using Broadcast Disks", *IEEE Personal Communications*, pp. 50-60, Dec 1995.

[2] Acharya, S. and Franklin, M. and Zdonik, S., "Prefetching from a Broadcast Disk", *Proceedings of 12th International Conference on Data Engineering*, pp. 276-285, Feb 1996.

[3] Acharya, S. and Muthukrishnan, S., "Scheduling on-demand broadcasts: New metrics and algorithms", *Proceedings of the Fourth Annual ACM/IEEE Mobicom*, pp. 43-54, 1998.

[4] Aksoy, D. and Franklin, M., "RxW: A Scheduling Approach for Large-Scale On-Demand Data Broadcast", *IEEE/ACM Transactions on Networking*, Volume 7, Number 6, pp. 846-860, Dec 1999.

[5] Bar-Noy, A. and Patt-Shamir, B. and Ziper, I., "Broadcast Disks with Polynomial Cost Functions", *IEEE INFOCOM 2000*, pp. 575-584, 2000.

[6] Baruah, S. and Bestavros, A., "Pinwheel Scheduling for Fault-Tolerant Broadcast Disks in Real-Time Database Systems", *Proceedings of IEEE International Conference on Data Engineering*, pp. 543-551, Apr 1997.

[7] Baruah, S. and Bestavros, A., "Real-Time Mutable Broadcast Disks", *Proceedings of Second International Workshop on Real-Time Databases*, 1997.

[8] Bennett, J. C. R. and Zhang, H., "Hierarchical Packet Fair Queueing Algorithms.", *Proceedings of ACM SIGMOD International Conference*, pp. 1-12, 1994.

[9] Bestavros, A., "Aida-Based Real-Time Fault-Tolerant Broadcast Disks", *Proceedings of the Second IEEE Real-Time Technology and Applications Symposium*, 1996.

[10] Boukerche, A., and Dash, T., "A Performance Analysis of a Refined Hybrid TDMA/CDMA Protocol for Wireless Networks with QoS Adaptations", *Proceedings of the 22nd IEEE International Performance Computing Communications Conference*, pp. 369-374, Apr 2003.

[11] Chatterjee, M. and Das, S. K., "Performance Evaluation of a Request-TDMA/CDMA Protocol for Wireless Networks", *Journal of Interconnection Networks*, Volume 2, Number 1, pp. 49-67, 2001.

[12] Demers, A. and Shenker, S., "Analysis and Simulation of a Fair Queuing Algorithm", *IEEE Proceedings of SIGCOM*, pp. 1-12, 1989.

[13] Gralla, P., *"How Wireless Works"*, Pearson Education, pp. 80-81, Sep 2001.

[14] Guo, Y. and Pinotti, M. C. and Das, S. K., "A New Hybrid Broadcast Scheduling Algorithm for Asymmetric Communication Systems", *ACM SIGMOBILE Mobile Computing and Communications Review*, Volume 5, Number 3, Jul 2001.

[15] Hameed, S. and Vaidya, N. H., "Efficient Algorithms for Scheduling Data Broadcast", *Wireless Networks*, Volume 5, Number 3, pp. 183-193, May 1999.

[16] Hameed, S. and Vaidya, N. H., "Scheduling Data Broadcast in Asymmetric Communication Environments", *Wireless Networks*, Volume 5, Number 3, pp. 171-182, May 1999.

[17] Hiller, F. S. and Lieberman, G. J., *Introduction to Operations Research*, 6th Ed., McGraw-Hill, Inc., 1995.

[18] Hu, J. and Yeung, K. L. and Feng, G. and Leung, K. F., "A Novel Push-and-Pull Hybrid Data Broadcast Scheme for Wireless Information Networks", *IEEE International Conference on Communications*, pp. 1778-1782, Jun 2000.

[19] Jain, R. and Werth, J., "Airdisks and Airraid: Modeling and Scheduling Periodic Wireless Data Broadcast (Extended Abstract)", *DIMACS Technical Report 95-11*, Rutgers University, DIMACS, 1995.

[20] King, P. J. B., *Computer and Communication Systems Performance Modeling*, Prentice Hall International (UK) Ltd., 1990.

[21] Li, W. and Chao, X., "A Novel Network Allocation Scheme", *Proceedings of the 5th ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWIM)*, Sep 2002.

[22] Lohi, M. and Mudani, K., "Prioritisation Queuing of User Services for Multi-Layer Wireless Systems", *Proceedings of the First IEEE International Workshop on Mobility and Wireless Access (MobiWac)*, Oct 2002.

[23] Lu, G., *Communication and Computing for Distributed Multimedia Systems*, Artech House, Boston, 1996.

[24] Motorola, Inc. Cellular Infrastructure Group, *CDMA Technology and Benefits An Introduction to the Benefits of CDMA for Wireless Telephony*, Mar 1996.

[25] Parekh, A. K. and Gallager, R. G., "A Generalized Processor Sharing Approach to Flow Control in Integrated Service Networks: The Single node case", *IEEE/ACM Transactions on Networking*, Volume 4, Number 3, pp. 344-357, 1993.

[26] Pinotti, M. C., "Push Less and Pull the Current Highest Demanded Data Item to Decrease the Waiting Time in Asymmetric Communication Environments", Technical Report in Progress.

[27] Wu, Y. and Cao, G., "Stretch-Optimal Scheduling for On-Demand Data Broadcasts", *IEEE International Conference on Computer Communications and Networks*, pp. 500-504, Oct 2001.