ANALYSIS OF AN INTERACTIVE VIDEO ARCHITECTURE

Thesis by

Petros N. Mouchtaris

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1993

(Defended June 10, 1992)

Acknowledgements

First of all, I would like to thank my advisor Professor Edward C. Posner. During my studies I have questioned my decisions of becoming an Electrical Engineer, coming to Caltech for graduate studies, doing research in traffic theory and many more. The only decision I never questioned was joining Professor Posner's research group. Among other things, I would like to thank him for suggesting the interactive video service that is discussed in this thesis and for his constant advice and encouregement throughout the course of this work.

The financial support of Pacific Bell is gratefully acknowledged. Pacific Bell supported my research at Caltech, provided funds for computer equipment and offered me summer employment in 1991. During my stay at Pacific Bell, I was allowed to use computer equipment for doing my research and this is where a great part of this thesis was written. I am really very thankful to all the people that made that possible and especially to Doctors Yuet Lee, Kuo-Hui Liu and Ning Zhang all of Pacific Bell.

Last but not least, I would like to thank my family in Greece and my friends at Caltech for their moral support. Especially, I would like to thank Nikos Bekiaris, Gamze Erten, Michael Mandell and Isaac Wong. I'm definitely going to miss them.

Abstract

A new residential application for interactive video is proposed. There is a service provider that prepares and distributes daily news programs customized to subscriber interest. The provider assembles the programs from short news clips and uses a profile data base of subscribers for selecting the appropriate clips. The time of viewing the program can be selected by the customers in near-real-time. We model this service and propose a network architecture that can support it. There is a main node that contains most of the storage and sourcing facilities, and an intermediate node to which all customers are connected. Multicasting is used as much as possible for reducing the traffic load on the network. In addition to that, popular material is stored in the intermediate node which is closer to the customers, which further decreases the traffic load.

Our main concern is the time that a customer has to wait until he starts getting his program. This time is a function of the capacity of the link that connects the main node to the intermediate node, the so-called main link. The case that the main link can only transport a single video connection is considered first. We propose a recurrent algorithm that calculates the probabilities of the states and uses them for evaluating the expected wait, and prove that there is a very simple relationship between the expected wait and the probabilities of the states. A simplified analysis that directly computes the expected wait is proposed next. This approach is computationally more efficient but does not give us any information about the probabilities of the states.

For the general case that the main link can transport more than one video connection, we generalize the recurrent algorithm that calculates the probabilities of the states and the simple relationship between the expected wait and the probabilities of the states. For the cases that the complexity of our algorithm is too large, we propose and evaluate three approximate techniques for estimating the expected wait. In the first technique we use the results for the case that a main link can only transport a single connection for estimating the results for the general case. In the second technique we use the idea of rescaling time. In the third, motivated by the fluid-flow theory, we solve a deterministic problem and use the results of that problem for estimating the expected wait for the problem we are interested in. We show that these approximate techniques compare well with simulations. Thus, we can now decide what the capacity of the main link should be so that our system has the desired performance, and we can do that even if the number of customers is very large.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Feasibility of Interactive Video

Fiber technology has been advancing rapidly in the last decade. This has resulted in a significant decrease in the cost of fiber. Today, fiber offers an alternative to the use of copper not only for long distance point to point transmission but also for transmissions up to the customer's end.

The use of fiber has three very important advantages. The first one is that it has low attenuation. This reduces the need for amplification, which usually results in signal distortion. This is very important especially for the cable industry. The second advantage is that the bandwidth of the fiber is almost unlimited. The only limitations, for the time being, are the optical devices and the devices that are used for converting electrical signals to optical and back (see [4]). The third advantage of fiber is its cost efficiency, which has drastically improved. In particular fiber has low maintenance costs. For all these reasons the use of fiber all the way to the customer's end is becoming more and more attractive (see [2, 3, 4, 6]).

At the same time, with advances in fiber technology, new video coding techniques have been developed. Vector quantization (see [5]) and subband coding (see [10]) are the most interesting among them. These new techniques together with the use of motion compensation have made possible transmission of video at unprecedented low

bit-rates. It has been reported that even High Definition TV (HDTV) signals can be transmitted at rates below 50 Mbps (see [1, 8]).

Storage technology has been also advancing rapidly. This is very important for Video on Demand. The reason is that even if coding techniques continue to improve, we will still need transfer rates of the order of 1 Mbps for NTSC video and more than that for HDTV. So, if we want to store a movie that is 100 minutes long, we will need storage of the order of 1 gigabyte (GB). Storing a library of movies will thus require enormous storage capacity. But even if we need to store only small video segments, that will still require a lot of memory. Although we may be able to use existing technology for doing so, we also need to make sure that this technology is cost effective. It has been predicted in [8] that by the year 2000 all the necessary storage equipment will be available, although it is not very clear yet whether their price will be low enough for practical applications.

Another important component of technology for video applications are the switching facilities. New high-bandwidth packet switches have been introduced which are based on Batcher-banyan technology (see [9]). It has been reported that 38.4 Gbps switching fabrics are under construction (see [8]). It has become clear that in the near future switches that can be used for video applications will be available.

In the previous paragraphs we discussed all the technological components that are necessary for the introduction of services that require video signals transmitted up to the customer's end. We have shown that within the next decade or so the technology will exist for applications of this kind (see also [8]). The exact architecture of the networks that will deliver video to the customer are yet unclear. It is possible that in the beginning some hybrid architectures will be used (see for example [2, 7]). It may be the phone or the cable company that will deliver the video.

It seems very likely that Broadband ISDN based on Asynchronous Transfer Mode (ATM) will be the basic ingredient in these architectures. This fact is becoming widely accepted and for that reason most of the researchers are working towards this direction. The basic idea of BISDN based on ATM is that information is divided into

small packets of fixed length, called cells. Cells from different sources are statistically multiplexed and transmitted through the same link. There is no dedicated connection for each source, which makes the introduction of variable bit-rate sources efficient. The reason is that capacity can be used more efficiently. We do not dedicate the maximum needed capacity to a connection for the duration of the connection because this is wasteful whenever the bit-rate of the source is smaller than the maximum. By allowing sources to send cells only if they need to, we can multiplex more sources through the same link and expect that the probability that all sources will need to transmit at their maximum rate will be negligible.

It is evident that video will be delivered to customers in the near feature and for that reason proposals for new architectures that can efficiently deliver video have started appearing in the literature. New applications that require the delivery of video signals have also been proposed. Among these, interactive applications are the ones that pose the hardest requirements on technology because they require switched video, but are clearly the ultimate desired services and we expect that they will become feasible in the near future.

1.2 Outline of Thesis

In Chapter 2 we propose a novel application for interactive video. There is a service provider that prepares and distributes daily news programs customized to subscriber interest. The provider assembles the programs from short news clips and uses a profile data base of subscribers for selecting the appropriate clips. The time of viewing the program can be selected by the customers in near-real-time. We model this service and propose a network architecture that can support it. There is a main node that contains most of the storage and sourcing facilities, and an intermediate node to which all customers are connected. Our main concern is the time that a customer has to wait until he starts getting his program which is a function of the capacity of the link that connects the main node to the intermediate node. We call this link the

main link.

In Chapter 3 we examine the case that the main link can only transport a single video connection. We propose a recurrent algorithm that calculates the probabilities of the states of the system and uses them for evaluating the expected wait. We prove that there is a very simple relationship between the expected wait and the probabilities of the states. We propose an alternative approach that directly computes the expected wait. This approach is computationally more efficient but does not give us any information about the probabilities of the states.

In Chapter 4 we examine the general case that the main link can transport more than one video connection. We generalize the recurrent algorithm proposed in Chapter 3 and also the simple relation that relates the probabilities of the states and the expected wait. We then propose some approximate techniques for estimating the expected wait for the cases that the complexity of our exact algorithm is too large. Thus, we can now decide what the capacity of the main link should be so that our system has the desired performance

Finally, in Chapter 5 we propose some directions for further research. Applications of interactive video are technologically very demanding and therefore we need to study them very carefully before we can actually implement them.

Bibliography

[1] T. C. Chen and K. H. Tzou, "HDTV Coding at below 45 Mbps for Digital Transmission," *GLOBECOM '91*, 3.5, pp. 96-100, December 1991.

[2] J. A. Chiddix and W. S. Ciciora, "Introduction of Optical Fiber Transmission Technology into Existing Cable Television Networks and its Impact on the Consumer Electronic Interface," *IEEE Transactions on Consumer Electronics*, Vol. 35, pp. 51-62, May 1989.

[3] S. A. Esty and D. E. Wolfe, "Cable TV Turns to Fiber for System Upgrades, Rebuilds," *Photonics Spectra*, pp. 79-84, June 1990.

[4] D. Large, "A Comparison of Various Fiber Optic Topologies for Delivery of Entertainment Video to Residencies," *IEEE Transactions on Consumer Electronics*, Vol. 35, pp. 72-80, May 1989.

[5] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Vol. COM-28, pp. 84-95, Jan. 1980.

[6] C. W. Lundgren and P. S. Venkatesan, "Applications of Video on Fiber Cable," *IEEE Communications Magazine,* Vol. 24, pp. 33-49, May 1986.

[7] T. S. Rzeszewski, "A Two Layer Fiber Network for Broadband Integrated Services," *IEEE Transactions on Consumer Electronics*, Vol. 35, pp. 81-85, May 1989.

[8] W. D. Sincoskie, "Video On Demand: Is It Feasible?" *GLOBECOM '90*, 305.3, pp. 201-205, December 1990.

[9] D. R. Spears, "Broadband ISDN Switching Capabilities from a Services Perspective," *IEEE Journal on Sel. Areas in Communications,* Vol. JSAC-5, pp. 1222-1230, October 1987.

[10] J. W. Woods and S. D. O'Neil, "Subband Coding of Images," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. ASSP-34, pp. 1278-1288, October 1986.

Chapter 2

Service Model

A new residential application for interactive video is proposed in this chapter. There is a service provider that prepares and distributes daily news programs customized to subscriber interest. The provider assembles the programs from short news clips and uses a profile data base of subscribers for selecting the appropriate clips. The customers may ask for their service any time in the evening but may have to wait for a few minutes until their programming starts. This is because the network may be able to serve only a subset of the whole customer population at the same time. A subscriber may also provide feedback on the program that he received and the network will try to learn his preferences. We model this service and propose a network architecture that can support it.

2.1 Service Description

Residential applications of real-time video usually distinguish between two types of services: broadcast and interactive (see [9]). In broadcast video, a program is transmitted through the network. There may be many channels available but the only control that a customer has is to select particular channels. After selecting a channel the customer doesn't have any control over the content of the program that is being viewed; all customers that are watching the same channel receive exactly the same

video signal. In interactive video, customers can control the beginning time of the program and have some control over its content. Each customer may receive a different program and for that reason switched video is required. Pay-Per-View service may be viewed as an intermediate service, where customers have modest control over the beginning of the program, since the same program is broadcasted in periodic intervals. Clearly, interactive video is the ultimate desired service and we expect that it will become feasible in the near future (see for example [8]).

We propose a residential application for interactive video (see also [5]). We assume that there is a service provider that prepares and distributes daily news programs customized to subscriber interest. The provider prepares and edits short news clips which we call **segments**. Each segment contains news about a specific event or a general news summary and lasts for about 3 minutes. Segments may have overlapping content and different segments about the same event may exist, because different subscribers may be interested in different aspects of that particular event. There may be supplemental segments that contain backround information. Each segment has some keywords attached to it, not transmitted to the subscriber, that describe the segment's content and demographic properties of potential viewers. The program of each subscriber is assembled from these segments.

The service provider maintains a profile data base of the subscribers. This profile may be based on direct input from the subscriber, on demographics and on learning subscriber preferences via feedback. The service provider uses these profiles and the keywords of the segments for deciding which segments to use for assembling the program of a particular subscriber. A similar idea is proposed in [1, 7]. Information about the status of the network and profiles of other subscribers may also be taken into account when assembling the program of a particular customer. The reason is that if for example it is expected that more than one customer would like to view a particular segment, that segment may be preferred, because the same segment may be multicast to more than one customer. We will talk more about this later on.

The subscriber tunes into the interactive channel some time during the evening.

He is allowed to do that during say a 3 hour time window. At first he gets some general broadcast continuing feed but within a couple of minutes his tailored feed is switched in. The reason for this delay is the fact that the network may be able to serve only a subset of the whole customer population. How long a customer will have to wait on the average will be our measure of performance of a system. Once transmission starts there are no interruptions.

After the tailored program is over, the channel is switched back to broadcast feed. The subscriber may then ask for further background information on the program that he just viewed. The subscriber can also send other feedback about the program. The service provider uses this feedback for learning subscriber preferences. By using this feedback, the service provider can update the profile of the subscriber and increase customer satisfaction and revenue.

2.2 System Architecture

We propose the hierarchical architecture shown in Figure 2.1 for supporting the service that we described in the previous section. This is a rather simple architecture but can be used as the basic building block for designing more complex networks (see for example [9]). The reason for concentrating on this system is that it gives rise to a simple analysis, as will become clear in the following chapters.

As we see in Figure 2.1, there is one node, called the **main node**. This node contains sourcing facilities and storage. We assume that most of the segments are stored in the main node. Customers are connected to the main node through an intermediate node. Each customer is assumed to have some sort of facilities for displaying video signals and sending data for making requests and providing feedback to the service provider.

The intermediate node contains a switch and some more sourcing facilities and storage. The introduction of sourcing facilities and storage in the intermediate node is a very important point. This is what makes our architecture different from the

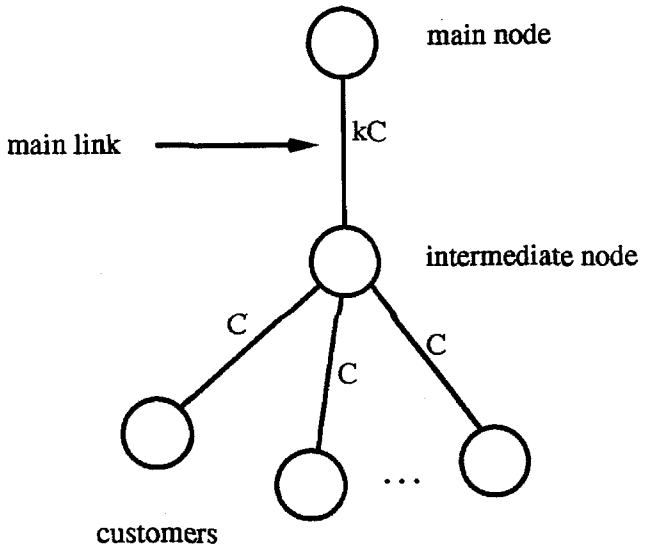

Figure 2.1: A Simple Network

architecture proposed in [9]. The idea of having more than one level for storing material and making popular material easily accessible is also proposed in [3] but in our architecture popular material is also stored closer to the subscribers.

The reason for adding these capabilities to the intermediate node is the following. Let us assume that a customer asks for his program and that in his program there is a segment that the service provider expects that more customers would like to see, for example a segment containing local news. When sending the segment to that customer the service provider arranges for that segment to be also stored in the intermediate node. In that way when the other customers are going to get their program, this segment won't have to be retransmitted from the main node again. This decreases the load on the transmission links that connect the main node to the intermediate node. We also assume that the switch in the intermediate node has multicasting capabilities. So, whenever possible, the same segment is multicast to more than one customer, which further decreases the load on the storage and the delay.

The problem of deciding which segments are going to be stored in the main node

and which are going to be stored in the intermediate node is a difficult problem, especially when the number of segments is large which is the case in the problem that we are considering. We will not try to solve this problem. One approach for solving this problem has been proposed in [6].

The link that connects the main node to the intermediate node is called the main link. This link may be able to transport more than one video connection at the same time. Deciding what the actual capacity of this link should be so that the system has the desired performance is a difficult question which we will answer in the following chapters.

The same architecture is shown in Figure 2.2 in greater detail. All the links that are shown in Figure 2.2 are assumed to have the same capacity, the capacity required for transporting video signal. Only the downstream links for transporting the video signal are shown in this figure. It is assumed that there are some upstream links that transport data and are used by the customers for making requests and sending evaluation and other feedback. The number of links that connect the main node with the intermediate node will be smaller than the total number of customers. Thus, it may be possible that if many customers ask for their program exactly at the same time, some of them will have to wait. How long they have to wait is the question that we will try to answer in the following chapters.

2.3 Traffic Model

We consider a circuit switching system. The reason for making this assumption is that we want customer's programming to continue without any interruptions. We further assume that time is slotted. This is a major assumption. The reason for assuming this is that in that way transmission of segments to different customers can be made to start at exactly the same time which makes multicasting much easier. Thus, we divide the time window of duration T during which requests are allowed into n slots. This window is the three-hour window that we mentioned in Section 2.1.

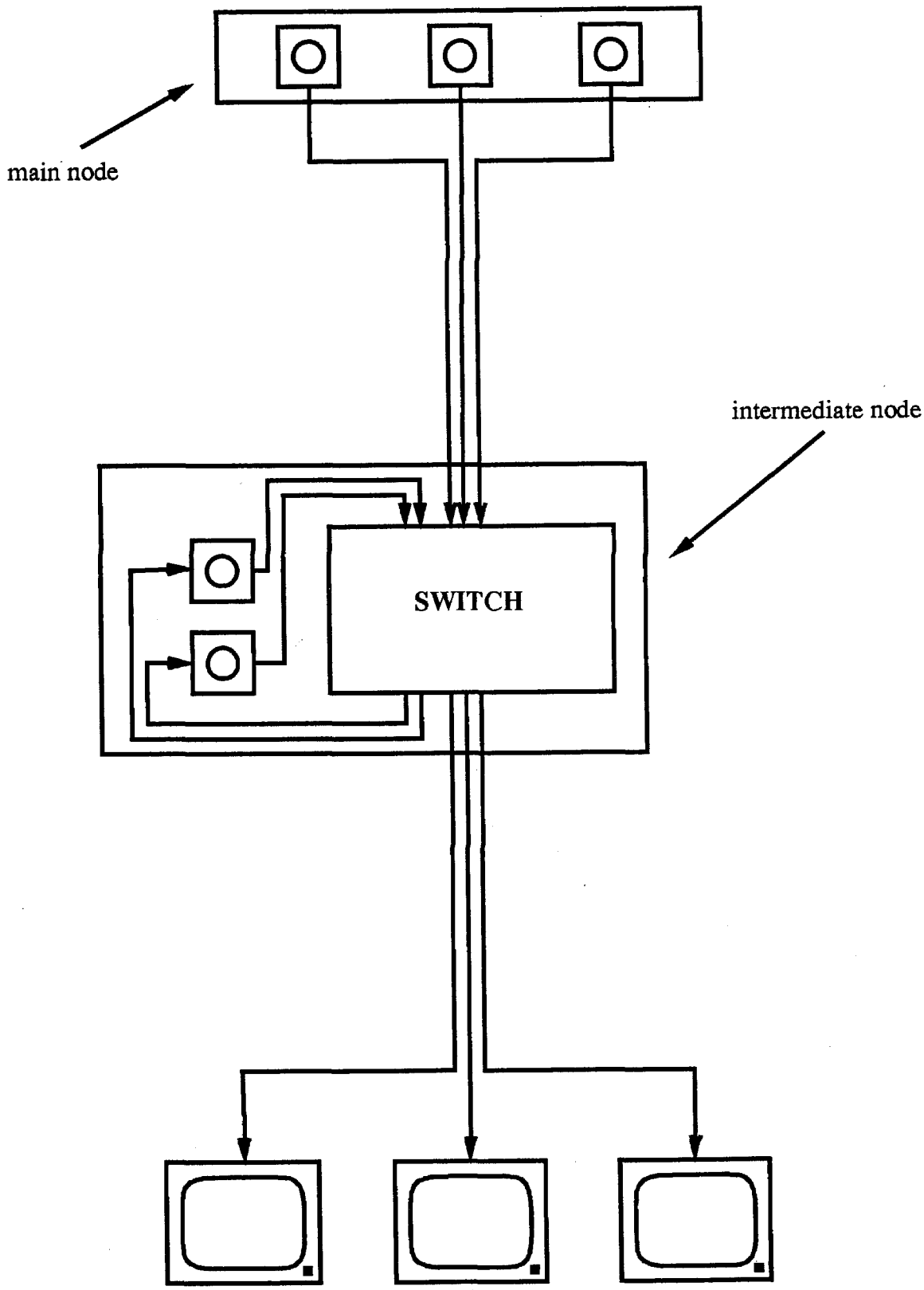

Figure 2.2: System Architecture

The duration of each slot is equal to d sec, the duration of each segment. During this time window of duration T each of the N customers will make exactly one request for his program. The algorithm that we are going to describe can be used even if each customer is allowed to make more than one request through the whole window but in that case we need to have a probability distribution of the number of requests that each customer makes. For simplicity we do not allow that here.

The request of a particular customer may occur at any one of the n slots; occurrences of the requests are independent of each other and uniformly distributed over the whole window. That means that the probability that r requests will occur at the i'th slot is $\binom{N}{r}\frac{(n-1)^{N-r}}{n^N}$.

We assume that requests arrive exactly in the beginning of a slot. In reality requests arrive througout the duration of a slot. When estimating the average delay of requests as seen by a subscriber, we will take into account that component too. Since we assume that requests arrive exactly at the beginning of a slot, it is possible that some requests will arrive exactly at the same time. In that case we order these requests at random. The same is done in [4]. Requests are then served in First-Come-First-Served order. That may be a restrictive assumption because in a real system we may want to serve requests in the way that will maximize multicasting but analysis of such a system is very difficult.

The number of links that connect the main node to the intermediate node is equal to k. We are mainly interested in the delay that requests are experiencing as a function of k. The reason for the delay is that if a large number of customers ask for their program, only k of them will be served at the same time and the rest of them will have to wait. We want k to be as small as possible for making the service more economical but we also want the average wait that the requests will have to wait to be fairly small too, so that customers will be satisfied by the service.

In our analysis we assume that each customer doesn't cancel his request even if he has to wait for a long time. In other words we assume infinitely patient customers as is explained in [2]. Some different types of customers are also described there, but

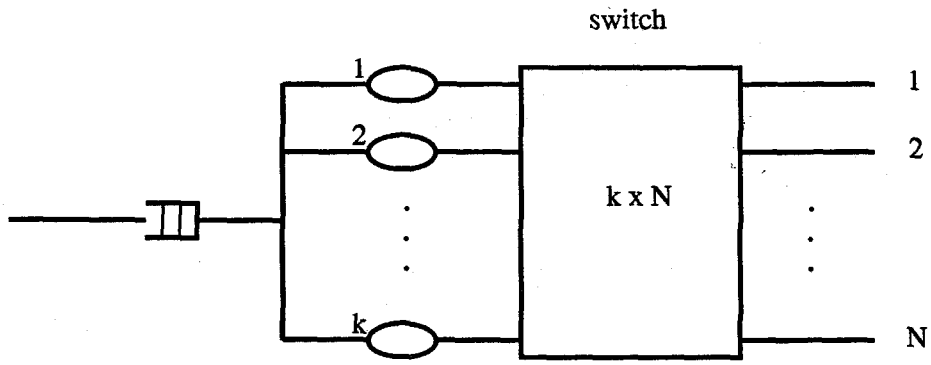

Figure 2.3: Traffic Model

we will not consider them here because they make the analysis very complicated.

Under these assumptions our system can be modeled as shown in Figure 2.3. The buffer that stores requests is assumed to have enough capacity to store all the requests. All k servers are identical. The service time of each request can be up to m slots and is assumed that after a server starts serving a particular request it will continue serving it until the request leaves the system. There is a probability vector $\mathbf{p} = (p_0, ..., p_m)$ that determines the duration of the service time of a request. What this means is that the probability that a request will require i slots of service is equal to p_i. The service time of each request is just the duration of the transmission between the main node and the intermediate node for the program of the particular customer. So, the program of a particular customer may last for up to m slots, i.e., contain up to m segments. How many segments will need to be transmitted from the main node to the intermediate node depends on the joint preferences of the subscribers, the segments that are stored in the intermediate node and the actual duration of the subscriber's program. All these factors have to be taken into account in the probability vector $\mathbf{p}$, which thus may be hard to estimate. We further assume that the segments that are transmitted from the main node to the intermediate node are directly transmitted to the customer and form the first part of his program. Immediately after this part is over, transmission of the segments that are stored in the intermediate node starts

and is assumed that enough storage and multicast facilities exist in the intermediate node for accommodating the remaining segments.

2.4 Summary

In this chapter we have proposed a new residential application for interactive video. The introduction of this service will be made possible with the deployment of Broadband ISDN. We proposed an architecture that can support this service efficiently. The novelty of this architecture is the introduction of storage closer to the customers, which increases the effectiveness of our system. The reason is that by storing popular material closer to the customer we decrease the traffic load in the system. We proposed a traffic model for quantitative analysis of this service. In the following chapters we will base our analysis on this model.

Bibliography

[1] H. E. Bussey, *et al.*, "Service Architecture, Prototype Description, and Network Implications of a Personalized Information Grazing System," *INFOCOM 1990*, pp. 1046-1053.

[2] A. D. Gelman and S. Halfin, "Analysis of Resources Sharing in Information Providing Services," *GLOBECOM '90*, 308.5, pp. 312-316, December 1990.

[3] A. D. Gelman, S. Halfin and W. Willinger, "On Buffer Requirements for Store-And-Forward Video on Demand," *GLOBECOM '91*, 28.5, pp. 976-980, December 1991.

[4] T.-C. Hou and A. K. Wong, "Queueing Analysis for ATM Switching of Mixed Continuous-Bit-Rate and Bursty Traffic," *INFOCOM '90*, pp. 660-667.

[5] E. C. Posner and P. N. Mouchtaris, "Interactive Video on Demand," *Joint FAW-IEEE Workshop*, Germany, September 1990.

[6] R. Ramarao and V. Ramamoorthy, "Architectural Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem," *ICC '91*, 17.6, pp. 506-510, June 1991.

[7] J. Rosenberg, *et al.*, "Multimedia Communications For Users," *IEEE Communications Magazine,* Vol. 30, pp. 20-36, May 1992.

[8] W. D. Sincoskie, " Video on Demand: Is it Feasible?" *GLOBECOM '90*, 305.3, pp. 201-205, December 1990.

[9] T.-S. P. Yum, "Hierarchical Distribution of Video with Dynamic Port Allocation," *IEEE Trans. on Communications*, Vol. 39, pp. 1268-1274, August 1991.

Chapter 3

The Single-Server Case

The case that the main link can only transport a single video connection is examined in this chapter. According to our traffic model this is the single-server case. A recurrent technique for the analysis of our system is proposed. This algorithm calculates the probabilities of the states and uses them for evaluating the expected wait. We prove that there is a very simple relationship between the probabilities of the states and the expected wait. We also propose an alternative approach that directly computes the expected wait. This approach is computationally more efficient but does not give us any information about the probabilities of the states.

3.1 Queueing Analysis

Our analysis is going to be based on the probabilities of the states. Therefore, we need to define what the states are. We consider the state of the system to be a vector $\mathbf{n} = (n_1, ..., n_m)$, where n_i for $1 \leq i \leq m-1$ is the number of requests that require i more slots of service. However the last component of the vector, n_m, is the number of requests that are waiting in the queue and it doesn't mean that each request waiting will actually require m slots of service. That is because requests that have not been served at all will require i slots of service with probability p_i, m slots with probability p_m. It is clear that $\sum_{i=1}^{m-1} n_i \leq 1$, since there is only 1 server, while $\sum_{i=1}^{m} n_i \leq N$, since

each one of the sources is allowed to make exactly one request through the whole window of duration T that we consider.

In the following sections we will describe a way of calculating the probability that the system is in state $(n_1, ..., n_m)$ at the end of a particular slot i, given that there were exactly j requests in the first i slots. We call these probabilities $P^{[i]}((n_1, ..., n_m)/j)$. We will use them for calculating the expected time that a request will have to wait until it starts getting served, which will be the criterion for measuring the performance of our system.

Before we start the analysis we need to introduce some notation. Let $\mathbf{n}^- = (n_1, ..., n_m)^-$ be the state of the system at the end of slot $i-1$ when we know that the state of the system at the end of the i'th slot is going to be $\mathbf{n} = (n_1, ..., n_m)$. This is assuming that no new requests are going to occur at the beginning of the i'th slot and that a request that enters the serving facility will require m slots of service. Suppose we know that $\sum_{i=1}^{m-1} n_i = 1$, i.e., one of the servers is busy and has some unfinished work to do. Then, if the request being served needs $m-1$ more slots of service, i.e., $n_{m-1} = 1$, then $(n_1, .., n_m)^- = (0, ..., 0, n_m+1)$. This is because the request that needs $m-1$ slots of service at the end of slot i needed m slots of service at the end of slot $i-1$. Similarly if $\sum_{i=1}^{m-1} n_i = 1$ but $n_{m-1} = 0$ then $(n_1, .., n_m)^- = (0, n_1, ..., n_{m-2}, n_m)$. On the other hand if $\sum_{i=1}^{m-1} n_i = 0$ then $(n_1, .., n_m)^- = (1, 0, ..., 0, n_m)$. We will also refer to the components of $\mathbf{n}^-$ as n_i^- for simplicity.

We should mention one more assumption that we are making. The above definition is a little ambiguous because $(0, .., 0)^-$ can be equal to either $(0, ..., 0)$, if the system was empty at the end of the previous slot, or $(1, 0, ..., 0)$, if there was only one request in the system that needed one more slot of service. The notation we are going to adopt is that $(0, ..., 0)^- = (1, 0, ..., 0)$ and $(0, ..., 0, 1, -1)^- = (0, ..., 0)$. The latter doesn't have any meaning in a real system but is a notational tool that we are going to use in order to make the proofs simpler. The problem arises from the fact that we want $\mathbf{n}^-$ to be unique. We can formally define $(n_1, ..., n_m)^-$ to be equal to $(1 - \sum_{i=1}^{m-1} n_i, n_1, ..., n_{m-2}, n_{m-1} + n_m)$. This definition unifies all the different cases that

we considered.

We will now prove a lemma which is quite general and is true for both constant and variable-service-times. We will need this lemma later on when we try to relate the $E(W)$ to the probabilities of the states.

Lemma 3.1 *Under the conditions that* $i \geq (j-1) \cdot m$ *and* $n_m \geq 1$*:*

$$P^{[i]}((n_1, ..., n_m)^-/j) = \sum_{a=0}^{j-n_m-1} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)^-/j-a).$$

Proof: The idea is that the set of cases that can lead the system to a particular state $\mathbf{n}^-$ at the end of slot $i-1$ is a subset of the set of cases that can lead to the same state at the end of slot i. This subset is the number of cases that can lead to the state $\mathbf{n}^-$ at the end of slot i provided that there was no arrival in the first slot. One has to consider the cases in which there were, say, a arrivals in the first slot. Since $i \geq (j-1) \cdot m$, these arrivals cannot affect the state that will occur at the end of slot i, because they were too far in the past. So, the number of cases that can lead to state $\mathbf{n}^-$ at the end of slot i is the number of cases that can lead to state $\mathbf{n}^-$ at the end of slot $i-1$ if there were only $j-a$ arrivals in the first $i-1$ slots. (A detail, though, is that we have to take a weighted average of those cases to take into account the probability distribution of the arrivals.) This completes the proof. The reason for considering $\mathbf{n}^-$ instead of $\mathbf{n}$ is because we are going to have $\mathbf{n}^-$'s in the proof of the theorems, where we are going to use this lemma.

3.1.1 Constant-Service-Times

Let us concentrate on the case where the service time of each request is constant and equal to m. This means that $\mathbf{p} = (0, ..., 0, 1)$. The special case $m = 1$ is treated in [11]. Actually that algorithm can be generalized to solve the case $m > 1$, but our technique, although not as efficient in the special case, can be generalized for the case of variable-service-times and multiple-servers as we will see in the next chapter. The work in [11] was motivated by a recursive approach for continuous-time periodic queues presented in [4, 5].

We are ready to express the probabilities $P^{[i]}((n_1,...,n_m)/j)$ of the states at the end of slot i, given that exactly j requests occurred in the first i slots, as a function of the same probabilities of the states at the end of slot $i-1$. Let us temporarily forget about the all-zero state and concentrate on the other states. If we know the number of arrivals (new requests) that occurred at the beginning of slot i, then we don't have any choices about the state of the system at the end of slot $i-1$. So, what we have to do is average over the different possible number of arrivals a at the beginning of slot i. We can easily do that because we know the number of arrivals in the first i slots, which is exactly j. Thus:

$$P^{[i]}((n_1,...,n_m)/j) = \sum_{a=0}^{n_m+n_{m-1}} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1,...,n_m-a)^-/j-a).$$

For the all-zero state things are more complicated. The reason is that we can go to the all-zero state from two different states. These are the all-zero state itself and the state $\mathbf{n} = (1,0,...,0)$. When $m = 1$, things are even more complicated because here it is actually possible to go from the all-zero state to the all-zero state in two ways. One way is if there are no arrivals at all at the beginning of the i'th slot, and the other is if there is exactly one arrival at the beginning of the i'th slot. Thus if $m > 1$:

$$P^{[i]}((0,...,0)/j) = [P^{[i-1]}((0,...,0)/j) + P^{[i-1]}((1,...,0)/j)](\frac{i-1}{i})^j.$$

So, for the case $m = 1$ we have to sum three terms:

$$\begin{aligned} P^{[i]}(0/j) &= [P^{[i-1]}(0/j) + P^{[i-1]}(1/j)](\frac{i-1}{i})^j \\ &\quad + P^{[i-1]}(0/j-1)j\frac{(i-1)^{j-1}}{i^j}. \end{aligned}$$

We should also note that $P^{[i]}((n_1,...,n_m)/j) = 0$ if $\sum_{i=1}^{m-1} n_i > 1$ or < 0, or $\sum_{i=1}^{m} n_i > j$ or < 0, or $n_m \geq j$, since the number of customers in service cannot be negative or larger than the number of servers, and the number of customers in the system cannot be larger than the total number of arrivals.

The question that remains is how to start the recursion. There are many ways to do so but there is a very simple answer. Since we know that everything starts at

the first slot, we also know that nothing happened before that. So, we can set all the $P^{[0]}((n_1, ..., n_m)/j)$ at the the end of the 0'th slot to zero, except for $P^{[0]}((0, ..., 0)/0)$. This is equal to 1 since there are no arrivals before the first slot and the system can only be in the all-zero state.

After finding a way of calculating the probabilities of the states, we can use these to find the expected time that an arrival will have to wait until it starts getting served. We call this $E(W)$. This expected wait depends on the slot in which the arrival occurred. So, we have to take the average over the different possibilities. If we call $E(W^i)$ the expected time that a request will have to wait given that it arrived at the i'th slot, we have:

$$E(W) = \sum_{i=1}^{n} \frac{1}{n} E(W^i).$$

The reason is that, because we assumed that occurrences of the requests are uniformly distributed over the whole time window, the probability that the request arrived at any of the slots is $\frac{1}{n}$, the same for all slots.

One point that we can make is that $E(W^i)$ increases as i increases. Actually it is strictly increasing until $i = m \cdot (N-1)$ and stays constant after that. This is because after that the number of cases that can force a request to wait stay the same. Requests that occurred more than $m \cdot (N-1)$ slots in the past cannot hurt the current request even if all the requests occurred at that slot. The consequence of the above statement that $E(W^i)$ increases as i increases is that requests that arrive in the last slots receive worse service than the ones that arrive in the first slots. For that reason from now on we will concentrate on evaluating only $E(W^n)$ because we know that arrivals in the last slot will receive the worst service. So $E(W^n)$ will be an upper bound for $E(W)$.

Another reason for concentrating on $E(W^n)$ is the fact that if we consider that the arrival pattern is periodic then the delay that all the requests are experiencing is equal to $E(W^n)$, because each request can be considered as arriving at the end of a window of duration $n \cdot d$ with exactly N arrivals in that window. The other $E(W^i)$'s give the delay that the arrivals experience when we start from a completely empty

system.

What we have to do is find out how much a request will have to wait given that it occurred at the n'th slot. It is possible that other requests occurred at the same slot, too. So, we will have to average over these possibilities using the random ordering assumption that we have made. After we know how many requests occurred at the last slot, we have to know in which state the system was at the end of the previous slot. Thus we have to take an average again. The result is

$$E(W^n) = \sum_{a=0}^{N-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} \frac{1}{a+1} \sum_{all\ states} P^{[n-1]}((n_1,...,n_m)/N-1-a) \cdot \sum_{t=0}^{a} [prod(\mathbf{n}) + t \cdot m] \tag{3.1}$$

where

$$prod(\mathbf{n}) = \sum_{i=1}^{m} n_i \cdot i.$$

The question now is whether we can find a somewhat simpler relation for $E(W^n)$. Hopefully such a simpler relation will generalize for the case of variable-service-times. The following theorem will answer this question.

Theorem 3.1 *When $n \geq m \cdot (N-1)$ then:*

$$E(W^n) = \frac{n}{N} \sum_{all\ states} P^{[n]}((n_1,...,n_m)/N) \cdot n_m.$$

This is proven in Appendix 3.A. The result of the following corollary which is a special case of Theorem 3.1 is mentioned in [1, 11]. Some similar results for a continuous-time system have also been mentioned in [2, 6, 7].

Corollary 3.1 *If $m = 1$ and $n \geq N-1$ then:*

$$E(W^n) = \frac{n}{N} \sum_{n_1=0}^{N-1} P^{[n]}(n_1/N) \cdot n_1.$$

Proof: Since $m = 1$, the state vector reduces to a simple number or index that we call n_1. Thus, the above relation follows.

3.1.2 Variable-Service-Times

We will now deal with arbitrary **p**. We can again express the probabilities of the states by using a recursive approach.

Let us temporarily forget about the states that have $n_{m-1} = 1$, and also the all-zero state. The other states can occur either because the request currently in service needed one more slot of service at the previous slot or because a new request that needed fewer than m slots of service entered the service facility. In the case of constant-service-times, only the first of these two cases can occur. In case a new request enters the system, this new request may not have been the first in line. Then all the requests that were in front of it didn't need any service at all. Thus

$$\begin{aligned} P^{[i]}((n_1,...,n_m)/j) &= \sum_{a=0}^{n_m} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1,...,n_m-a)^-/j-a) \\ &+ \sum_{r=0}^{j} \binom{j}{r} \frac{(i-1)^{j-r}}{i^j} \sum_{s=\max(n_m+1-r,0)}^{\max(j-r-1,0)} P^{[i-1]}((0,...,0,s)/j-r) \\ &\cdot p_0^{s+r-n_m-1} \prod_{i=1}^{m-1} p_i^{n_i^-}. \end{aligned}$$

For the states in which $n_{m-1} = 1$, things are simpler because we know that a new request that needed m slots of service entered the system. Thus

$$\begin{aligned} P^{[i]}((0,...,0,1,n_m)/j) &= \sum_{r=0}^{j} \binom{j}{r} \frac{(i-1)^{j-r}}{i^j} \sum_{s=\max(n_m+1-r,0)}^{\max(j-r-1,0)} \\ &\cdot P^{[i-1]}((0,...,0,s)/j-r) \cdot p_m \cdot p_0^{s+r-n_m-1}. \end{aligned}$$

We can use similar arguments for the all-zero state, remembering the special properties of the all-zero state that we discussed in the case of constant-service-times. We have

$$\begin{aligned} P^{[i]}((0,...,0)/j) &= [P^{[i-1]}((0,...,0)/j) + P^{[i-1]}((1,...,0)/j)](\frac{i-1}{i})^j \\ &+ \sum_{r=0}^{j} \binom{j}{r} \frac{(i-1)^{j-r}}{i^j} \sum_{s=\max(1-r,0)}^{\max(j-r-1,0)} P^{[i-1]}((0,...,0,s)/j-r) \\ &\cdot [p_0^{s+r} + p_0^{s+r-1} p_1]. \end{aligned}$$

We will again concentrate on $E(W^n)$ for the reasons explained earlier. We have

$$E(W^n) = \sum_{a=0}^{N-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} \frac{1}{a+1} \sum_{all\ states} P^{[n-1]}((n_1,...,n_m)/N-1-a) \cdot \sum_{t=0}^{a} [prod(\mathbf{n}) - n_m \cdot m + (n_m+t) \cdot prod(\mathbf{p})]. \quad (3.2)$$

The only difference from the constant-service-time case is that when $t+n_m$ requests are in front of the request that we are considering we have to take the average by using the probability vector $\mathbf{p}$. The waiting time of that request is not just $(t+n_m) \cdot m$.

We are now ready to express $E(W^n)$ as a simple function of the probabilities of the states. The proof of the theorem can be found in Appendix 3.B.

Theorem 3.2 *When $n \geq m \cdot (N-1)$ then:*

$$E(W^n) = \frac{n}{N} \sum_{all\ states} P^{[n]}((n_1,...,n_m)/N) \cdot n_m.$$

Let us now try to make our algorithm a little more efficient. The expression that we have from Lemma 3.1 gives us an alternative for calculating the probabilities of the states. This expression is more efficient and simpler to use than the general recursive relations that we have. The only problem is that we cannot always use it since there are those two conditions that have to be satisfied: these are $i \geq (j-1) \cdot m$ and $n_m \geq 1$. What we will do now is to try to relax those conditions as much as possible. If we do that we will be able to use the simpler relations more often and thus increase the performance of our algorithm.

Lemma 3.2 *Under the condition that $i \geq (j-n_m) \cdot m + 1 - \sum_{l=1}^{m-1} n_l \cdot l$ and $\mathbf{n}$ is not the all-zero state:*

$$P^{[i]}((n_1,...,n_m)/j) = \sum_{a=0}^{j-n_m-1} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1,...,n_m)/j-a).$$

Proof: The reason for this expression holding is the same as the one that we produced in the proof of Lemma 3.1. The difference is that in that proof we were interested in treating all the states together and for that reason our result was more conservative. Here we concentrate on each state separately. In fact, the result of Lemma 3.1 is the worst-case result of this lemma and corresponds to the $(0,...,0,1)$ state.

Lemma 3.3 *If* $\mathbf{n}$ *is the all-zero state and* $i \geq j \cdot m + 1$, *we have:*

$$P^{[i]}((n_1, ..., n_m)/j) = \sum_{a=0}^{j} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((0, ..., 0)/j - a).$$

Proof: The proof is the same as the proof of the previous lemma. The only difference is the fact that the summation goes from $a = 0$ up to $a = j$ instead of going up to $a = j - 1$. The reason is that for the all-zero state we are not certain that the server was busy at the previous slot.

It is very important, as was mentioned in [11], that the results of the lemmas relate the probability of a particular state to the probabilities of the same state at the previous slot for various numbers of arrivals. What it means is that, because of the lemmas, when calculating the probabilities of a particular state, we can forget about the probabilities of the other states. So, instead of having to operate on a huge matrix, we only have to operate on vectors. This decreases both the number of operations and the storage requirements.

Let us now take advantage of another fact. We have mentioned in the previous sections that $E(W^i)$ is strictly increasing up to one point and then stays the same. We may be able to use that fact to stop our algorithm before it actually reaches the last slot, because after $E(W^i)$ stops increasing the algorithm doesn't give us any more information. This is indeed true and will be clear after the next theorem.

We will need to prove a lemma before proving the theorem.

Lemma 3.4 *Let* $n_* = (N-1) \cdot m$. *If* $n \geq n_*$:

$$P^{[n]}((n_1, ..., n_m)/N) = \sum_{r=n_m+1}^{N} \binom{N}{r} \frac{(n_*)^r (n - n_*)^{N-r}}{n^N} P^{[n*]}((n_1, ..., n_m)/r).$$

Proof: By using similar arguments to the ones that we used for proving Lemma 3.1 we can write:

$$P^{[n]}((n_1, ..., n_m)/N) = \sum_{a=0}^{N-n_m-1} \binom{N}{a} \frac{(n_*)^{N-a} (n - n*)^a}{n^N} P^{[n*]}((n_1, ..., n_m)/N - a).$$

The only difference is that, instead of expressing the probabilities of the states at the end of slot n by using the states at the end of slot $n-1$, we do that by using the

probabilities of the states at the end of slot n_*. By making a change of variable the result follows.

Now we are ready to prove the theorem.

Theorem 3.3 *Let $n_* = (N-1)\cdot m$. If $n \geq n_*$ then:*

$$E(W^n) = \frac{n}{N} \sum_{all\ states} \sum_{r=n_m+1}^{N} \binom{N}{r} \frac{(n_*)^r (n-n_*)^{N-r}}{n^N} P^{[n_*]}((n_1,...,n_m)/r)\cdot n_m.$$

Proof: By using the result of Theorem 3.2 and the previous lemma the result follows.

It is now clear that in order to calculate $E(W^n)$, we only need to use the recursive relations and the results of the lemmas to calculate the probabilities of the states at the end of slot n_*. After we do that, we can use the result of the last theorem to get $E(W^n)$.

3.2 A Simplified Analysis

Let us now concentrate a little more on relation 3.2 in page 25 for $E(W^n)$. It seems possible that by using the results of Lemma 3.1 and Theorem 3.2 we will be able to get a relationship between $E(W^n)$ when there are N customers, which we call $E(W^n, N)$, and $E(W^n)$ when there are $N-1$ customers, which we call $E(W^n, N-1)$. This is indeed true and this is what we are going to prove next. It is clear that

$$\begin{aligned}
E(W^n, N) = & \\
& \sum_{a=0}^{N-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} \frac{1}{a+1} \sum_{all\ states} P^{[n-1]}((n_1,...,n_m)/N-1-a) \\
& \quad \cdot \sum_{t=0}^{a} [prod(\mathbf{n}) - n_m \cdot m + (n_m + t)\cdot prod(\mathbf{p})] \\
& = \sum_{a=0}^{N-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} \frac{1}{a+1} \sum_{all\ states} P^{[n-1]}((n_1,...,n_m)/N-1-a) \\
& \quad \cdot (a+1)\cdot [prod(\mathbf{n}) - n_m \cdot m + n_m \cdot prod(\mathbf{p}) + \frac{a}{2} prod(\mathbf{p})].
\end{aligned}$$

Let us now assume that $n \geq m\cdot(N-1)$ so that Lemma 3.1 and Theorem 3.2 hold. By using Lemma 3.1, Theorem 3.2 and the fact that the sum of the probabilities of

all the possible states for a given number of arrivals is equal to 1, we get:

$$\begin{aligned}
E(W^n,N) &= \sum_{a=0}^{N-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}\sum_{all\ states}P^{[n-1]}((n_1,...,n_m)/N-1-a)\\
&\quad\cdot[prod(\mathbf{n})-n_m\cdot m]\\
&\quad+\frac{N-1}{n}prod(\mathbf{p})E(W^n,N-1)\\
&\quad+\frac{prod(\mathbf{p})}{2}\sum_{a=0}^{N-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}a\\
&= \sum_{a=0}^{N-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}\sum_{all\ states}P^{[n-1]}((n_1,...,n_m)/N-1-a)\\
&\quad\cdot[prod(\mathbf{n})-n_m\cdot m]\\
&\quad+\frac{N-1}{n}prod(\mathbf{p})E(W^n,N-1)\\
&\quad+\frac{prod(\mathbf{p})}{2}\cdot\frac{1}{n^{N-1}}\sum_{a=1}^{N-1}\binom{N-1}{a-1}(N-a)(n-1)^{N-1-a}\\
&= \sum_{a=0}^{N-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}\sum_{all\ states}P^{[n-1]}((n_1,...,n_m)/N-1-a)\\
&\quad\cdot[prod(\mathbf{n})-n_m\cdot m]\\
&\quad+\frac{N-1}{n}prod(\mathbf{p})E(W^n,N-1)\\
&\quad+\frac{prod(\mathbf{p})}{2}\cdot\frac{N-1}{n}. \qquad (3.3)
\end{aligned}$$

The following theorem will help us simplify the first term.

Theorem 3.4 *When $n \geq m\cdot(N-1)$ then:*

$$\sum_{all\ states}P^{[n]}((n_1,...,n_m)/N)\cdot[prod(\mathbf{n})-n_m\cdot m]=$$
$$\frac{N}{n}\Big(p_m\frac{m(m-1)}{2}+p_{m-1}\frac{(m-1)(m-2)}{2}+...+p_2\Big).$$

This is proven in Appendix 3.C. We can use the result of this theorem in equation 3.3. What we get is:

$$\begin{aligned}
E(W^n,N) &= \frac{N-1}{n}\Big(p_m\frac{m(m-1)}{2}+p_{m-1}\frac{(m-1)(m-2)}{2}+...+p_2\Big)\\
&\quad+\frac{N-1}{n}prod(\mathbf{p})E(W^n,N-1)+\frac{prod(\mathbf{p})}{2}\cdot\frac{N-1}{n}.
\end{aligned}$$

By using the fact that $prod(\mathbf{p}) = p_1 + 2p_2 + ... + mp_m$, we get that

$$E(W^n, N) = \frac{N-1}{n} prod(\mathbf{p}) E(W^n, N-1) + \frac{N-1}{2n} \sum_{i=0}^{m} p_i \cdot i^2. \qquad (3.4)$$

This is a very simple and efficient relation for calculating $E(W^n)$. If we are only interested in $E(W^n)$ we do not need to run the whole algorithm and obtain the probabilities of the states. We can use this relation which is much faster and can give us the answer for all practical cases. In the case that we need the probabilities of the states the whole algorithm has to be used, though.

From our analysis it is clear that, if we let N and n become large then our system looks like a continuous-time system and our results should agree with the known results for the M/G/1 queue. The relation that we just proved gives us a good opportunity to check this agreement. Let $N, n \to \infty$ by keeping their ratio constant. This ratio is the arrival rate that is usually called λ in the continuous-time systems. What our relation gives us is:

$$E(W^n) = \frac{\lambda \sum_{i=0}^{m} p_i \cdot i^2}{2(1 - \lambda \cdot prod(\mathbf{p}))}.$$

This result is in direct agreement with the known Pollaczek-Khinchine formula for $E(W)$ that says that:

$$E(W) = \frac{\lambda E(\tau^2)}{2(1-\rho)}$$

where $\rho = \lambda \cdot E(\tau)$ and τ is the service time of arrivals which is a random variable (see [8], page 58). This is therefore another derivation of that formula.

In the special case that $m = 1$ and $\mathbf{p} = (0, 1)$ equation 3.4 reduces to the following relation:

$$E(W/N) = \frac{N-1}{n} E(W/N-1) + \frac{N-1}{2n}.$$

It can be easily proven (e.g. by induction) by using this relation that:

$$E(W/N) = \frac{1}{2} \sum_{i=1}^{N-1} \frac{(N-1)!}{(N-1-i)!} (\frac{1}{n})^i.$$

Amazingly enough, this is almost the same relation that is mentioned in [3] for the expected wait in an nD/D/1 queue. That relation is derived from the Laplace-Stieltjes

transform of the waiting time distribution for that system. The only difference between that relation and the relation derived here is due to the fact that in [3] the service time of each request is equal to D and the duration of the time window that we consider is equal to the time unit, while in our case the service time of each request is equal to the time unit and the duration of the window is equal to n. There is another difference in the exhibited relation but clearly it is due to a typo in [3].

Similarly, in the case of arbitrary m and $\mathbf{p}$ it can be proven by using equation 3.4 that:

$$E(W/N) = \frac{1}{2}\sum_{i=1}^{N-1} \frac{(N-1)!}{(N-1-i)!}(\frac{1}{n})^i (prod(\mathbf{p}))^{i-1} \cdot \sum_{i=0}^{m} p_i \cdot i^2.$$

This result agrees with the expected wait in an nD/G/1 queue (see [9]). The only difference is due to the fact that the average service time is taken to be the time unit in [9].

Let us now consider what happens when we rescale time; what we mean by that will become clear later on. We know that for the M/M/1 queue, when we rescale time (divide both the arrival and the service rates by the same constant,) $E(W)$ gets rescaled (gets multiplied by the same constant). The following theorem will show us what happens when we rescale time in the system that we are considering.

Theorem 3.5 *Consider two systems. In the first system the total number of slots is n_1, the total number of arrivals is N and the probability vector that determines the service times is $\mathbf{p_1} = (p_{1,0}, ..., p_{1,m})$. In the second system the total number of slots is n_2, the total number of arrivals is N and the probability vector that determines the service times is $\mathbf{p_2} = (p_{2,0},\ p_{2,1})$, where $p_{2,1} = \frac{1}{m}\sum_{i=0}^{m} p_{1,i} \cdot i$ and $p_{2,0} = 1 - p_{2,1}$. Let $n_1 = n_2 \cdot m$ and $n_2 \geq N - 1$. Let $E_1(W^{n_1})$ and $E_2(W^{n_2})$ be the expected wait that requests in the first and second system experience respectively. Then*

$$E_1(W^{n_1}) = E_2(W^{n_2}) \cdot \frac{1}{m \cdot p_{2,1}} \sum_{i=0}^{m} p_{1,i} \cdot i^2.$$

Proof: The proof is by induction on N. For $N = 2$ it is trivially true. By assuming that it is true for N it can be proven that the same relation holds for $N + 1$ by using relation 3.4. This completes the proof.

For the special case that we have constant-service-times, Theorem 3.5 says that $E_1(W^{n_1}) = E_2(W^{n_2}) \cdot m$. What that means is that by scaling down the total number of slots and the service time by m the expected wait gets multiplied by m. This is reasonable because a similar effect occurs in the continuous-time case. Something even stronger is true for this case, though. The probabilities that the number of requests in the queue is l, i.e., $n_m = l$, is the same for both systems. We will give the proof of this statement when we talk about the multiple-servers case.

3.3 Numerical Examples

Let us now try a few numerical examples. Let us assume that $m = 7$ and that all p_i's are equal, i.e., $p_i = 0.125$. We are going to keep the ratio $\frac{N}{n}$ constant and equal to $\frac{1}{7}$. In other words we are going to keep the arrival rate constant. Let us see what happens to $E(W^n)$ as both N and n increase. We showed in the previous section that $E(W^n)$ will approach a specific value and that value is 2.5 in this case. In Table 3.1 we see that indeed $E(W^n)$ approaches that value, although N and n need to become fairly large before that value is achieved. We calculated the values in this table by using the simplified analysis. This is the reason why we could calculate $E(W^n)$ for such large values of N and n.

If we are interested in the probabilities of the states we can use the full algorithm. In Figure 3.1 we see the probability distribution for n_m for the case that $N = 50$, $n = 350$ and for the same probability vector $\mathbf{p}$ as before. What we see in that figure is $\sum_{n_1,...,n_{m-1}} P^{[350]}(n_1, ..., n_m)$ plotted versus n_m. The probability that $n_m = 0$ is fairly large, 0.7875. The probability that n_m is larger than 16 drops below 10^{-10}.

3.4 Summary

A new recurrent algorithm for the analysis of our system for the case that the main link can only transport a single video connection was proposed. We showed how this

N	n	$E(W^n)$
10	70	1.794
50	350	2.317
100	700	2.405
500	3,500	2.480
1,000	7,000	2.490
5,000	35,000	2.498
10,000	70,000	2.499
50,000	350,000	2.500

Table 3.1: $E(W^n)$ in number of slots

algorithm can be used to obtain the probabilities that our system is at a particular state. We proved that there is a simple relation which we can use for calculating the expected wait after we have obtained the probabilities of the states.

We also proposed a simplified analysis. In the case that we are not interested in the probabilities of the states we can directly obtain the expected wait with a very simple and efficient algorithm. With this approach we can calculate the expected wait for a very large number of customers, which we cannot do with the previous algorithm. We also showed that in the limit that the number of customers becomes very large our formula reduces to the well-known Pollaczek-Khinchine formula.

3.A Appendix: Proof of Theorem 3.1

Let us define

$$f(\mathbf{n}) \triangleq \sum_{t=0}^{n_m-1} [1 + prod(\mathbf{n}) - (t+1)m],$$

$$F(\mathbf{n}, N, n) \triangleq \frac{n}{N} \sum_{\substack{all\ states \\ with\ n_m \geq 1}} [P^{[n]}(\mathbf{n}/N) - P^{[n]}(\mathbf{n}^-/N)] f(\mathbf{n}).$$

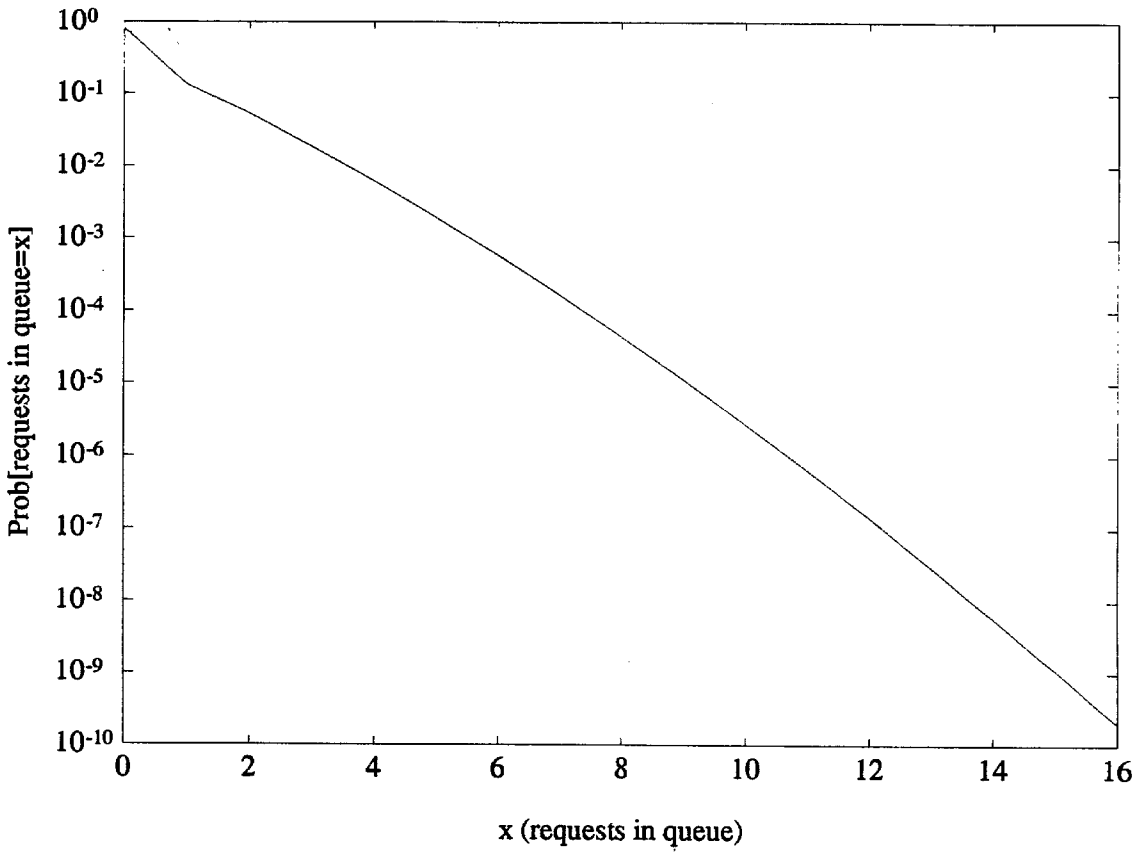

Figure 3.1: Probability Distribution for $N = 50$

What $f(\mathbf{n})$ expresses is the sum of the delays that the n_m requests will experience if all of them arrive at the beginning of the same slot and the state at the end of that slot is $\mathbf{n}$. We have

$$\begin{aligned} F(\mathbf{n},N,n) \;=\; & \frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\;\sum_{a=0}^{n_{m-1}+n_m}\binom{N}{a}\frac{(n-1)^{N-a}}{n^N} \\ & \cdot P^{[n-1]}((n_1,\ldots,n_m-a)^-/N-a)f(\mathbf{n}) \\ & -\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\;\sum_{j=0}^{N-n_m-1}\binom{N}{j}\frac{(n-1)^{N-j}}{n^N} \\ & \cdot P^{[n-1]}((n_1,\ldots,n_m)^-/N-j)f(\mathbf{n}). \end{aligned}$$

The first sum holds because of the recurrence algorithm and the second because

of Lemma 3.1; the condition for the lemma is indeed satisfied because $n_m \geq 1$. Continuing,

$$\begin{aligned} F(\mathbf{n},N,n) &= \frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{a=1}^{n_{m-1}+n_m}\binom{N}{a}\frac{(n-1)^{N-a}}{n^N} \\ &\quad \cdot P^{[n-1]}((n_1,...,n_m-a)^-/N-a)f(n_1,...,n_m) \\ &\quad -\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{j=1}^{N-n_m-1}\binom{N}{j}\frac{(n-1)^{N-j}}{n^N} \\ &\quad \cdot P^{[n-1]}((n_1,...,n_m)^-/N-j)f(n_1,...,n_m). \end{aligned}$$

(There were some cancellations and algebraic manipulations in deriving the above relation.) Continuing,

$$\begin{aligned} F(\mathbf{n},N,n) &= \frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{a=0}^{n_{m-1}+n_m-1}\binom{N}{a+1}\frac{(n-1)^{N-1-a}}{n^N} \\ &\quad \cdot P^{[n-1]}((n_1,...,n_m-a)^-/N-1-a)f(n_1,...,n_m) \\ &\quad -\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{j=0}^{N-n_m-2}\binom{N}{j+1}\frac{(n-1)^{N-1-j}}{n^N} \\ &\quad \cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)f(n_1,...,n_m) \\ &= \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{a=0}^{n_{m-1}+n_m-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}} \\ &\quad \cdot P^{[n-1]}((n_1,...,n_m-a-1)^-/N-1-a)\frac{1}{a+1}f(n_1,...,n_m) \\ &\quad -\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{j=0}^{N-n_m-2}\binom{N-1}{j}\frac{(n-1)^{N-1-j}}{n^{N-1}} \\ &\quad \cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)\frac{1}{j+1}f(n_1,...,n_m). \end{aligned}$$

One important point is that $n_m - a - 1$ can be negative in the above expression. The reason why this makes sense was mentioned earlier in the definition of $\mathbf{n}^-$ about the all-zero state; only the m'th component of $\mathbf{n}^-$ is negative and not the m'th component of $\mathbf{n}$. We can now write:

$$F(\mathbf{n},N,n) = \sum_{a=0}^{N-1}\sum_{\substack{\text{all states with}\\ n_m\geq a+1-n_{m-1}\\ n_m\geq 1}}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m-a-1)^-/N-1-a)\frac{1}{a+1}f(n_1,...,n_m)$$

$$-\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{j=0}^{N-n_m-2}\binom{N-1}{j}\frac{(n-1)^{N-1-j}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)\frac{1}{j+1}f(n_1,...,n_m)$$

$$=\sum_{a=0}^{N-1}\sum_{\substack{n_m\leq N-2-a\\ \text{all states with}\\ n_m\geq -n_{m-1}\\ \text{and}\\ n_m\geq -a}}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-a)\frac{1}{a+1}f(n_1,...,n_m+a+1)$$

$$-\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{j=0}^{N-n_m-2}\binom{N-1}{j}\frac{(n-1)^{N-1-j}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)\frac{1}{j+1}f(n_1,...,n_m)$$

$$=\sum_{\substack{\text{all states with}\\ n_m\geq -n_{m-1}}}\sum_{\substack{a=0\\ \text{with}\\ a\geq -n_m}}^{N-n_m-2}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-a)\frac{1}{a+1}f(n_1,...,n_m+a+1)$$

$$-\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1}}\sum_{j=0}^{N-n_m-2}\binom{N-1}{j}\frac{(n-1)^{N-1-j}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)\frac{1}{j+1}f(n_1,...,n_m)$$

$$=\sum_{\substack{\text{all states with}\\ n_m\geq -n_{m-1}}}\sum_{\substack{a=0\\ \text{with}\\ a\geq -n_m}}^{N-n_m-2}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-a)$$

$$\cdot\frac{1}{a+1}\sum_{t=0}^{a}[1+prod(n_1,..,n_m)+t\cdot m]$$

$$=\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0}}\sum_{a=0}^{N-n_m-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)/N-1-a)$$

$$\cdot\frac{1}{a+1}\sum_{t=0}^{a}[prod(n_1,..,n_m)+t\cdot m].$$

The last step follows because the time that a request would have to wait if the system

were at state $\mathbf{n}^-$ is one slot more than the time that it would have to wait when the system is at state $\mathbf{n}$.

Thus we finally conclude that $E(W^n) = F(\mathbf{n}, N, n)$. One detail is that we assume, as usual, that whenever the lower limit of a summation is larger than the upper limit then the whole summation is equal to zero. It is also easy to see that

$$f(\mathbf{n}^-) - f(\mathbf{n}) = n_m^-.$$

That is because of the meaning of $f(\mathbf{n})$ that we mentioned before. Each of the n_m^- requests will have to wait one more slot if the system is at state $\mathbf{n}^-$ rather than state $\mathbf{n}$. This completes the proof of the theorem.

The main point of the proof is that we can use the probabilities of the states $\mathbf{n}$ to directly calculate $E(W^n)$. The problem is that when we calculate the probability of a particular state we allow cases in which there were no arrivals at the last slot. For finding $E(W^n)$, our assumption is that there was certainly at least one arrival and we are trying to find the expected wait for this arrival. So, we have to subtract the probability of state $\mathbf{n}^-$, which is the probability that we are going to enter state $\mathbf{n}$ if there was no arrival at the last slot. This is what $\mathbf{n}^-$ means, after all. This is why $F(\mathbf{n}, N, n)$ is defined as above. There are some more details that just work out in the combinatorial algebra.

3.B Appendix: Proof of Theorem 3.2

Let us define

$$\begin{aligned}
f(\mathbf{n}) &\triangleq \sum_{t=0}^{n_m-1} [1 + prod(\mathbf{n}) - n_m \cdot m + t \cdot prod(\mathbf{p})], \\
g(\mathbf{n}) &\triangleq \Big[\sum_{j=0}^{n_m-1} p_0^j \sum_{i=0}^{m-1} [f(0,...,0,1,n_m - j) - (n_m - j) \cdot i \,] p_{m-i} \Big] \\
&\quad \cdot \frac{1}{f(0,...,0,1,n_m)}, \\
F(\mathbf{n}, N, n) &\triangleq \frac{n}{N} \sum_{\substack{\text{all states} \\ \text{with } n_m \geq 1 \\ \text{and } n_{m-1}=0}} [P^{[n]}(\mathbf{n}/N) - P^{[n]}(\mathbf{n}^-/N)] f(\mathbf{n})
\end{aligned}$$

$$+\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}[P^{[n]}(\mathbf{n}/N)-g(\mathbf{n})P^{[n]}(\mathbf{n}^-/N)]f(\mathbf{n}).$$

We are now going to use the lemma and the recursive relations that we have for the probabilities of the states. By omitting some of the steps that are identical to the ones taken to prove Theorem 3.1, we get

$$\begin{aligned}
F(\mathbf{n},N,n) = &\\
&\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0\\ \text{and } n_{m-1}=0}}\sum_{a=0}^{N-n_m-2}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}\\
&\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-a)\frac{1}{a+1}f(n_1,...,n_m+a+1)\\
&-\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0\\ \text{and } n_{m-1}=0}}\sum_{j=0}^{N-n_m-2}\binom{N-1}{j}\frac{(n-1)^{N-1-j}}{n^{N-1}}\\
&\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)\frac{1}{j+1}f(n_1,...,n_m)\\
&+\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=0}}\sum_{r=0}^{N}\binom{N}{r}\frac{(n-1)^{N-r}}{n^N}\sum_{s=\max(n_m+1-r,0)}^{\max(N-r-1,0)}P^{[n-1]}((0,...,0,s)/N-r)\\
&\cdot p_0^{s+r-n_m-1}\prod_{i=1}^{m-1}p_i^{n_i^-}f(\mathbf{n})\\
&+\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}\sum_{r=0}^{N}\binom{N}{r}\frac{(n-1)^{N-r}}{n^N}\sum_{s=\max(n_m+1-r,0)}^{\max(N-r-1,0)}P^{[n-1]}((0,...,0,s)/N-r)\\
&\cdot p_m\cdot p_0^{s+r-n_m-1}f(\mathbf{n})\\
&-\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}\sum_{j=0}^{N-n_m-1}\binom{N}{j}\frac{(n-1)^{N-j}}{n^N}P^{[n-1]}((n_1,...,n_m)^-/N-j)f(\mathbf{n})g(\mathbf{n})\\
=&\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0\\ \text{and } n_{m-1}=0}}\sum_{a=0}^{N-n_m-2}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}P^{[n-1]}((n_1,...,n_m)^-/N-1-a)\\
&\cdot\frac{1}{a+1}\sum_{t=0}^{a}[1+prod(n_1,..,n_m)-n_m\cdot m+(n_m+t)prod(\mathbf{p})]\\
&+\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=0}}\sum_{r=0}^{N-1}\binom{N}{r+1}\frac{(n-1)^{N-1-r}}{n^N}\sum_{s=\max(n_m-r,0)}^{\max(N-r-2,0)}p_0^{s+r-n_m}\prod_{i=1}^{m-1}p_i^{n_i^-}f(\mathbf{n})
\end{aligned}$$

$$\cdot P^{[n-1]}((0,...,0,s)/N-1-r)$$

$$+\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}\sum_{r=0}^{N-1}\binom{N}{r+1}\frac{(n-1)^{N-1-r}}{n^N}\sum_{s=\max(n_m-r,0)}^{\max(N-r-2,0)} p_m\cdot p_0^{s+r-n_m} f(\mathbf{n})$$

$$\cdot P^{[n-1]}((0,...,0,s)/N-1-r)$$

$$-\frac{n}{N}\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}\sum_{j=0}^{N-n_m-2}\binom{N}{j+1}\frac{(n-1)^{N-1-j}}{n^N}\cdot f(\mathbf{n})\cdot g(\mathbf{n})$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j).$$

We have so far performed some algebraic manipulations that are similar to the ones that we did for proving Theorem 3.1. Continuing on, we have

$$F(\mathbf{n},N,n)=$$

$$\sum_{\substack{\text{all states with}\\ \sum_{i=1}^{m-1} n_i=1}}\sum_{a=0}^{N-n_m-2}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}P^{[n-1]}((n_1,...,n_m)/N-1-a)$$

$$\cdot\frac{1}{a+1}\sum_{t=0}^{a}[prod(n_1,..,n_m)-n_m\cdot m+(n_m+t)prod(\mathbf{p})]$$

$$+\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=0}}\sum_{r=0}^{N-1}\binom{N-1}{r}\frac{(n-1)^{N-1-r}}{n^{N-1}}\sum_{s=\max(n_m-r,0)}^{\max(N-r-2,0)}$$

$$\cdot P^{[n-1]}((0,...,0,s)/N-1-r)\cdot p_0^{s+r-n_m}\prod_{i=1}^{m-1}p_i^{n_i^-}\frac{1}{r+1}f(\mathbf{n})$$

$$+\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}\sum_{r=0}^{N-1}\binom{N-1}{r}\frac{(n-1)^{N-1-r}}{n^{N-1}}\sum_{s=\max(n_m-r,0)}^{\max(N-r-2,0)}$$

$$\cdot P^{[n-1]}((0,...,0,s)/N-1-r)\cdot p_m\cdot p_0^{s+r-n_m}\frac{1}{r+1}f(\mathbf{n})$$

$$-\sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=1}}\sum_{j=0}^{N-n_m-2}\binom{N-1}{j}\frac{(n-1)^{N-1-j}}{n^{N-1}}$$

$$\cdot P^{[n-1]}((0,...,0,n_m+1)/N-1-j)\cdot\frac{1}{j+1}f(\mathbf{n})g(\mathbf{n}).$$

Let us concentrate on a particular state of the form $(0,...,0,n_m)$. Each of these states occurs a certain number of times with the plus sign and a certain number of times with the minus sign. In the second and the third summation, this happens

whenever the $(0,...,0,n_m)$ state can lead to the state over which we are summing. In the last summation, each of the states of the form $(0,...,0,n_m)$ occurs once but it has a large weight because it gets multiplied by $g(\mathbf{n})$. Because of the form of $g(\mathbf{n})$, what we get is:

$$\begin{aligned} F(\mathbf{n},N,n) \;=\; & \sum_{\substack{\text{all states with} \\ \sum_{i=1}^{m-1} n_i=1}} \sum_{a=0}^{N-n_m-2} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} \\ & \cdot P^{[n-1]}((n_1,...,n_m)/N-1-a) \\ & \cdot \frac{1}{a+1} \sum_{t=0}^{a} [prod(n_1,..,n_m) - n_m m + (n_m+t)prod(\mathbf{p})] \\ & + \sum_{n_m \geq 1} \sum_{r=0}^{N-1} \binom{N-1}{r} \frac{(n-1)^{N-1-r}}{n^{N-1}} P^{[n-1]}((0,...,0,n_m)/N-1-r) \\ & \cdot \frac{1}{r+1} [f(0,...,0,1,n_m+r) - f(0,...,0,1,n_m-1)]. \end{aligned}$$

Thus we can conclude that $E(W^n) = F(\mathbf{n},N,n)$. This completes the proof. It is again true that $f(\mathbf{n}^-) - f(\mathbf{n}) = n_m^-$ for the states that have $n_{m-1} = 0$ and $f(\mathbf{n}^-) - g(\mathbf{n})f(\mathbf{n}) = n_m^-$ for the states that have $n_{m-1} = 1$.

3.C Appendix: Proof of Theorem 3.4

Let us state a different theorem first. This theorem is stated and proven in [10] (Section 2, Theorem 3, page 4) as an extension of the classical Ballot Theorem. The same theorem is mentioned in [1]. We will state the theorem here because we are going to use it later.

Theorem 3.6 *Suppose that an urn contains n cards marked with non-negative integers $k_1, k_2, ..., k_n$ respectively where $k_1 + k_2 + ... + k_n = k \leq n$. All the n cards are drawn without replacement from the urn. Denote by ν_r, $r = 1, 2, ..., n$, the number on the card drawn at the rth drawing. Then*

$$\Pr[\nu_1 + \nu_2 + ... + \nu_r < r \text{ for } r = 1,...,n] = 1 - \frac{k}{n},$$

provided that all the possible orders of drawing are equally probable.

Let us now state a lemma. We are going to use this lemma in the proof of Theorem 3.4 too.

Lemma 3.5 *Let v_j be the number of slots of service that the jth arrival requires. Then*

$$E(\sum_{j=1}^{N} v_j) = (p_1 + 2p_2 + ... + mp_m)N.$$

Proof: Since service times of requests are independent from each other, the result follows.

We will now use Theorem 3.6 and Lemma 3.5 to prove Theorem 3.4. Let a_i be the number of arrivals in the beginning of a particular slot i. We know that $\sum_{i=1}^{n} a_i = N$. These arrivals are ordered somehow. Let v_j be the number of slots of service that the j'th arrival requires. We can now form a vector $\mathbf{u} = (a_1, ..., a_n; v_1, ..., v_N)$. In order to get the probability of a particular state at the end of the n'th slot we need to find all the vectors $\mathbf{u}$ that can lead to that particular state and then take the weighted sum by taking into account the probability that the vector $\mathbf{u}$ occurred. Thus

$$P^{[n]}(\mathbf{n}/N) = \sum_{\substack{all\ \mathbf{u}\ that \\ lead\ to\ \mathbf{n}}} \Pr(\mathbf{u}).$$

Let us now concentrate on a particular vector $\mathbf{u}$ that leads to state $(0, ..., 0, 1, n_m)$ at the end of the n'th slot. In this case the request that is being served at the last slot needed m slots of service. If that request only needed $m-1$ slots instead of m, the state at the end of slot n would have been $(0, ..., 0, 1, 0, n_m)$. Let us consider the vector $\mathbf{u}' = (a_2, ..., a_n, a_1; v_{1+a_1}, ..., v_{N+a_1})$ where indices are considered mod$N$. In the case that $\mathbf{u}'$ occurs instead of $\mathbf{u}$ then the state at the end of slot n is going to be $(0, ..., 0, 1, 0, n'_m)$ where n'_m may be different from n_m depending on the number of arrivals, a_i, that occurred in the first slot. This is true because of the condition that $n \geq m \cdot (N-1)$. It is clear that by considering all the possible vectors $\mathbf{u}$ that lead to state $(0, ..., 0, 1, n_m)$ we get vectors that lead to state $(0, ..., 0, 1, 0, n'_m)$. Not only that; by considering all the possible vectors $\mathbf{u}$ that can lead to state $(0, ..., 0, 1, n_m)$ for all possible n_m's with the last request entering the system requiring m slots of service,

we actually get all the vectors that lead to state $(0,...,0,1,0,n'_m)$ for all possible n'_ms with the last request entering the system requiring $m-1$ slots of service. Thus we can conclude that

$$\sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,1,0,n_m)/N) = \sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,1,n_m)/N) + \frac{p_{m-1}}{p_m} \sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,1,n_m)/N).$$

By using similar arguments, we can prove that

$$\sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,n_i=1,0,...,0,n_m)/N) = \frac{(p_m+...+p_{i+1})}{p_m} \sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,1,n_m)/N). \tag{3.5}$$

We also have

$$\sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,n_m)/N) = \frac{(p_m+...+p_1)}{p_m} \sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,1,n_m)/N) + \Big[1-(p_1+2p_2+...+mp_m)\frac{N}{n}\Big].$$

We get the first term by using the same arguments that we used for obtaining relation 3.5. The difference in this case is that we have to take into account more cases. These are the cases in which the server was idle during the last slot. This is why we need the second term. We can get that second term by applying Theorem 3.6 to our problem and using the result of Lemma 3.5.

It is now clear that, since $\sum_{all\ states} P^{[n]}((n_1,...,n_m)/N) = 1$,

$$\sum_{n_m=0}^{N-1} P^{[n]}((0,...,0,1,n_m)/N) = p_m \frac{N}{n}.$$

By using these results we have:

$$\sum_{all\ states} P^{[n]}((n_1,...,n_m)/N)\cdot[prod(\mathbf{n}) - n_m \cdot m] = \frac{N}{n}p_m(m-1) + \frac{N}{n}(p_m+p_{m-1})(m-2) + \frac{N}{n}(p_m+...+p_2) = \frac{N}{n}\Big(p_m\frac{m(m-1)}{2} + p_{m-1}\frac{(m-1)(m-2)}{2} + ... + p_2\Big).$$

This completes the proof.

Bibliography

[1] A. Bhargava, P. Humblet and M. G. Hluchyj, "Queueing Analysis of Continuous Bit-Stream Transport in Packet Networks," *GLOBECOM 1989*, 25.6, pp. 903-907.

[2] V. E. Benes, *General Stochastic Processes in the Theory of Queues.* Reading, MA: Addison Wesley, 1963.

[3] L. G. Dron, G. Ramamurthy and B. Sengupta, "Delay Analysis of Continuous Bit Rate Traffic Over an ATM Network," *IEEE Journal on Sel. Areas in Communications*, Vol. 9, pp. 402-407, April 1991.

[4] A. E. Eckberg, Jr., "The Single Server Queue With Periodic Arrival Process and Deterministic Service Times," *IEEE Trans. on Communications*, Vol. 27, pp. 556-562, March 1979.

[5] A. E. Eckberg, Jr., "Response Time Analysis For Pipelining Jobs In a Tree Network of Processors," *Applied Probability - Computer Science: The Interface*, Volume 1, R. L. Disney and T. J. Ott, Eds. Boston, MA: Birkhaüser, 1982.

[6] I. Norros, J. W. Roberts, A. Simonian and J. T. Virtamo, "The Superposition of Variable Bit Rate Sources in an ATM Multiplexer," *IEEE Journal on Sel. Areas in Communications*, Vol. 9, pp. 378-387, April 1991.

[7] J. W. Roberts and J. T. Virtamo, "The Superposition of Periodic Cell Arrival Streams in an ATM Multiplexer," *IEEE Trans. on Communications*, Vol. 39, pp. 298-303, February 1991.

[8] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley Publishing Company, 1987.

[9] B. Sengupta, "A Queue with Superposition of Arrival Streams with an Application to Packet Voice Technology," *PERFORMANCE '90,* P.J.B. King, I. Mitrani and P.J. Pooley, Eds. Amsterdam, The Netherlands: North-Holland, 1990, pp. 53-60.

[10] L. Takacs, *Combinatorial Methods in the Theory of Stochastic Processes*, Wiley (New York), 1967.

[11] A. K. Wong, "Queueing Analysis for ATM Switching of Continuous-Bit-Rate Traffic–A Recursion Computation Method," *GLOBECOM 1990*, 801.2, pp. 1438-1444.

Chapter 4

The Multiple-Servers Case

The case that the main link can transport more than one connection is examined in this chapter. According to our traffic model this is the multiple-servers case. A recurrent technique for the analysis of our system is proposed. This algorithm calculates the probabilities of the states and uses them for evaluating the expected wait. We prove that there is a very simple relationship between the probabilities of the states and the expected wait. This relationship is an extension of what we proved in the previous chapter for the single-server case. For cases in which the complexity of the proposed algorithm is too large, we propose some approximate techniques for estimating the expected wait of the multiple-servers system.

4.1 Queueing Analysis

Our analysis in this chapter will be an extension of our analysis for the single-server system that we presented in the previous chapter. The state of our system is again a vector $\mathbf{n} = (n_1, ..., n_m)$, where n_i for $1 \leq i \leq m-1$ is the number of requests that require i more slots of service. The last component of the vector, n_m, is the number of requests that are waiting in the queue and it doesn't mean that each request waiting will actually require m slots of service.

We will again describe a way of calculating the probability that the system is in

state $(n_1, ..., n_m)$ at the end of a particular slot i, given that there were exactly j requests in the first i slots. We call these probabilities $P^{[i]}((n_1, ..., n_m)/j)$. We will use them for calculating the expected time that a request will have to wait until it starts getting served, which will be the criterion for measuring the performance of our system.

Before we start the analysis we need to introduce some notation. Let $\mathbf{n}^- = (n_1, ..., n_m)^-$ be the state of the system at the end of slot $i-1$ when we know that the state of the system at the end of the i'th slot is going to be $\mathbf{n}$, assuming that no new arrivals occurred at the beginning of the i'th slot and all requests that entered the service facility required m slots of service. We will refer to the components of $\mathbf{n}^-$ as n_i^- for simplicity.

The above definition is a little ambiguous for some cases. For example $(0, .., 0)^-$ can be equal to $(0, ..., 0)$, if the system was empty at the end of the previous slot, or to $(1, 0, ..., 0)$, if there was only one request in the system that needed one more slot of service. It can be actually equal to various other states depending on the value of k. The notation we are going to adopt is that $(0, ..., 0)^- = (k, 0, ..., 0)$. We formally define $(n_1, ..., n_m)^-$ to be equal to $(k - \sum_{i=1}^{m-1} n_i, n_1, ..., n_{m-2}, n_{m-1} + n_m)$. This definition has some implications. It says, for example, that $(0, ..., 0, k, -k)^- = (0, ..., 0)$. The latter doesn't have any meaning in a real system but is a notational tool that we are going to use in order to make the proofs simpler. The problem arises from the fact that we want $\mathbf{n}^-$ to be unique. This is the reason why we defined $\mathbf{n}^-$ as $(k - \sum_{i=1}^{m-1} n_i, n_1, ..., n_{m-2}, n_{m-1} + n_m)$. Our definition here is an extension of our definition in the previous chapter.

4.1.1 Constant-Service-Times

We will now concentrate on the case where the service time of each request is constant and equal to m. This means that $\mathbf{p} = (0, ..., 0, 1)$. The special case $m = 1$ was treated in [8]. The work there was an extension of the analysis for the single-server case that was similar to the continuous-time analysis in [1, 3, 4, 6].

For the probabilities of the states we will consider two cases. The first case is the case where either $n_m \geq 1$ or $\sum_{i=1}^{m-1} n_i = k$ and the second case is the case where $n_m = 0$ and also $\sum_{i=1}^{m-1} n_i < k$. The difference between these two cases is that in the first case if we know the number of arrivals that occurred in the beginning of the current slot and we also know the state at the end of the current slot then we can determine the state at the end of the previous slot. The same is not true for the second case. The reason is that we know that at the end of the current slot some of the servers are going to be idle but we don't know whether they were idle at the end of the previous slot nor whether they are about to finish serving some requests. Thus, for the first case we have:

$$P^{[i]}((n_1,...,n_m)/j) = \sum_{a=0}^{n_m+n_{m-1}} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1,...,n_m-a)^-/j-a).$$

For the second case, the case where $n_m = 0$ and $\sum_{i=1}^{m-1} n_i < k$, things are more complicated for another reason. When $m = 1$, a request can arrive at the beginning of a slot and leave at the end of the same slot, if there is a free server. Thus if $m = 1$:

$$P^{[i]}(n_1/j) = \sum_{r=0}^{k} \sum_{a=0}^{n_1} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}(r-a/j-a).$$

When $m > 1$ we have:

$$P^{[i]}((n_1,...,n_m)/j) = \sum_{r=0}^{k-\sum_{i=1}^{m-1} n_i} \sum_{a=0}^{n_m+n_{m-1}} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((r,n_1,...,n_{m-2},n_{m-1}-a)/j-a).$$

We will now introduce some notation that will help us express $E(W^n)$ as a function of the states in a somewhat simple way. Let us call $wait(n_1,...,n_m,k)$ the number of slots that a request will have to wait until it starts getting served, when there are n_i requests in the service facility that require i more slots of service (i takes all values between 1 and $m-1$), n_m requests waiting in the queue, and the number of servers is equal to k. This is equivalent to the amount of time that a request will have to wait if the system is in state $(n_1,...,n_m)$ at the time of its arrival and no other request arrived at the same slot, or, even if more requests arrived at the same slot, the request that we

consider was ordered first. Actually this function is an extension of $prod(n_1, ..., n_m)$, the function that we introduced in the previous chapter for the single-server case, i.e., when there is only one server. That function was simply equal to $\sum_{i=1}^{m} n_i \cdot i$. It is more difficult to express $wait(n_1, ..., n_m, k)$ as a function of the n_i's in the general case, though. However, we do not need to do that, because of a theorem that we will prove below, which states that $E(W^n)$ can be expressed as a simple function of the probabilities of the states. This difficulty of expressing $wait(n_1, ..., n_m, k)$ as a function of the n_i's is the reason why we cannot carry out a simplified analysis for calculating the expected wait without obtaining the probabilities of the states, as we did for the single-server case.

Using the above notation we can write:

$$E(W^n) = \sum_{a=0}^{N-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} \frac{1}{a+1} \sum_{all\ states} P^{[n-1]}((n_1, ..., n_m)/N-1-a) \cdot \sum_{t=0}^{a} wait(n_1, ..., n_{m-1}, n_m + t, k).$$

We should mention that the $E(W^i)$'s are strictly increasing as i increases up to a point, and then they stay constant. The reason is the fact that, after we have looked far enough in the past for cases that can force a request to wait, looking further in the past doesn't add any new cases.

Before proving the theorem that we mentioned above, we will need to prove two lemmas.

Lemma 4.1 *Under the conditions that* $i \geq \lceil \frac{(j-1)}{k} \rceil m$ *and* $n_m \geq 1$*:*

$$P^{[i]}((n_1, ..., n_m)^-/j) = \sum_{a=0}^{j-n_m-k} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)^-/j-a).$$

Proof: The reason why this lemma is true is merely the fact that arrivals that occurred too far in the past cannot affect the current state. A more detailed proof can be obtained by using the same arguments that were used for proving Lemma 3.1.

Lemma 4.2 *Let us define*

$$f(\mathbf{n}, k) \triangleq \sum_{t=0}^{n_m-1} [1 + wait(n_1, ..., n_{m-1}, t, k)].$$

Let $\mathcal{N}_l$ be the set of all states $(n_1, ..., n_m)$ with $f(\mathbf{n}, k) = l$. Under the condition that $i \geq \lceil \frac{j-k}{k} \rceil m$ and for all possible $\mathcal{N}_l$ we have:

$$\sum_{\mathbf{n} \in \mathcal{N}_l} P^{[i]}((n_1, ..., n_m)^-/j) = \sum_{\mathbf{n} \in \mathcal{N}_l} \sum_{a=0}^{j-n_m-k} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)^-/j-a).$$

Proof: The function $f(\mathbf{n}, k)$ is an extension of the function $f(\mathbf{n})$ that we defined in the proof of Theorem 3.1. We will use this function again when we prove the next theorem which is an extension of Theorem 3.1 for the multiple-servers case. For the cases that $i \geq \lceil \frac{(j-1)}{k} \rceil m$ we don't need to prove anything. The result follows from Lemma 4.1. For the remaining cases Lemma 4.1 doesn't hold and the reason for that is that the statement of Lemma 4.1 is stronger than the statement of this lemma. The proof for these cases is similar to the proof of Lemma 3.1. What happens in these cases is that, although the arrivals that occurred in the first slot may affect the state that will occur at the end of the i'th slot the resulting state is going to remain within the same set $\mathcal{N}_l$. This completes the proof.

We will now prove the theorem we mentioned above.

Theorem 4.1 *When $n \geq \lceil \frac{(N-k)}{k} \rceil m$ then:*

$$E(W^n) = \frac{n}{N} \sum_{all\ states} P^{[n]}((n_1, ..., n_m)/N) \cdot n_m.$$

Proof: Let us define

$$\begin{aligned} f(\mathbf{n}, k) &\triangleq \sum_{t=0}^{n_m-1} [1 + wait(n_1, ..., n_{m-1}, t, k)], \\ F(\mathbf{n}, N, n, k) &\triangleq \frac{n}{N} \sum_{\substack{all\ states \\ with\ n_m \geq 1}} [P^{[n]}(\mathbf{n}/N) - P^{[n]}(\mathbf{n}^-/N)] f(\mathbf{n}, k). \end{aligned}$$

The function $f(\mathbf{n}, k)$ is the same function that we defined in Lemma 4.2. The rest of the proof is similar to the proof of Theorem 3.1. Instead of using the lemma that we used for proving Theorem 3.1 we use Lemma 4.2 in this proof. Also the upper limits of the summations change. It is again clearly true that

$$f(\mathbf{n}^-, k) - f(\mathbf{n}, k) = n_m^-.$$

This completes the proof.

4.1.2 Variable-Service-Times

Let us now consider the most general case where we have k servers and the service times of the requests are determined by the probability vector $\mathbf{p}$.

Before we can express the probabilities of the states with the usual recursive approach we have to introduce some more notation that will make the expressions look simpler. Let us define $prob(v_1, ..., v_m, \mathbf{p}, w)$ to be the probability that among the $\sum_{i=1}^{m} v_i$ requests, v_i need i slots of service and all of these requests will leave the queue and start getting served if they need to. The probability vector $\mathbf{p}$ determines probabilistically the number of slots of service that a request will require and w is the number of servers. It is clear that if $\sum_{i=1}^{m} v_i > w$ then the value of this function is equal to zero. On the other hand it is possible that the sum of the v_i's from $i = 0$ up to m exceeds w if v_0 is large enough, i.e., there are enough requests that don't require any service at all. These requests cannot be last in the queue, though, because we are assuming that requests get served in a First-Come-First-Served order. So, even if a request doesn't need any service at all it has to wait in line until it becomes first in line. It is clear that part of this function is going to consist of products of the form ${p_i}^{v_i}$, but the main question is how many different choices there are for the different ways that we can order the requests and get an allowable ordering. It is not difficult to see that in the case that $\sum_{i=1}^{m} v_i = w$, by using the multinomial coefficient notation, we have:

$$prob(v_0, ..., v_m, \mathbf{p}, w) = \binom{\sum_{i=1}^{m} v_i}{v_1, ..., v_m} \cdot \binom{w - 1 + v_0}{v_0} \cdot \prod_{i=0}^{m} {p_i}^{v_i}.$$

When $\sum_{i=1}^{m} v_i < w$ things are a little easier because requests that don't require any service at all can be last in the queue. Thus:

$$prob(v_0, ..., v_m, \mathbf{p}, w) = \binom{\sum_{i=0}^{m} v_i}{v_0, ..., v_m} \cdot \prod_{i=0}^{m} {p_i}^{v_i}.$$

In order to be able to express the probabilities of the states with the usual approach we have to distinguish among different cases. Let us consider the case where $n_m \geq 1$ or $\sum_{i=1}^{m-1} n_i = k$ first. We have to consider two cases again. The first case is the case

where $n_{m-1} = 0$. In this case we have:

$$P^{[i]}((n_1,...,n_m)/j) =$$
$$\sum_{a=0}^{n_m} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1,...,n_m-a)^-/j-a)$$
$$+\sum_{r=0}^{j} \binom{j}{r} \frac{(i-1)^{j-r}}{i^j} \sum_{n_1^*=0}^{n_1^-} \cdots \sum_{\substack{n_{m-1}^*=0 \\ \prod_{i=1}^{m-1}(n_i^- - n_i^*)\neq 0}}^{n_{m-1}^-} \sum_{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)+n_m-r,0)}^{\max(j-r-k,0)}$$
$$\cdot P^{[i-1]}((n_1^*,...,n_m^*)/j-r)$$
$$\cdot prob(r+n_m^*-n_m-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*, ..., n_{m-1}^- - n_{m-1}^*, 0, \mathbf{p}, k-\sum_{i=1}^{m-1} n_i^*).$$

Let us consider a simple example that will show the validity of this formula. Let $N = 3$, $k = 2$, $m = 2$, $n = 3$ and $\mathbf{p} = (0.2,\ 0.3,\ 0.5)$. Let us concentrate on the probability that the system is at state $(0,1)$ at the end of the third slot and we know that there were three arrivals in the first three slots, i.e., we are looking for $P^{[3]}((0,1)/3)$. There are four different cases that can lead the system to the state that we are considering. The first case is when the system is at the all-zero state at the end of the second slot, there was no arrival in the first two slots and the first two arrivals in the beginning of the third slot required one slot of service each. The second case is when the system is at state $(1,0)$ at the end of the second slot, there was one arrival in the first two slots and the first arrival in the beginning of the third slot required one slot of service. The third case is when the sytem is at state $(2,0)$ at the end of the second slot and there were two arrivals in the first two slots. The fourth case is when the system is at state $(2,1)$ at the end of the second slot and there were three arrivals in the first two slots. In other words we have:

$$P^{[3]}((0,1)/3) = P^{[2]}((0,0)/0) \cdot \frac{(0.3)^2}{3^3} + P^{[2]}((1,0)/1) \cdot \frac{6}{3^3} \cdot 0.3$$
$$+P^{[2]}((2,0)/2) \cdot 3 \cdot \frac{2^2}{3^3} + P^{[2]}((2,1)/3) \cdot (\frac{2}{3})^3.$$

We can get the same result by using the general formula that we gave above. It is clear that $(0,1)^- = (2,1)$ and $(0,0)^- = (2,0)$. By substituting the values of our

example in that formula we get:

$$P^{[3]}((0,1)/3) = P^{[2]}((1,0)/1)\cdot\frac{6}{3^3}\cdot prob(0,1,0,\mathbf{p},1) + P^{[2]}((2,1)/3)\cdot(\frac{2}{3})^3$$
$$+P^{[2]}((2,0)/2)\cdot 3\cdot\frac{2^2}{3^3} + P^{[2]}((0,0)/0)\cdot\frac{1}{3^3}\cdot prob(0,2,0,\mathbf{p},2).$$

This completes the example. We now return to the general study of cases.

The remaining cases are similar to the one that we just described. In the case where $n_{m-1} > 0$ but still $n_m \geq 1$ or $\sum_{i=1}^{m-1} n_i = k$ we have:

$$P^{[i]}((n_1,...,n_m)/j) =$$
$$\sum_{r=0}^{j}\binom{j}{r}\frac{(i-1)^{j-r}}{i^j}\sum_{n_1^*=0}^{n_1^-}\cdots\sum_{n_{m-1}^*=0}^{n_{m-1}^-}\sum_{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)+n_m+n_{m-1}-r,0)}^{\max(j-r-k,0)} P^{[i-1]}((n_1^*,...,n_m^*)/j-r)$$
$$\cdot prob(r+n_m^*-n_m-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*,...,n_{m-1}^- - n_{m-1}^*, n_{m-1}, \mathbf{p},$$
$$k-\sum_{i=1}^{m-1} n_i^*).$$

In the case where $n_m = n_{m-1} = 0$ and $\sum_{i=1}^{m-1} n_i < k$ we have:

$$P^{[i]}((n_1,...,n_{m-2},0,0)/j) =$$
$$P^{[i-1]}((k-\sum_{i=1}^{m-2} n_i, n_1,...,n_{m-2},0)/j)(\frac{i-1}{i})^j$$
$$+\sum_{r=0}^{j}\binom{j}{r}\frac{(i-1)^{j-r}}{i^j}\sum_{\substack{n_1^*=0\\ n_1^*\neq k-\sum_{i=1}^{m-2} n_i \text{ when}\\ k-\sum_{i=2}^{m-1}(n_{i-1}-n_i^*)=0}}^{k-\sum_{i=1}^{m-2} n_i}\sum_{n_2^*=0}^{n_1}\cdots\sum_{n_{m-1}^*=0}^{n_{m-2}}\sum_{n_m^*=\max(\sum_{i=2}^{m-1}(n_{i-1}-n_i^*)-r,0)}^{\max(j-r-k,0)}$$
$$\cdot P^{[i-1]}((n_1^*,...,n_m^*)/j-r)\sum_{q=0}^{\min(k-\sum_{i=1}^{m-2} n_i - n_1^*,\ n_m^*+r-\sum_{i=2}^{m-1}(n_{i-1}-n_i^*))} prob(r+n_m^*-n_m-$$
$$\sum_{i=2}^{m-1}(n_{i-1}-n_i^*)-q, q, n_1-n_2^*,...,n_{m-2}-n_{m-1}^*, 0, \mathbf{p}, k-\sum_{i=1}^{m-1} n_i^*).$$

In the case where $n_{m-1} > 0$ but $n_m = 0$ and $\sum_{i=1}^{m-1} n_i < k$ we have:

$$P^{[i]}((n_1,...,n_{m-1},0)/j) =$$

$$\sum_{r=0}^{j}\binom{j}{r}\frac{(i-1)^{j-r}}{i^j}\sum_{n_1^*=0}^{k-\sum_{i=1}^{m-1}n_i}\sum_{n_2^*=0}^{n_1}\cdots\sum_{n_{m-1}^*=0}^{n_{m-2}}\sum_{n_m^*=\max(n_{m-1}+\sum_{i=2}^{m-1}(n_{i-1}-n_i^*)-r,0)}^{\max(j-r-k,0)}$$

$$\cdot P^{[i-1]}((n_1^*,...,n_m^*)/j-r)\sum_{q=0}^{min(k-\sum_{i=1}^{m-1}n_i-n_1^*,n_m^*+r-n_{m-1}-\sum_{i=2}^{m-1}(n_{i-1}-n_i^*))} prob(r+n_m^*-n_m-$$

$$\sum_{i=2}^{m-1}(n_{i-1}-n_i^*)-n_{m-1}-q,q,n_1-n_2^*,...,n_{m-2}-n_{m-1}^*,n_{m-1},\mathbf{p},k-\sum_{i=1}^{m-1}n_i^*).$$

We can now use these probabilities to write an expression for $E(W^n)$:

$$E(W^n) = \sum_{a=0}^{N-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}\frac{1}{a+1}\sum_{all\ states}P^{[n-1]}((n_1,...,n_m)/N-1-a)$$

$$\cdot\sum_{t=0}^{a} wait(n_1,...,n_{m-1},n_m+t,k).$$

We will now prove a lemma which is similar to Lemma 4.1 but holds for variable-service-times.

Lemma 4.3 *Under the conditions that* $i \geq \lceil\frac{(j-2)}{k}\rceil m+m$ *and* $n_m \geq 1$*:*

$$P^{[i]}((n_1,...,n_m)^-/j) = \sum_{a=0}^{j-n_m-k}\binom{j}{a}\frac{(i-1)^{j-a}}{i^j}P^{[i-1]}((n_1,...,n_m)^-/j-a).$$

Proof: The reason why this lemma is true is again merely the fact that arrivals that occurred too far in the past cannot affect the current state. In this case though we have to be a little more careful. Although requests that occurred far in the past may not have been able to affect the current state of the system if all the requests had constant-service-times, they may affect the current state if variable-service-times are allowed. The reason for that is that they may allow requests with shorter service times to occur in between. For that reason the conditions under which Lemma 4.3 holds are different from the conditions under which Lemma 4.1 holds.

By using this lemma we can now prove a theorem that holds for variable-service-times. The theorem is proven in Appendix 4.A.

Theorem 4.2 *When* $n \geq \lceil\frac{(N-2)}{k}\rceil m+m$ *then:*

$$E(W^n) = \frac{n}{N}\sum_{all\ states}P^{[n]}((n_1,...,n_m)/N)\cdot n_m.$$

This expression that relates the expected wait with the probabilities of the states can be proved by using Little's result. The difference is that Little's result can be used only under a stricter condition, i.e., $n \geq \lceil \frac{N}{k} \rceil m + m$ (see Appendix 4.B).

4.2 Improving the Efficiency of the Algorithm

The expression that we have from Lemma 4.3 gives us an alternative for calculating the probabilities of the states. This expression is more efficient and simpler to use than the general recursive relations that we have. The only problem is that we cannot always use it since there are those two conditions that have to be satisfied: these are $i \geq \lceil \frac{(j-2)}{k} \rceil m + m$ and $n_m \geq 1$. What we will do now is to try to relax those conditions as much as possible. If we do that we will be able to use the simpler relation more often and thus increase the performance of our algorithm.

Let us introduce some notations first. Consider a vector $(n_0, ..., n_{m-1})$ of dimension m with non-negative components. Define n_i as the number of objects that have length i. Let us order these objects according to their length by putting the ones with shorter length first. We assume that this ordering is cyclic, which means that if there are k objects in total, then the first one is also the $k+1$'st one. We define $length(n_0, ..., n_{m-1}, j)$ to be the length of the j'th object in that ordering.

We will first concentrate on the case of constant-service-times.

Lemma 4.4 *Under the conditions that* $i \geq \lceil \frac{(j-n_m)}{k} \rceil m - length(k - \sum_{l=1}^{m-1} n_l, n_1, ..., n_{m-1}, k - ((j - n_m - k) mod\ k) + 1) + 1$ *and* $n_m \geq 1$*:*

$$P^{[i]}((n_1, ..., n_m)/j) = \sum_{a=0}^{j-n_m-k} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)/j - a).$$

Lemma 4.5 *Under the conditions that* $i \geq \lceil \frac{(j-\sum_{l=1}^{m-1} n_l)}{k} \rceil m + m - length(k - \sum_{l=1}^{m-1} n_l, n_1, ..., n_{m-1}, k - ((j - \sum_{l=1}^{m-1} n_l) mod\ k) + 1) + 1$ *and* $n_m = 0$*:*

$$P^{[i]}((n_1, ..., n_m)/j) = \sum_{a=0}^{j-\sum_{l=1}^{m-1} n_l} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)/j - a).$$

The proofs of these two lemmas are the same as the proofs of Lemmas 3.2 and 3.3 and therefore omitted.

It is very important that, especially when the service times of the requests is constant, the conditions of the two lemmas are almost always satisfied. What is actually true in the case of constant-service-times is that the probabilities of the states are zero up to a particular slot. After that slot the results of the lemmas hold. So we only have to use the complicated recursions for finding the probabilities of the states at the slots that take a non-zero value for the first time.

Another important point is this. The results of the lemmas relate the probability of a particular state to the probabilities of the same state at the previous slot for various number of arrivals. What it means is that, because of the lemmas, when calculating the probabilities of a particular state, we can forget about the probabilities of the other states. So, instead of having to operate on a huge matrix, we only have to operate on vectors. This decreases both the number of operations and the storage requirements, which is why the algorithm is so efficient for the special case considered in [8].

What we mentioned for the constant-service-times case is not true for the general service time case. The reason is that the probabilities of the slots are non-zero even from the first slot. This means that we have to use the complicated relations for several times before we can use the results of the lemmas.

Let us now take advantage of another fact. We have mentioned in the previous sections that $E(W^i)$ is strictly increasing up to one point and then stays the same. We may be able to use that fact to stop our algorithm before it actually reaches the last slot, because after $E(W^i)$ stops increasing the algorithm doesn't give us any more information. This is indeed true and will be clear after the next theorem.

We will need to prove a lemma before proving the theorem.

Lemma 4.6 *Let* $n_* = \lceil \frac{(N-k)}{k} \rceil m$. *Let* $\mathcal{N}_l$ *be the set of all states* $(n_1, ..., n_m)$ *with*

$f(\mathbf{n},k)=l$ *and* $n_m \geq 1$. *If* $n \geq n_*$ *and for all possible* $\mathcal{N}_l$ *we have:*

$$\sum_{\mathbf{n}\in\mathcal{N}_l} P^{[n]}((n_1,...,n_m)/N) = \sum_{\mathbf{n}\in\mathcal{N}_l} \sum_{r=n_m+k}^{N} \binom{N}{r} \frac{(n_*)^r(n-n_*)^{N-r}}{n^N} P^{[n*]}((n_1,...,n_m)/r).$$

Proof: By using similar arguments to the ones that we used for proving Lemma 4.2 we can write:

$$\sum_{\mathbf{n}\in\mathcal{N}_l} P^{[n]}((n_1,...,n_m)/N) =$$
$$\sum_{\mathbf{n}\in\mathcal{N}_l} \sum_{a=0}^{N-n_m-k} \binom{N}{a} \frac{(n_*)^{N-a}(n-n*)^a}{n^N} P^{[n*]}((n_1,...,n_m)/N-a).$$

The only difference is that, instead of expressing the probabilities of the states at the end of slot n by using the states at the end of slot $n-1$, we do that by using the probabilities of the states at the end of slot n_*. By making a change of variable the result follows.

Now we are ready to prove the theorem.

Theorem 4.3 *Let* $n_* = \lceil \frac{(N-k)}{k} \rceil m$. *If* $n \geq n_*$ *then:*

$$E(W^n) = \frac{n}{N} \sum_{all\ states} \sum_{r=n_m+k}^{N} \binom{N}{r} \frac{(n_*)^r(n-n_*)^{N-r}}{n^N} P^{[n_*]}((n_1,...,n_m)/r) \cdot n_m.$$

Proof: By using the result of Theorem 4.1 and the previous lemma the result follows.

It is now clear that in order to calculate $E(W^n)$, we only need to use the recursive relations and the results of the lemmas to calculate the probabilities of the states at the end of slot n_*. After we do that, we can use the result of the last theorem to get $E(W^n)$.

We will now consider the variable-service-times case.

Lemma 4.7 *Under the conditions that* $i \geq \lceil \frac{(j-n_m)}{k} \rceil m - length(k - \sum_{l=1}^{m-1} n_l, n_1, ...,$ $n_{m-1}, k-((j-n_m-k) mod\ k)+1)+1$, $j-n_m-k \leq k$ *and* $n_m \geq 1$*:*

$$P^{[i]}((n_1,...,n_m)/j) = \sum_{a=0}^{j-n_m-k} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1,...,n_m)/j-a).$$

Proof: The proof of this lemma is similar to the proof of Lemma 4.3. There is a difference though. The difference is that the total number of arrivals in this case is very small. For that reason the phenomenon that we described in the proof of Lemma 4.3 cannot occur.

Lemma 4.8 *Under the conditions that* $i \geq \lceil \frac{(j-n_m-1)}{k} \rceil m + m - length(k - \sum_{l=1}^{m-1} n_l, n_1, ..., n_{m-1}, k - ((j - n_m - k - 1) mod\ k) + 1) + 1$, *and* $n_m \geq 1$:

$$P^{[i]}((n_1, ..., n_m)/j) = \sum_{a=0}^{j-n_m-k} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)/j - a).$$

Proof: The proof is again based on the proof of Lemma 4.3. The worst case occurs if one request occurs at the first slot. Because of the conditions of the lemma, requests that occurred even in the next $m-1$ slots cannot have any effect on the current state. Actually even if all the remaining requests occurred at the m'th slot they cannot affect the current state. The reason for being so conservative is the phenomenon that we described in the proof of Lemma 4.3. This completes the proof.

In the same way we can prove the next two lemmas for the case that n_m is equal to 0. We will omit the proofs.

Lemma 4.9 *Under the conditions that* $i \geq \lceil \frac{(j-\sum_{l=1}^{m-1} n_l)}{k} \rceil m + m - length(k - \sum_{l=1}^{m-1} n_l, n_1, ..., n_{m-1}, k - ((j - \sum_{l=1}^{m-1} n_l) mod\ k) + 1) + 1$, $j - \sum_{l=1}^{m-1} n_l \leq k$ *and* $n_m = 0$:

$$P^{[i]}((n_1, ..., n_m)/j) = \sum_{a=0}^{j-\sum_{l=1}^{m-1} n_l} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)/j - a).$$

Lemma 4.10 *Under the conditions that* $i \geq \lceil \frac{(j-\sum_{l=1}^{m-1} n_l-1)}{k} \rceil m + m - length(k - \sum_{l=1}^{m-1} n_l, n_1, ..., n_{m-1}, k - ((j - \sum_{l=1}^{m-1} n_l - 1) mod\ k) + 1) + m + 1$ *and* $n_m = 0$:

$$P^{[i]}((n_1, ..., n_m)/j) = \sum_{a=0}^{j-\sum_{l=1}^{m-1} n_l} \binom{j}{a} \frac{(i-1)^{j-a}}{i^j} P^{[i-1]}((n_1, ..., n_m)/j - a).$$

From Lemma 4.7 it is clear that for the case $N-1-k \leq k$ we can make a stronger statement than the one we did in Theorem 4.2.

Theorem 4.4 *When* $n = \lceil \frac{(N-1)}{k} \rceil m$ *and* $N - 1 - k \leq k$.

$$E(W^n) = \frac{n}{N} \sum_{all\ states} P^{[n]}((n_1, ..., n_m)/N) \cdot n_m.$$

We will omit the proof of this theorem because it is identical to the proof of Theorem 4.2.

By using similar ideas to the ones that we used for proving Theorem 4.3 we can prove the following theorem. The proof is very similar to the proof of Theorem 4.3 and therefore will be omitted.

Theorem 4.5 *Let* $n_* = \lceil \frac{(N-2)}{k} \rceil m + m$. *If* $n \geq n_*$ *then:*

$$E(W^n) = \frac{n}{N} \sum_{all\ states} \sum_{r=n_m+k}^{N} \binom{N}{r} \frac{(n_*)^r (n-n_*)^{N-r}}{n^N} P^{[n_*]}((n_1, ..., n_m)/r) \cdot n_m.$$

4.3 A Simple Example

We are now going to give a simple example that will make the ideas that we have expressed in the previous sections more concrete. We are going to assume that there are two servers, three customers and only three slots. The requests need two slots of transmission with probability 0.5, one slot with probability 0.3 and don't require any transmission with probability 0.2. In other words $k = 2$, $N = 3$, $n = 3$, $m = 2$ and $\mathbf{p} = (0.2,\ 0.3,\ 0.5)$. Let us concentrate on the probabilities of the states at the end of the last slot given that three arrivals occurred at the first three slots. The reason for doing that is that, because of our queueing model, we know that exactly three arrivals are going to occur in the first three slots. We have called these probabilities $P^{[3]}((n_1, n_2)/3)$. There are six possible sates: $(0,0)$, $(1,0)$, $(2,0)$, $(0,1)$, $(1,1)$ and $(2,1)$. We are only going to consider the last three because these are the only ones that we need for $E(W^3)$.

The only way that the system can be at state $(2,1)$ at the end of the third slot is if all the arrivals occurred at the beginning of the third slot and the first two arrivals required two slots of service. Thus:

$$P^{[3]}((2,1)/3) = P^{[2]}((0,0)/0) \cdot \frac{1}{3^3}(0.5)^2.$$

The system can be at state $(1,1)$ at the end of the third slot if either all the arrivals occurred at the beginning of the third slot and one of the first two arrivals required

j \ n	(0,0)	(1,0)	(2,0)	(0,1)	(1,1)	(2,1)
0	1.000	0.000	0.000	0.000	0.000	0.000
1	0.500	0.500	0.000	0.000	0.000	0.000
2	0.250	0.500	0.250	0.000	0.000	0.000
3	0.080	0.180	0.100	0.090	0.300	0.250

Table 4.1: Values of $P^{[i]}((n_1,n_2)/j)$ for $i=1$

two slots of service and the other one just one, or the system was at state $(1,0)$ at the end of the second slot, two arrivals occurred at the beginning of the third slot and the first of them required two slots of service. Thus:

$$P^{[3]}((1,1)/3) = P^{[2]}((0,0)/0)\cdot\frac{1}{3^3}\cdot 2\cdot 0.3\cdot 0.5 + P^{[2]}((1,0)/0)\cdot 3\cdot\frac{2}{3^3}\cdot 0.5.$$

We have already shown that

$$\begin{aligned} P^{[3]}((0,1)/3) &= P^{[2]}((0,0)/0)\cdot\frac{(0.3)^2}{3^3} + P^{[2]}((1,0)/1)\cdot\frac{6}{3^3}\cdot 0.3 \\ &\quad + P^{[2]}((2,0)/2)\cdot 3\cdot\frac{2^2}{3^3} + P^{[2]}((2,1)/3)\cdot(\frac{2}{3})^3. \end{aligned}$$

In Tables 4.1, 4.2 and 4.3 we show the actual values of the probabilities of the states for $i=1$, 2 and 3 respectively obtained by our algorithm.

From Theorem 6 we know that

$$E(W^3) = P^{[3]}((0,1)/3) + P^{[3]}((1,1)/3) + P^{[3]}((2,1)/3).$$

Because of Lemma 10 we also have

$$P^{[3]}((2,1)/3) = P^{[2]}((2,1)/3)\cdot(\frac{2}{3})^3.$$

By using the above relations we get:

$$\begin{aligned} E(W^3) &= P^{[2]}((0,0)/0)\cdot\frac{1}{3^3}[2\cdot(0.5)^2 + 2\cdot 0.5\cdot 0.3 + (0.3)^2] \\ &\quad + P^{[2]}((1,0)/1)\cdot 3\cdot\frac{2}{3^3}[0.3+0.5] + P^{[2]}((2,0)/2)\cdot 3\cdot\frac{4}{3^3}. \end{aligned}$$

j \ n	(0,0)	(1,0)	(2,0)	(0,1)	(1,1)	(2,1)
0	1.000	0.000	0.000	0.000	0.000	0.000
1	0.750	0.250	0.000	0.000	0.000	0.000
2	0.563	0.375	0.063	0.000	0.000	0.000
3	0.286	0.300	0.059	0.193	0.131	0.031

Table 4.2: Values of $P^{[i]}((n_1, n_2)/j)$ for $i = 2$

j \ n	(0,0)	(1,0)	(2,0)	(0,1)	(1,1)	(2,1)
0	1.000	0.000	0.000	0.000	0.000	0.000
1	0.833	0.167	0.000	0.000	0.000	0.000
2	0.694	0.278	0.028	0.000	0.000	0.000
3	0.498	0.352	0.045	0.057	0.039	0.009

Table 4.3: Values of $P^{[i]}((n_1, n_2)/j)$ for $i = 3$

We can get the same result by directly using the definition of $E(W^3)$. As we have explained before $E(W^3)$ is the average waiting time that a request that arrives at the third slot will experience. So, we are given that a request will arrive at the beginning of the third slot. In this case it is easy to directly use the definition of $E(W^3)$ because the example is simple. What we get is:

$$\begin{aligned} E(W^3) &= \frac{1}{3^2}\cdot\frac{1}{3}\cdot P^{[2]}((0,0)/0)[2\cdot(0.5)^2+2\cdot 0.5\cdot 0.3+(0.3)^2] \\ &\quad +2\cdot\frac{2}{3^2}\cdot\frac{1}{2}P^{[2]}((1,0)/1)\cdot[0.3+0.5]+\frac{4}{3^2}P^{[2]}((2,0)/2). \end{aligned}$$

It is clear that both relations for $E(W^3)$ are equivalent.

4.4 Numerical Example

Let us assume that the number of customers is equal to 45, the total number of slots is equal to 64 and the number of servers is equal to 3, i.e., $N = 45$, $n = 64$ and $k = 3$. The probability vector $\mathbf{p}$ that determines the service times of requests is equal to (0.2, 0.2, 0.2, 0.2, 0.2), so $m = 4$. In Figure 4.1 we see the probability distribution for n_m. What we have in that figure is $\sum_{n_1,\ldots,n_{m-1}} P^{[64]}(n_1,\ldots,n_m)$ plotted versus n_m. The probability that $n_m = 0$ is very large, 0.9183. The probability that n_m is larger than 15 drops below 10^{-10}.

4.5 Some Approximate Techniques

The algorithm that we proposed and discussed in the previous sections can be used for solving the queueing problem that we are considering for arbitrary values of N, k, m and $\mathbf{p}$. The complexity of the algorithm however can be very large if N, k and m are fairly large. The reason is that the number of states, i.e., the number of allowable vectors $(n_1,\ldots,n_m)$, is equal to $\binom{k+m-1}{m-1}\cdot(N-k+1)$.

In this section we will propose some approximate techniques that can be used in the case that the complexity of our algorithm is too large. We will concentrate

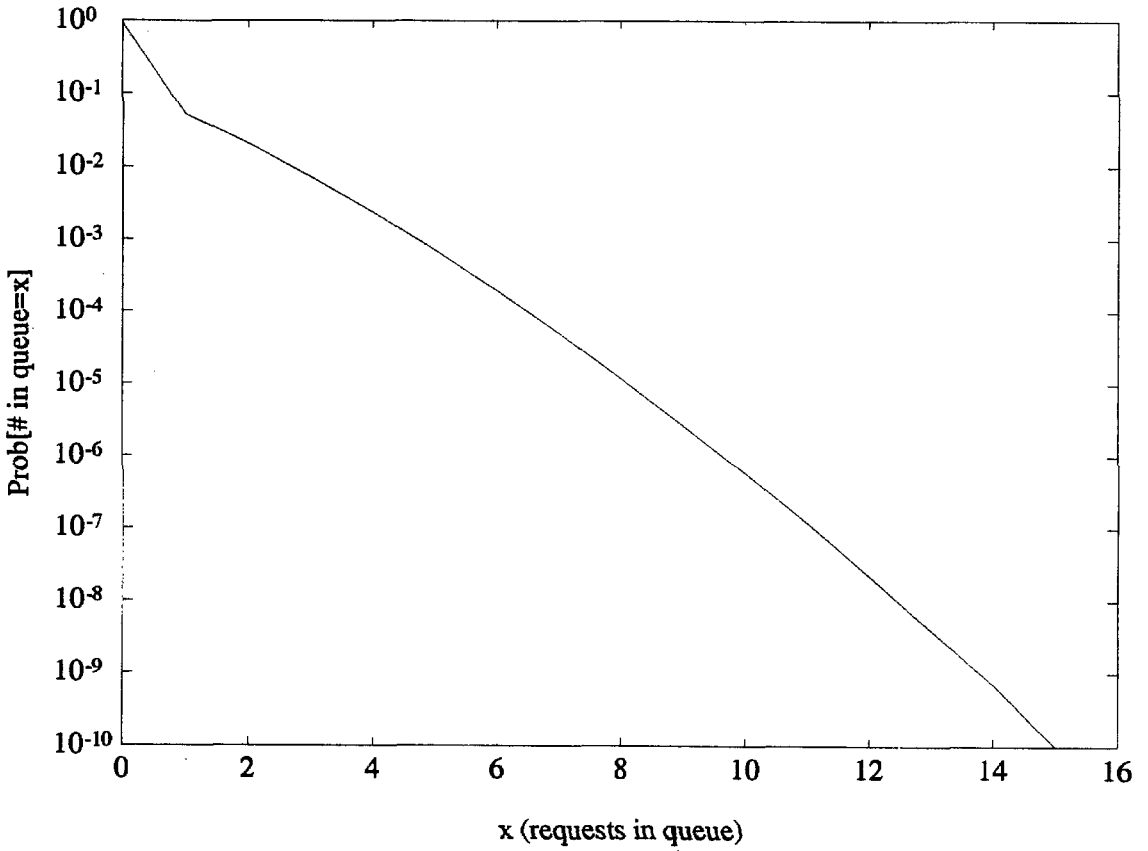

Figure 4.1: Probability Distribution for $N = 45$

on the expected wait although these approximate techniques can also be used for approximating the probabilities of the states. The reason is that we are mainly interested in the expected wait.

Let us assume that $n \geq \lceil \frac{(N-2)}{k} \rceil m + m$. The reason for doing that is that in order to be able to get the expected wait we need to use Theorem 4.2. That theorem holds only if this condition is satisfied. Later in this section we will discuss what happens when this condition is not satisfied.

Let us define $E(W^n, N, k, \mathbf{p})$ to be the expected wait that requests that arrive at the last slot experience when there are N customers, the system has k servers and service times of arrivals are determined by the probability vector $\mathbf{p}$. We have been

calling this quantity just $E(W^n)$, and the dependencies on the other parameters were only implicit.

We are going to consider what happens if instead of having k servers, we actually have only one server that is k times faster. The way we are going to do that is by considering a system with a single server and observe it for a window that has $n \cdot k$ slots. The duration of one slot in the system with k servers is equivalent to the duration of k slots in the system with one server. Our claim is that $E(W^n, N, k, \mathbf{p}) \leq \frac{1}{k} E(W^{n \cdot k}, N, 1, \mathbf{p})$. The reason is that if we consider a particular arrival pattern that in the second system leads to a state $\mathbf{n}$ at the end of the last slot, this same arrival pattern leads to state $\mathbf{n}'$ in the first system with $n'_m \leq n_m$. Therefore, by using Theorem 4.2 we get that $E(W^n, N, k, \mathbf{p}) \leq \frac{1}{k} E(W^{n \cdot k}, N, 1, \mathbf{p})$. We call this approximation Model 1. The reason for the scale factor $\frac{1}{k}$ is that the expected wait is measured in number of slots and the duration of a slot in the first system is equivalent to the duration of k slots in the second system.

It is indeed mentioned in [5], that if we want to minimize the expected wait that arrivals experience, it is better to have many servers than having just one fast server. The reason is that when there are few arrivals, let us say less than k, they won't need to wait at all in the first system, while they may need to wait in the second.

Furthermore, as N and n increase with their ratio remaining constant the ratio of the expected wait for the two systems approaches 1. As we mentioned earlier, a particular arrival pattern that in the system with the single-server leads to a state $\mathbf{n}$ will lead to state $\mathbf{n}'$ in the multiple-servers system with $n'_m \leq n_m$. This n'_m, though, is not much smaller than n_m. Therefore, as N and n become large this difference loses its significance.

The reason for approximating the multiple-servers system with a single-server system is that we can use the simplified analysis proposed in the previous chapter. This analysis is only valid for the single-server case but can give us the expected wait for very large numbers of N and m.

Let us now consider what happens if we rescale time. Instead of observing a

system with k servers, N customers and a probability vector $\mathbf{p_1} = (p_{1,0}, ..., p_{1,m})$ for a time window that has n slots, we observe a system that has k servers and N customers for a time window that has only $\frac{n}{m}$ slots. The probability vector that determines the service times in the second system is $\mathbf{p_2} = (p_{2,0},\ p_{2,1})$ where $p_{2,1} = \frac{1}{m}\sum_{i=0}^{m} p_{1,i}$ and $p_{2,0} = 1 - p_{2,1}$. The reason for doing something like this is that the complexity of our algorithm for the second system is smaller than the complexity of our algorithm for the first one. The number of states for the second system is only $N - k + 1$ compared to $\binom{k+m-1}{m-1} \cdot (N - k + 1)$ for the first. So, if we can get an estimate of the expected wait in the first system by using the result for the second system, we will be able to analyze systems that we couldn't analyze otherwise because of the complexity of the algorithm.

We showed in the previous chapter that if we do something similar for the single-server case, we find that the expected wait for the first system is a scaled version of the expected wait for the second system. One would expect something like that to hold because the second system is very similar to the first system except for a rescaling of time. A similar relation is not true in general for the multiple-servers case, though. By using this rescaling argument we only get an approximate expression in this case. Our claim is that

$$E(W^n, N, k, \mathbf{p_1}) \approx E(W^{\frac{n}{m}}, N, k, \mathbf{p_2}) \frac{\sum_{i=0}^{m} p_{1,i} \cdot i^2}{\sum_{i=0}^{m} p_{1,i} \cdot i}.$$

We call this approximation Model 2. The reason why this approximation makes sense, is that as we mentioned earlier in this section, the expected wait for the first system will approach the expected wait of a particular single-server system and the expected wait of the second system will approach the expected wait of a different single-server system. By using Theorem 3.5 we can find the ratio of the expected waits of the single-server systems. Thus, the ratio of the two multiple-servers systems will approach that ratio as N and n increase with the ratio of N and n remaining constant. The reason why this relation is not exact in general is the fact that the expected waits of the muliple-servers systems may not approach the expected waits of the single-server systems at the same speed.

There are some cases, though, where the previous relation is exact as the following theorem states.

Theorem 4.6 *Consider two systems. In the first system the total number of slots is* n_1*, the total number of arrivals is* N*, the number of servers is* k *and the probability vector that determines the service times is* $\mathbf{p_1} = (p_{1,0},\ 0, ...,\ 0,\ p_{1,m})$*. In the second system the total number of slots is* n_2*, the total number of arrivals is* N*, the number of servers is* k *and the probability vector that determines the service times is* $\mathbf{p_2} = (p_{2,0},\ p_{2,1})$*, where* $p_{2,1} = p_{1,m}$ *and* $p_{2,0} = p_{1,0}$*. Let* $n_1 = n_2 \cdot m$ *and* $n_2 \geq \lceil \frac{(N-2)}{k} \rceil + 1$*. Let* $E_1(W^{n_1})$ *and* $E_2(W^{n_2})$ *be the expected wait that requests in the first and second system experience respectively. Then*

$$E_1(W^{n_1}) = E_2(W^{n_2}) \cdot m.$$

Proof: The main idea is to group slots in the first system in groups of m. The first m slots belong to the first group, the second m to the second and so on. A request that arrives at the i'th slot in the second system may arrive at any of the slots of the i'th group in the first system. No matter in which of the m slots of that group the request arrived, the last component of the state vector of the first system at the end of the window of n_1 slots that we are considering is going to be unaffected. Not only that: this last component of the state vector is going to be the same as the last component of the state vector at the last slot of the second system. This completes the proof.

Let us now consider what happens if the condition $n \geq \lceil \frac{(N-2)}{k} \rceil m + m$ is *not satisfied.* If $k \geq \frac{N \cdot \sum_{i=0}^{m} p_i \cdot i}{n}$ we expect Model 1 and 2 to still give us quite accurate results. Theorem 4.2 may not be exactly true but it still gives us an accurate estimate of the expected wait. The reason is that the proof of the theorem is based on the assumption that our system has enough capacity to empty the queue within a time window of n slots if all requests arrived in the first slot. The worst case is if all requests needed m slots of service. In that case k will need to be greater or equal to $\frac{N \cdot m}{n}$ which is approximately the condition of Theorem 4.2. It is very unlikely, though,

that all requests will need m slots of service. We expect that on the average each request will only need about $\sum_{i=0}^{m} p_i \cdot i$ slots of service, i.e., will need the average service time. Thus we expect that our theorem and thus our two models will be a good approximation for cases in which $k \geq \frac{N \cdot \sum_{i=0}^{m} p_i \cdot i}{n}$.

We now need to consider what happens if $k \leq \frac{N \cdot \sum_{i=0}^{m} p_i \cdot i}{n}$. In this case the rate at which new work arrives in the system is faster than the service rate. Clearly, this situation results in long waiting times for the requests. Motivated by the fluid-flow approximation for continuous-time queues (see [5] Section 2.7 pp. 56-62) we will propose another approximation model which we call Model 3. According to the fluid-flow approximation we can replace stochastic processes by their average values as a function of time. The reason is that the average values are large compared to the fluctuations.

In our approximate model, we assume that all the arrivals are uniformly distributed over the time window of duration n in a deterministic manner, i.e., in each slot there are $\frac{N}{n}$ arrivals. The results that we are going to get by making this assumption are not going to be very different from the exact results. This is because the servers are almost always busy, so the exact time of occurrences of requests does not much change the expected wait of requests that arrive at the last slot. We make another assumption, too. We assume that service times of requests are exactly equal to the average service time and are not determined by the probability vector $\mathbf{p}$. This is especially justified for large N because then the expected wait of requests at the last slot is affected by the service time of a large number of other requests. So, the average service time of those requests is very close to the actual average service time of requests.

By using the two assumptions that we made, i.e., that there are $\frac{N}{n}$ arrivals in each slot and that the service time of each request is equal to the average service time of requests, we can determine the state of the system at the end of slot $n-1$. By using the random ordering assumption for the $\frac{N}{n}$ requests that arrive at the last slot, we can get the expected wait for these requests. This can be done easily because of the

constant-service-time assumption. Therefore, this model can be used for obtaining the expected wait under heavy load even for a large number of requests.

There are some problems with this approach. The first problem comes from the fact that $\frac{N}{n}$ may not be an integer. So, it may not be very easy to uniformly distribute the N arrivals in the n slots. In case N is fairly large the different ways of distributing the requests will give very similar results, though. The next problem comes from the fact that the average service time of requests may not be an integer. In that case we can average the result for the two closest integers and that will give us a good estimate of the average wait for the case in which we are interested.

4.6 Numerical Examples of Approximate Models

Let us now concentrate on a particular example. We assume that there are 100 customers, i.e., $N = 100$. As we mentioned in Chapter 2 we expect that the segments will last for about 3 minutes. For this reason we assume that the duration of each slot is 3 minutes. The probability vector that determines the service times of the requests has the form $\mathbf{p} = (\ 0,\ 0,\ 0,\ 0,\ 0.1,\ 0.2,\ 0.4,\ 0.2,\ 0.1,\ 0,\ 0)$. The program of a particular customer lasts for 10 slots but from the form of $\mathbf{p}$ it is clear that at most 8 of the segments will need to be transmitted from the main node to the intermediate node. The fact that the program of each customer consists of 10 segments is realistic because that means that the duration of the program will be 30 minutes. The choice of $\mathbf{p}$ is arbitrary but we expect that in reality it will have the form that we assume in this example. We believe that there will be a peak and the components of the probability vector will decrease rapidly as we move to the left and the right of that peak. The reason is that we think that there will be a number of segments that most of the customers would like to get and these segments are going to be stored in the intermediate node. In our example this number is 4. The rest of the segments that cover the personal preferences of the different subscribers are going to be stored in the main node.

The form of $\mathbf{p}$ enables us to use $m = 8$ in our algorithm what decreases the amount of calculations that we need to do. The total number of slots is 64, i.e., $n = 64$. As we explained in Chapter 2, customers are allowed to tune into the network within a 3 hour window. Since the duration of each slot is 3 minutes, 60 would be a good choice for the total number of slots. We chose 64 because we would prefer this number to be divisible by m which is 8 in our case.

In Figure 4.2 we compare the results that we obtained from Models 1 and 2 and from our simulation. Both in the theoretical models and simulation we assumed that requests arrive exactly in the beginning of a slot. This is not realistic because requests of the customers may occur anytime during a slot. For that reason we added an extra half slot delay to all the results that we obtained. This is the reason why the expected wait approaches 0.5 and not 0 as the number of servers increases.

In Figure 4.2 we can see that the results of Model 2 are in excellent agreement with the simulation results for $k \geq 10$, i.e., $k \geq \frac{N \cdot \sum_{i=0}^{m} p_i \cdot i}{n}$. The results of Model 1 overestimate the expected wait but even this model is in good agreement with the behavior of the expected wait as a function of the number of servers. On the other hand we see that for $k < 10$ neither of our models is in good agreement with the simulation results, as we expected. For the cases that $10 \leq k < 15$, the results of Models 1 and 2 compare well with simulation, although Theorem 4.2 does not hold. The reasons for this agreement were explained in the previous section.

Let us now discuss whether we may be interested in the cases that $k < 10$. As we see from Figure 4.2, the expected wait increases exponentially as k decreases. For example the expected wait measured in number of slots goes from 2 for $k = 10$ to 5 for $k = 9$. This is to be expected because the system does not have enough capacity and the arrival rate is larger than the service rate, which results in requests accumulating in the queue. In this region, by a very small increase in the number of servers, we can get a large improvement in the performance of our system, i.e., the expected wait will decrease drastically.

Let us now consider what happens if we increase the number of subscribers N to

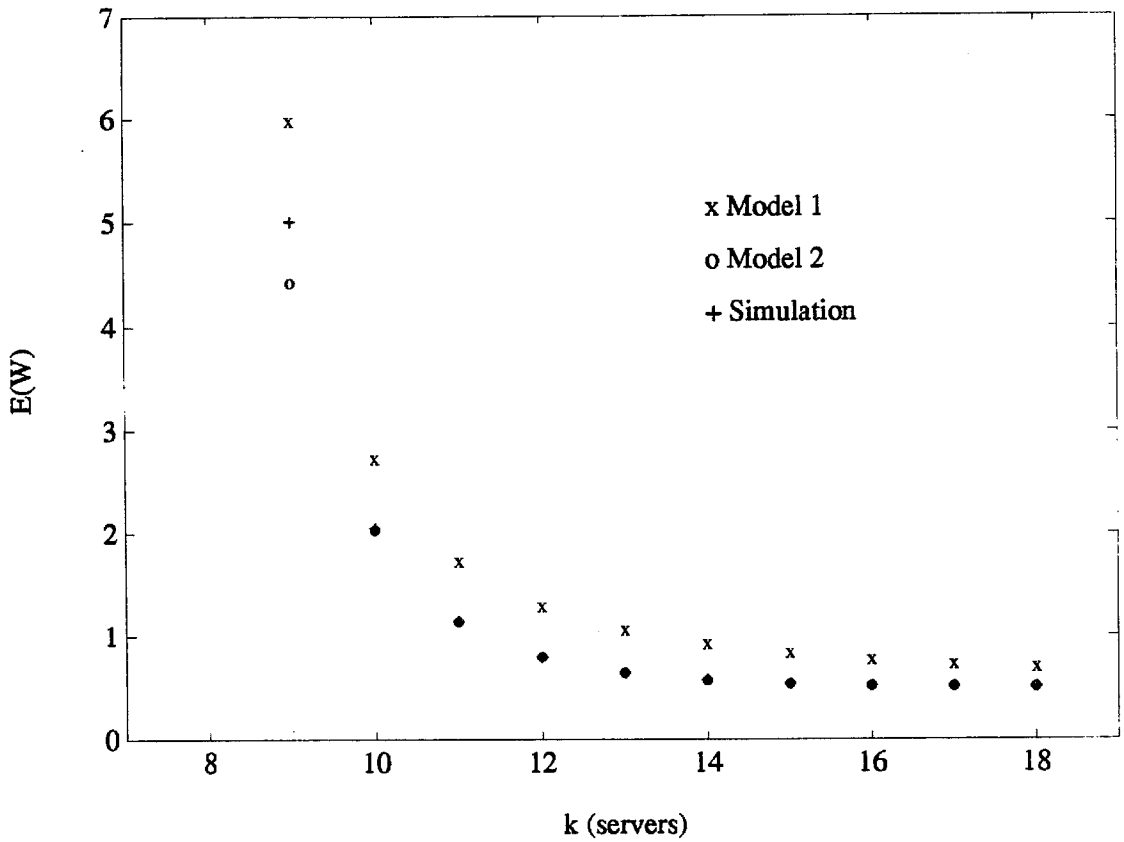

Figure 4.2: $E(W)$ in number of slots for $N = 100$

1000. We keep all other parameters the same. In this case Model 2 cannot be used because the complexity of the algorithm is prohibitory. In Figure 4.3 we compare the results of Models 1 and 3 with simulation. One important point is that now we are interested in the cases $k < 94$, i.e., $k < \frac{N \cdot \sum_{i=0}^{m} p_i \cdot i}{n}$. The reason is that although in the beginning the expected wait increases exponentially, as we move away from the case $k = 94$ the increase becomes only linear. We didn't observe this phenomenon in the case $N = 100$ because we stopped within the area of the exponential increase. As we increase N, the increase in the expected wait becomes smoother when we decrease the number of servers. We observe in Figure 4.3 that Model 1 captures the behavior of the system when $k \geq 94$ and Model 3 captures the behavior of the system when

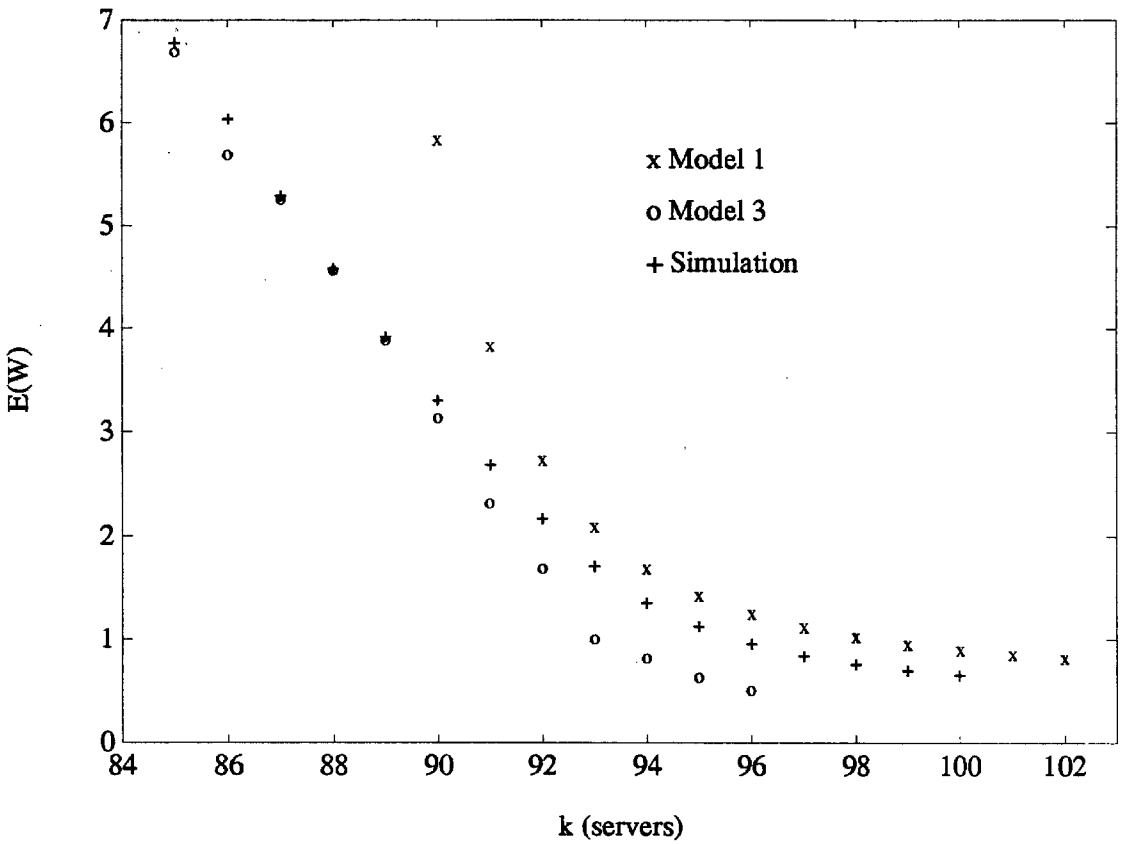

Figure 4.3: $E(W)$ in number of slots for $N = 1000$

$k < 94$.

From the numerical examples it is clear that we should choose the number of servers k to be around $\frac{N \cdot \sum_{i=0}^{m} p_i \cdot i}{n}$. The exact choice of k depends on the desirable level of performance and on economical factors. In order to make the choice we are equipped with our three approximate models. In the case that N is of the order of 100 subscribers, Models 1 and 2 can be used for obtaining good approximations. When N is of the order of 1000, Models 1 and 3 can be used instead.

4.7 Summary

A new recurrent algorithm for the analysis of our system for the cases that the main link can transport more than one video connection was proposed. We showed how this algorithm can be used to obtain the probabilities that our system is at a particular state. We proved that there is a simple relation which we can use for calculating the expected wait after we have obtained the probabilities of the states.

For the cases that the complexity of our algorithm is too large, we proposed some approximate techniques. We showed that the results of these approximations compare well with simulations. Equipped now with these approximations, we can decide what the number of servers k should be so that the customers obtain the desirable level of service even if the number of customers N is fairly large.

4.A Appendix: Proof of Theorem 4.2

Let us define

$$f(\mathbf{n},k) \triangleq \sum_{t=0}^{n_m-1}[1+wait(n_1,...,n_{m-1},t,k)],$$

$$g(\mathbf{n},k) \triangleq \Big[\sum_{n_1^*\geq n_2^-}\cdots\sum_{n_{m-2}^*\geq n_{m-1}^-}\sum_{n_{m-1}^*=0}^{k-\sum_{i=1}^{m-2}n_i^*-n_1^-}\sum_{n_m^*=1}^{n_m^--n_{m-1}^*-\sum_{i=1}^{m-1}(n_i^*-n_{i+1}^-)} f(n_1^*,...,n_m^*,k)$$
$$\cdot prob\Big(\sum_{i=1}^{m-1}(n_i^*-n_{i+1}^-)-n_{m-1}^*-n_m^*, n_1^*-n_2^-,...,n_{m-2}^*-n_{m-1}^-,$$
$$n_{m-1}^*,\mathbf{p},k-\sum_{i=1}^{m-1}(n_i)^-)\Big]\cdot\frac{1}{f(\mathbf{n},k)},$$

$$F(\mathbf{n},N,n,k) \triangleq \frac{n}{N}\sum_{\substack{all\ states\\ with\ n_m\geq 1\\ and\ n_{m-1}=0}}[P^{[n]}(\mathbf{n}/N)-P^{[n]}(\mathbf{n}^-/N)]f(\mathbf{n},k)$$
$$+\frac{n}{N}\sum_{\substack{all\ states\\ with\ n_m\geq 1\\ and\ n_{m-1}\geq 1}}[P^{[n]}(\mathbf{n}/N)-g(\mathbf{n},k)P^{[n]}(\mathbf{n}^-/N)]f(\mathbf{n},k).$$

Since the proof is similar to the proof of Theorem 3.2 we will go through some of the steps with little explanation. By omitting a few of the steps that we have seen

before and using Lemma 4.3 we get:

$$
\begin{aligned}
F(\mathbf{n},N,n,k) = & \\
& \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0\\ \text{and} n_{m-1}=0}} \sum_{a=0}^{N-n_m-k-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} P^{[n-1]}((n_1,...,n_m)^-/N-1-a) \\
& \cdot \frac{1}{a+1} f(n_1,...,n_m+a+1,k) \\
& - \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0\\ \text{and} n_{m-1}=0}} \sum_{j=0}^{N-n_m-k-1} \binom{N-1}{j} \frac{(n-1)^{N-1-j}}{n^{N-1}} P^{[n-1]}((n_1,...,n_m)^-/N-1-j) \\
& \cdot \frac{1}{j+1} f(n_1,...,n_m,k) \\
& + \frac{n}{N} \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}=0}} \sum_{r=0}^{N} \binom{N}{r} \frac{(n-1)^{N-r}}{n^N} \sum_{n_1^*=0}^{n_1^-} \cdots \sum_{\substack{n_{m-1}^*=0\\ \prod_{i=1}^{m-1}(n_i^- - n_i^*)\neq 0}}^{n_{m-1}^-} \sum_{\substack{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)\\ +n_m-r,0)}}^{\max(N-r-k,0)} \\
& \cdot P^{[n-1]}((n_1^*,...,n_m^*)/N-r) \cdot f(\mathbf{n},k) \\
& \cdot prob(r+n_m^*-n_m-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*,...,n_{m-1}^- - n_{m-1}^*, 0, \mathbf{p}, \\
& \qquad k-\sum_{i=1}^{m-1} n_i^*) \\
& + \frac{n}{N} \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}>0}} \sum_{r=0}^{N} \binom{N}{r} \frac{(n-1)^{N-r}}{n^N} \sum_{n_1^*=0}^{n_1^-} \cdots \sum_{n_{m-1}^*=0}^{n_{m-1}^-} \sum_{\substack{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)\\ +n_m+n_{m-1}-r,0)}}^{\max(N-r-k,0)} \\
& \cdot P^{[n-1]}((n_1^*,...,n_m^*)/N-r) \cdot f(\mathbf{n},k) \\
& \cdot prob(r+n_m^*-n_m-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*,...,n_{m-1}^- - n_{m-1}^*, n_{m-1}, \mathbf{p}, \\
& \qquad k-\sum_{i=1}^{m-1} n_i^*) \\
& - \frac{n}{N} \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 1\\ \text{and } n_{m-1}>0}} \sum_{j=0}^{N-n_m-1} \binom{N}{j} \frac{(n-1)^{N-j}}{n^N} P^{[n-1]}((n_1,...,n_m)^-/N-j) \\
& \cdot f(\mathbf{n},k) g(\mathbf{n},k) \\
= & \sum_{\substack{\text{all states}\\ \text{with } n_m\geq 0\\ \text{and } n_{m-1}=0}} \sum_{a=0}^{N-n_m-k-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} P^{[n-1]}((n_1,...,n_m)^-/N-1-a)
\end{aligned}
$$

$$\cdot\frac{1}{a+1}\sum_{t=0}^{a}[1+wait(n_1,..,n_m+t,k)]$$

$$+\frac{n}{N}\sum_{\substack{all\ states\\ with\ n_m\geq 1\\ and\ n_{m-1}=0}}\sum_{r=0}^{N-1}\binom{N}{r+1}\frac{(n-1)^{N-1-r}}{n^N}\sum_{n_1^*=0}^{n_1^-}\cdots\sum_{\substack{n_{m-1}^*=0\\ \prod_{i=1}^{m-1}(n_i^- - n_i^*)\neq 0}}^{n_{m-1}^-}$$

$$\sum_{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)+n_m-r-1,0)}^{\max(N-r-k-1,0)} P^{[n-1]}((n_1^*,...,n_m^*)/N-r-1)\cdot f(\mathbf{n},k)$$

$$\cdot prob(r+1+n_m^*-n_m-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*,...,n_{m-1}^- - n_{m-1}^*, 0, \mathbf{p},$$

$$k-\sum_{i=1}^{m-1}n_i^*)$$

$$+\frac{n}{N}\sum_{\substack{all\ states\\ with\ n_m\geq 1\\ and\ n_{m-1}>0}}\sum_{r=0}^{N-1}\binom{N}{r+1}\frac{(n-1)^{N-1-r}}{n^N}\sum_{n_1^*=0}^{n_1^-}\cdots\sum_{n_{m-1}^*=0}^{n_{m-1}^-}\sum_{\substack{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)\\ +n_m+n_{m-1}-r-1,0)}}^{\max(N-r-k-1,0)}$$

$$\cdot P^{[n-1]}((n_1^*,...,n_m^*)/N-r-1)\cdot f(\mathbf{n},k)\cdot prob(r+1+n_m^*-n_m$$

$$-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*,...,n_{m-1}^- - n_{m-1}^*, n_{m-1}, \mathbf{p}, k-\sum_{i=1}^{m-1}n_i^*)$$

$$-\frac{n}{N}\sum_{\substack{all\ states\\ with\ n_m\geq 1\\ and\ n_{m-1}>0}}\sum_{j=0}^{N-n_m-k-1}\binom{N}{j+1}\frac{(n-1)^{N-1-j}}{n^N}$$

$$\cdot P^{[n-1]}((n_1,...,n_m)^-/N-1-j)\cdot f(\mathbf{n},k)\cdot g(\mathbf{n},k)$$

$$=\sum_{\substack{all\ states\ with\\ \sum_{i=1}^{m-1}n_i=k}}\sum_{a=0}^{N-n_m-k-1}\binom{N-1}{a}\frac{(n-1)^{N-1-a}}{n^{N-1}}P^{[n-1]}((n_1,...,n_m)/N-1-a)$$

$$\cdot\frac{1}{a+1}\sum_{t=0}^{a}wait(n_1,..,n_m+t,k)$$

$$+\sum_{\substack{all\ states\\ with\ n_m\geq 1\\ and\ n_{m-1}=0}}\sum_{r=0}^{N-1}\binom{N-1}{r}\frac{(n-1)^{N-1-r}}{n^N}\sum_{n_1^*=0}^{n_1^-}\cdots\sum_{\substack{n_{m-1}^*=0\\ \prod_{i=1}^{m-1}(n_i^- - n_i^*)\neq 0}}^{n_{m-1}^-}$$

$$\sum_{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*)+n_m-r-1,0)}^{\max(N-r-k-1,0)} P^{[n-1]}((n_1^*,...,n_m^*)/N-r-1)\cdot\frac{1}{r+1}\cdot f(\mathbf{n},k)$$

$$\cdot prob(r+1+n_m^*-n_m-\sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*,...,n_{m-1}^- - n_{m-1}^*, 0, \mathbf{p},$$

$$k - \sum_{i=1}^{m-1} n_i^*)$$

$$+ \sum_{\substack{\text{all states} \\ \text{with } n_m \geq 1 \\ \text{and } n_{m-1} > 0}} \sum_{r=0}^{N-1} \binom{N-1}{r} \frac{(n-1)^{N-1-r}}{n^N} \sum_{n_1^*=0}^{n_1^-} \cdots \sum_{n_{m-1}^*=0}^{n_{m-1}^-} \sum_{n_m^*=\max(\sum_{i=1}^{m-1}(n_i^- - n_i^*) + n_m + n_{m-1} - r - 1, 0)}^{\max(N-r-k-1,0)}$$

$$\cdot P^{[n-1]}((n_1^*, ..., n_m^*)/N - r - 1) \cdot \frac{1}{r+1} \cdot f(\mathbf{n}, k) \cdot prob(r + 1 + n_m^* - n_m$$

$$- \sum_{i=1}^{m-1}(n_i^- - n_i^*), n_1^- - n_1^*, ..., n_{m-1}^- - n_{m-1}^*, n_{m-1}, \mathbf{p}, k - \sum_{i=1}^{m-1} n_i^*)$$

$$- \sum_{\substack{\text{all states} \\ \text{with } n_m \geq 1 \\ \text{and } n_{m-1} > 0}} \sum_{j=0}^{N-n_m-k-1} \binom{N-1}{j} \frac{(n-1)^{N-1-j}}{n^N} P^{[n-1]}((n_1, ..., n_m)^-/N - 1 - j)$$

$$\cdot \frac{1}{j+1} \cdot f(\mathbf{n}, k) \cdot g(\mathbf{n}, k)$$

$$= \sum_{\substack{\text{all states with} \\ \sum_{i=1}^{m-1} n_i = k}} \sum_{a=0}^{N-n_m-k-1} \binom{N-1}{a} \frac{(n-1)^{N-1-a}}{n^{N-1}} P^{[n-1]}((n_1, ..., n_m)/N - 1 - a)$$

$$\cdot \frac{1}{a+1} \sum_{t=0}^{a} wait(n_1, .., n_m + t, k)$$

$$+ \sum_{\substack{\text{all states with} \\ \sum_{i=1}^{m-1} n_i < k}} \sum_{r=0}^{N-n_m-k-1} \binom{N-1}{r} \frac{(n-1)^{N-1-r}}{n^{N-1}} P^{[n-1]}((n_1, ..., n_m)/N - 1 - r)$$

$$\cdot \frac{1}{r+1}[f(n_1, ..., n_{m-1} + k - \sum_{i=1}^{m-1} n_i, n_m + r, k)$$

$$- f(n_1, ..., n_{m-1} + k - \sum_{i=1}^{m-1} n_i, n_m - 1, k)].$$

It is again true that $f(\mathbf{n}^-, k) - f(\mathbf{n}, k) = n_m{}^-$ for the states that have $n_{m-1} = 0$ and $f(\mathbf{n}^-, k) - g(\mathbf{n}, k)f(\mathbf{n}, k) = n_m{}^-$ for the states that have $n_{m-1} > 0$. This completes the proof.

4.B Appendix: Expression for $E(W^n)$ using Little's Result

In this Appendix, we will prove by using Little's result that when $n \geq \lceil \frac{N}{k} \rceil m + m$,

$$E(W^n) = \frac{n}{N} \sum_{\text{all states}} P^{[n]}((n_1, ..., n_m)/N) \cdot n_m.$$

Our proof will be based on the following version of Little's result that is stated and

proven in [2, 7].

Theorem 4.7 *Let $L(x)$ be the number of customers present at time x, and define the mean number L of customers present throughout the time interval $[0, \infty)$ as*

$$L = \lim_{t \to \infty} \frac{1}{t} \int_0^t L(x)dx;$$

let $N(t)$ be the number of customers who arrive in $[0, t]$, and define the arrival rate λ as

$$\lambda = \lim_{t \to \infty} \frac{N(t)}{t};$$

let W_i be the waiting time of the i'th customer, and define the mean waiting time W as

$$W = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} W_i.$$

If λ and W exist and are finite, then so does L, and

$$L = \lambda W.$$

Up to now we have been considering a time window with n slots. Requests are allowed to occur within this window. Let us now consider what happens if we require that each customer makes a request every n slots, i.e., if a customer makes a request at the i'th slot, he also makes a request at the $n + i$'th slot, the $2n + i$'th and so on. The system starts operating at the beginning of the first slot and keeps operating forever. If we consider n consecutive slots, we see that the arrival distribution is the same as in the first n slots. Exactly N requests arrive at this arbitrary window of n slots and they are uniformly distributed throughout this window. Therefore, we can use the $P^{[n]}((n_1, ..., n_m)/N)$'s as the probabilities that the system is at state $(n_1, ..., n_m)$ at any of the slots after the n'th.

When computing $P^{[n]}((n_1, ..., n_m)/N)$ for the first n slots, we assume that the system is empty in the beginning. In general this may not be true for an arbitrary window of n slots. Since $n \geq \lceil \frac{N}{k} \rceil m + m$, the initial state has no effect on the state of the system at the last slot of the time window that we are considering. Therefore,

the probability that the system is at state $(n_1, ..., n_m)$ at any of the slots after the n'th slot is just $P^{[n]}((n_1, ..., n_m)/N)$. Similarly, the average wait stays the same after the n'th slot.

In order to use Theorem 4.7, we need to prove that the arival rate and the mean waiting time are finite. The arrival rate is $\frac{N}{n}$ since we know that there are N arrivals every n slots. The mean waiting time is finite for the following reason. The queue empties at least once within a time window of n slots. This is because $n \geq \lceil \frac{N}{k} \rceil m + m$. Therefore, none of the requests stays in the queue for more than n slots. Since the waiting time of each request is bounded by n, the average waiting time is bounded by n, too.

By applying Theorem 4.7 in our problem we get that

$$\sum_{all\ states} \text{Prob(system at state } (n_1, ..., n_m)) \cdot n_m = \frac{N}{n} E(W).$$

Since the probabilities of the states and the average wait stay the same after the n'th slot, we can ignore the first $n-1$ slots and conclude that

$$E(W^n) = \frac{n}{N} \sum_{all\ states} P^{[n]}((n_1, ..., n_m)/N) \cdot n_m.$$

Our arguments here are however valid only if $n \geq \lceil \frac{N}{k} \rceil m + m$. Therefore, Theorem 4.2 is slightly stronger. One example is the case $m = 1$, $\mathbf{p} = (0,\ 1)$. Theorem 4.2 says that n can be equal to $N - 1$ and the expression for the expected wait is still valid. In that case, our proof based on Little's result does not apply because the arrival rate is faster than the service rate and the queue for the system with customers that make requests every n slots grows beyond any bounds; moreover, the average waiting time is not finite.

Bibliography

[1] V.E. Benes, *General Stochastic Processes in the Theory of Queues.* Reading, MA: Addison Wesley, 1963.

[2] R.B. Cooper, *Introduction to Queueing Theory,* CEEPress Books, Washington D.C., 1990.

[3] A.E. Eckberg, Jr., "The Single Server Queue With Periodic Arrival Process and Deterministic Service Times," *IEEE Trans. on Communications*, Vol. 27, pp. 556-562, March 1979.

[4] A.E. Eckberg, Jr., "Response Time Analysis For Pipelining Jobs In a Tree Network of Processors," *Applied Probability - Computer Science: The Interface*, Volume 1, R. L. Disney and T. J. Ott, Eds. Boston, MA: Birkhaüser, 1982.

[5] L. Kleinrock, *Queueing Systems Volume II: Computer Applications,* Wiley Interscience, 1976.

[6] J.W. Roberts and J.T. Virtamo, "The Superposition of Periodic Cell Arrival Streams in an ATM Multiplexer," *IEEE Trans. on Communications*, Vol. 39, pp. 298-303, February 1991.

[7] S. Stidham, Jr., "A Last Word on $L = \lambda W$," *Operations Research*, Vol 22, pp. 417-421, March-April 1974.

[8] A.K. Wong, "Queueing Analysis for ATM Switching of Continuous-Bit-Rate Traffic–A Recursion Computation Method," *GLOBECOM 1990*, 801.2, pp. 1438-1444.

Chapter 5

Epilogue

In this thesis, we proposed a residential application for interactive video. After describing the service, we modeled it and proposed a network architecture that can support it. The question that we considered was what the capacity of the main link should be so that the customers receive a good grade of service. The measure of the goodness is the delay that customer requests experience.

There are some questions related to this architecture that we didn't consider. We assumed that the segments that are used for creating the programs of the subscribers are stored either in the main node or in the intermediate node. We did not explain how the service provider is going to determine which segments are going to be stored in the intermediate node and which in the main node. This is a difficult problem; some initial attempts have been made in [2] but the problem is far from solved.

We asserted that popular material should be stored in the intermediate node. It is not clear though what a good measure of popularity is and what level of popularity justifies introducing more storage in the intermediate node rather than having larger capacity in the main link. The answer depends heavily on the actual cost of storage and capacity. We have not studied this further.

After it has been decided where each segment is going to be located, we can compute the probability vector $\mathbf{p}$ that determines the service times of the requests. This probability vector was very important in our analysis. We made some comments

on the form of $\mathbf{p}$ but it is clear that more work needs to be done for getting good estimates on $\mathbf{p}$. It also needs to be determined whether $\mathbf{p}$ is actually constant in time as we assumed or not. In a practical system, we may even want to make $\mathbf{p}$ depend on the actual traffic. What we mean by that is that we may want to offer better service when there are only a few requests but if the system gets congested, popular segments may be sent to every customer so that the customers will not have to wait for a long time. In that case the analysis becomes much more involved and it is not clear that the expected wait can be expressed as a simple function of the probabilities of the states anymore.

Another interesting problem is to consider different ways for selecting segments that are going to be used for creating the program of a particular subscriber. One simple approach has been proposed in [1]. A neural network is used for ranking segments for each customer. Only the profile of the particular customer and the keywords of the segments are taken into account. It may be better to take into account profiles of other customers too, so that multicasting can be used more often.

There are many problems that need to be considered before interactive video services can be introduced. Our contribution is that we proposed an approach for analyzing such services. Although we used this approach for analyzing a particular service, the only significant assumption that we made was that video information is delivered in segments. We believe that this is going to be the case in many interactive video applications, becauses it makes multicasting much easier. Therefore, we hope that our approach will be used for analyzing a wide class of such services and that it will stimulate further research and development in this area.

Bibliography

[1] E. C. Posner and P. N. Mouchtaris, "Interactive Video on Demand," *Joint FAW-IEEE Workshop*, Germany, September 1990.

[2] R. Ramarao and V. Ramamoorthy, "Architectural Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem," *ICC '91*, 17.6, pp. 506-510, June 1991.