A COMPARATIVE STUDY OF CENTRALISED PROTOCOLS FOR MULTIPLE ACCESS.

JOHN BOYD

B. Eng., McGill University
(1980)

Submitted in Partial Fulfillment
of the Requirements for the
Degree of

MASTER OF ELECTRICAL ENGINEERING

at

McGill University
© (December 1982)

Sommaire

On traite un cas simple mais fondamental du problème de la gestion d'un système de télécommunications à accès multiple. On suppose un ensemble fini de terminaux, un trafic aléatoire, un seul canal binaire et synchronisé, et un contrôle centralisé. Un message soumis au système lorsque le canal est occupé doit attendre. On suppose que *l'état* dū système – soit le nombre et les origines des messages en attente – est connu du contrôleur. Le problème est de déterminer comment ordonner les transmissions et comment signaler au récepteur les origines et les délimitations des messages, tel que regler efficacement la durée moyenne d'attente.

Dans ce mémoire, on évalue et compare systématiquement une gamme de stratégies de contrôle et de signalisation. La durée d'une *trame* – soit le temps écoulé entre deux instants consecutifs auxquels on peut commencer transmettre d'un terminal donné – peut être fixe ou variable. Pour des modes d'accès fonctionnant à partir d'une répartition dans le temps à trames fixes, et pour de différents codes signalant la présence d'une mémoire tampon vide, on calcule les statistiques du nombre de messages en attente et du retard subi par un message. Pour des modes d'accès dont la durée d'une trame varie selon l'état du système, on obtient le retard moyen à partir d'une combinaison d'analyse et de simulations. On présente la performance retard-débit pour les stratégies optimales de chaque classe, les paramètres optimaux dépendant en général du trafic.

Abstract

The multi-access problem is considered in its simplest setting: a set of sources obtain access to a synchronous channel via a centralised arbiter with complete state information. The role of the arbiter is to select the next user to access the channel and to include along with the user-data sufficient information for the receiver to determine the origin, beginning and end of the current transmission.

The work consists of a systematic evaluation and comparision of a variety of fixed-frame strategies and variable-cycle schemes. The steady-state moment-generating functions of queue size and virtual delay are derived for generalised *TDMA* systems employing different techniques for encoding underflow. Mean delay results are obtained, using a combination of simulation and analysis, for various state-dependent polling disciplines. The analytical work is based on the theory of Markov renewal processes.

Table of Contents

List of Figures

Notation

$A^{(n)}(t)$	–	number of Poisson arrivals in the interval $[0,t)$ of the n^{th} frame or cycle.
$B^{(n)}(t)$	–	total amount of work brought by the arrivals in the interval $[0,t)$ of the n^{th} frame or cycle.
$\mathrm{E}[z^F]$	–	$\sum_{j=0}^{\infty} z^i \Pr\{F=i\}$
	–	steady-state moment-generating function of the discrete random variable F.
$(x)^+$	–	$\max(0,x)$.
$\lfloor x \rfloor$	–	integer portion of x.
$\lceil x \rceil$	–	nearest integer greater than or equal to x.
ω_l	–	$e^{i2\pi l/K}$.
z^*	–	complex conjugate of z.

Acknowledgements

I would like to express my gratitude to Professor Michael Kaplan for his interest, assistance and guidance in this endeavour. Also, I wish to thank all those affiliated with Bell-Northern Research/INRS-Telecommunications who made my two year sojourn there pleasant, stimulating and informative. Finally, special thanks are owed to Dr. Maier Blostein, director of the institute, whose efforts made it possible for myself and others from McGill University to be associated with such a unique concept in graduate education.

Chapter I: Introduction

Whenever there is more than one demand for a limited resource, a rationing policy is needed to assure the fair and efficacious distribution of the restricted quantity. The limited resource of interest here is channel bandwidth in a communications network; demands for it are made by the subscribers to the service, while the selection of the rationing policy forms the crux of what is known as the multi-access problem.

Network access schemes can be categorized according to the quantity and utility of state information available to the decision-making process. If the state information is local, delayed or prohibitvely expensive to gather — as can be the case with a large, dispersed population of bursty users — then the access policies must operate at a distributed (local) level. On the other hand, if complete global state information is available at little or no cost and without significant delay, then centralised arbitration can be employed.

The basic difference between the two types of policies is that a centralised algorithm grants access to only a single user at any given time, whereas, in a distributed system, access is open to all users, and consequently, conflicts between users will occur. Because of this difference, centralised control policies are aimed at finding efficient methods of selecting the next user to be granted access, while distributed algorithms seek to resolve conflicts, once detected, as soon as possible.

Throughout this work we shall be concerned solely with centralised access schemes. The basic model will be that of a synchronous, bandlimited channel accessed by a finite number of sources proximate to a central arbiter. The controller has complete state-information which can be used to schedule access to the channel and is obtained without affecting any of the system performance measures with which we shall be concerned. The role of the arbiter is to select the next user to be granted access to the channel and to determine the allotment for that user. Also, the controller must include along with the user-data sufficient information[†] to enable the receiver

i) to determine the origin and length of the current transmission;

ii) and, because the channel is synchronous, to separate data transmissions from channel idle periods.

The inclusion of supervisory information requires a portion of the available capacity and, is therefore, the overhead in the system.

The purpose of this thesis is to provide a systematic evaluation and comparision of a variety of reasonable access policies based on time division multiple access (*TDMA*) and polling strategies, with specific attention paid to the structure and effect of the overhead necessary to convey control information to the receiver.

This thesis differs from most other work on multi-access for a combination of reasons. First, the control is centralised; thus, it is removed from the domain of the *ALOHA*-type problem. Second, overhead is accounted for explicitly and its effects are the central focus; this distinguishes it from work on *CPU*-scheduling and from most of the work on cyclic queuing systems. Finally, no special characters are reserved exclusively for control purposes in the manner of byte-oriented protocols.

In the remainder of this chapter we make precise our model and assumptions, review related literature and outline the work.

[†] A qualitative discussion of some of the possible methods of achieving these goals is given in [11], which provides an in-depth introduction to the problem we consider.

1. Model

As mentioned previously, the basic model employed in this work is that of a finite number of sources communicating with a single receiver via a common resource: the transmission medium. Access to the resource is determined by a central arbiter. The sources are independent of each other and all produce messages according to a Poisson process. Associated with each source is a buffer of infinite size. The average arrival rates are, in general, heterogeneous. Message lengths are deterministic and have unit length, measured in bits or slots.

The central arbiter, alternatively known as the concentrator, switch or server, allocates the resource to each source according to a fixed conflict-free strategy. Throughout this work we shall be concerned solely with cyclical channel allocation strategies; the quantity of bandwidth allocated to each source at any given time may be either fixed or variable. The concentrator also inserts — before, after or within the user-data — the overhead necessitated by the protocol. The arbiter may obtain perfect state information as to the contents of each buffer without cost.

The channel is synchronous, errorless, has zero-propagation delay and is slotted. The capacity of the channel, measured in bits per slot ($b/slot$) is one.

Finally, we shall assume that the interarrival times between messages arriving at a buffer are of no interest to the network or the receiver. Consequently, source idle periods can be ignored; this is *not* true of server idle periods.

In summary, the assumptions we have made are as follows:

i) independent, heterogeneous Poisson sources;

ii) infinite buffers;

iii) messages of length 1 bit;

iv) perfect state information available to the controller;

v) cyclic source-selection strategy;

vi) single destination;

vii) error-free, zero-delay, synchronous channel with capacity 1 *b/slot*;

viii) source idle periods contain no information.

Assumptions *i*) and *ii*) were made to facilitate analysis; *iii*) not only simplifies the analysis, it causes the role of the overhead to be emphasized, due to the near parity of source message lengths and overhead. Number *iv*) gives us the information necessary to employ adaptive or state-dependent service stategies. That the acquisition of this knowledge will not have any adverse effect on the performance or cost of the system is, in general, not the case.

The reason for restricting ourselves to a cyclic service strategy is that of all the strategies one can imagine it is the "fairest", in that the buffer which has been waiting longest for service is always served next. This eliminates the problem of one buffer being "frozen-out" of service by another. Also, by judiciously employing the state information and transmitting it to the receiver, cyclic strategies are easily modified to "skip-over" certain buffers, thereby allowing us to model other selection stategies.

That we allow only one destination is not an insurmountable restriction if we consider only point-to-point exchanges; for any number of source-receiver pairs can modelled by an increased number of sources. However, increasing the number of sources cannot accomodate broadcast or multi-destination transmissions.

The penultimate assumption is in fact four: the channel being error-free and without delay implies that our problem is one of source coding, not channel coding. That the channel is synchronous forms the heart of our problem, since the protocol must communicate to the receiver not only the data, but also the location of server idle periods. A slotted channel is in accord with common practice.

Finally, that source idle periods are neglected has two implications. The first is that our work is not applicable to voice traffic, since idle periods contain information. The second is that the source itself must provide the receiver with sufficient information to delimit its bursts, if required.

2. Related Work

The body of work concerned with the systematic evaluation and comparision of the effects of state-dependent protocol overhead in multi-access systems is exiguous. Most of the recorded work simply proceeds under the assumption that the overhead requires a portion of the channel capacity which is independent of the state of the system and accounts for the supervisory information either by increasing the message length or by incorporating a fixed or random 'switching' time into the model. As a result, there exists a vast amount of work which has as its central focus the calculation of system penalty functions (for example, delay) rather than the overhead required to implement the service protocols. Examples of this type of work can be found in [5,6,7,18,27], which are related to our work only in that they provide potential analytical techniques for the evaluation of system performance.

The first inquiry to quantify the effects of protocol overhead was performed by Gallager [10]. He derived basic limits on the overhead necessary for the encoding of source busy and idle periods as a function of the service delay, assuming the user population is large and there is no queuing delay. and unlimited channel capacity. Futhermore, he observed that source addressing amounts to sorce coding the starting times of messages. Thus, by suitably delaying the transmission of messages, it is possible to reduce the average protocol overhead employed in the network. In Gallager's work the delay was a coding delay and did not include the inevitable queuing delays of a bandlimited and finite user system.

Humblet [15] provided a more detailed examination of source coding for concentrators under less restrictive assumptions. The problem he considered was the encoding of message beginnings, lengths and origins when the system is comprised of a finite number of sources and a bandlimited, synchronous channel. Humblet provided numerous contributions to the study of protocol overhead. One was to recognise that for a synchronous channel the average number of overhead bits (that is, those which are not data bits) per data bit is a meaningless measure: if idle bits are counted as protocol bits, this number is the same for all stable systems. Also, he was the first to treat comprehensively the issues involved in using flags to encode message length. However, his principal achievement was a study of source addressing using both information and queuing theoretic techniques. Using a simplified heavy-traffic model, Humblet demonstrated that employing state-dependent service strategies and overhead reduced the entropy of the addressing process, and hence, the average number of bits expended upon it. The second portion of his work on addressing overhead consisted of mean delay calculations for cyclic polling systems, from which he was able to conclude that, among the service strategies and protocols considered, exhaustive service yielded the lowest average delay.

3. Thesis Outline

In this thesis we consider the joint encoding of message start/stop information and source addresssing, as well as the more general issue of state-dependent overhead in cyclic systems. In Chapter 3, we examine fixed-frame strategies which require neither addressing nor message length overhead; it is simply necessary to encode buffer underflows. In Chapter 4, we study two variable-cycle polling schemes employing state-dependent protocols which jointly encode origins and lengths of transmissions using flag-based overhead. Chapter 2, describing the structure and performance of flags and sets of flags, provides the necessary preliminaries to our study.

Chapter II: Flags

This chapter is a study of the structure and performance of flags and sets of flags. Our aim is to determine conditions for flag sequences to be unambiguously decodable, and to calculate the equilibrium probability of insertion for such sequences. Some of the results contained within are well known, while others are new. The chapter has the merit that it presents in one place a comprehensive study of flags.

1. Single Flag

In many systems it is necessary for the transmitter to convey to the receiver the information that an event apart from data transmission has occurred. A simple and effective method of relaying this information is to place a known sequence of characters, a flag, into the data stream whenever the event occurs.

In order that only the desired event be consistantly recorded by the receiver, the chance replication of the flag by the data must be avoided. This can be accomplished by inserting or 'stuffing' a bit into the data stream in such a way as to destroy the similarity between the data and the flag, while retaining the information content of the data. The insertion rule that is usually employed is to insert the complement of the last (F^{th}) bit of the flag if the data replicates the first $F-1$ bits of the flag.

A consequence of this convention is that certain sequences cannot be used as flags because they are not unambiguously decodable (that is, given a sequence of bits there is a unique and unambiguous parsing of data, stuff and flag bits in that sequence). To illustrate, consider the sequence $S = (1,0,1,0)$ when it is used to encode idle characters. If the source produces (1,0,*idle*), the coder will generate (1,0,1,0,1,0), which the receiver will decode as (*idle*,1,0); in this case the data is not lost, just misplaced in time. However, if instead of inserting a 1 whenever the data is (1,0,1), a 0 is inserted after every occurance of (1,0), it is readily verified that the bit sequence arriving at the receiver will be unambiguously decodable. This decodability is obtained at a cost: an increased number of insertions. Since this is undesirable, we shall restrict our attention to sequences which are unambiguously decodable when the first insertion convention is in effect.

It is known [15] that the feature which distinguishes unambiguously decodable sequences of length F from others such as S is that the former contain no prefixes which are also suffixes within the first $F-1$ bits. In the case of S, the first and penultimate bits are identical, hence S has a prefix (1) which is also a suffix (1). This observation leads to the following proposition.

Proposition 2.1:

Let $S_F = (f_1, f_2, \ldots, f_F)$ be a flag sequence, $S_{F-1} = (f_1, f_2, \ldots, f_F)$ be the flag root and $\bar{f}_F$ be the complement of the last bit. If the insertion rule is to insert $\bar{f}_F$ whenever the data replicates the flag root, S_{F-1}, and if the sequence of bits is to be unambiguously decodable, then the flag root can contain no suffix which is also a prefix.

Proof : Let $x \circ y$ denote the concatenation of two sequences x and y, and assume to the contrary that $S_{F-1} = x \circ y = y \circ z$ for some x, y, and z; then the source sequence $x \circ idle$ has an ambiguous decoding, since

i) if $z_1 = f_F$ then $x \circ idle \rightarrow x \circ y \circ z \circ f_F \rightarrow idle \circ z_1, z_2 \cdots$, *ii)* if $z_1 = \bar{f}_F$ then $x \circ idle \rightarrow x \circ y \circ z \circ f_F \rightarrow x \circ y \circ z_2 \cdots$, where z_1 has been falsely decoded as a stuff bit. ▮

Sequences which contain no prefixes which are also suffixes are known as 'bifix-free.' They were first studied by Neilson [21], who calculated the number of bifix-free sequences of a given length. Since the last bit of the flag plays no role in the performance of the flag, it can be chosen arbitrarily. Thus, the number of allowable (unambiguously decodable) flags of length F is twice the number of bifix-free sequences of length $F-1$, as calculated in [21].

Although unambiguous decodability is a desirable feature of flags, a performance measure of greater interest is the average time between insertions. It is this measure which will determine the efficiency of capacity utilization. For the purpose of examing this issue, we consider a memoryless source which produces in each slot either a 0 or 1 with probability $\frac{p}{2}$, or an idle character with probability $1-p$. Futhermore, we denote by $\{X_i\}$ the number of consecutive source symbols up to time i which have replicated the flag root; thus, $\{X_i\}$ is the memory of the insertion process. Since insertions are renewal points, $\{X_i\}$ is a finite Markov chain assuming values in $C = \{0, 1, \ldots, F-1\}$, where F is the length of the flag and $F-1$ is the insertion state.

The two chains for the flags $S_1(F) = (1 \circ 0^{F-1})$ and $S_1(F) = (\circ 1^{F-2}, 0, 0)$, ($\circ x^n =$ n-fold concatenation of x), are displayed in Fig. 2.1. The corresponding stationary distributions $\{\pi_j^i\}_{j \in C}, i \in \{1,2\}$, are easily calculated [8] as a function of F. Once the distributions are known, it is a simple matter to compute the mean recurrence time between stuff bits ($\overline{T}_i, i \in \{1,2\}$) for the two flags:

$$\overline{T}_1 = \frac{1}{\pi_{F-1}^1} = \left(\frac{2}{p}\right)^{F-1} + 1, \tag{2.1a}$$

$$\overline{T}_2 = \frac{1}{\pi_{F-1}^2} = \left(\frac{2}{p}\right)^{F-1} + 1, \tag{2.1b}$$

This shows that the average time between insertions is the same for both flags, even though the chains are quite different. Indeed, the mean recurrence time

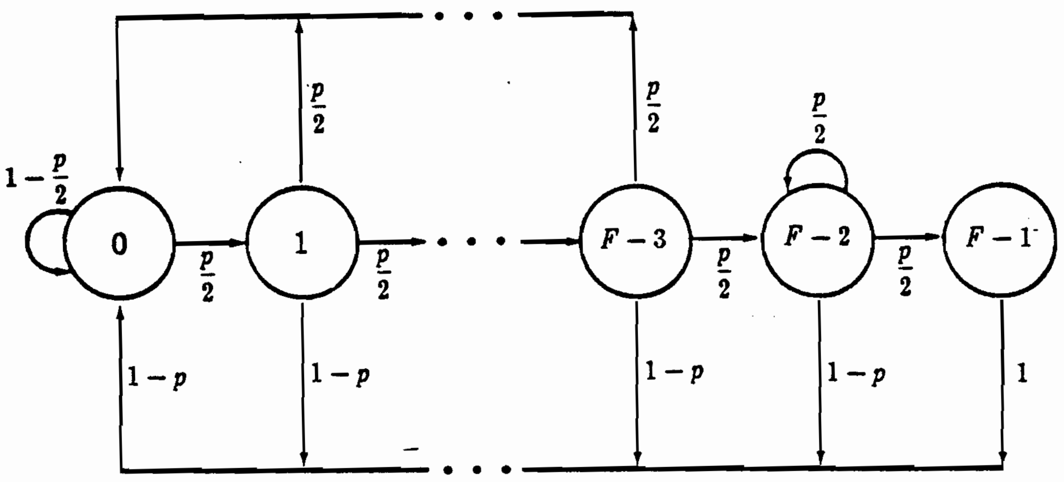

Figure 2.1a: Markov Chain Model for the Sequence $S_1(F) = (1, \circ 0^{F-1})$.

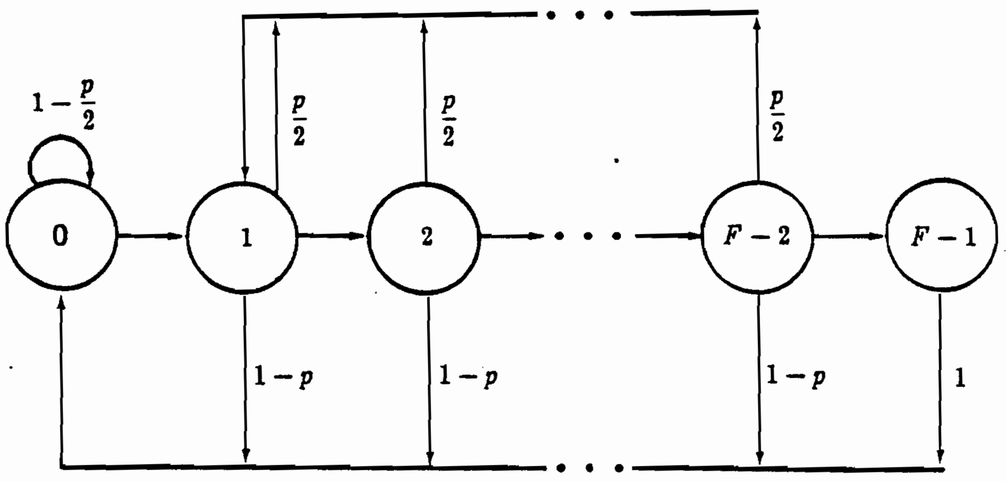

Figure 2.1b: Markov Chain Model for the Sequence $S_2(F) = (\circ 1^{F-2}, 00)$.

between stuff bits for any allowable sequence is identical to that given above.

The reason for this is that the insertion state can only be reached from the zero state by progressively passing through each state in between; regardless of the interruptions along the way, it is always necessary that $F-1$ data bits arrive consecutively so as to imitate the flag root. Since the probability of the source producing a particular data bit is $\frac{p}{2}$, we have that

$$\Pr\{insertion\} = \left(\frac{p}{2}\right)^{F-1}, \tag{2.2}$$

with a mean recurrence time given by the inverse of the insertion probability. The additive factor of 1 in Eq. 2.1 accounts for the time to transmit the stuff bit.

We close this section by pointing out that the probability of insertion which we have calculated is an equilibrium result for a *memoryless* and *synchronous* source. The result does not apply if the source is not memoryless or if some of the characters are discarded. We shall have more to say on this issue in subsequent chapters.

2. Multiple Flags

Employment of a set of flags is indicated when there is more than one event to be encoded. Specifically, we are interested in using multiple flags to encode jointly both the end of a message and the address of the transmitting source.

The simplest method of creating M flags is to select a single prefix (that is, an allowable flag of the last section) and to append to it $\lceil \log_2 M \rceil$ bits to indicate the flag number. This effectively creates a set of codewords with a common prefix and at least one flag with a sub-prefix which is also a suffix of the flag; consequently, insertions must be made whenever the data replicates the prefix root (*i.e.*, the first $F - \lceil \log_2 M \rceil - 1$ bits), and not after replication of $F-1$ flag bits. Thus, the equilibrium probability of stuffing is given by

$$P_s^{(\#)} = \left(\frac{p}{2}\right)^{F-\lceil \log_2 M \rceil - 1}, \tag{2.3}$$

where F is the total length of a flag in the flag set ($F = \| prefix \| + \lceil \log_2 M \rceil$).

An alternative construction is to produce a set of flags which encode the flag number within the structure of the flag. In order to do this, we must first determine the criteria for multiple flags to be allowable.

If all flags in a flag set are to be unambiguously decodable then, each flag root in the set must be bifix-free. Furthermore, in order that certain data sequences concatenated with one flag not be decoded as another flag, there can be no flags in the set which have a suffix of length 2 or more which is also a prefix of another flag in the set. For example, the two sequences $S_1 = (0,0,1,1)$ and $S_2 = (1,1,0,1)$ cannot both be members of the same flag set since, the data sequence (0,0) concatenated with S_2 would yield, when decoded,

$$(0,0,S_2) \rightarrow (0,0,1,1,0,1) \rightarrow (S_1,0,1),$$

and undermine our intention of encoding message length and origin.

In order to construct a flag set for a given length F, one could list all 2^F possibilities and strike from that list those not satisfying the admissibility criteria. However, the flag length is often not the deciding factor when forming a flag set; instead, the magnitude of the set is the constraint. For this reason we give a method for the construction of M flags:

> Select a first flag bit f_1, set the penultimate flag bit f_{F-1} to the complement $\overline{f}_1$ of f_1 , and set the last flag bit f_F to f_1. The i^{th} flag is given by
>
> $$F_i = (f_1 \circ (f_1)^{M-i} \circ (\overline{f}_1)^{i-1} \overline{f}_1 f_1).$$

From this we conclude that the flag length necessary to construct M fixed-length flags is no longer than $M+2$. Figure 2.2 shows the structure of the flags when $M = 5$. Notice that the internal bits f_2, f_3, f_4, f_5, f_6 encode the flag number, not in its binary form, but by the number of 1's within these bits.

f_1	$f_2 f_3 f_4 f_5 f_6$	f_7
0	0 0 0 0 1	0
0	0 0 0 1 1	0
0	0 0 1 1 1	0
0	0 1 1 1 1	0
0	1 1 1 1 1	0

Figure 2.2: Structure for 5 Flags.

The insertion process for the flag set just described can be modelled, assuming the source produces data without interruption, by the binary tree shown in Fig. 2.3. The numbers on the branches (0,1) indicate the bit value which will cause the transition to occur. The memory of the process is such that there is a single main path from which all address paths subtend. Once on an address path, only one of two possibilities can occur:

i) a sufficient number of 1's are received to reach the insertion state;

ii) return to the 1-memory state when a 0 is received.

It is not possible to move from one address path to another without first returning to the 1-memory state. This indicates that the equilibrium probability of insertion for this set of flags is composed of the M disjoint probabilities of insertion for each flag in the set. Since the probability of receiving $F-1$ bits in a specified order is $\left(\frac{p}{2}\right)^{F-1}$, we have that the time independent probability of insertion for the flag set is

$$P_s^{(c)} = M\left(\frac{p}{2}\right)^{F-1} \tag{2.4}$$

Comparing this result with that obtained for the prefix plus flag number scheme (Eq. 2.3), we find that the likelihood of an insertion is less when the flag number is encoded within the flag; since, if we fix the flag length at $M+2$ for both cases,

$$P_s^{(\#)} - P_s^{(c)} = \left(\frac{p}{2}\right)^{M+1}\left\{\left(\frac{2}{p}\right)^{\lceil \log_2 M \rceil} - M\right\} \geq 0, \qquad \textit{for all } M, p;$$

with equality if and only if $p = 1$ and $M = 2^N$ for some integer N.

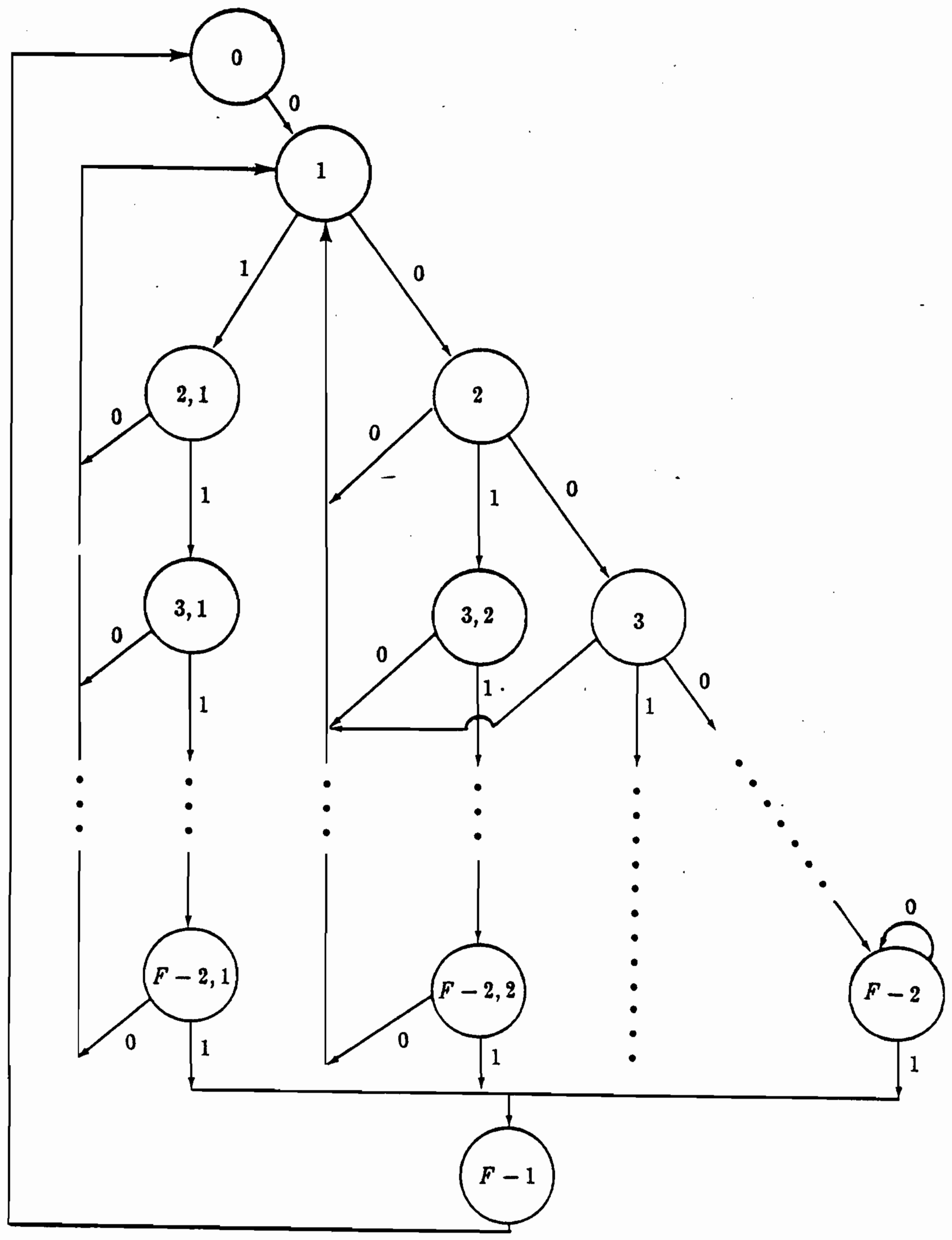

Figure 2.3: Insertion Model for a Multiple Flag Set.

Before closing, we mention that fixed-length flags are optimal in terms of the number of bits per flag number only in the case where the flags occur equiprobably. If the flag choice is made according to any other distribution, the structure of flag-plus-Huffman code would yield a lower average number of bits per flag number. However, because of the variable length, it is not possible to include the codeword for the flag number within the flag without reducing the probability of insertion below that of flags plus codeword.

Chapter III: Fixed-Frame Service Strategies

In this chapter, we examine service strategies which periodically allocate a fixed portion of channel capacity to each source. In all the schemes we shall analyse, the time axis is slotted. A constant number of consecutive slots forms a group and a constant number of consecutive groups constitutes a frame. The word sub-frame will be reserved for that portion of a frame allocated for transmission of data and overhead from a particular source.

Since both frames and groups are of constant duration, the receiver, by simply counting slots, can determine the source currently transmitting. This eliminates the need to expend capacity on explicit source addressing. However, because the frame structure is inflexible, there will be occasions when slots are idle. This is due to the possibility that a source will not have the requisite amount of data to fill its sub-frame, a situation which must be conveyed to the receiver and requires overhead. In this chapter we examine three fixed-frame strategies that encode underflow information and demonstrate a trade-off between the ratio of overhead-to-data and the delay.

1. *TDMA(K)*

TDMA(K) is a general form of time division multiple access (*TDMA*); it includes protocol overhead to indicate buffer underflow and allows up to K units of work, from one source, to be served within a sub-frame. In this section, we describe the system, obtain the steady-state moment-generating functions (*mgf*) for various quantities of interest, and discuss the effects of the protocol overhead on system performance.

System:

The time axis is divided into frames of length T, each of which is partitioned into M sub-frames, where M is the number of sources. The length of a sub-frame is $K+1$ slots. In this strategy a group is identical to a sub-frame. The division of the time axis is shown in Fig. 3.1.1.

If, at the instant the server arrives at a buffer there are K or more units of work queued, exactly K are removed and a 1 is appended to the beginning of the data before transmission. If instead, the server finds fewer than K units waiting, the buffer is emptied, and the data flanked by a 0 at the beginning and a 1 at the end; 0's are used to pad the sub-frame. In the first case the 1 informs the receiver that a full sub-frame has been transmitted, and in the second case the first 0 indicates a partial sub-frame, while the trailing 1 marks the end of the data.

Assumptions:

We require two assumptions other than those given previously:

i) The sources are homogeneous: $\lambda_i = \lambda$, $i = 1, 2, \ldots, M$.
This justifies using the same sub-frame lengths for all sources.

ii) Source (i) work arriving within the i^{th} sub-frame must wait at least until the next frame before being served. Consequently, arrivals which see a non-full sub-frame cannot be served within the empty portion. This is equivalent to saying that at the instant the server arrives, it removes up to K units of work and immediately departs to process them, becoming available to the next source $K+1$ slots later.

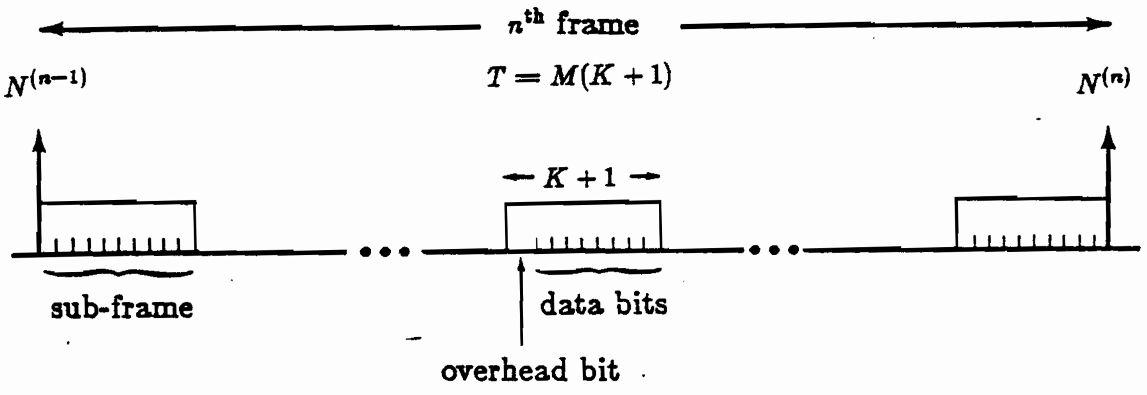

Figure 3.1.1: *TDMA*(K) Frame Structure.

Notation:

Throughout this section we shall focus on a single source and buffer; for convience, we shall assume that the first sub-frame of each frame is dedicated to the source of interest. Bearing this in mind, we now define the various symbols used in our analysis:

$N^{(n)}$	–	the number in the queue at the beginning of the nth frame.
$N^{(n)}(t)$	–	the number in the queue t seconds into the nth frame, $t \in [0,T)$; note $N^{(n)} \neq N^{(n)}(0)$.
$A^{(n)}(t)$	–	the number of arrivals in $[0,t)$ of the nth frame; A follows a Poisson distribution.

The same symbols, without the time dependent superscripts, will be used for the equilibrium values of the above random variables.

Stability:

Observing Fig. 3.1.1 and using assumption *ii*) , the backlog at the n^{th} frame boundary, $N^{(n)}$, is given by

$$N^{(n)} = (N^{(n-1)} - K)^{+} + A^{(n)}(T). \tag{3.1.1}$$

Sufficient conditions for ergodicity of a Markov Chain [22] are

i) Aperiodicity and irreducibility;

ii) $\mathrm{E}[N^{(n)} \mid N^{(n-1)} = i] < \infty, \quad \forall\, i;$

iii) $\mathrm{E}[N^{(n)} - N^{(n-1)} \mid N^{(n-1)} = i] < -C \quad \forall\, i > N$, *for positive constants C and N.*

Conditions *i*) and *ii*) are satisfied when A is a Poisson arrival process. Condition *iii*) is satisfied when $i > K$ and $\lambda < K/T$. Thus a sufficient condition for equilibrium is

$$\lambda < \frac{K}{T} = \lambda_{max}(K). \tag{3.1.3}$$

Equation 3.1.2 is also necessary for ergodicity [17].

Delay Analysis:

Equation 3.1.1 is a variant of the classical bulk-departure problem [2]; thus, assuming that the stability condition is met, the steady-state *mgf* of the number, N, in the queue at frame boundaries is given by

$$\mathrm{E}[z^{N}] = \frac{\mathrm{E}[z^{A(T)}] \sum_{i=0}^{K-1} (z^{K} - z^{i}) \Pr\{N = i\}}{z^{K} - \mathrm{E}[z^{A(T)}]}. \tag{3.1.4}$$

If we let θ_i, $i \in \{0, 1, \ldots, K-1\}$, denote the roots in unit disc, $|z| \leq 1$, of the characteristic equation

$$z^{K} - \mathrm{E}[z^{A(T)}], \tag{3.1.5}$$

with $\theta_0 = 1$, Eq. 3.1.4 can be rewritten as

$$\mathrm{E}[z^N] = \frac{\mathrm{E}[z^{A(T)}]\left(\prod_{i=1}^{K-1} \frac{z-\theta_i}{1-\theta_i}\right)(z-1)(K-\overline{A})}{z^K - \mathrm{E}[z^{A(T)}]}. \tag{3.1.6}$$

Within a frame, the state equation for the backlog at time t, $t \in [0,t)$, is given by

$$N^{(n)}(t) = (N^{(n-1)} - K)^+ + A^{(n)}(t). \tag{3.1.7}$$

Since $(N^{(n-1)} - K)^+ = N^{(n)} - A^{(n)}$, the steady-state mgf of the number in the queue at time t, $N(t)$, is given by

$$\mathrm{E}[z^{N(t)}] = \frac{\mathrm{E}[z^N]\mathrm{E}[z^{A(t)}]}{\mathrm{E}[z^{A(T)}]} \tag{3.1.8}$$

Substitution of Eq. 3.1.3 into 3.1.6 yeilds

$$\mathrm{E}[z^{N(t)}] = \frac{\mathrm{E}[z^{A(t)}]\left(\prod_{i=1}^{K-1} \frac{z-\theta_i}{1-\theta_i}\right)(z-1)(K-\overline{A})}{z^K - \mathrm{E}[z^{A(T)}]}. \tag{3.1.9}$$

We are now in a position to give the main result of this section.

Theorem 3.1:

i) The transmission delay, as seen by a test customer inserted into the system at time t, $t \in [0,T)$, of a frame is given by

$$D(t) = \alpha N(t) - \beta N(t)_{modK} + T - t + 2, \tag{3.1.10}$$

where $\alpha = T/K$ and $\beta = \alpha - 1$.

ii) If $\omega_l = \exp(\frac{i2\pi l}{K})$, and $\lambda < \lambda_{max}(K)$ then the steady-state mgf of the transmission delay for a test customer is

$$\mathrm{E}[e^{-sD(t)}] = e^{-s(T-t+2)}\frac{1}{T}\sum_{l=0}^{K-1} \frac{1-e^{s\beta K}}{1-e^{s\beta}\omega_l^*}\mathrm{E}[(\omega_l e^{-s\alpha})^{N(t)}]. \tag{3.1.11}$$

iii) The expected value of the delay as seen by a test customer, averaged over all t, is defined by

$$\overline{D} \doteq \frac{1}{T}\int_0^T \mathrm{E}[D(t)]dt,$$

and is

$$\overline{D} = \alpha\left\{\frac{\lambda T}{2} + \sum_{i=1}^{K-1}\frac{1}{1-\theta_i} + \frac{(\lambda T)^2 - K(K-1)}{K-\lambda T}\right\}$$

$$-\beta\left\{\frac{(K-1)}{2} - \frac{1}{T}\sum_{l=1}^{K-1}\frac{\int_0^T \mathrm{E}[\omega_l^{N(t)}]dt}{1-\omega_l^*}\right\} \tag{3.1.12}$$

$$+\frac{T}{2} + 2.$$

Proof:

i) Figure 3.1.2 shows the components of the delay for a test customer arriving at time t. They are 1) the elapsed time from the instant of arrival until the next frame $(T-t)$; 2) the time to discharge the backlog the test customer observes $(\lfloor\frac{N(t)}{K}\rfloor T + 1 + N(t)_{modK})$; 3) the service time (1). Noting that

$$\left\lfloor\frac{N(t)}{K}\right\rfloor = \frac{N(t)}{K} - \frac{N(t)_{modK}}{K},$$

and summing the components of the delay yields the first conclusion.

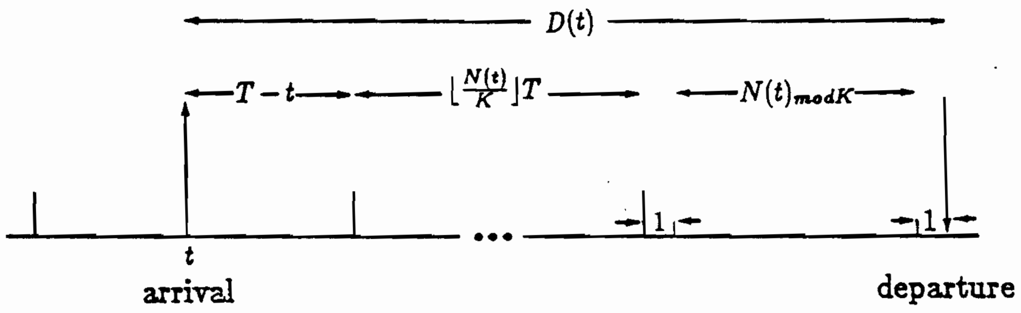

Figure 3.1.2: Delay for a Test Customer.

ii) If the equilibrium condition is satisfied, then the *mgf* of the delay is well defined and given by

$$\mathrm{E}[e^{-sD(t)}] = e^{-s(T-t+2)}\mathrm{E}[e^{-s(\alpha N(t)-\beta N(t)_{modK})}].$$

In Appendix (1), we calculate the expectation on the right hand side; substitution of the result gives Eq. 3.1.11.

iii) Assuming a stable system and taking expectations of both sides of Eq. 3.1.9 results in

$$\mathrm{E}[D(t)] = \alpha\mathrm{E}[N(t)] - \beta\mathrm{E}[N(t)_{modK}] + T - t + 2.$$

The mean of the the backlog at t can be obtained by differentiating Eq. 3.1.8 and setting $z = 1$. The mean of the backlog *modulo K* is found in Appendix (1). The final result is obtained by interchanging the order of integration and summation. ∎

Remarks:

If we eliminate the overhead (that is, if $T = MK$), we obtain the delay-throughput curves of Fig. 3.1.3, from which one can see that the $K = 1$ curve is the lower bound for all stable values of throughput. This situation corresponds to conventional *TDMA*. If protocol overhead is included, the optimal value of K is no longer $K = 1$, but is determined by the trade-off between the fraction of channel capacity lost to overhead, which diminishes in K, and the fraction of channel capacity lost to idle bits, which increases in K. The optimal compromise is represented by the envelope of the family of curves indexed by K, as evinced by Figure 3.1.4.

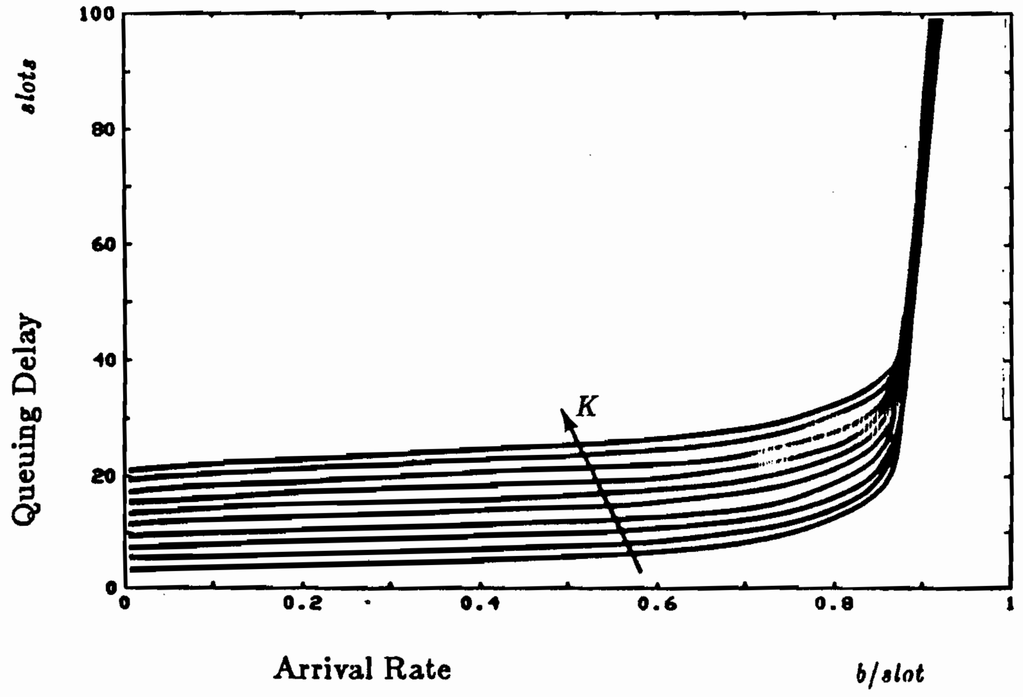

Figure 3.1.3: $TDMA(K)$ with No Overhead; $M = 4, K : 1 \rightarrow 10$.

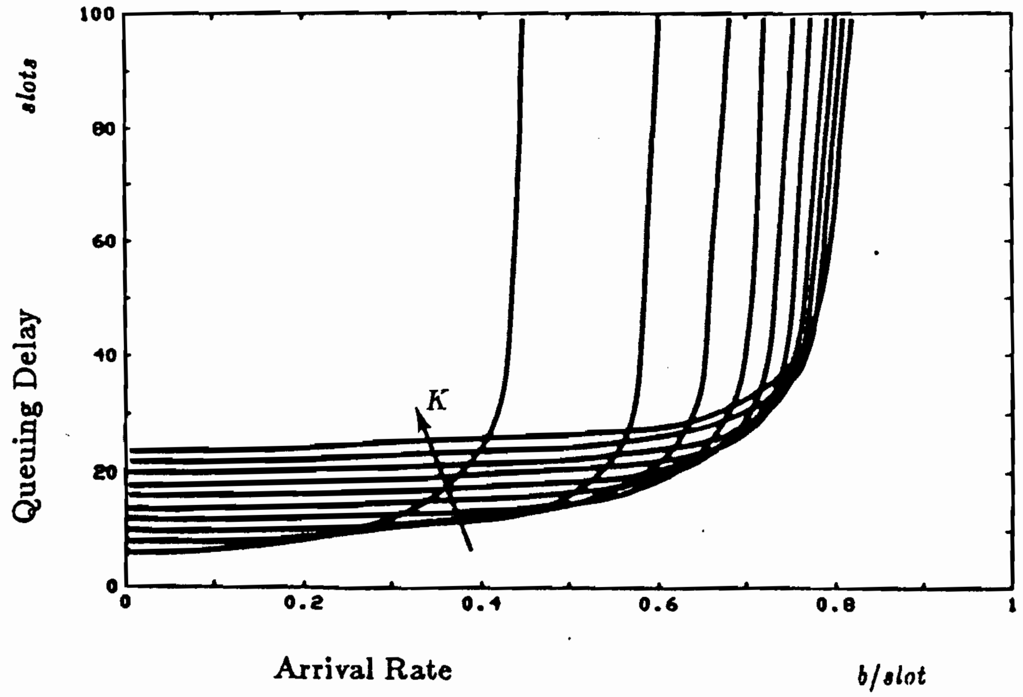

Figure 3.1.4: $TDMA(K)$; $M = 4, K : 1 \rightarrow 10$.

2. $TDMA(K,U)$

$TDMA(K,U)$ is similar to $TDMA(K)$ in that both strategies include protocol overhead to indicate buffer underflow and allow up to K units of work per source to be served in each frame. The difference between the two systems is in the location within a frame of the data slots assigned to each source. In $TDMA(K)$, the K slots per frame allocated to each source are consecutive and service is gated; in $TDMA(K,U)$, the K slots are distributed uniformly within the frame, and the restriction on channel access for new arrivals is relaxed. In this section, we describe the model precisely, and calculate the steady-state *mgf* for the number in the queue at sub-frame boundaries, from which we compute the average transmission delay.

System:

The time axis is divided into frames of length T, and each frame split into $K+1$ groups of M consecutive slots each. The sub-frame corresponding to the slots in a frame allocated to Source (i) consists of the i^{th} slot in each group. A sub-frame is thus a sequence of $K+1$ equi-spaced slots in each frame. The last slot of each sub-frame is reserved for the overhead bit.The division of the time axis is shown in Fig. 3.2.1.

The overhead bit is set to 1 when all K preceding slots of the sub-frame contain data, and set to 0 otherwise. In the event that a slot in a sub-frame cannot be filled with data, a 1 is inserted into that slot and the remainder of the sub-frame is padded with 0's. The difference between the coding procedures employed in $TDMA(K)$ and $TDMA(K,U)$ is simply that in the former the 'sub-frame full' bit precedes the data and in the latter the 'sub-frame full' bit follows the data. As a result the protocol overhead appended is indicative of the same events as described for $TDMA(K)$.

The advantage of the $TDMA(K,U)$ service policy is that new arrivals have the possibility of obtaining service in the frame in which they arrive. For example, let the buffer contain only one unit of work when the server arrives. The buffer is then depleted; however, if another unit of work arrives at that buffer before the server returns it will be transmitted in the next sub-frame slot. Thus, provided that the server never sees the buffer empty, an arrival need never be forced to wait for service until the next frame as in $TDMA(K)$.

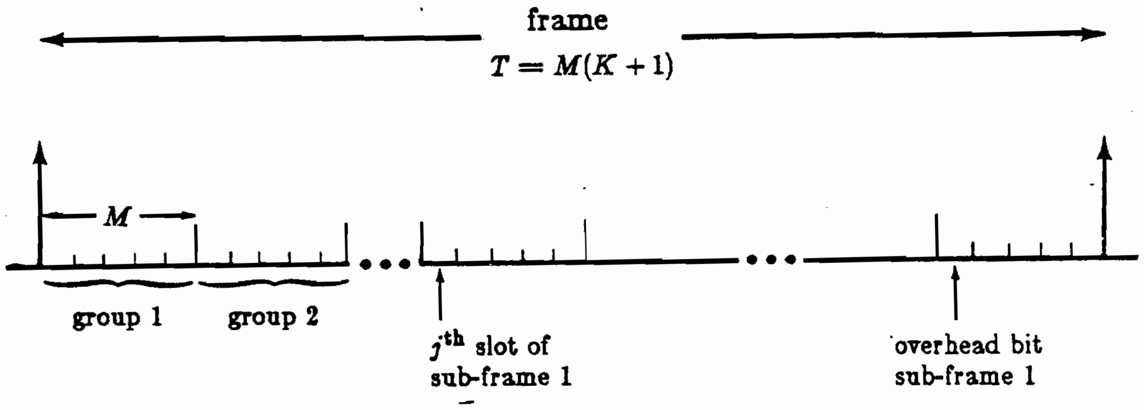

Figure 3.2.1: $TDMA(K,U)$ Frame Structure.

Assumptions:

We require two assumptions in addition to those given in Chapter 1:

i) The sources are homogeneous: $\lambda_i = \lambda$, $i = 1, 2, \ldots, M$. This facilitates the choice of the sub-frame size.

ii) Arrivals to a buffer which occur after its sub-frame has been terminated must wait at least until the next frame for service. This is necessary to insure the decodability of the protocol overhead inserted within the data portion of a sub-frame, since termination can only be determined at the receiver by backtracking to the last 1 contained in a sub-frame.

Notation:

We assume the source of interest is allocated the first slot in each group of every frame. The following notation is used in this section.

$N_j^{(n)}$ – the number in the queue at the beginning of the j^{th} slot of a sub-frame in the n^{th} frame.

$N_j^{(n)}(t)$ – the number in the queue at time $jM + t$ of the n^{th} frame.

$A_j^{(n)}$ – the number of arrivals in the interval $[(j-1)M, jM)$ of the n^{th} frame; A has a Poisson distribution.

$A_j^{(n)}(t)$ – the number of arrivals in the interval $[(j-1)M, t)$, $t \in [0, M)$, of the n^{th} frame; $A_j^{(n)}(t)$ also has a Poisson distribution.

The above symbols without the frame superscripts will denote equilibrium values.

Stability:

The number in the queue at frame boundaries constitutes a Markov chain. From Figure 3.2.1, we see that the number in the queue at the beginning of the n^{th} frame is given by

$$N_o^{(n)} = N_o^{(n-1)} - [\textit{departures in } (n-1)^{th} \textit{ frame}] + A^{(n)}(T). \tag{3.2.1}$$

Using the sufficiency conditions for ergodicity given previously, we find that

$$\lambda < \frac{K}{T} = \lambda_{max}(K), \tag{3.2.2}$$

is the requirement for the imbedded chain to be ergodic.

The system is stable everywhere if it is stable at frame boundaries ([4], Theorem 6.6, p.124).

Delay Analysis:

Because in any sub-frame the server's ability to serve depends on the content of preceding data slots within that sub-frame (assumption *ii*)), the number in the queue at sub-frame boundaries is not a sufficient descriptor of the system. In order to achieve a complete description of the system, we introduce a dependent binary random variable, $U_j^{(n)}$, which characterizes the state of the server at the beginning of the jth sub-frame of the nth frame. A value of 1 enables the server to remove one unit of work from the buffer and place it in the jth slot of the sub-frame. To indicate that the server is disabled, $U_j^{(n)}$ is set to 0; this implies that the data portion of the sub-frame has been terminated. The introduction of this random variable expands the dimensionality of the system description in order to make the process Markov at sub-frame boundaries.

From Figure 3.2.1, and using the augmented state description, we find that the evolution of the system is governed by the following set of equations:

$$\begin{aligned} N_j^{(n)} &= (N_{j-1}^{(n)} - U_{j-1}^{(n)})^+ + A_{j-1}^{(n)}, \qquad j = 1,2,\ldots,K; \\ N_o^{(n+1)} &= N_K^{(n)} + A_{K+1}^{(n)}; \\ N_o^{(n+1)} &= N_{K+1}^{(n)}; \\ U_j^{(n)} &= \min[N_{j-1}^{(n)}, U_{j-1}^{(n)}], \qquad j = 1,2,\ldots,K; \\ U_o^{(n)} &= 1. \end{aligned} \tag{3.2.3}$$

The following theorem summarises the results.

Theorem 3.2:

Let $P_{0,1}^{(j)} = \Pr\{N_j = 0, U_j = 1\}$. If $\lambda < \lambda_{max}(K)$,

i) The steady-state *mgf* of the number in the queue at frame boundaries, N_o, is

$$\mathrm{E}[z^{N_o}] = \mathrm{E}[z^{A(T)}] \frac{\sum_{j=0}^{K-1} P_{0,1}^{(j)} \mathrm{E}[z^{A(-jM)}]}{z^K - \mathrm{E}[z^{A(T)}]}. \tag{3.2.4}$$

ii) The steady-state *mgf* for the number in the queue at the beginning of the j^{th} service slot, $j \in \{1, 2, \ldots, K\}$ can be computed in terms of the *mgf* for N_o and the probabilities $P_{0,1}^{(i)}$, and is given by

$$\mathrm{E}[z^{N_j}] = \mathrm{E}[z^{N_o}]z^{-j}\mathrm{E}[z^{A(jM)}] + \sum_{i=0}^{j-1} P_{0,1}^{(i)}\mathrm{E}[z^{-A(M(j-i))}](1 - z^{-(j-i)}). \qquad (3.2.5)$$

iii) The average number in the queue is given by

$$\begin{aligned}\mathrm{E}[N] &\doteq \frac{1}{T}\int_0^T \mathrm{E}[N(t)]dt \\ &= \mathrm{E}[N_o] - K(1 - M\lambda) + \frac{\lambda M}{2} \div \frac{K}{K+1} \qquad (3.2.6) \\ &+ \frac{1}{K+1}\left\{\sum_{j=0}^{K-1}\sum_{i=0}^{j} P_{0,1}^{(i)}(j+1-i) + \sum_{j=0}^{K-1} P_{0,1}^{(j)}(K-j)\right\},\end{aligned}$$

where

$$\begin{aligned}\mathrm{E}[N_o] = &\sum_{j=0}^{K-1} P_{0,1}^{(j)}[K(K-1) - j(j-1)] \\ &+ 2M\lambda \sum_{j=0}^{K-1} P_{0,1}^{(j)}[(K-j)(K+1-j)]; \qquad (3.2.7) \\ &+ \frac{K(K-1) + T\lambda^2}{2(K - T\lambda)}\end{aligned}$$

Proof: See Appendix (2).

The average transmission delay can be calculated by applying Little's Theorem to the average value of the number in the queue [19,26].

Remarks:

The *TDMA*(K, U) transmission delay versus throughput curves with K as a parameter are plotted in Fig. 3.2.2. As with *TDMA*(K), the performance lower bound is given by the envelope of a family of curves. This is again due to the trade-off between the fraction of capacity expended on overhead bits and the portion given over to idle bits.

Figure 3.2.3 compares the delay envelopes for $TDMA(K,U)$ and $TDMA(K)$ with the number of sources, M, as the control variable. One can see that the delay for $TDMA(K,U)$ is always less than that of $TDMA(K)$. In fact, $TDMA(K,U)$ is at least marginally better than $TDMA(K)$ for all values of K and throughput (compare Figure 3.2.2 with Figure 3.1.4).

This result is valid only if the decoding delays are not included in the calculation of the mean delays. If on the other hand we were to assume that as a resonable approximation, the mean decoding delay is half a sub-frame in $TDMA(K)$ and half a frame in $TDMA(K,U)$ (that is, $(K+1)/2$ and $M(K+1)/2$, respectively), then $TDMA(K)$ would yield the lower set of delay-throughput curves. However, because the total delay from the instant of transmission to the instant bits are declared data bits is influenced by many additional factors, we shall restrict ourselves in the sequel to the comparision of transmission delays.

As a final remark, we note that for $K > 1$, the $TDMA(K,U)$ delay curves are not strictly non-decreasing. The reason for this is that by allowing some work arriving within a frame to be served within that frame, the delay must necessarily decrease with respect to the delay of a test customer inserted into a zero-throughput system. Since there does exist a minimum, one might be tempted to insert artificial work into the system when the throughput is less than that which extremises the delay for a particular value of K; however, additional protocol overhead would be required to separate the real data from the artificial. This would have an adverse effect on delay.

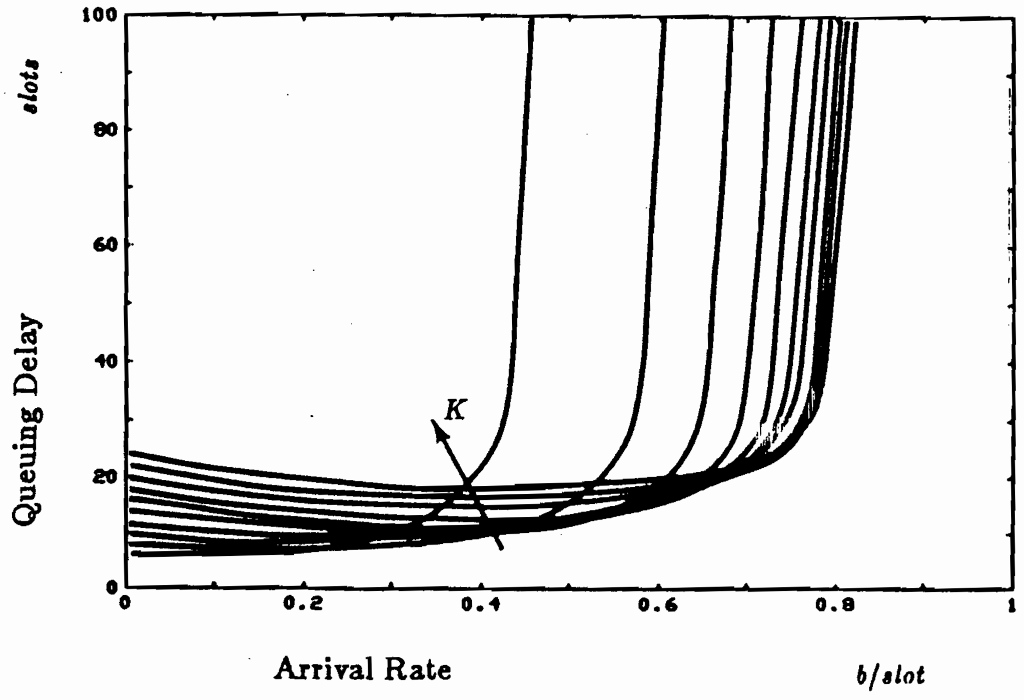

Figure 3.2.2: $TDMA(K,U)$; $M = 4, K : 1 \rightarrow 10$.

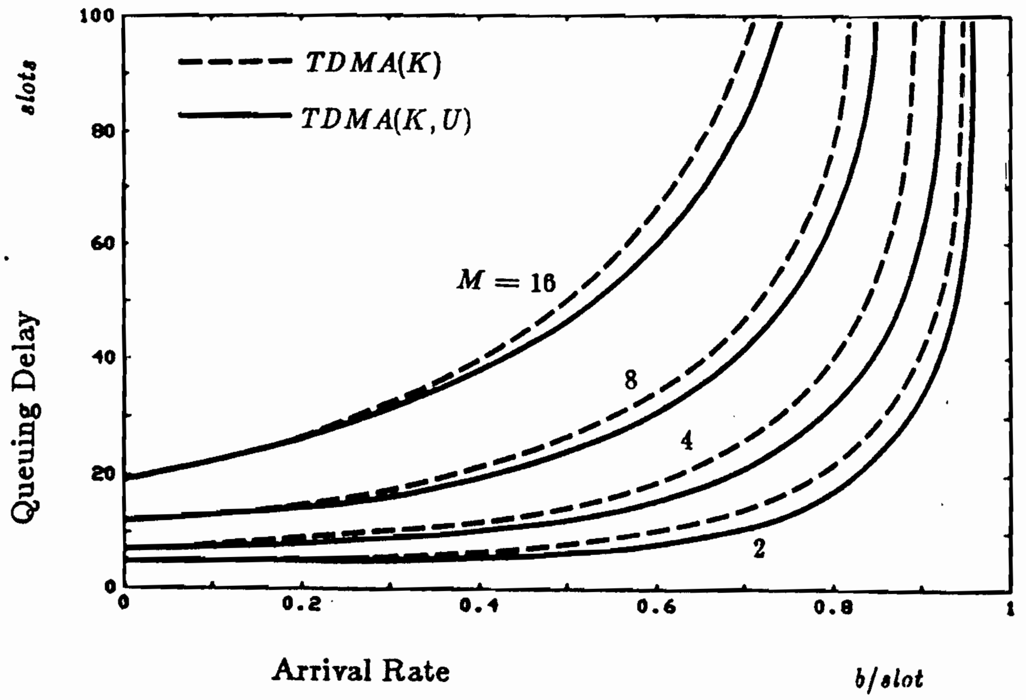

Figure 3.2.3: Delay Envelopes For $TDMA(K)$ and $TDMA(K,U)$.

3. *TDMA(K,F)*

In the previous two sections, message length information was encoded using a fixed portion of each sub-frame. An alternative is to transmit a flag whenever a buffer underflows. In order for a flag to be consistently and correctly decoded at the receiver, the data must be inhibited from imitating the flag sequence. If the flag is bifix-free, this can be accomplished by inserting the complement of the last bit into the data stream whenever the data replicates the flag root.

In this section we describe a fixed-frame system which uses a single flag to encode buffer depletions, discuss the effect of insertions on system performance, and using an independent-insertion model calculate the steady-state *mgf* for the number in the queue and for the delay. From the latter, we calculate the average transmission delay.

System:

The division of the time axis into frames and sub-frames is the same as for *TDMA(K)*. Each source is apportioned K consecutive slots per frame for data transmission. If the number of sources is M, the frame is composed of M sub-frames of length K. There are no dedicated overhead slots.

Upon arriving at a buffer the server first determines if there are K or more units of work awaiting transmission. If there are, then K units are removed and sent. If there are fewer than K units buffered, then a flag of length F is appended to the end of the data. A flag is sent each time the server sees the buffer empty. Due to the fixed sub-frame size, there will be occasions when a portion of the flag will require buffering. Whenever this occurs, new arrivals from the source join the queue behind the buffered flag bits.

We now turn our attention to the issue of stuffing. Ideally, insertions should only be made when the data has replicated the flag root (that is, the first $F-1$ bits of the flag). However, an exact model of stuffing would introduce memory into the system description, since the probability of insertion at a particular instant of time is linked to past insertions or the lack thereof [13]. For example, if the flag is bifix-free and an insertion has just been made, another insertion will not be necessary until at least $F-1$ data bits have been processed.

In order to simplify the analysis, we shall assume that the probability of an insertion at any point in time is independent of insertions at any other point in time. This can be achieved [23] by selecting the flag at each occurence and in i.i.d. fashion from the set of all 2^F binary sequences of length F. In this case, the probability of an insertion is time independent and given by $P_s = 2^{-(F-1)}$. This approximation will give an upper bound for whatever measures of performance one wishes to calculate, since the independence assumption will result in more insertions than the exact procedure.

Assumptions:

i) The sources are homogeneous: $\lambda_i = \lambda$, $i = 1, 2, \ldots, M$. This serves to simplify the selection of the sub-frame size.

ii) Service is gated; that is, Source (i) customers arriving anywhere within a frame must wait until the next frame for service. This assumption simplifies the calculation and can be relaxed.

iii) Each arrival brings with it either 1 or 2 units of work with probabilities $(1-P_s)$ and P_s, respectively, where $P_s = 2^{-(F-1)}$ is the probability of an insertion, assuming independent decisions at each point in time. This models the approximate stuffing procedure outlined previously.

Notation:

The notation employed in this section is identical to that given in the section on $TDMA(K)$, with the following additions:

i) for $K \geq F$,

$$\overline{D}_{K\geq F} = M\left\{+\sum_{i=1}^{K-1}\frac{1}{1-\theta_i}+\left(\frac{[\lambda T(1+P_s)]^2+2\lambda TP_s-K(K-1)}{2[K-\lambda T(1+P_s)]}\right)\right\}$$

$$+M\left\{\frac{\lambda T(1+P_s)}{2}+\frac{F-1}{2}\right\}$$

$$-(M-1)\left\{\frac{K-1}{2}-\frac{1}{T}\sum_{l=1}^{K-1}\frac{\int_0^T \mathrm{E}[\omega_l^{N^{K\geq F}(t)}]dt}{1-\omega_l^*}\right\}$$

$$+\frac{T}{2}+1; \qquad (3.3.6a)$$

ii) for $K < F$,

$$\overline{D}_{K<F}, = M\left\{\frac{\lambda T(1+P_s)}{2}+\frac{[\lambda T(1+P_s)]^2+2\lambda TP_s-K(K-1)}{2[K-\lambda T(1+P_s)]}\right.$$

$$+\frac{\sum_{j=0}^{K-F}[K(K-1)-j(j-1)]\Pr\{N_o=j\}}{2[K-\lambda T(1+P_s)]}$$

$$\left.+\frac{\sum_{j=K-F+1}^{K-1}[F(F-1)+2Fj]\Pr\{N_o=j\}}{2[K-\lambda T(1+P_s)]}\right\}$$

$$-(M-1)\left\{\frac{K-1}{2}-\frac{1}{T}\sum_{l=1}^{K-1}\frac{\int_0^T \mathrm{E}[\omega_l^{N^{K<F}(t)}]dt}{1-\omega_l^*}\right\}+\frac{T}{2}+1. \qquad (3.3.6b)$$

Proof: The proof is identical in procedure to that given for *TDMA*(K) (see Appendix 1).

Remarks:

Figure 3.3.1 shows the throughput versus delay curves for *TDMA*($K, F = 2$). In contrast to the previous two systems, the delay is lower-bounded by the case $K = 1$. This result is true for all values of F, M and λ. The explanation for this is that for $K > 1$, the probability that a sub-frame will not be full is greater than that for $K = 1$. As a result, flags occur with a greater frequency in the systems with $K > 1$, which in turn causes the data backlog to be larger in these systems.

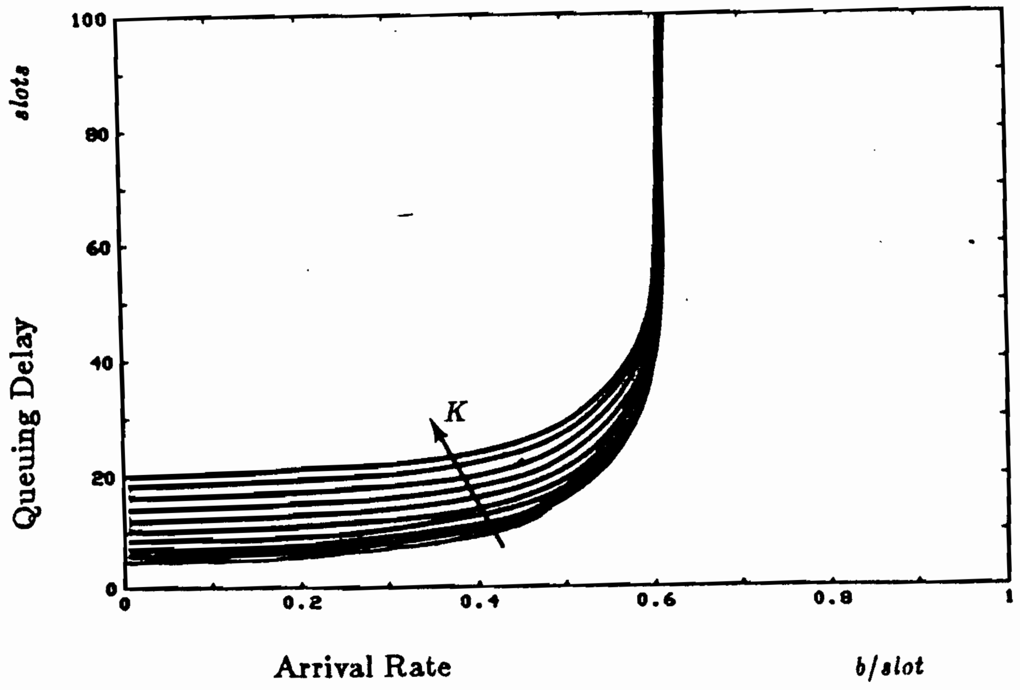

Figure 3.3.1: $TDMA(K,2)$: $M = 4, K : 1 \rightarrow 10$.

Figure 3.3.2 displays the delay plots for $TDMA(1,F)$, from which we see that as the total system throughput increases, so must the flag length in order that the delay remain bounded. This is due to the fact that as the flag is made larger, the frequency of stuff bits diminishes. It should also be noted that the $TDMA(1,F)$ system approaches the maximum allowable throughput rate (1) much faster than either $TDMA(K)$ or $TDMA(K,U)$.

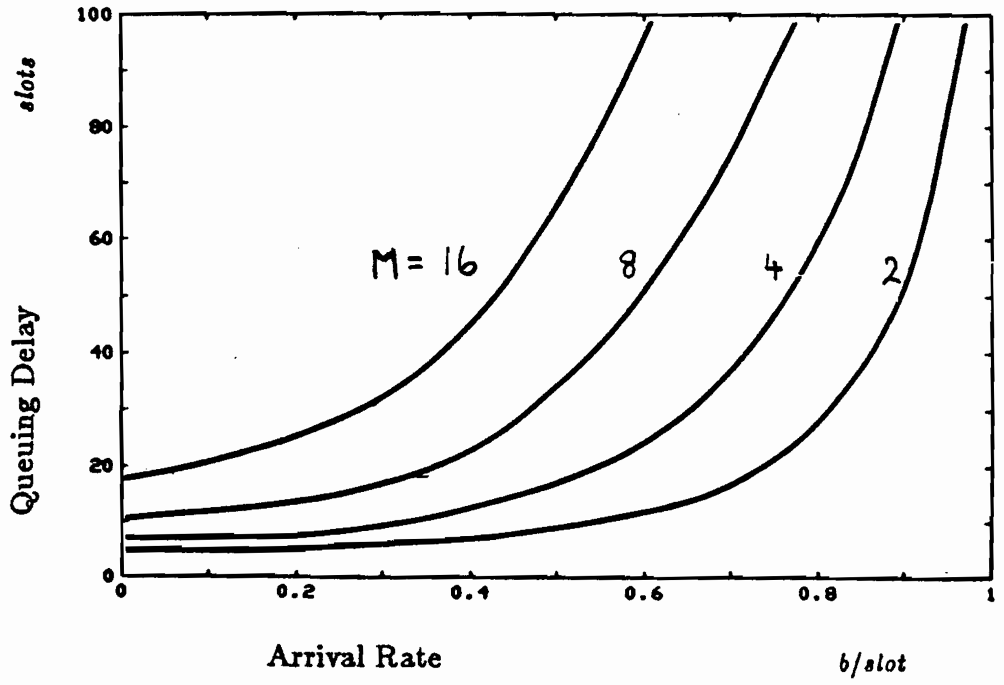

Figure 3.3.2: $TDMA(1,F)$

4: Summary and Conclusions

The delay analyses in this chapter suggest several conclusions.

First, in the absence of protocol overhead, the optimal sub-frame size for the standard $TDMA$-type system is identical to the message length[†] ($TDMA(K = 1)$).

Second, the protocol overhead is the deciding factor in determining the maximum allowable throughput rate of the system. In both $TDMA(K)$ and $TDMA(K,U)$ where there is a fixed allotment of channel capacity per frame for protocol overhead, we have demonstrated that there is an optimal compromise between the fraction of capacity expended on overhead (which determines $\lambda_{max}(K)$), and the fraction lost to idle bits (which contributes to the delay). On the other hand, in the $TDMA(K,F)$ scheme, where the insertion of a flag occurs only if a buffer underflows, the optimal sub-frame size is unity, regardless of the flag length, and the limiting factor is the fraction of capacity given over to insertions.

Third, we have seen that $TDMA(K,U)$ has a lower average delay than $TDMA(K)$ for all values of throughput and number of sources. Comparision of $TDMA(K,U)$ and $TDMA(1,F)$ (Figure 3.4.1), shows that for small to medium values of throughput, both sytems have practically the same delay performance; however, at higher levels of throughput the transmission time of $TDMA(1,F)$ is markedly better than that of $TDMA(K,U)$.

The reason for this is that the steady-state fraction of capacity expended on overhead decreases exponentially in the flag scheme ($O(P_s)$), and only proportionately in the coding schemes. Therefore, $TDMA(1,F)$ has a lower delay in the high throughput range.

[†] If the messages are of random length with mean $\bar{\mu}$, the analysis of $TDMA(K)$, neglecting protocol, can be redone to show that the optimal sub-frame length is $\bar{\mu}$ bits.

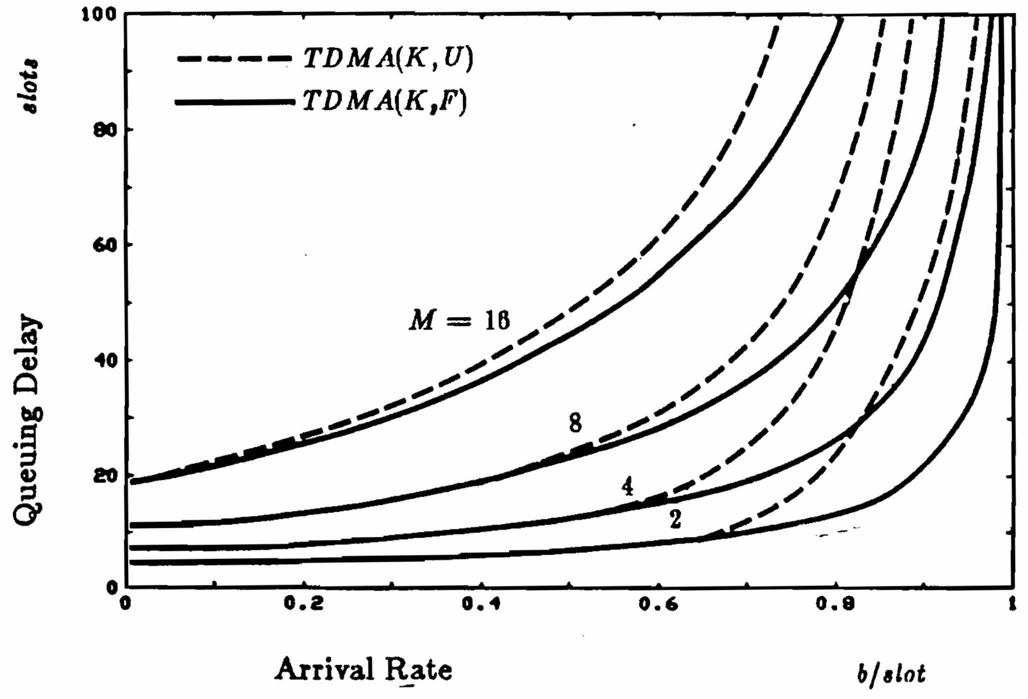

Figure 3.4.1: Delay Envelopes for $TDMA(K,U)$ and $TDMA(1,F)$.

Chapter IV: Variable-Cycle Service Strategies

The service strategies studied in the previous chapter do not share the resource dynamically. Since the capacity apportioned to each source is fixed for all time and is independent of the state of the system, a portion of the available channel capacity will occasionally be wasted on the transmission of idle bits, necessary to maintain synchronisation. This wastage can be reduced by allocating to each source a variable-length service period, which depends on the amount of work awaiting transmission at a buffer when its service period begins.

The variable length of source transmissions makes it necessary for the transmitter to include overhead which enables the receiver to determine the beginning and the end of each burst. It is also necessary to encode the source addresses since the channel is synchronous and some, or all, of the buffers may be empty. Under a cyclic selection strategy, the encoding of message lengths and source addressing are not separate issues, since the longer a transmission from one source, the smaller the uncertainty as to the origin of the next transmission.

In this chapter we study four protocols which provide the receiver with the information necessary to demarcate messages and determine their origin. The work focuses on two common service strategies: 'gated' and 'exhaustive.' Using a combination of analysis and simulation, we calculate the mean delay for these systems.

1. Model

There are M sources visited by the server in a cyclic manner. A cycle, $T_i^{(n)}$, is the length of time between the $(n-1)^{\text{th}}$ and n^{th} arrival of the server at the i^{th} buffer. A cycle, as seen by the i^{th} source, consists of a service period, $S_i^{(n)}$, and a vacation period, $V_i^{(n)}$. A service period is comprised exclusively of data and stuff bits from one source; the vacation period contains all the overhead of the cycle plus the service periods of the other $M-1$ sources.

We study two service strategies: gated, wherein only that work present in the i^{th} buffer at the instant the server arrives to it is served during the ensuing cycle; and exhaustive, where the server remains at a buffer until the buffer has been depleted. Since the duration of the service periods in both strategies is contingent on the number in the queue at the instant the server arrives, the service periods, and hence the cycles, will be of variable length.

2. Protocols

All of the protocols considered in this chapter use at least one flag to terminate the transmissions from each source; consequently, the data must be inhibited from replicating the flag. This requires an insertion rule which we shall take to be the same as the one discussed in Chapters 2 and 3.

The protocols differ according to the manner in which empty buffers are treated. Since information as to which buffers are empty is not *in itself* important to the receiver, it is apparently undesirable to expend capacity on the encoding of this information. However, not conveying this information to the receiver causes the regularity of the polling order to be destroyed, which in turn necessitates some form of origin addressing. Futhermore, because the channel is synchronous, it becomes necessary to include message-start information so that message beginnings can be differentiated from channel idle characters.

The following describes in detail the four protocols examined in this chapter which explore the interrelationship of the encoding of idle information and source addressing in variable-cycle strategies. In all cases the server visits the buffers cyclically.

Repeat Flag (RF): An empty buffer is considered to have a message of zero length. Service periods, including those which are null, are terminated with a single fixed length flag. This strategy, judging by the the length of the codewords assigned to each event, considers an empty buffer to be no more likely than a non-empty buffer.

Flag + Address (FA): The server selects the next non-empty buffer in the cycle, disregarding any intervening empty buffers. Each service period is prefaced by $\lceil \log_2 M \rceil$ bits containing the absolute address of the source currently transmitting. If all the buffers are empty, 0's are transmitted until there is an arrival to the system. In order to differentiate between an idle 0 and an address 0, each transmission begins with a 1. The end of a mesage is terminated with a flag. This policy uses state information and explicit addressing.

Flag + Unary Code (FU): Service periods having non-zero length are terminated with a flag. If an empty buffer is encountered during a cycle, a 0 is transmitted. A 1 is transmitted at the beginning of each service period to indicate its start. The unary code consists of a string of consecutive 0's terminated by a 1. In effect, message origins are signalled by unary coding of relative source addresses.

Multiple Flags (MF): There is a set of flags which are used to encode jointly the end of a source transmission and the number of intervening empty buffers (alternatively, the next source to transmit). Since only $M-1$ buffers can be examined without finding the system empty, the M^{th} flag is reserved to indicate that the system is empty. After the transmission of the M^{th} flag, the system generates a unary code (0's) until such time as the server encounters a non-empty buffer. At this point a 1 is transmitted, and the service period commences. We note that the prefacing of a 1 to a message is only necessary after the transmission of the M^{th} flag, since message start information is contained implicitly within all of the other flags in the set.

3. Mean Delay Computations

A complete delay analysis (that is, computation of the distribution) for each of the protocols described previously is beyond the scope of this work. The principal reason is the complexity introduced by incorporating a variety of state- and data-dependent overhead into the systems. In order to simplify our task, we focus solely on the mean delay in each of the systems. While this statistic is less satisfactory than the complete delay distribution, it is nonetheless an important performance measure in systems where queuing occurs, and allows us to compare the various protocols.

We begin with a theorem which enables the delay to be calculated from the first two moments of the cycle time (vacation time) in a gated (exhaustive) system. The formulae contained in the theorem are not new results; only the manner in which they are obtained is novel. Equation 4.2b is given in [7] and Eq. 4.1 can be found in [14].

Theorem 4.1:

Let $\mathrm{E}[T_i^j]$, $\mathrm{E}[V_i^j]$, and $\mathrm{E}[S_i^j]$ be the j^{th} steady-state moment of the cycle time, vacation time, and service period as seen by Source (i). Also, let $\overline{\mu_i^j}$ be the j^{th} moment of Source (i) message lengths. Then the mean delay, $\mathrm{E}[D_i]$, incurred in steady-state by a Source (i) customer, is

i) in a gated system,

$$\mathrm{E}[D_i^g] = (1 + \lambda_i \overline{\mu_i^1}) \frac{\mathrm{E}[T_i^2]}{2\mathrm{E}[T_i^1]}; \tag{4.1}$$

ii) in an exhaustive system,

$$\mathrm{E}[D_i^e] = \frac{(1 + \lambda_i \overline{\mu_i^1})\mathrm{E}[V_i^2] + (1 - \lambda_i \overline{\mu_i^1})\mathrm{E}[S_i^2]}{2\mathrm{E}[T_i^1]} \tag{4.2a}$$

$$= \frac{\lambda_i \overline{\mu_i^2}}{2(1 - \lambda_i \overline{\mu_i^1})} + \frac{\mathrm{E}[V_i^2]}{2\mathrm{E}[V_i^1]}. \tag{4.2b}$$

Proof:

Assume that the system is in equilibrium.

i) Gated System:

Under the gated service policy, a customer arriving in one cycle must wait until the next for service. This means that the delay of a Source (i) customer is composed of two parts: the first is the time, $D_1^i(t)$, from his arrival at time t until the next visit of the server to his buffer (or the end of the cycle), and the second is the time, $D_2^i(t)$, he must wait on queue after the server has arrived at his buffer. If we assume that customers in a buffer are served in the order in which they arrive, then this delay is identical to the total amount of work arriving at the Source (i) buffer before the arrival of the test customer.

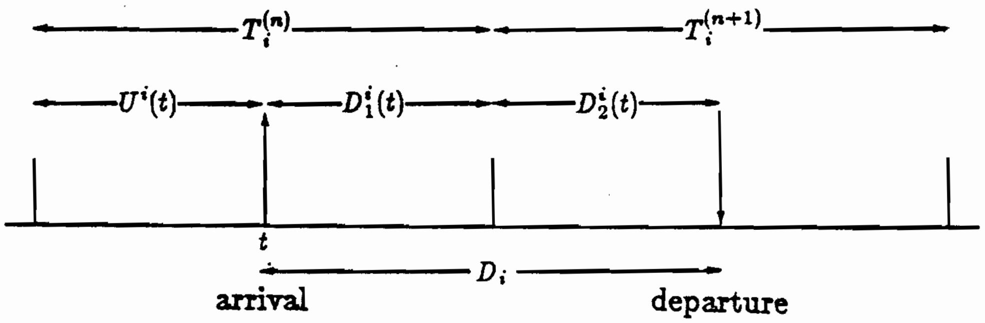

Figure 4.1 Delay Components for Gated System.

Referring to Figure 4.1, we see that

$$\mathrm{E}[D_i^g] = \mathrm{E}[D_1^i(t) + D_2^i(t)] = \mathrm{E}[D_1^i(t)] + \mathrm{E}[D_2^i(t)].$$

The expected amount of Source (i) work arriving before the tagged customer is $\lambda_i \overline{\mu_i^1} \mathrm{E}[U^i(t)]$, and since the test customer arrives according to a Poisson process and is chosen randomly, the two intervals, $D_1^i(t)$ and $U^i(t)$, are identically distributed; therefore

$$\mathrm{E}[D^i] = (1 + \lambda_i \overline{\mu_i^1}) \mathrm{E}[D_1^i(t)].$$

Since,

$$\mathrm{E}[D_1^i(t)] = \frac{\mathrm{E}[T_i^2]}{2\mathrm{E}[T_i^1]},$$

the proof is complete.

ii) Exhaustive System:

Referring to Figures 4.2a and 4.2b, one can see that whether a test customer is served in the cycle in which he arrives or in the next cycle depends on when he arrives in the cycle. If a Source (i) customer arrives before the completion of a Source (i) service period, then he will be served during that period. If on the other hand he arrives during a vacation period, it will be necessary for him to wait until the next cycle. Letting Q denote the event that a Source (i) customer arrives during a Source (i) service period, and Q^c the complementary event, we have

$$\mathrm{E}[D_i^e] = \mathrm{E}[D_i|Q]\Pr\{Q\} + \mathrm{E}[D_i|Q^c]\Pr\{Q^c\}.$$

The calculation of $\mathrm{E}[D_i|Q^c]$ is identical to that given for the gated system if the cycle time (T) is replaced by the vacation time (V) (compare Figure 4.1 and Figure 4.2a); thus

$$\mathrm{E}[D^i|Q^c] = (1 + \lambda_i\overline{\mu_i^1})\frac{\mathrm{E}[V_i^2]}{2\mathrm{E}[V_i^1]}.$$

In Figure 4.2b, the random variable $B_i(t)$ is the length of a busy period initiated by the remaining Source (i) work at time t plus the tagged arrival plus the necessary stuff bits. In order to find a relation between $\mathrm{E}[D_i|Q]$ and $\mathrm{E}[B_i(t)]$, we require a detailed knowledge of the arrival and insertion process. To overcome this difficulty, we shall assume that insertions are made independently at each point in time. This can be modelled by an increased service distribution as was done in Chapter 3, with $\overline{\mu_i^1} = (1 + P_s)$, and $\overline{\mu_i^2} = (1 + 3P_s)$.

This enables us to use the M/G/1 formula for the expected length of of a busy period initiated by $\mathrm{E}[D_i|Q]$ customers; that is

$$\mathrm{E}[B_i(t)] = \frac{\mathrm{E}[D_i|Q]}{1 - \lambda_i\overline{\mu_i^1}}.$$

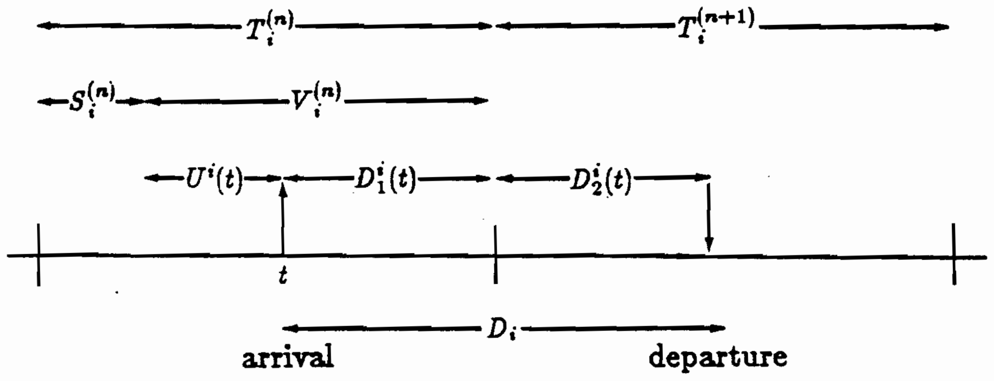

Figure 4.2a: Arrival in Vacation Period.

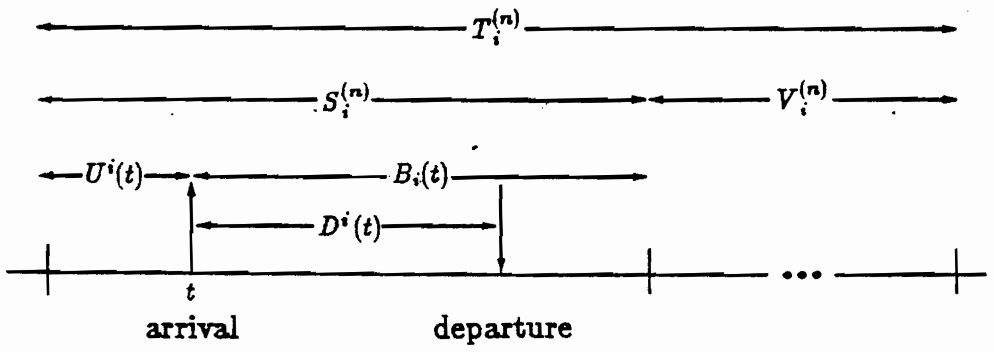

Figure 4.2b: Arrival in Service Period.

Since the arrival process is Poisson, the probability of an arrival occurring in a sub-interval of a larger interval can be calculated using a renewal-type arguement; therefore

$$\Pr\{Q^c\} = \frac{\mathrm{E}[V_i^1]}{\mathrm{E}[T_i^1]},$$

and

$$\Pr\{Q\} = \frac{\mathrm{E}[S_i^1]}{\mathrm{E}[T_i^1]}.$$

Rearranging terms and bringing all the components together yields

$$\mathrm{E}[D_i] = (1 + \lambda_i\overline{\mu_i^1})\frac{\mathrm{E}[V_i^2]}{2\mathrm{E}[V_i^1]}\frac{\mathrm{E}[V_i^1]}{\mathrm{E}[T_i^1]} + (1 - \lambda_i\overline{\mu_i^1})\mathrm{E}[B_i(t)]\frac{\mathrm{E}[S_i^1]}{\mathrm{E}[T_i^1]}.$$

Using an arguement identical to that given for $\mathrm{E}[D_i^1(t)]$ in the gated system, $\mathrm{E}[B_i(t)]$ can be written as $\mathrm{E}[S_i^2]/2\mathrm{E}[S_i^1]$. Substitution of this result and simplification gives Equation 4.2a.

Noting that the length of a Source (i) service period is identical to the length of a busy period initiated by the number of Source (i) customers arriving during a vacation period, we can compute $\mathrm{E}[S_i^2]$ in terms of $\mathrm{E}[V_i^2]$, and obtain Equation 4.2b.

The results apply for systems operating under all four protocols which we have described, and many others; however, protocols can be found where the theorem will not be applicable. The requirement is that an ergodic distribution of the cycle (vacation) time exist.

The utility of Theorem 4.1 is directly related to our ability to calculate the required moments. In systems where the overhead is independent of both the current state and the data, it is possible to calculate the distribution of the cycle or vacation time, from which the moments can be obtained [7]. An alternate approach used to calculate the moments of these simplified systems is to solve a set of M^2 linear equations for the auto- and cross-correlation values of the system , as detailed in [15,25,27].

The systems we wish to analyse do not, however, possess the simplifying independence property. This makes the analysis difficult and protocol dependent. In order that a wide variety of protocols can be compared, we obtain the required moments by simulation.

dent. In order that a wide variety of protocols can be compared, we obtain the required moments by simulation.

The moments are computed using the Law of Large Numbers; that is

$$E[T_i] = \frac{1}{N}\sum_{n=1}^{N} T_i^{(n)},$$

and

$$E[(T_i)^2] = \frac{1}{N}\sum_{n=1}^{N} (T_i^{(n)})^2.$$

The duration of the simulation is increased, as the throughput is increased in order to assure a confidence interval of at least 85 percent in each run.

While it is true that we could simulate the system directly, Theorem 4.1 simplifies the simulation considerably. First, it obviates tracking of individual customers; this facilitates the programming. Second, it reduces the CPU-time required for accurate results.

The minimum delay envelopes for the various protocols and service strategies are shown graphically in the subsequent section. The delays were calculated using moments obtained by simulation.

4. Observations

Figures 4.3 to 4.6 show the minimum delay envelopes of the four protocols under both exhaustive and gated service. The conclusion one can draw from these curves is that regardless of the protocol, the delay under an exhaustive service policy is always less than that of a gated system operating under the same protocol.

Initially, one might be tempted to attribute this to the fact that in an exhaustive system some customers will receive service in the cycle in which they have arrived, instead of being automatically forced to wait until the next cycle, as is the case in a gated system. However, upon reflection this alone cannot account for the improvement, since in the absence of protocol overhead both systms are conservative, and therefore, have the same mean delay [17].

The correct reason for the better performance is that less overhead is expended on source addressing in systems with exhaustive service than in those

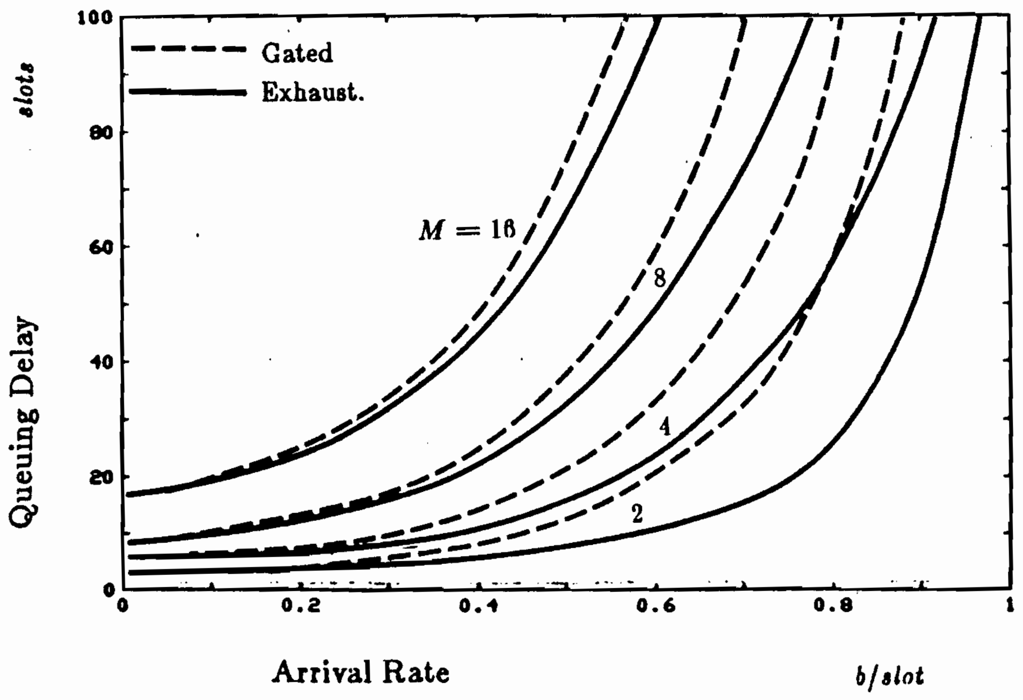

Figure 4.3: Delay Envelopes for Repeat-Flag Protocol.

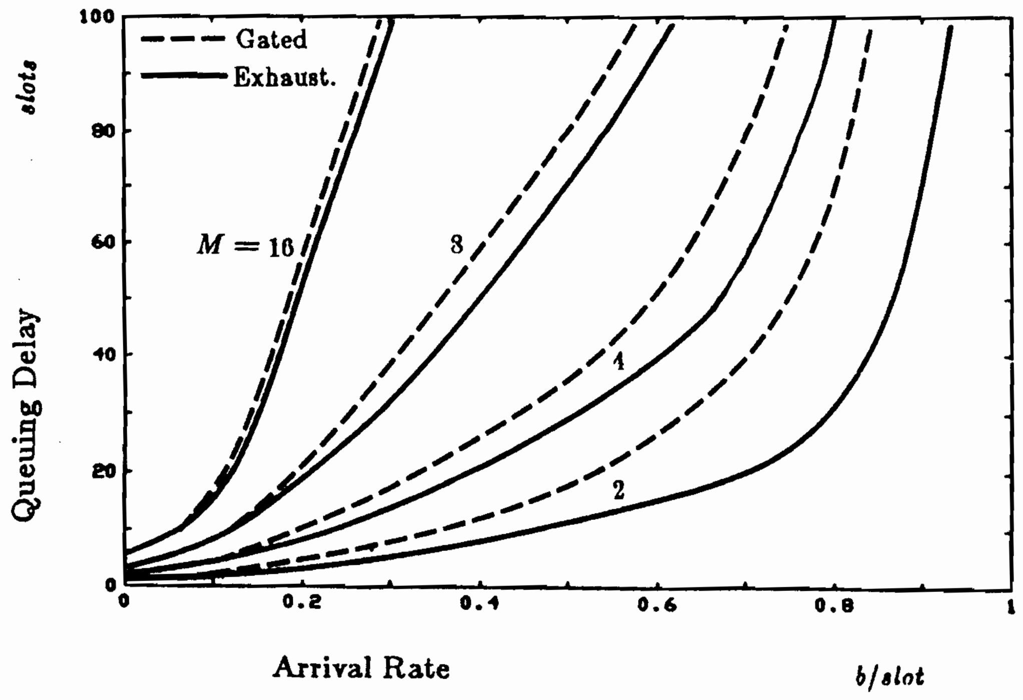

Figure 4.4: Delay Envelopes for Flag + Address Protocol.

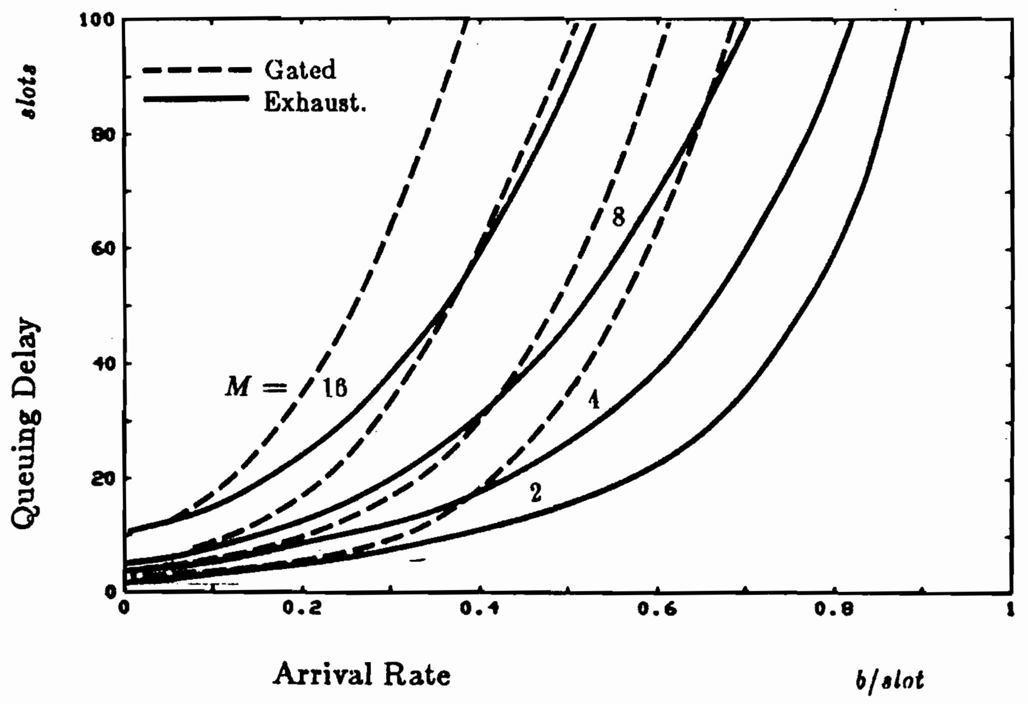

Figure 4.5: Delay Envelopes for Flag + Unary Code Protocol.

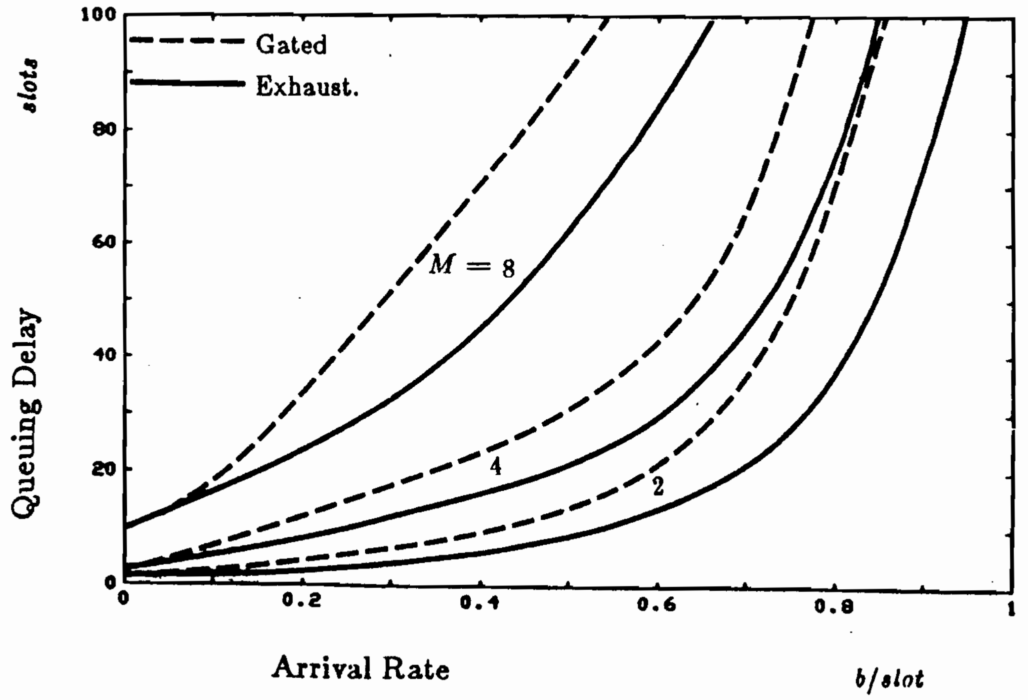

Figure 4.6: Delay Envelopes for Multiple Flags Protocol.

with gated service. More explicitly, whenever the server encounters an empty buffer, the system is assessed a penalty in the form of addressing overhead. This overhead retards the entry into service of subsequent buffers in the polling cycle and, negatively affects the delay of each succeeding customer. We conjecture that the probability of a server encountering an empty buffer is smaller in an exhaustive system than in a gated system. This would result in addressing overhead being transmitted less frequently, since for cyclic polling, the encoding of idle buffers is a form of source addressing.

The notion that addressing overhead adds to the delay can be further justified by comparing the performance of the various protocols (Figure 4.7).

The first piece of evidence is the performance of the Flag + Address protocol. This strategy has the worst mean delay of all the protocols considered, and as the number of sources is increased, the separation between it and the other protocols grows. Since the addressing overhead increases as the number of sources increases, this demonstrates the negative effect of source addressing on delay.

The second piece of supporting evidence is that, up to a certain point, the Flag + Unary Code (FU) protocol outperforms the Multiple Flags (MF) protocol and then the reverse is true. This is also a result of source addressing and can be seen by considering the two protocols at various throughput levels.

At low throughput, the MF and FU protocols operate similarly, except that due to source addressing, the flags in the MF scheme are longer than the flag in the FU strategy. This accounts for the better performance of the FU protocol in the low throughput range. In the medium range, the mean delay of the MF protocol approaches that of the FU protocol, since the flag lengths become comparable. In the higher throughput, range the MF strategy outperforms the FU protocol because it requires message start bits only when the system is empty, an event whose likelihood diminishes with increasing arrival rates. In contrast, the FU protocol requires a message start bit at the beginning of each message.

Finally, the last piece of evidence is the performance of the Repeat-Flag

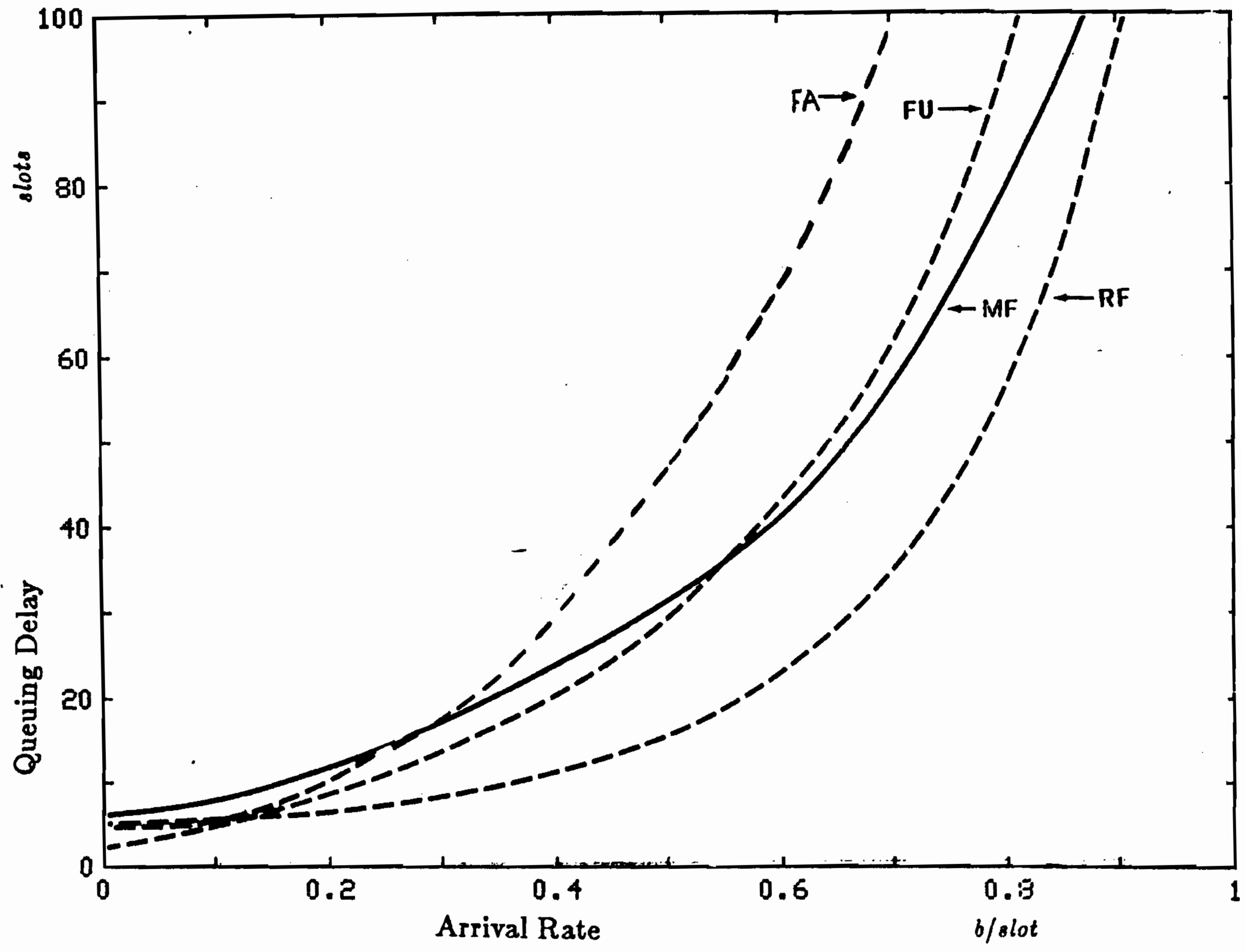

Figure 4.7: Delay for 4 Protocols, Exhaustive Service; $M = 4$.

(RF) protocol, which uses no special overhead to encode either an address or idle buffers. This strategy yields the lowest mean delay of all the protocols considered, in all but the very low throughput region.

Figure 4.7 indicates that for very low throughput a slight improvement in mean delay can be obtained by employing relative addressing (FU); however, the improvement is marginal at best.

5. Comparision of TDMA(K,F) and Exhaustive-RF.

Our final observation is based on a comparision of the best of the fixed-frame strategies and the best of the variable-cycle strategies (that is, $TDMA(K,F)$ and exhaustive-RF). Referring to Figure 4.8, one can see that the mean delay envelopes for $TDMA(K,F)$ are either equal to or lower than those for exhaustive-RF. Since each protocol uses a single flag to mark the end of a source transmission, the difference in the performance must be due to the due to the differences in the two service strategies.

In order to explain the improved performance of $TDMA(K,F)$, we first consider some known results for single source systems. In [14] it was shown that for a single source using a repetition of a single flag to mark an empty buffer, the delay is proportional to $\log\left(\frac{1}{1-p}\right)$, where $1-p$ is the probability of the source producing an idle character. In later work [23], a new strategy was developed, which was demonstrated to have a delay proportional to $\log\log\left(\frac{1}{1-p}\right)$. The new strategy improved the performance of the system by reducing the frequency at which flags are required. This was achieved by inserting artificial work into the system at periodic points in time. The purpose of the artificial work was to allow the queue to build so that it rarely underflowed, thereby reducing the frequency of flag use.

In the case of $TDMA(K,F)$, the artificial work is replaced by the *fixed* period when the server is away from any one buffer. This absent period allows time for arrivals to occur, so that upon the next visit it is unlikely that the buffer will be empty, and therefore unlikely that a flag will be required. In contrast, the exhaustive-RF strategy uses a flag each time a buffer exhausts. This is the reason for the beter delay performance of $TDMA(K,F)$.

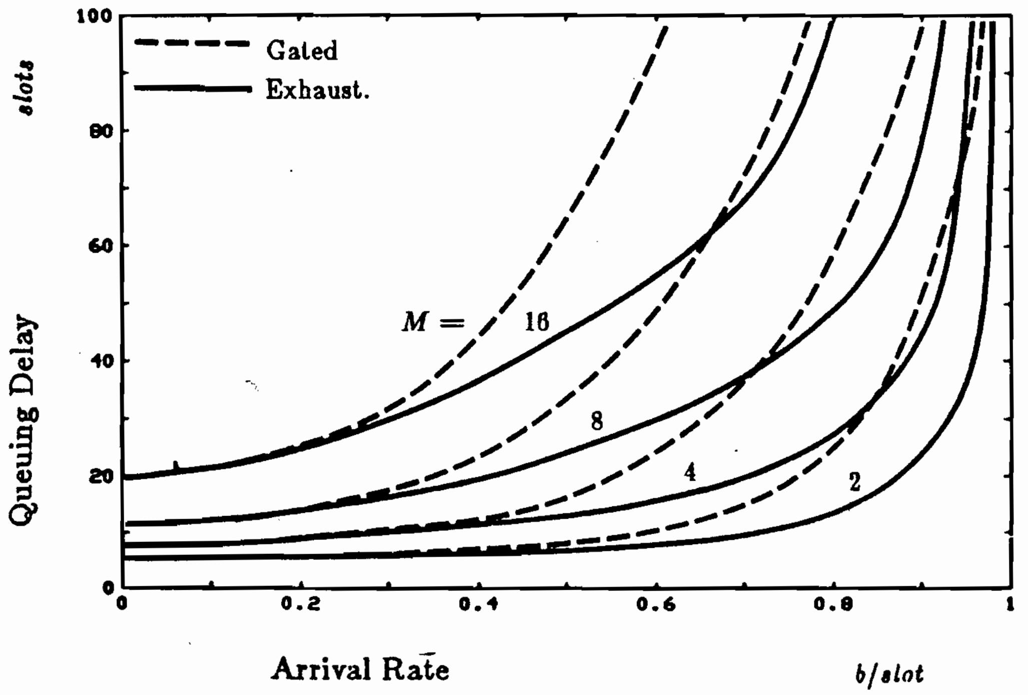

Figure 4.8: Comparision of *TDM-A*(K,F) and Exhaustive-RF.

Chapter V: Conclusion

1. Summary of Work

The intent of this work was to provide a comparative study of centralised, multiple access protocols. We were primarily interested in the interaction between service policies and the protocol overhead required to convey to the receiver information concerning source addressing and message length. This interaction was evaluated by comparing the mean delays for a variety of service strategies and protocols.

We began by analysing a set of protocols in which each source was periodically allocated a fixed portion of channel capacity. These protocols, dubbed fixed-frame strategies, required that buffer underflows be encoded, but did not, however, require any explicit overhead for the encoding of message origins. This was due to the periodic and immutable nature of the allocation policy. Of the three protocols we analysed, two required one slot in each sub-frame to encode idle information, while in the third, a flag was used to mark buffer underflows. The analysis of fixed-frame strategies consisted of computing the distributions of the number in the queue and delay. Also, stability criteria were given.

By noting that fixed-frame strategies apportion bandwidth independently of the current state of the system, we were led to examine service policies that shared the resource more dynamically. The two we studied were gated and exhaustive service. In these schemes, the length of the service period allotted to each source depended on the amount of work in a buffer when the server arrives to it, and had a variable length. Consequently, it was necessary to include message length information along with each transmission. For all the protocols we examined, this was accomplished by means of a flag. Also because one or more of the buffers could be empty and because the channel was taken to be synchronous, it was necessary to provide for source addressing. This accounted for the differences in the four protocols we considered. A mean delay comparision of these variable-cycle protocols was made using a combination of analysis and simulation.

Also included at the beginning of our work was a chapter on flags which provided the background for the flag based protocols. Necessary conditions for unambigious decodability of flags were given, as were the equilibrium probabilities of insertion for flags and sets of flags.

2. Summary of Major Results

In our study of fixed-frame service strategies, we found that in the low throughput range all three policies had comparable mean delays; however, in the high throughput range, the encoding of buffer underflows by inserting a flag into the data stream (that is, $TDMA(K,F)$) resulted in a substantially lower mean delay than either $TDMA(K)$ or $TDMA(K,U)$. The reason for the better performance of the flag strategy was that as the throughput increased, the average fraction of capacity expended on overhead in $TDMA(K,F)$ could be diminished at a faster rate than in either of the other two fixed-frame protocols.

Our comparision of gated and exhaustive service showed that, regardless of the protocol, exhaustive service always yielded a lower mean delay. We gave evidence that this was due to the lesser amount of source addressing overhead expended in exhaustive systems. Futhermore, we found that the Repeat-Flag protocol resulted in the lowest mean delay of the four protocols considered, except for very low throughputs. This was also attributed to a smaller expenditure of capacity on source addressing.

A comparision of the best of the fixed-frame protocols ($TDMA(K,F)$) and the best of the variable-cycle strategies (exhaustive-RF) showed that $TDMA(K,F)$ gave a lower mean delay for all throughput values. This result was attributed to the reduced variability of the TDMA(K,F) strategy, and consequent reduction in the amount of protocol overhead required to convey the desired information to the receiver.

Bibliography

[1] Arthurs, E. and Stuck, B.W., "Traffic Analysis Tools For Local Area Network Media Access Methods," *IEEE National Telecommunications Conference*, New Orleans, La., Vol. 1, A2.5/1-6, Dec. 1981.

[2] Bailey, N.T., "On Queuing Processes with Bulk Service," *J. Royal Statistical Society*, **B16**, pp. 80-87, 1955.

[3] Cinlar, E., *Introduction To Stochastic Processes*, New Jersey: Prentice-Hall, 1975.

[4] Cohen, J.W., *The Single Server Queue*, New York: North Holland, 1969.

[5] Cooper, R.B. and Murray, G., "Queues Served in Cyclic Order," *B.S.T.J.* **48**, pp. 675-689, Mar. 1969.

[6] Cooper, R.B., "Queues Served in Cyclic Order: Waiting Times," *B.S.T.J.* **49**, pp. 399-413, Mar. 1970.

[7] Eisenberg, M., "Queues with Periodic Service and Changeover Times," *Opns. Res.* **20**, pp. 440-451, 1972.

[8] Feller, W., *An Introduction to Probability and Its Applications*, Volume I, New York: Wiley, 1968.

[9] ————; *Idem.*, Volume II, New York: Wiley, 1966

[10] Gallager, R.G., "Basic Limits on Protocol Information in Data Communication Networks," *IEEE Trans. Inform. Theory*, **IT-22**, pp. 385-398, July, 1976.

[11] ____________; "Applcations of Information Theory to Data Communication Networks," M.I.T. Technical Report LIDS-P-1017, 1980; also presented at NATO Advanced Study Institute Univ. of East Anglia, Norwich, England, Aug.4-14, 1980

[12] ____________ and Camrass, J., "Encoding of Message Lengths for Data Transmission," *IEEE Trans. Inform. Theory*, **IT-21**, pp. 228-230, Mar. 1975.

[13] Hayes, J.F., " Performance Models of an Experimental Computer, Communications Network," *B.S.T.J.* **53**, pp. 225-259, 1974.

[14] __________, and Boorstyn, R.R., "Delay and Overhead in the Encoding of Data Sources," *IEEE Trans. Commun.*, **Com-29**, pp. 1678-1683, Nov. 1981.

[15] Humblet, P.A., "Source Coding for Communication Concentrators," Ph.D. Thesis and M.I.T. Technical Report ESL-R-78, Jan. 1978.

[16] Kaplan, M.A., "A Sufficient Condition for the Non-Ergodicity of a Markov Chain," *IEEE Trans. Inform. Theory*, **IT-25**, pp. 470- 471, 1979.

[17] Kleinrock, L., "*Queueing Systems, Volume II: Computer Applications*, New York: Wiley, 1976.

[18] Konheim, A.G., and Meister, B., "Waiting Lines and Times In a System with Polling," *J. of the Ass. for Computing Machinery*, **21**, pp. 470-490, July 1974.

[19] Little, J.D.C., "A Proof of the Queuing Formula $L = \lambda W$," *Opns. Res.* **9**, pp. 383-387, 1961.

[20] MacKenthun, K.M., "Bounds On Message Start Information For Messages Subject To Queueing Delay," *Proceedings 17 Ann. Allerton Conf. on Comm., Ctl, and Comp.*, John Hopkins University, Baltimore, Maryland. pp. 820-829, 1979.

[21] Neilson, P.T., " A Note on Bifix-Free Sequences," *IEEE Trans. Inform. Theory*, **IT-19**, pp. 704-706, 1973.

[22] Pakes, A.G., "Some Conditions for Ergodicity and Recurrence of Markov Chains," *Opns. Res.* **17**, pp. 1059-1061, 1969.

[23] Roskind, J.A., "Protocols for Encoding Idle Characters in Data Streams," *Master's Thesis*, M.I.T. University, June, 1980.

[24] Ross, S., *Applied Probability Models with Optimization Applications*, Holde: San Francisco, 1970.

[25] Rubin, I., and DeMoraes, L.F., "Polling Schemes for Local Communication Networks," *ICC '81 Conference Record*, pp. 33.5.1-7, Denver, June 1981.

[26] Stidham, S., "A Last Word on $L = \lambda W$," *Opns. Res.* **22**, pp. 417-421, 1974.

[27] Sykes, J.S., "Simplified Analysis of an Alternating-Priority Queuing Model With Setup Times," *Opns. Res.* **19**, pp. 1182-1192, 1978.

Appendix 1:
Proof of Theorem 3.1

In this section we calculate the quantities necessary to complete the proof of *Theorem* 3.1.

Lemma A1:

Let X be a discrete random variable, then the steady-state mgf of the random variable defined by

$$Y = \alpha X - \beta X_{(modK)}, \qquad \alpha,\, \beta \; rational, \tag{A1}$$

is given by

$$\mathrm{E}[z^Y] = \frac{1}{T}\sum_{r=0}^{K-1} \frac{1 - z^{\beta K}}{1 - z^{\beta}\omega_r^*}\mathrm{E}[(\omega_r z^{-\alpha})^X], \tag{A2}$$

where ω_r is defined as $e^{i2\pi r/K}$.

Proof :

$$\mathrm{E}[z^Y] = \mathrm{E}[z^{(\alpha X - \beta X_{(modK)})}]$$

$$= \sum_{i=0}^{\infty}\sum_{j=0}^{K-1} z^{(\alpha i - \beta j)} \Pr\{X = i, X_{(modK)} = j\}$$

$$= \sum_{l=0}^{\infty}\sum_{j=0}^{K-1} z^{(\alpha(lK+j) - \beta j)} \Pr\{X = lK + j\}$$

$$= \sum_{m=0}^{K-1} z^{(\alpha-\beta)m} \sum_{l = multiple\, K} z^{\alpha l} \Pr\{X = l + m\} \tag{A3}$$

Since

$$\frac{1}{K}\sum_{r=0}^{K-1} e^{i2\pi l/K} = \begin{cases} 1, & \text{if } l = nK, \quad n = 0, 1, 2, \ldots; \\ 0, & \text{otherwise;} \end{cases} \tag{A4}$$

$$E[z^Y] = \frac{1}{K}\sum_{m=0}^{K-1} z^{(\alpha-\beta)m} \sum_{l=0}^{\infty}\sum_{r=0}^{K-1} z^{\alpha l} e^{i2\pi rl/K}\,\Pr\{X = l+m\}$$

$$= \frac{1}{K}\sum_{r=0}^{K-1}\sum_{m=0}^{K-1} z^{(\alpha-\beta)m} \sum_{j=m}^{\infty} z^{\alpha(j-m)} e^{i2\pi(j-m)/K}\,\Pr\{X = j\}$$

$$= \frac{1}{K}\sum_{r=0}^{K-1}\sum_{m=0}^{K-1}\left(z^{-\beta m} e^{i2\pi rm/K}\right)\bullet$$

$$\left\{\sum_{j=0}^{\infty} z^{\alpha rj} e^{i2\pi rj/K}\,\Pr\{X = j\} - \sum_{j=0}^{m-1} z^{\alpha rj} e^{i2\pi rj/K}\,\Pr\{X = j\}\right\}$$

$$= \frac{1}{K}\sum_{r=0}^{K-1}\sum_{m=0}^{K-1} z^{-\beta m} e^{i2\pi r\underline{m}/K}\, E[(z^{\alpha} e^{i2\pi r/K})^{X}]$$

$$- \frac{1}{K}\sum_{r=0}^{K-1}\sum_{m=0}^{K-1} z^{-\beta m} e^{i2\pi rm/K} \sum_{j=0}^{m-1} z^{\alpha j} e^{i2\pi rj/K}\,\Pr\{X = j\}. \qquad (A5)$$

By examining the indices r and j, we find that the second term of (A5) is equal to 0. Therefore, we have

$$E[z^Y] = \frac{1}{K}\sum_{r=0}^{K-1}\sum_{m=0}^{K-1} z^{-\beta m} e^{i2\pi rm/K}\, E[(z^{\alpha} e^{i2\pi r/K})^{X}] \qquad (A6)$$

Summing over m and substituting ω_r for $e^{i2\pi r/K}$ gives

$$E[z^Y] = \frac{1}{T}\sum_{r=0}^{K-1} \frac{1 - z^{-\beta K}\omega_r^{\beta K}}{1 - z^{-\beta}\omega_r^{*}} E[(z^{-\alpha}\omega_r)^{X}].$$

Noting that $\omega_r^{\beta K} = 1$ completes the proof. ∎

Note: The result shows that the transform of a random variable which is the modulus of some other random variable is given by the sum of filtered, equi-spaced samples on the unit circle of the original random variable's transform. This is a discrete version of the Poisson Summation Formula [9] and the sampling theorm.

Corollary A1:

The mean of the random variable $Y = X_{(modK)}$ is given by

$$\overline{Y} = \frac{K-1}{2} - \sum_{r=1}^{K-1} \frac{\mathrm{E}[\omega_r^X]}{1-\omega_r^*} \tag{A7}$$

Proof: Set $\alpha = 0$ and $\beta = -1$ in (A1), compute the derivative with respect to z, and evaluate the result at $z = 1$. Since the term corresponding to $r = 0$ is indeterminate, it must be separated from the other terms in the sum and l'Hopital's rule applied. ∎

Appendix 2:
Proof of Theorem 3.2

In this section, we demonstrate the validity of the results contained in Theorem 3.2. We begin by calculating the steady-state mgf of the number in the queue at the beginning of the i^{th} slot of a sub-frame N_i. For convenience we repeat the state equations given in (3.2):

$$N_i^{(n)} = (N_{i-1}^{(n)} - U_{i-1}^{(n)})^+ + A_{j-1}^{(n)}(M), \qquad i = 1,2,\ldots,K; \qquad (3.2.3a)$$

$$N_{K+1}^{(n)} = N_K^{(n)} + A_{K+1}^{(n)}(M); \qquad (3.2.3b)$$

$$N_o^{(n+1)} = N_{K+1}^{(n)}; \qquad (3.2.3c)$$

$$U_i^{(n)} = \min[N_{i-1}^{(n)}, U_{i-1}^{(n)}], \qquad i = 1,2,\ldots,K; \qquad (3.2.3d)$$

$$U_o^{(n)} = 1. \qquad (3.2.3e)$$

We assume that there exists an equilibrium distribution and remove the superscript n; then by conditioning on the value of the server-state variable, U_i, we can write

$$\mathrm{E}[z^{N_i}] = \mathrm{E}[z^{N_i}|U_i = 1]\Pr\{U_i = 1\} + \mathrm{E}[z^{N_i}|U_i = 0]\Pr\{U_i = 0\}. \qquad (A2.1)$$

We first concentrate on the second term of the summand. If $U_i = 0$, then for some j, $j \in \{0,1,\ldots,i-1\}$, $N_j = 0$ and $U_j = 1$ (*i.e.*, the buffer in question emptied). Consequently, the number in the queue at the i^{th} slot of the sub-frame will be the number of arrivals in $[j,i)$, thus

$$\mathrm{E}[z^{N_i}|U_i = 0]\Pr\{U_i = 0\} = \sum_{j=0}^{i-1} \mathrm{E}[z^{A(M)}]^{(i-j)} \Pr\{N_j = 0, U_j = 1\}, \qquad (A2.2)$$

where we have assumed the arrival process is identical in each group; that is $A_i \sim A$, for all i.

Turning to the first term of (A2.1) and employing (3.2.3a), we have, for $i \in \{1,2,\ldots,K\}$,

$$\mathrm{E}[z^{N_i}|U_i = 1]\Pr\{U_i = 1\} = \mathrm{E}[z^{(N_{i-1}-U_{i-1})^+}|U_i = 1]\mathrm{E}[z^{A(M)}]\Pr\{U_i = 1\}. \qquad (A2.3)$$

But if $U_i = 1$, then $U_{i-1} = 1$ and $N_{i-1} > 0$; therefore

1) $\mathrm{E}[z^{(N_{i-1}-U_{i-1})^+}|U_i = 1] = \mathrm{E}[z^{(N_{i-1}-1)}|U_{i-1} = 1, N_{i-1} > 0]$;

2) $\Pr\{U_i = 1\} = \Pr\{N_{i-1} > 0, U_{i-1} = 1\}$.

Substituting these results in (A2.3) yields

$$\begin{aligned}\mathrm{E}[z^{N_i}|U_i = 1]\Pr\{U_i = 1\} = z^{-1}\mathrm{E}[z^{N_{i-1}}|U_{i-1} = 1, N_{i-1} > 0]\mathrm{E}[z^{A(M)}]\bullet \\ \Pr\{N_{i-1} > 0, U_{i-1} = 1\}.\end{aligned} \tag{A2.4}$$

Noting that

1) $\mathrm{E}[z^{N_i-1}|U_{i-1} = 1] = \mathrm{E}[z^{N_{i-1}}|U_{i-1} = 1, N_{i-1} > 0]\Pr\{N_{i-1} > 0|U_{i-1} = 1\}$ $+ \Pr\{N_{i-1} = 0|U_{i-1} = 1\}$;

2) $\Pr\{U_{i-1} = 1, N_{i-1} > 0\} = \Pr\{N_{i-1} > 0|U_{i-1} = 1\}\Pr\{U_{i-1} = 1\}$;

yields, for $i \in \{1, 2, \ldots, K\}$,

$$\begin{aligned}\mathrm{E}[z^{N_i}|U_i = 1]\Pr\{U_i = 1\} = z^{-1}\mathrm{E}[z^{A(M)}]\bullet \\ \left\{\mathrm{E}[z^{N_i-1}|U_{i-1} = 1]\Pr\{U_i - 1 = 1\} - \Pr\{N_{i-1} = 0, U_{i-1} = 1\}\right\}.\end{aligned} \tag{A2.5}$$

Letting $P_{1,0}^{(i)} = \Pr\{N_{i-1} = 0, U_{i-1} = 1\}$ and using succesive substitution, we obtain

$$\begin{aligned}\mathrm{E}[z^{N_i}|U_i = 1]\Pr\{U_i = 1\} = & z^{-i}\mathrm{E}[z^{A(M)}]\mathrm{E}[z^{N_0}|U_0 = 1]\Pr\{U_0 = 1\} \\ & - \sum_{j=0}^{i-1}(z^{-1}\mathrm{E}[z^{A(M)}])^{i-j}P_{1,0}^{(j)};\end{aligned} \tag{A2.6}$$

however, $U_0 = 1$, $w.P.\,1$ (3.2.3e); therefore (A2.6) is equivalent to

$$\mathrm{E}[z^{N_i}|U_i = 1]\Pr\{U_i = 1\} = z^{-i}\mathrm{E}[z^{A(M)}]\mathrm{E}[z^{N_0}] - \sum_{j=0}^{i-1}(z^{-1}\mathrm{E}[z^{A(M)}])^{i-j}P_{1,0}^{(j)}. \tag{A2.7}$$

Combining (A2.7) and (A2.2), we have that the *mgf* of the number in the queue at the beginning of the i^{th} slot of a sub-frame is

$$\mathrm{E}[z^{N_i}] = z^{-i}\mathrm{E}[z^A(M)]\mathrm{E}[z^{N_0}] - \sum_{j=0}^{i-1}(P_{1,0}^{(j)}\mathrm{E}[z^{A(M)}])^{i-j}(1 - z^{-(i-j)}). \tag{A2.8}$$

Taking the transform of (3.2.3b), using (3.2.3c) and rearranging terms, we find

$$\mathrm{E}[z^{N_K}] = \frac{\mathrm{E}[z^{N_0}]}{\mathrm{E}[z^{A(M)}]}. \qquad (A2.9)$$

Setting $i = K$ in (A2.8), substituting the expression for $\mathrm{E}[z^{N_K}]$, (A2.9), and rearrangement yields

$$\mathrm{E}[z^{N_0}] = \frac{\mathrm{E}[z^{A(M)}]^{K+1} \sum_{j=0}^{K-1} P_{1,0}^{(j)}\, \mathrm{E}[z^{A(M)}]^{-j}(z^K - z^j)}{z^K - \mathrm{E}[z^{A(M)}]^{K+1}}; \qquad (A2.10)$$

which is the first conclusion of *Theorem* 3.2 if we note that $\mathrm{E}[z^{A(M)^{K+1}}] = \mathrm{E}[z^{A(T)}]$. By inserting (A2.10) in (A2.8), we obtain the second conclusion. ▮

Equation (A2.10) contains K constants, $(P_{1,0}^{(i)},\ i \in \{0, 1, \ldots, K-1\})$, which remain to be determined. Using Rouché's Theorem , together with the fact that the *mgf* of a probability distribution function is analytic in the unit disc, one can show that the unknown probabilities satisfy the following set of linear equations:

$$\sum_{j=0}^{K-1} P_{1,0}^{(j)}\, \mathrm{E}[\theta_i^{A(-jM)}](\theta_i^K - \theta_i^j) = 0, \qquad i : 1, 2 \ldots, K-1;$$

$$\sum_{j=0}^{K-1} (K-j) P_{1,0}^{(j)} = K - T\lambda;$$

where θ_i, $i \in \{0, 1, \ldots, K-1\}$, is the i^{th} root of the characteristic equation,

$$z^K - \mathrm{E}[z^A(T)]$$

lying within the unit disc, $|z| \leq 1$, with $\theta_0 = 1$. ▮

Because the arrival process is an independent-increment process, we are able to interpolate between the i^{th} and $(i+1)^{\text{th}}$ *mgf*. Performing the calculations (identical to those of § 3.1) and defining $N_i(t)$ as the amount of work awaiting transmission at an instant $iM + t$, $t \in [0, T)$, within a frame, gives

$$\mathrm{E}[z^{N_i(t)}] = \frac{\mathrm{E}[z^{N_{i+1}(t)}]\mathrm{E}[z^{A(t)}]}{\mathrm{E}[z^{A(M)}]}; \qquad i : 0, 1, \ldots, K. \qquad (A2.14)$$

The average number in the queue, as seen by an average arrival, is computed according to the following formula:

$$\overline{N} = \frac{1}{T}\int_0^T \mathrm{E}[N(t)]dt = \frac{1}{T}\sum_{i=0}^{K}\int_0^M \mathrm{E}[N_i(t)]dt; \qquad (A2.15)$$

where

$$\mathrm{E}[N_i(t)] = \frac{\mathrm{d}}{\mathrm{dz}}\Big[\mathrm{E}[z^{N_i(t)}]\Big]_{z} = 1.$$