Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Title of Thesis: Integrated Voice/Data Services on Multihop Radio Networks

Chialun Lu, Master of Science in Electrical Engineering, 1989

Thesis directed by: Irving Wang

Department of Electrical Engineering

Integrated voice/data is the first step toward Integrated Services Digital Networks (ISDN). Various techniques have been proposed and analyzed for multiplexing voice and data over a large bandwidth channel. Network efficiency is what we are most concerned about. In this paper, we address the problem of determining optimal access policies for a tandem ISDN network by using Dynamic Programming. Computer simulation shows that optimal policy minimizes the data delay.

Integrated Voice/Data Services on Multihop Radio Networks

by

Chialun Lu

Thesis submitted to the Faculty of the Graduate School of the New Jersey Institute of Technology in partial fulfillment of the requirements for the degree of master of Science in Electrical Engineering

1989

APPROVAL SHEET

Title of Thesis: Integrated Voice/Data Services on Multihop Radio Networks

Name of Candidate: Chialun Lu

Degree: Master of Science in Electrical Engineering

Thesis and Abstract Approved : _____ _____

Dr. Irving Wang *Date*
Assistant Professor
Department of Electrical Engineering

: _____ _____

Dr. Anthony Robbi *Date*
Department of Electrical Engineering

: _____ _____

Dr. John Carpinelli *Date*
Department of Electrical Engineering

New Jersey Institute of Technology, Newark, New Jersey.

VITA

Name: Chialun Lu.

Permanent address:

Degree and date to be conferred: MSEE, 1989.

Date of birth:

Place of birth:

Secondary education: Taipei Vocational High School, June 1980.

Collegiate institutions attended	Date	Degree	Date of Degree
National Taiwan College of Education	9/80	B. Ed.	6/84
University of Texas at El Paso	9/87		
New Jersey Institute of Technology	1/88	MESS	10/89

Major: Electrical Engineering.

CONTENTS

CHAPTER 1 INTRODUCTION

1-1 ISDN Evolution

Thousands of Data Communication Networks are deployed worldwide. These systems range from small networks within a building to large ones covering entire country or even spanning the globe. No matter how these networks are operated, privately or publicly, they have the same object, that is, to provide communication ability from users to others.

Some networks use packet-switched technology, in which blocks of data called packets are transmitted from a source to a destination. Source and destination can be any type of data communicating and/or data-handling devices, such as user terminals, computers or printers.

Other networks use circuit-switched technology. The most common example is the telephone networks to which we are accustomed. In this type of networks, which usually transmit voice or data, a private transmission path is established between any pair or group of users and is held as long as required. Integrated networks, which combines aspects of both packet- and circuit- switched technology, are now beginning to be deployed.

Nowadays, the real-time voice transmission is still the main mode of communication and all projections indicate that voice will continue to be the heaviest user of communication facilities worldwide. In addition to the voice traffic, the other types of traffic such as data, files, facsimile and images, should also be considered for transmission as well. However, most of the telephone networks are still designed for voice analog only; thus, data signals must normally be converted to voice-type analog signals using devices called modems. This limits the rate of transmission of data to at most 9600-bps, and then only using private, specially conditioned transmission facilities. Typically, in the public telephone network, data bit rates can only achieve 1200 bps or 2400 bps.

In early 1960s, American Telephone & Telegraph (AT&T) first introduced a digital carrier system to the world and then the telephone networks began a new era, in which the telephone network began to use digital transmission and switching facility. Once telephone networks become all digital, digital voice, digital images and data files could be transmitted in the same network. At present time, telephone networks are partly analog and partly digital, only providing analog voice transmission but inadequate for data transmission, facsimile and video. Therefore, a new system is necessary to provide all users with all types of traffic services.

Integrated Services Digital Networks (ISDN) is the new system redesigned for the above purpose and this new advanced digital system is expected to replace a major portion of worldwide telephone system by early 21 century. However, ISDN will not happen overnight because present investment in current telephone system is so huge. So, ISDN have to

coexist with present analog system for many years. This reason has had a major influence on final form of ISDN. Even when ISDN is a brand new redesigned system, it still has to consider current voice telephone system.

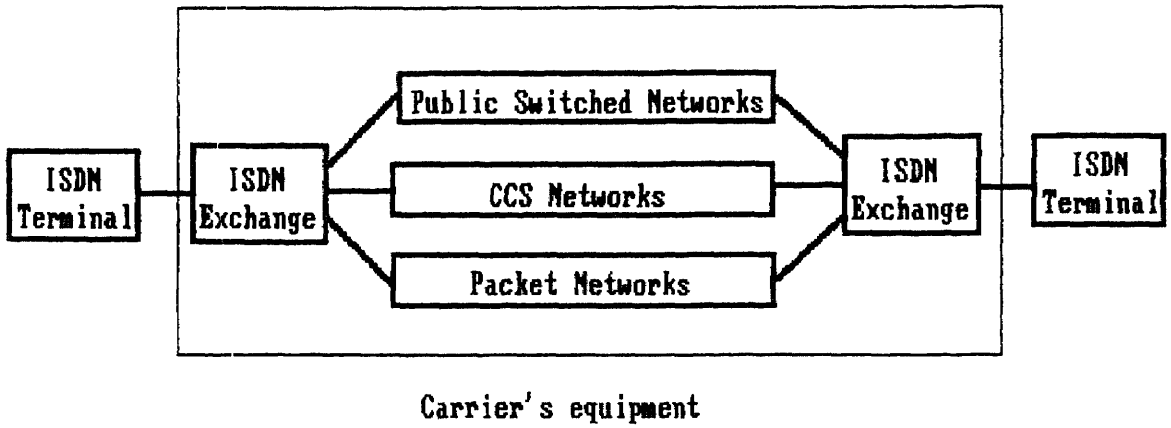
The analog voice telephone system (the public switched network in telephone jargon) originally sent all its control information in the same 4 kHz channel used for voice. Pure tones at various frequencies were used for signaling by the system itself. This scheme, known as in-band signaling, meant that in theory users could interfere with the internal signaling system.

In 1976 AT&T built and installed a packet switching network separate from the main public switched network. The network, called Common Channel Interoffice Signaling (CCIS), ran at 2.4 kbps and was designed to move the signaling traffic out-of-band and these signals used dedicated signaling slots of Time Division Multiplexing (TDM) frame.

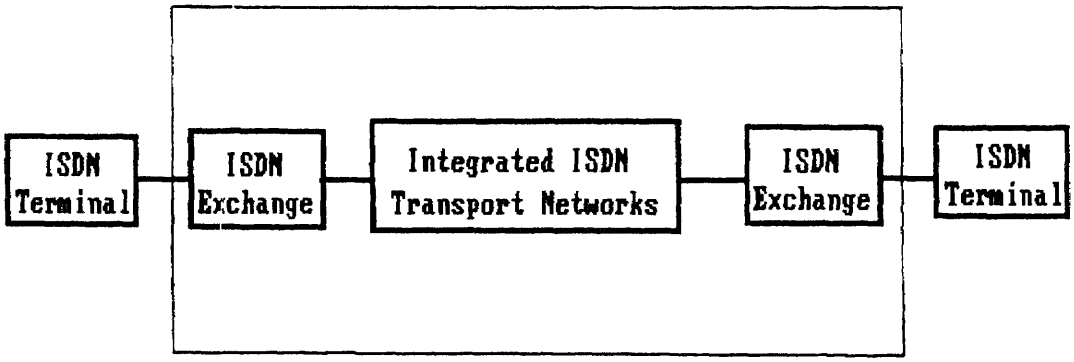
Another development that has been taking place since the mid 1970s is the growth of packet switching networks, which are dedicated for data and provide local terminal to access a remote database or time-sharing system.

These facts mean that the early stage of ISDN was designed based on the limitation of the existing public circuit switching networks, CCS network and packet networks. The first step toward ISDN was to define and standardize the user to ISDN interface. Next step was to slowly start replacing existing end offices with ISDN exchanges that support the ISDN

interface as shown in Fig. 1-1 (a). Eventually the existing transmission and switching networks will be replaced by an integrated one, as shown in Fig. 1-1 (b), but this will not occur until next century.



(a)



(b)

Figure 1-1(a) Initial stage of ISDN evolution

(b) Later ISDN

The principal benefits of ISDN is that the users can be expressed in terms of cost saving and flexibility. The integration of voice and a variety of data on a single transport system means that the user does not have to buy multiple services to meet multiple needs. The efficiency and economics of scale of an integrated network allows these services to be offered at lower cost than if they were provided separately. Further, the user needs to bear the expense of just a single access line to those multiple service.

The ISDN will provide a variety of services supporting existing voice and data application as well as providing for applications now being developed. Table 1-1 shows the type of services that could be supported by ISDN. These services fall into the broad categories of voice, digital data, text and image.

1-2 Multiplexing and Switching Techniques

FDM

Frequency Division Multiplexing (FDM) is possible when useful bandwidth of the medium exceeds the required bandwidth of signals to be transmitted. A number of signals can be carried simultaneously if each signals is modulated on to a different carrier frequency.

Fig. 1-2 shows how the voice-grade telephone channels are multiplexed in FDM. Filters limit the usable bandwidth to about 3 kHz per voice

grade channel. When many channels are multiplexed together, carrier frequency allocate every 4 kHz will keep them separate well.

BW / Service	64kbps	Wideband (>64kbps)
TELEPHONE	Telephone Leased Circuits Information retrieval	Music
DATA	Packet switched Circuit switched Leased circuits Telemtary Funds transfer Informatino retrieval Mailbox Alarms	High speed computer communication
TEXT	Telex Videotex Electronic mail	
IMAGE	Facsimile Surveillance	TV conferencing Teletex Videophone Cable TV distribution

Table 1-1 Candidate Service for ISDN

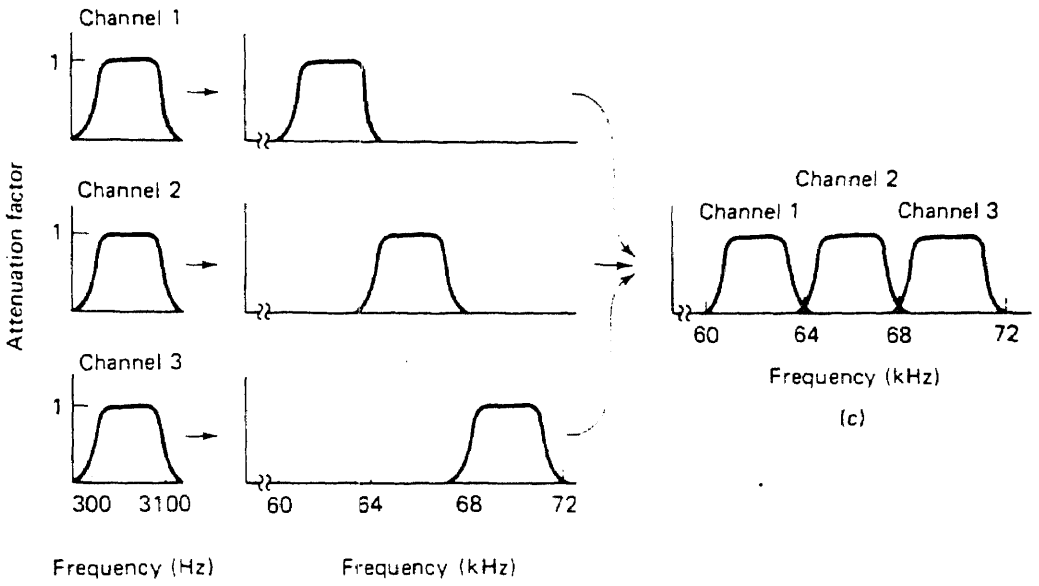


Fig.1-2 Frequency Division Multiplexing

TDM

Time Division Multiplexing is possible when the achievable data rate of the medium exceed the data rate of digital signals to be transmitted. Multiple digital signals or analog signals carrying digital data can be carried on a single transmission path by interleave portions of each signal in time.

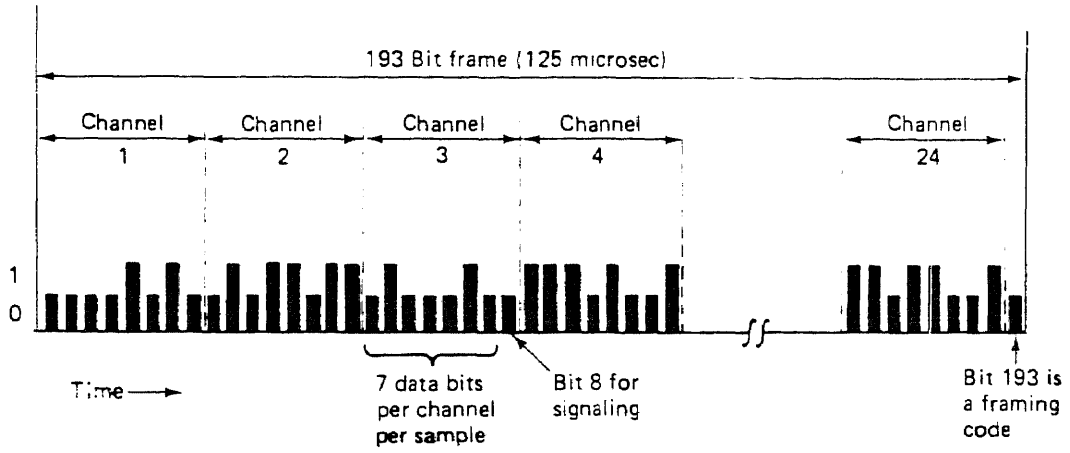


Fig 1-3 Time Division Multiplexing (TDM).

Fig. 1-3 shows DS1 transmission format which multiplexing 24 channels. Each frame contains eight bits per channel plus a control bit for 24x8+1=193 bits. For voice transmission, each channel bandwidth is 4 kHz. According to Nyquist's theory, sample rate must be no less than 8 kHz. With a frame length 193 bits, we have a data rate 8kx193=1.5444 Mbps to provide 24 voice channel services.

Circuit Switching

Communication via circuit switching implies that there is a dedicated communication path between two users. That path is connected sequence of links between nodes. On each physical link, a channel is dedicated to connection. The most common example of circuit switching is the public telephone network.

In circuit switching, before any data or voice can be transmitted, an end-to-end circuit must be established. The transmitting signals can be digital or analog. Channel capacity is dedicated for the duration of a connection, even if no data are being transferred.

Packet Switching

There are no dedicated path in packet switching which only can transmit packet data. Packets are stored until being delivered, and each packet has a route, even if they have the same source and destination. With packet switching, any unused bandwidth may be utilized by other packets from unrelated sources going to unrelated destination, because circuits are never dedicated.

1-3 Multihop Packet Radio Networks

Broadcast radio networks carry data between nodes equipped with radio transceivers. If two end users are not within communication range, intermediate radio will rebroadcast the data, thus creating a multihop radio networks.

Various technology and routing protocol have been proposed to establish reliable end-to-end path for maximum network throughput, or we can say best sharing. Sharing can be occurred in three domains, frequency, time and space.

One of the major problems in the multihop radio network is how to assign the available channel capacity among the users in some optional or efficient manner.

Multihop radio networks operating under random access protocol are susceptible to two types of interference--primary conflict and secondary conflict.

Primary Conflict

In the same channel, a node can not receive and transmit simultaneously, nor can a node receive from more than one transmitter simultaneously. As shown in Fig. 1-4, assume each link only has one channel and the same frequency. If A transmits data to B, then C can not transmit data to B or A; otherwise, it will cause primary conflict. A line graph can be developed from Fig.1-4 shown in Fig. 1-5.

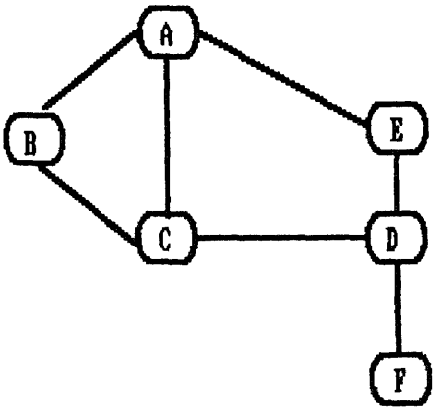


Fig 1-4 Primary Conflict

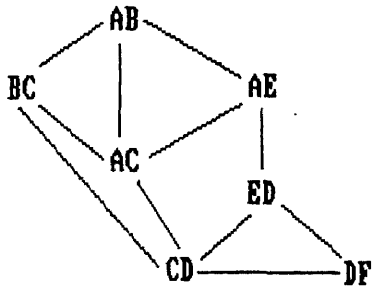


Fig 1-5 Line Graph

In the line graph, if any link is active (transmission and receiver), then its neighbor link can not be active. For example, if AB link is active, no matter A transmits to B or B transmits to A, link BC, AC and AE can not be active.

Secondary Conflict

Frequently it is possible for a receiver to be blocked by cumulative power of its transmitting neighbors. For a given link threshold, K, any node that is in neighborhood of more than K active transmitter cannot receive.

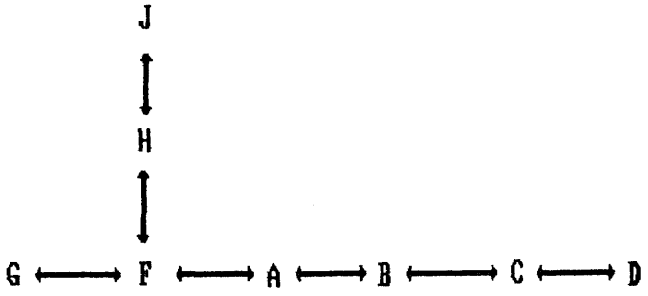


Fig 1-6 Secondary Conflict

In fig. 1-6, if threshold K=1, suppose link A-B is active, node A is transmitting to B when C transmits to D or H transmits F will cause secondary conflict.

CHAPTER 2 MODEL DESCRIPTION

2-1 Fixed boundary and movable boundary scheme

Various techniques have been proposed and analyzed for multiplexing (either in time or frequency) voice and data over a large bandwidth. In general, there are two kinds of techniques used to multiplex voice and data: fixed boundary scheme and movable boundary scheme.

In the fixed boundary scheme, the communication channels on each link are divided into two groups, one of which is assigned to the voice and the other one is dedicated to the data. Both voice and data are not allowed to use the channels, no matter they are busy or not, that do not belong to their own groups. However, in the movable boundary scheme, data traffic is allowed to use idle channels of voice. But voice is still not allowed to use the idle slots of data. The frame structure suggested is indicated in Fig. 2-1. The blocking probabilities of voice calls remain the same in either case. Queueing delay of data packets are reduced in the movable boundary scheme since advantage is taken by using the idle slot of voice call to transmit more data packets. Detail descriptions of the time-division-multiplexing frame structure for such scheme appear in [1] and [2].

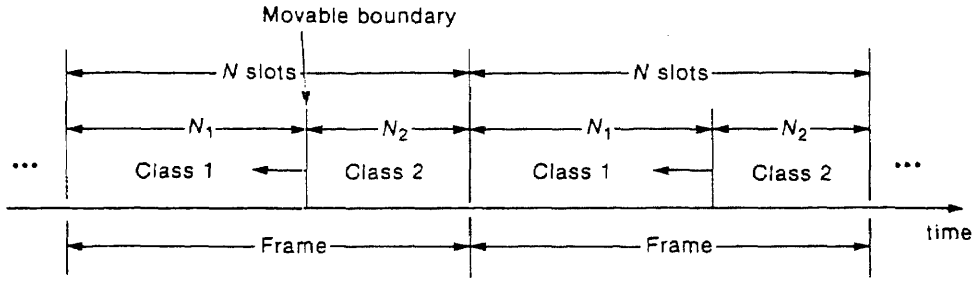


Fig. 2-1 Movable-boundary strategy

2-2 Model assumption

Consider a 3-node 2-link network, illustrated in Fig.2-2, operating in multihop radio environment. Frequency Division Multiplexing (FDM) scheme is used to multiplex two types of digital packets, data and voice, into the same communication links. A model similar to the one described in [3] is adopted in this paper. Let Ni denote the number of channels available on link i, i= 1 or 2. Each of these channels occupies the same amount of bandwidth C_0 which is the capacity required to serve one voice call and assumed to be 1 for simplicity. Both links are assumed to operate at the same frequency range, ie. if $N_1 \rangle N_2$ ($N_1 \langle N_2$), then link 2 (link 1) channels are said to be subset of link 1 (link 2) channels in terms of frequency bandwidths. If $N_1 = N_2$, frequencies used on both links are exactly the same.

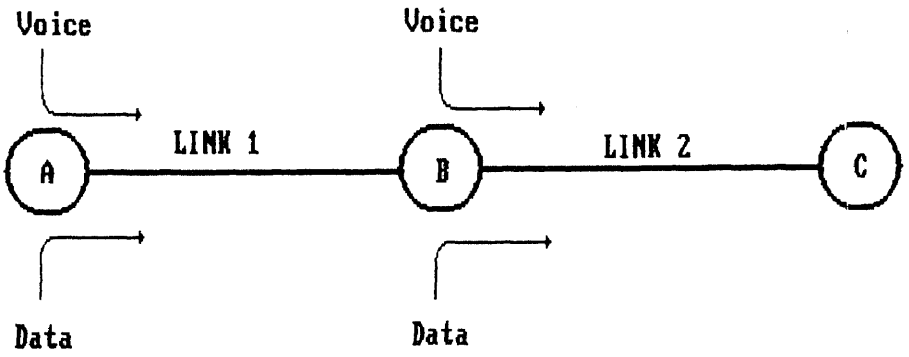


Fig. 2-2 A tandem communication networks

At each time instant, N_{c1} of the N_1 channels are utilized to N_{c1} to serve voice calls. To avoid the primary conflict in the radio network, N_{c1} channels of the same frequencies on link 2 have to be relinquished and stay unused. Amount of channels on link 1 are also sacrificed due to the voice call usage of channels on link 2. Therefore, only N_1-N_{c1}-N_{c2} and N_2-N_{c1}-N_{c2} channels are left for the data service on link 1 and link 2, respectively. Primary conflict is, again, taken into consideration when data are sharing these residual bandwidths. On each link, under the assumption in this paper, simply one data message could receive service from all the remaining channels, if there is any available, at any instance.

In the network of Fig. 2-2, both voice calls and data messages transmitted from node A are destined for node C via node B. At node B, more voice and data join the above communication bounded for C which is a plain receiver. When a voice call occurs at node A or B, a decision has to be made whether it is accepted or not. This is due to the attribute of voice that it can endure a block, but not a delay. Two factors could affect the decision: 1. if there are available channels, on both links, can accommodate this voice call. 2. if the acceptance of the call would seriously hamper the service of data. The cost function will be introduced in the next section.

The arrivals of voice call and data message at node A, B follow the Poisson process with average rate λ_{c1}, λ_{m1}, λ_{c2} and λ_{m2}, respectively. The exponentially distributed service times of data messages on link 1 and link 2 are of average rate μ_{m1} and μ_{m2}. The average holding times of voice calls on link 1 and 2 are $1/\mu_{c1}$ and $1/\mu_{c2}$, both of which also form exponential processes.

Movable boundary scheme has overload problem [4]. When datas' utilization (traffic intensity) is less than data slots N_2 ($\rho_2 \langle N_2$), the system will be in underload region and it will be stable. When $\rho_2 \rangle N_2$, the system will be in overload region and then data queue increase very fast and large. If voice call continues, occupying the slots, the system will be unstable [5].

In our model, the system avoids the overload region by blocking calls. So the system always in stable state and will not have infinite queue of data.

Dynamic Programming technique is employed in our model's control to decide whether to block a arriving call or not. We will discuss this in Chapter 3.

A tandem network with 2 links (2 hops) interconnects voice and data users via Frequency Division Multiplexing (FDM). We assume that available bandwidth C_l of link l is divided to N_l units. C_0, each channel's bandwidth, is equal to C_l divided by N_l. C_0 also is capacity required to serve a voice call.

We assume every call from link 1 must go through link 2. For the reason of avoiding interference, any call in link 1 also has to occupy another channel in link 2. Therefore, a call from link 1 to link 2 (or node 1 to node 3) could occupy 2 channels. And then a call from node 2 to node 3 will only use one channel without any interference problems.

Link 1 may serve i_1 calls, which $0 \leq 2i_1 \leq N_1$, at any time. Link 2 may serve i_1 and i_2 calls, which $0 \leq 2i_1 + i_2 \leq N_2$. The data traffic will use the remaining channels. Each node has a very large (infinite) buffer for data queue. So there is no blocking problem for data traffic.

When a call arrives, the system is either given a channel immediately or blocked, decided by control equation. When a call is accepted at node

1 or node 2, the service rate for the data traffic is instantaneously decreased by one or two units of channels.

In our model, we assume all call arrivals and departure are poisson distribution with the rate λ_{cl} and μ_{cl}. Data traffic arrivals and departure also form a possion distribution with the rate λ_{ml} and μ_{ml}.

Under these assumption, the number of messages in the system (or in the node) and the number of calls being served, at any particular node, constitute a state transition problem of this system. We can now formulate the control problem, using the Markov decision processes and Dynamic Programming to find an optimal solution for channel assignment.

CHAPTER 3 CONTROL PROBLEMS AND CONTROL EQUATION

3-1 Control problems

In this section, channel assignment problems, for either voice call or data message, are studied in detail. When a message arrives at a node and finds that there is no channel available, it will join the queue and wait for the service based on the first-come-first-serve policy. This unavailability could be due to the fact that all the channels are occupied by the voice calls or that one data message is receiving service on the link. The queue size is assumed infinite, therefore no data blocking problem will be considered. At the time the data locate channels for its service, a decision that how many it will grasp for its own use has to be made. This decision should lead to the best benefits of the whole system.

When a voice call attempts a transmission from A to C, it will immediately be granted two channels on link 1 and two channels on link 2 (one channel for transmission, the other one for avoiding primary conflict), if it is accepted at A. If the call is from B to C, one channel on each link is given for its use.

On the occasion that a voice call finishes its service, it releases the channels it occupied and donates them to the data message at service. Once a data message completes its transmission, the first one waiting in the data queue takes over all the channels left behind immediately.

The voice call is said to have higher priority than the data message in the sense that it can preempt channels from the data message at service. But it has to be noted that the voice call would probably be blocked at the entrance of a node because too many messages are waiting desperately for the service in data queue.

3-2 Markovian Decision

In this tandem network model, there exist two nodes and three links. Under the assumption that the global information of voice call and data message is known to each node, let $s_t=(i_1,i_2,j_1,j_2)$ be the state of the system at time $t\in [0,\infty)$, where

i_l: number of calls being served at link l. $l=1,2$

j_l: number of message being served at link l. $l=1,2$

and $s_t \in S = \{0,1,...,N1\}x\{0,1,...,N2\}x\{0,1,2,...\}$

$x\{0,1,2,...\}$. Here S is the state space of the system.

The following operators which map s_t into S are adopted in this paper:

$A_{c1}(i_1,i_2,j_1,j_2)=(i_1+1,i_2,j_1,j_2)$
$A_{c2}(i_1,i_2,j_1,j_2)=(i_1,i_2+1,j_1,j_2)$
$A_{m1}(i_1,i_2,j_1,j_2)=(i_1,i_2,j_1+1,j_2)$

$$
\begin{aligned}
&A_{m2}(i_1,i_2,j_1,j_2)=(i_1,i_2,j_1,j_2+1)\\
&D_{c1}(i_1,i_2,j_1,j_2)=(i_1-1,i_2,j_1,j_2)\\
&D_{c2}(i_1,i_2,j_1,j_2)=(i_1,i_2-1,j_1,j_2)\\
&D_{m1}(i_1,i_2,j_1,j_2)=(i_1,i_2,j_1-1,j_2+1)\\
&D_{m2}(i_1,i_2,j_1,j_2)=(i_1,i_2,j_1,j_2-1)
\end{aligned}
\qquad (3\text{-}1)
$$

Here A and D represent an arrival and a departure, respectively, and subscripts ci, mi stand for the voice call and the data message on node i, i=1 or 2.

Under statistical assumptions adopted in Chapter 2, s_t is a four dimensional continuous time Markov process. The transition rates out of a nonboundary state are shown in Fig. 3-1.

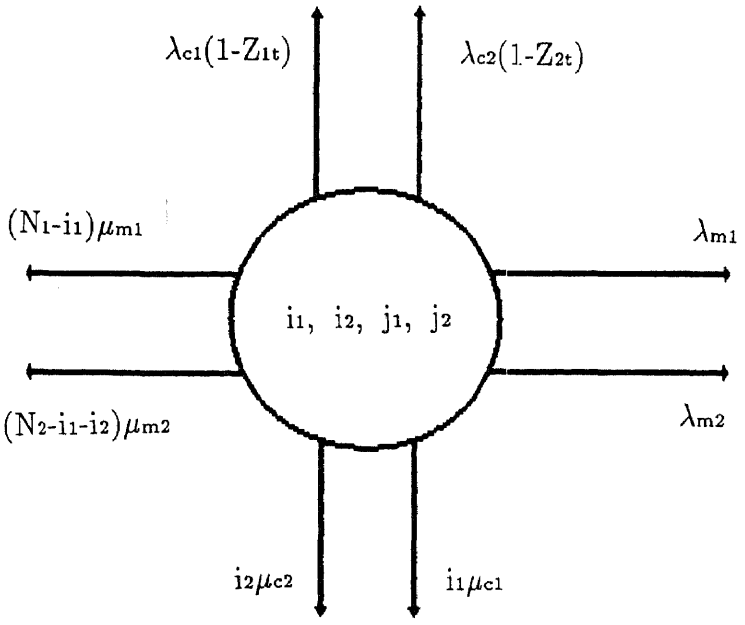


Fig. 3-1 Transition rates

Let Zi, in the range of [0,1], denote the probability that a call which arrives at node i is blocked and the "total event rate" r be defined by

$$r=\lambda_{c1}+\lambda_{c2}+\lambda_{m1}+\lambda_{m2}+N_1(\mu_{m1}+\mu_{c1})+N_2(\mu_{m2}+\mu_{c2}) \qquad (3\text{-}2)$$

Also, let K_1 and K_2 represent the number of channels assigned to the data in link 1 and link 2, respectively. Note that if N_1, the total number of channels in link 1, is larger than N_2, then $N_1=i_1+i_2+K_1+K_2$. For each control value $Z=(Z_1, Z_2)$, the transition probability function can be written as

$$\begin{aligned} p(y|x,z)\cdot r \;=\; & (1-z^1)\lambda_{c1}I(y=A_{c1}x) \\ & +z^1\lambda_{c1}I(y=x) \\ & +(1-z^2)\lambda_{c2}I(y=A_{c2}x) \\ & +\lambda_{m1}I(y=A_{m1}x) \\ & +\lambda_{m2}I(y=A_{m2}x) \\ & +z^2\lambda_{c2}I(y=x) \\ & +i_1\mu_{c1}I(y=D_{c1}x) \\ & +i_2\mu_{c2}I(y=D_{c2}x) \\ & +(N_2-2i_1-i_2-k_1)\mu_{m2}I(y=D_{m2}x) \\ & +k_1\mu_{m1}I(y=D_{m1}x) \\ & +\{(N_1-i_1)\mu_{c1}+(N_2-i_2)\mu_{c2} \\ & +(2i_1+i_2+k_1)\mu_{m2}+(N_1-k_1)\mu_{m1}\}I(y=x) \end{aligned} \qquad (3\text{-}3)$$

Where I(A) is the indicator function of the event A and y is the next state of x. Our goal is to minimize the time delay of data messages and blocking probability of voice calls.

3-3 Control Equation

Let (i_1, i_2, j_1, j_2) be the system state at time t, and (i_1, i_2, j_1, j_2) be the state at t=0. The instantaneous cost is given by $j_{1t}+j_{2t}+\alpha_1 Z_{1t}+\alpha_2 Z_{2t}$, where Z_{lt} is the probability of blocking a call that originates at node l at time t and α_l is the corresponding weighting factor, $0 \leq \alpha_l \leq \infty$. Let δ be a discount factor. Let τ_n denote the (random) time at which the state s_t jumps for the n-th time. Then, the cost for a control policy f over the random time interval $[0,\tau_n)$ is

$$E_f \int_0^{\tau n} e^{-\delta t}(j_{1t} + j_{2t} + \alpha_1 z_{1t} + \alpha_2 z_{2t})dt \qquad (3\text{-}4)$$

From [6], this cost is found identical to

$$E_f \sum_{k=0}^{n-1} \beta^k (j_{1k} + j_{2k}\alpha_1 z_{1k} + \alpha_2 z_{2k}) \qquad (3\text{-}5)$$

where expectation is taken with respect to the process generated by f when starting at (i_1, i_2, j_1, j_2) initially.

The cost given in (3-5) can be considered as the cost derived over n time steps in a discrete time decision process with discount factor β, the same state space S and action space A=[0,1]*[0,1]. The transition probability $p=p(x_{k+1}|x_k,z_k)$ are given below

$$\begin{array}{lll} p(x_{k+1}|x_k,z_k) = \lambda_{cl}/r & & x_{k+1} = A_{cl}x_k, z_{lk} = 0 \\ \lambda_{cl}/r & & x_{k+1} = x_k, z_{lk} = 1 \\ \lambda_{ml}/r & & x_{k+1} = A_{ml}x_k \\ i_l\mu_{cl}/r & & x_{k+1} = D_{cl}x_k \end{array}$$

$$
\begin{array}{ll}
(N_l - \sum_{n=1}^{l} i_n)\mu_{ml}/r & x_{k+1} = D_{ml}x_k \\
\Delta/r & x_{k+1} = x_k
\end{array}
\qquad (3\text{-}6)
$$

where xk=(i_1, i_2, j_1, j_2), l=1,2 and $\Delta=(N_1\text{-}i_1)\mu_{c1}+(N_2\text{-}i_2)\mu_{c2}+(2i_1+i_2+k_1)\mu_{m2}+(N_1\text{-}k_1)\mu_{m1}$. For convenience, we may assume that r=1 and ignore the constant factor; thus the expression (3-5) can be considered as the cost incurred by policy f.

With the initial state (i_1, i_2, j_1, j_2), let Jn(i_1, i_2, j_1, j_2) denote the cost incurred at the n-th step by optimal policy. Then Jn(i_1, i_2, j_1, j_2) is characterized by following Dynamic Programming Equation.

Dynamic Programming Equation:

$$
\begin{aligned}
J_{n+1}(i_1,i_2,j_1,j_2) = {} & j_1 + j_2 \\
& + min_{z1\in[0,1]}\{\alpha z1 + \beta\lambda_{c1}z1J_n(i_1,i_2,j_1,j_2) + \\
& (1-z1)\beta\lambda_{c1}J_n(i_1+1,i_2,j_1,j_2)\} \\
& + min_{z2\in[0,1]}\{\alpha z2 + \beta\lambda_{c2}z2J_n(i_1,i_2,j_1,j_2) + \\
& (1-z2)\beta\lambda_{c2}J_n(i_1,i_2+1,j_1,j_2)\} \\
& + \beta\lambda_{m1}J_n(i_1,i_2,j_1+1,j_2) + \beta\lambda_{m2}J_n(i_1,i_2,j_1,j_2+1) \\
& + \beta i_1\mu_{c1}J_n(i_1-1,i_2,j_1,j_2) + \beta i_2\mu_{c2}J_n(i_1,i_2-1,j_1,j_2) \\
& + min_{k1\in[0,N1-2i1-i2]}\{\beta k_1\mu_{m1}J_n(i_1,i_2,j_1-1,j_2+1) + \\
& \beta(N_2-2i_1-i_2-k_1)\mu_{m2}J_n(i_1,i_2,j_1,j_2-1) + \\
& \beta[(N_1-i_1)\mu_{c1} + (N_2-i_2)\mu_{c2} + \\
& (2i_1+i_2+k_1)\mu_{m2} + (N_1-k_1)\mu_{m1}]J_n(i_1,i_2,j_1,j_2)\}
\end{aligned}
$$

(3-7)

$$
\begin{aligned}
Z1 &= 1(block) \\
&\quad if \\
&\quad \alpha_1 - \beta\lambda_{c1}[J_k(i_1+1, i_2, j_1, j_2) - J_k(i_1, i_2, j_1, j_2)] < 0
\end{aligned}
$$

$$
\begin{aligned}
Z1 &= 0(accept) \\
&\quad if \\
&\quad \alpha_1 - \beta\lambda_{c1}[J_k(i_1+1, i_2, j_1, j_2) - J_k(i_1, i_2, j_1, j_2)] >= 0
\end{aligned} \tag{3-8}
$$

$$
\begin{aligned}
k_1 &= N_1 - 2i_1 - i_2 \\
&\quad if \\
&\quad \{\mu_{m1} J_n(i_1, i_2, j_1 - 1, j_2 + 1) - \mu_{m2} J_n(i_1, i_2, j_1, j_2 - 1) \\
&\quad +(\mu_{m2} - \mu_{m1}) J_n(i_1, i_2, j_1, j_2)\} < 0
\end{aligned}
$$

$$
\begin{aligned}
k_1 &= 0 \\
&\quad if \\
&\quad \{\mu_{m1} J_n(i_1, i_2, j_1 - 1, j_2 + 1) - \mu_{m2} J_n(i_1, i_2, j_1, j_2 - 1) \\
&\quad +(\mu_{m2} - \mu_{m1}) J_n(i_1, i_2, j_1, j_2)\} >= 0
\end{aligned} \tag{3-9}
$$

From the above equation, two decisions can be made:

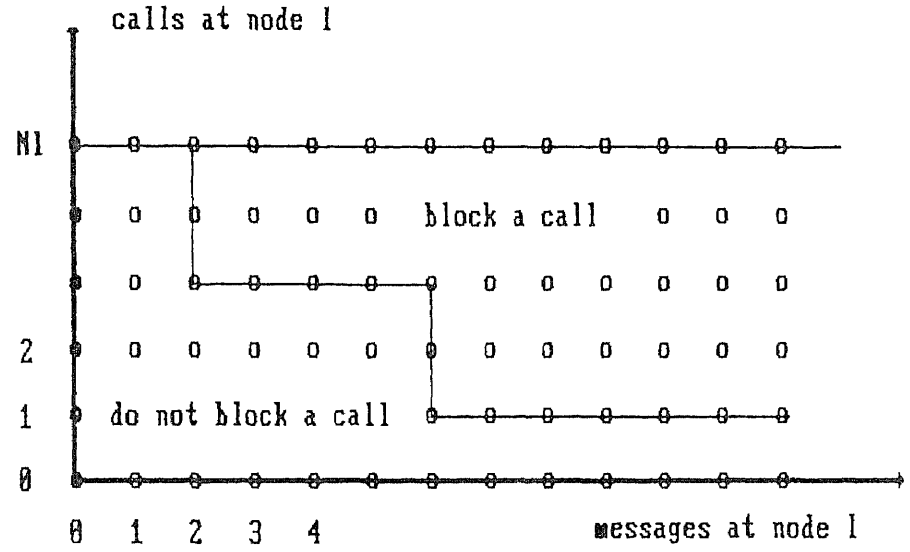
1) the acceptance of an arriving voice call with a similar expression for z_2.

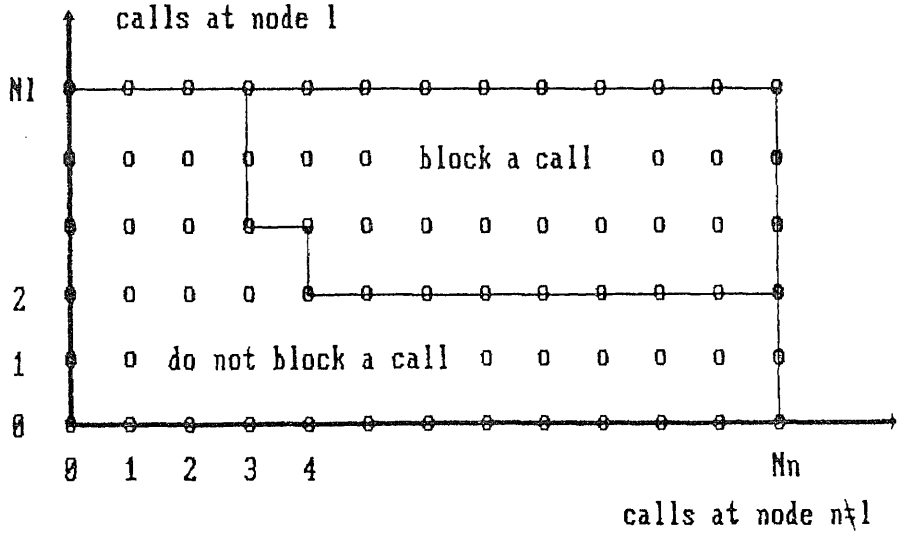
2) the number of channels assigned to the data message

In contrast to 2), an alternate assignment which looks more instinctive is also brought to the simulation. The results are shown in the next section.

A typical optional switching curve shown in Fig. 3-2. Dynamic Programming Equation partitions transition state into two regions by switching curve. Over the curve, Z_1 value=1, the Control Equation will

block the arriving call. Under the curve, $Z_1=0$, the Control Equation will accept the arriving call.





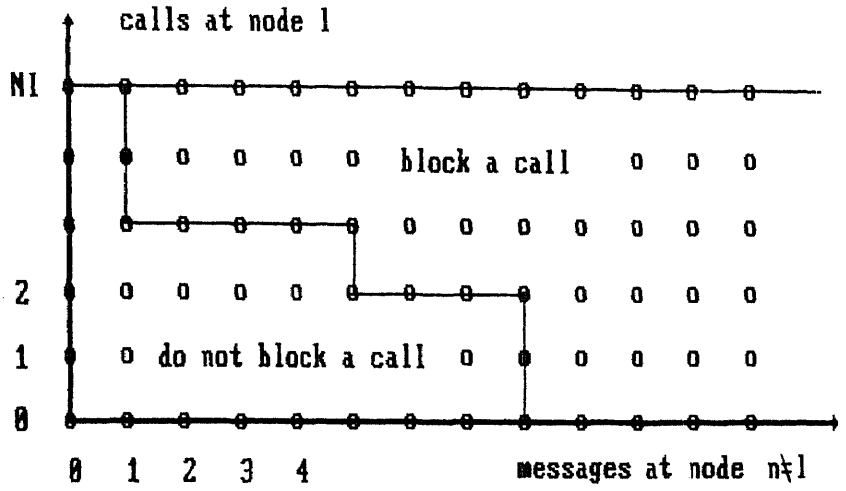


Fig. 3-2 General form of the optional switching curve

CHAPTER 4 SIMULATION RESULTS

4-1 The Alternate Data Channel Assignment Policy

Besides the decision of data channel assignment obtained from Equation (3-8) and (3-9), an alternate policy is adopted in our simulation. In this policy, the channel assignment depends on the ratio of J_1 and J_2 which are the numbers of data waiting for the service at node A and node B, respectively. If the queue length at node A is greater than that at node B ($J_1 \rangle J_2$), then more channels in link 1 are apportioned for the service of data at node A. By assuming that $N_1 \langle N_2$, the equation can be written explicitly as

$$\frac{K_1}{N_2-2i_1-i_2} = \frac{J_1}{J_1+J_2}$$

Here $N_2-2i_1-i_2$ is the number of channels available for the data service in link 2 and K_1 is the number of channels allocated to the data at node A. Then $N_2-2i_1-i_2-K_1$ will be the capacity assigned for the service at node B.

For comparison, the movable boundary scheme is also introduced in our simulation. Four channels in each link are allocated for voice call service, i.e. if less than four voice calls are receiving service, the new voice call attempt is accepted definitely. No decision-making for voice call is needed in this scheme because its number can never exceed four. The data message could steal the channel allocated to voice if it is not occupied. The alternate assignment policy for data is used in this scheme. Under the assumption that the four channels dedicated to the voice call starting at node B have no duplicate frequencies in link 2 (because N_1(N_2), in the worst case (four voice calls are being served in each link), there is still one available channel for data service in link 2.

Our simulation model is portrayed as following: link 1 and link 2 possess channel capacities of 9 and 14, respectively (i.e. N_1=9 and N_2=14). At node A, λ_{c1}, the arrival rate of voice call, is 0.03; μ_{c1}, the service rate of voice call, is 0.008. At node B, λ_{c2}=0.035, μ_{c2}=0.008.

Decision tables are set up in node A and B according to the system parameters given above. The simulation lasts for 5000 seconds and the results, in terms of blocking probabilities and time delay, are shown below.

4-2 Simulation Results

Blocking Probability

In the decision-making scheme, the acceptance of voice calls depends on the number of data messages j_1 and j_2. When j_1 and j_2 (queue length) are large, voice call tends to be blocked. It is shown in Fig. 4-1 and 4-2 that when ρ_1 and ρ_2, defined as the ratio of arrival rate over departure rate, is increasing, the blocking probabilities in link 1 and link 2 are also increasing. When $\rho_1 \geq 6$ or $\rho_2 \geq 6$, the system blocks all the arriving calls to provide all channels for message transmission.

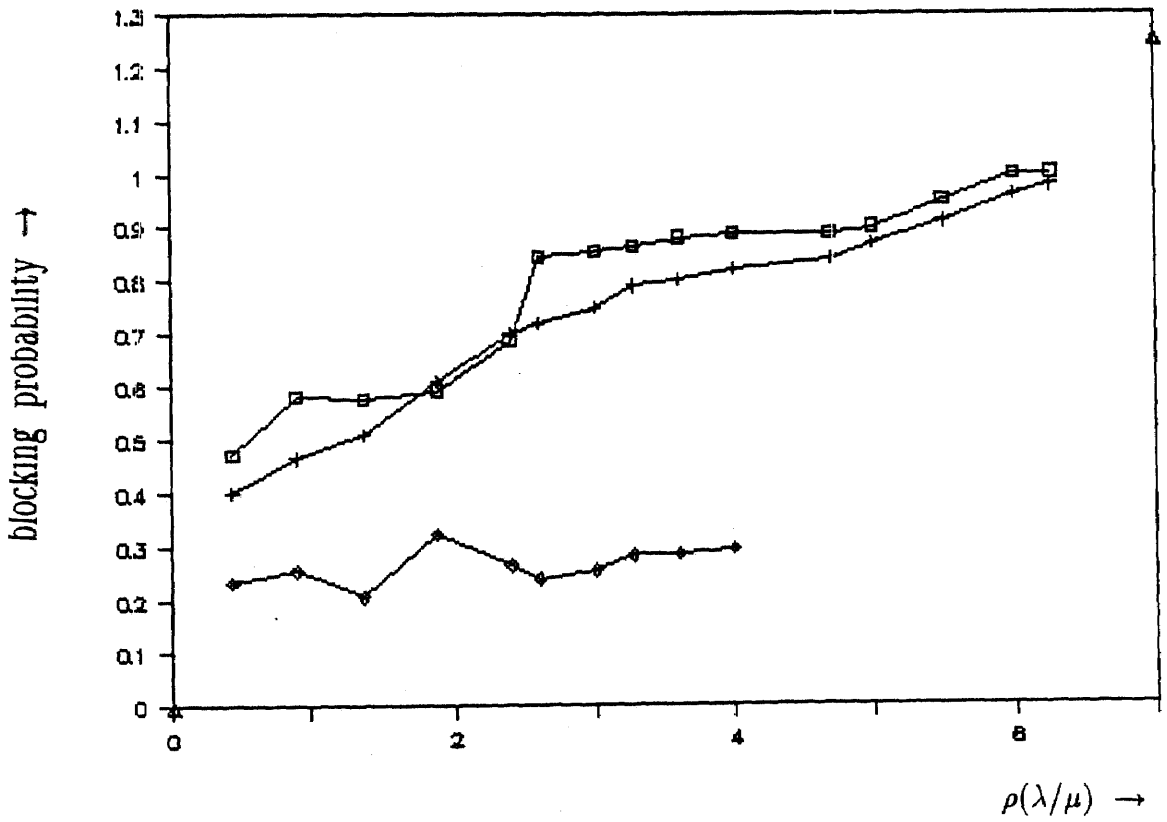


Fig. 4-1 Link 1 blocking probability

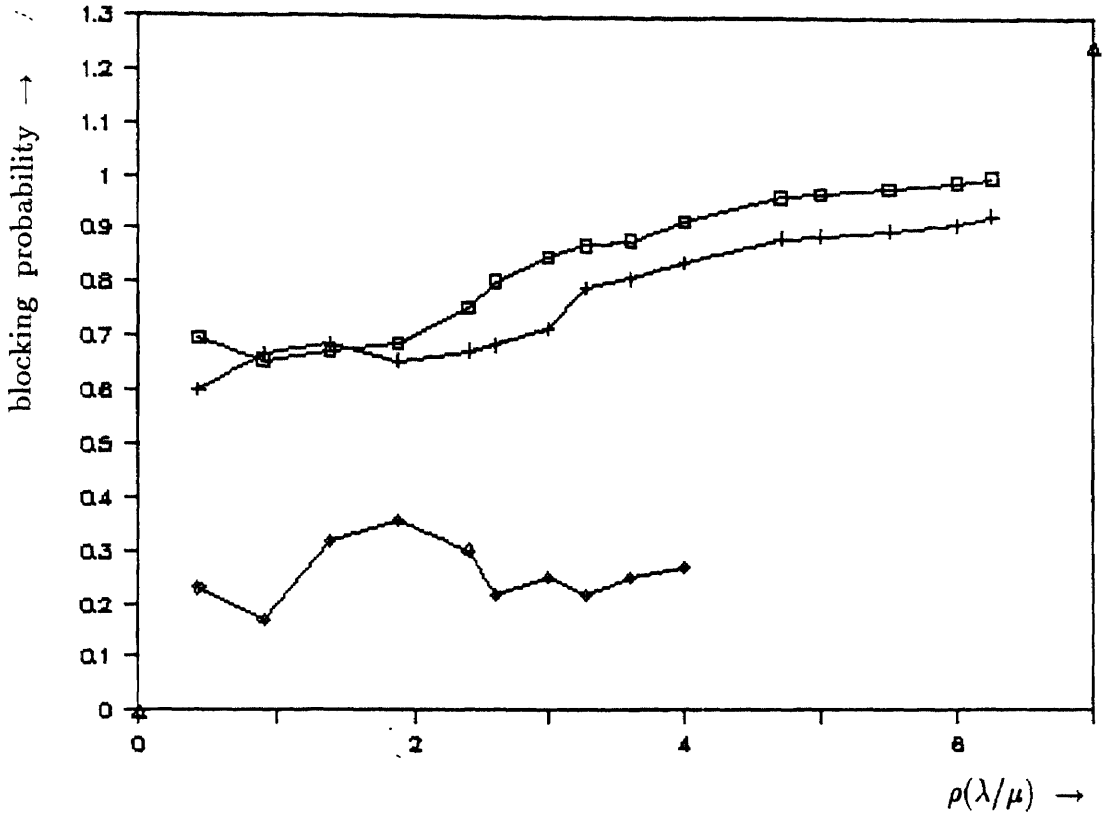


Fig. 4-2 Link 2 blocking probability

In the movable boundary scheme, voice calls can preempt data packets occupying their allocated channels. Thus, no matter what the value of ρ is and what the queue length is, the values of the blocking probability always keep at the level which is much lower than those in the decision-making scheme, especially when ρ is large.

Data Time Delay

In the decision-making scheme, it is possible that no voice call could access the transmission facility if too many data messages are expecting services. But data messages are always preempted in the movable boundary scheme, even if the condition is critical. As expected, the time delay in

the movable boundary scheme increases abruptly when the utilization rate reaches certain value. In link 1, as depicted in Fig. 4-3, this value stands at 1.5. In link 2, as illustrated in Fig. 4-4, 3.5 is the threshold. When ρ =4.5 or greater, the queue size will be greater than 50,000.

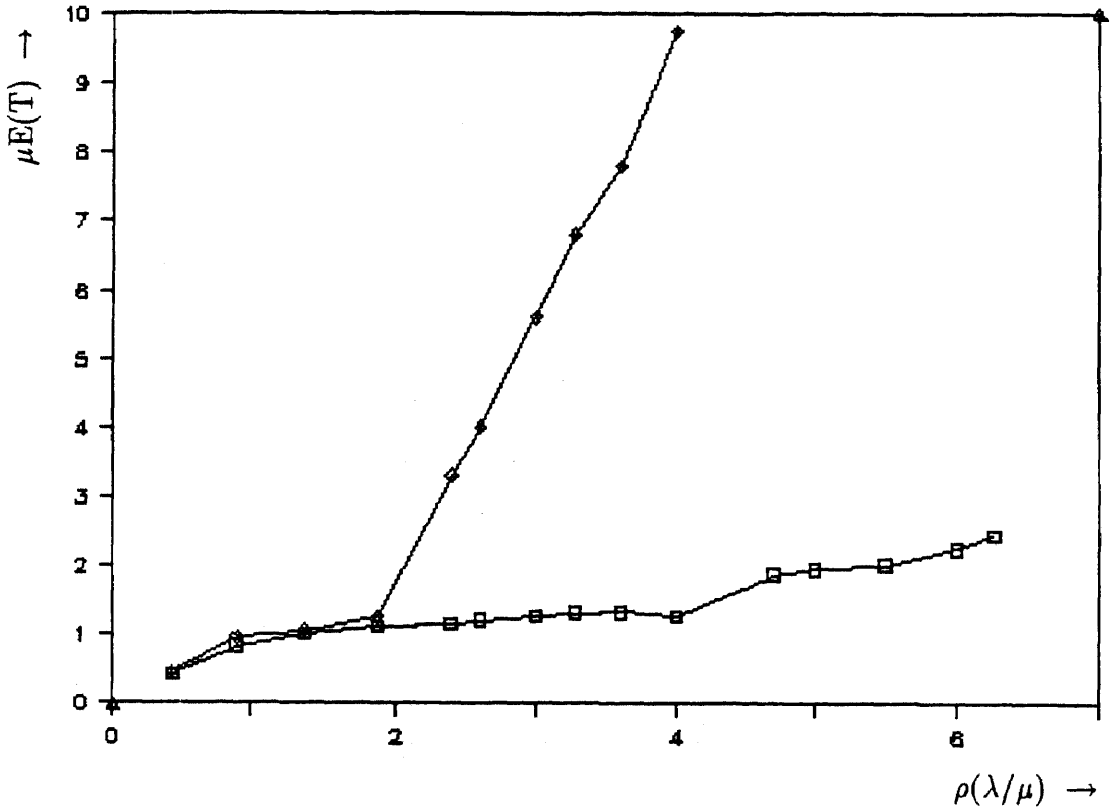


Fig. 4-3 Link 1 normalized time delay
□ decision-making scheme and
◇ movable boundary scheme

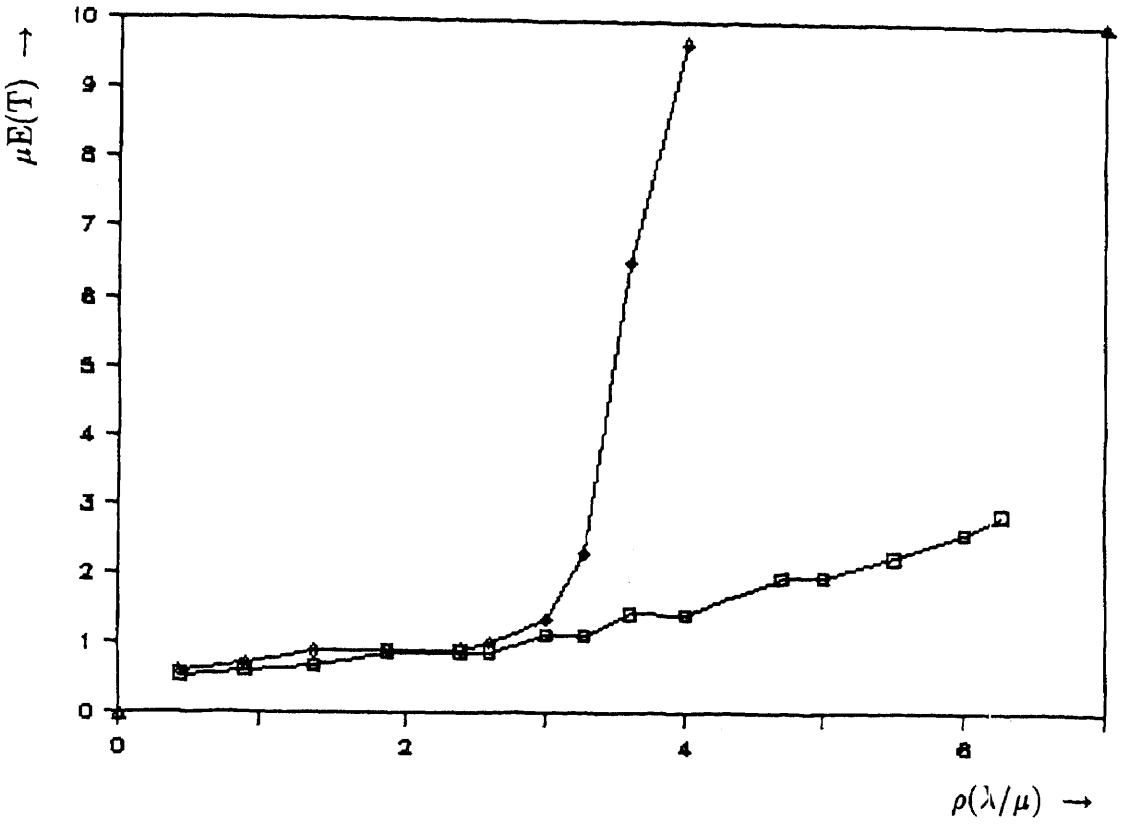


Fig. 4-4 Link 2 normalized time delay
□ decision-making scheme and
◇ movable boundary scheme

It is shown, in Fig. 4-5 and 4-6, that data time delay will not be affected much whether the decision is made from the dynamic programming or from the alternate policy. But in link 2 the alternate policy still yields better performance than the dynamic programming.

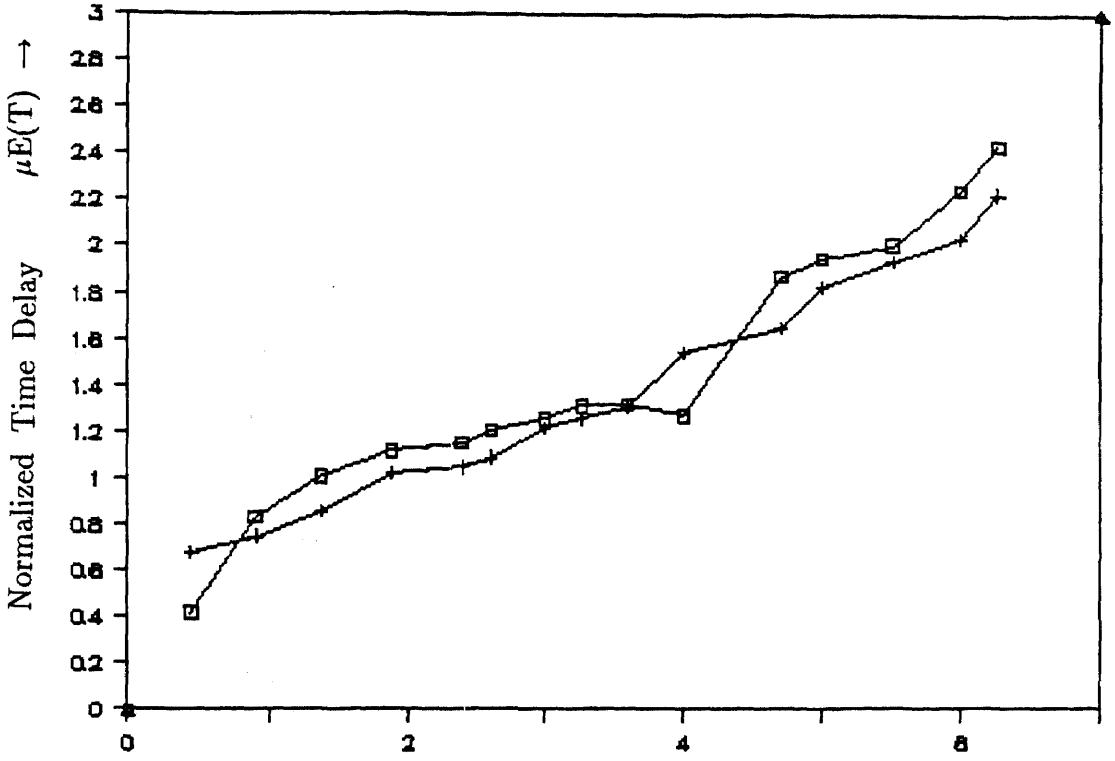


Fig. 4-5 Link 1 normalized time delay dynamic programming equation scheme alternate policy (ratio) scheme

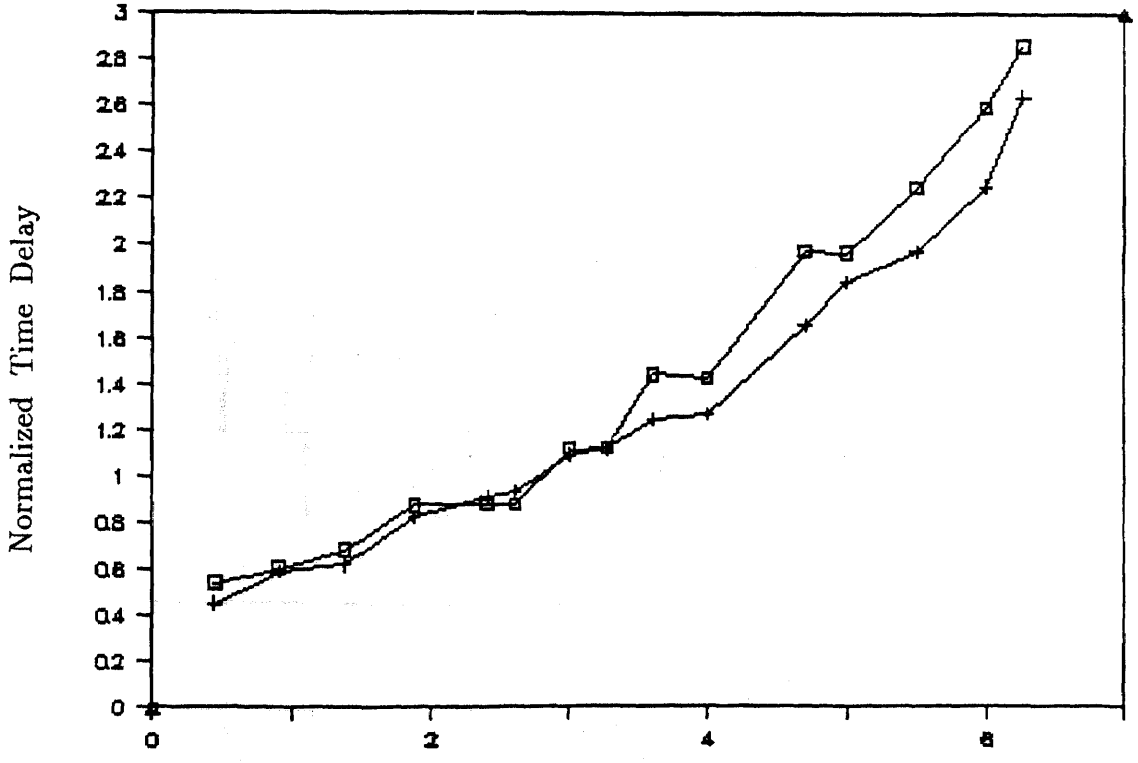


Fig. 4-6 Link 2 normalized time delay dynamic programming equation scheme alternate policy (ratio) scheme

Fig. 4-7 illustrates a real-time plot of voice calls and the corresponding data queue length, as a function of time, for the movable boundary scheme with ρ=3.5. During the intervals of time in which two or fewer voice calls are present, the data queue is underloaded. When there are more than two calls present in either link 1 or link 2, the data queue is overloaded and the length increases steeply to several hundreds of packets, as shown in Fig. 4-7(a), (b) (link1). Since N_2 is greater than N_1, data in link 2 have more channels available than those in link 1. Thus the queue length is always shorter in link 2.

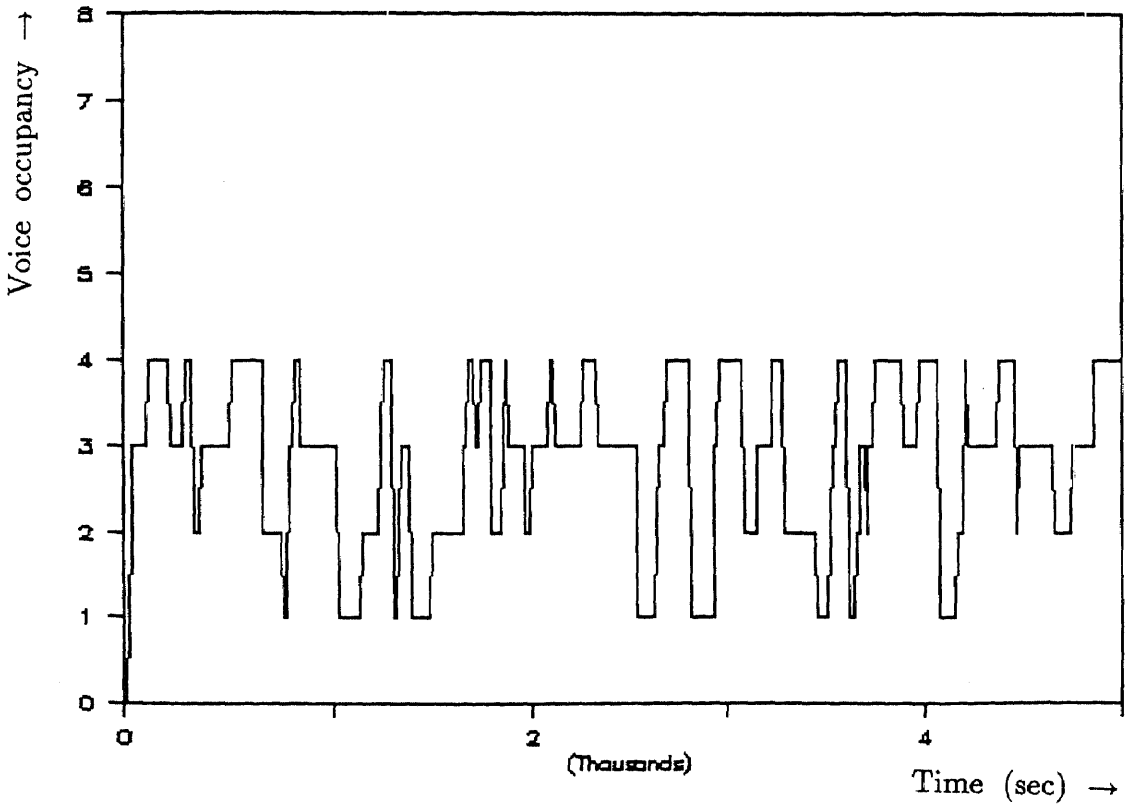


Fig. 4-7(a) Link 1 voice calls vs time

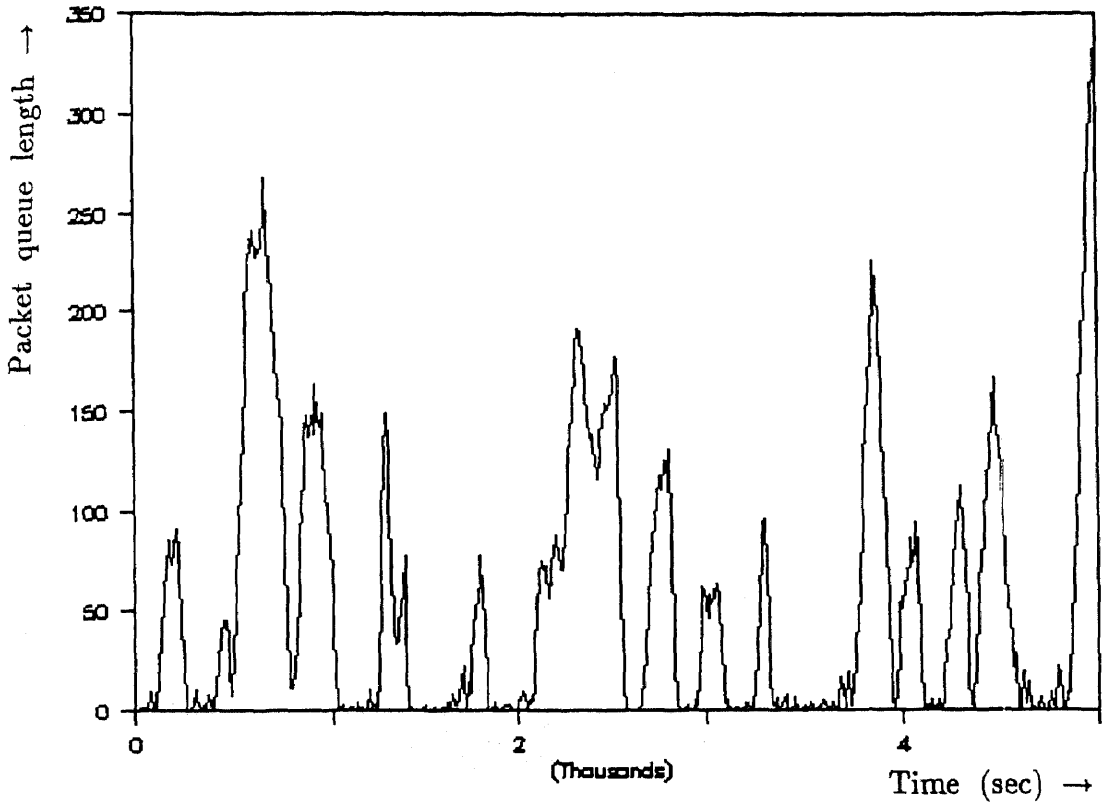


Fig. 4-7(b) Link 1 data queue length vs time

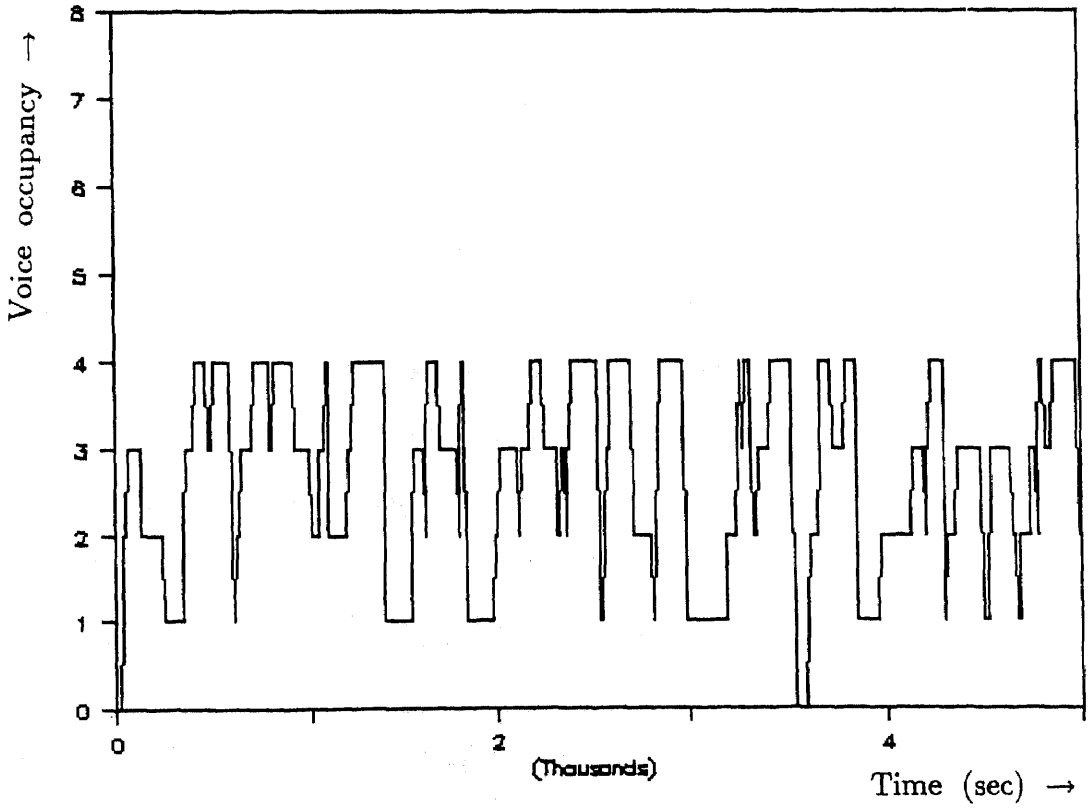


Fig. 4-7(c) Link 2 voice calls vs time

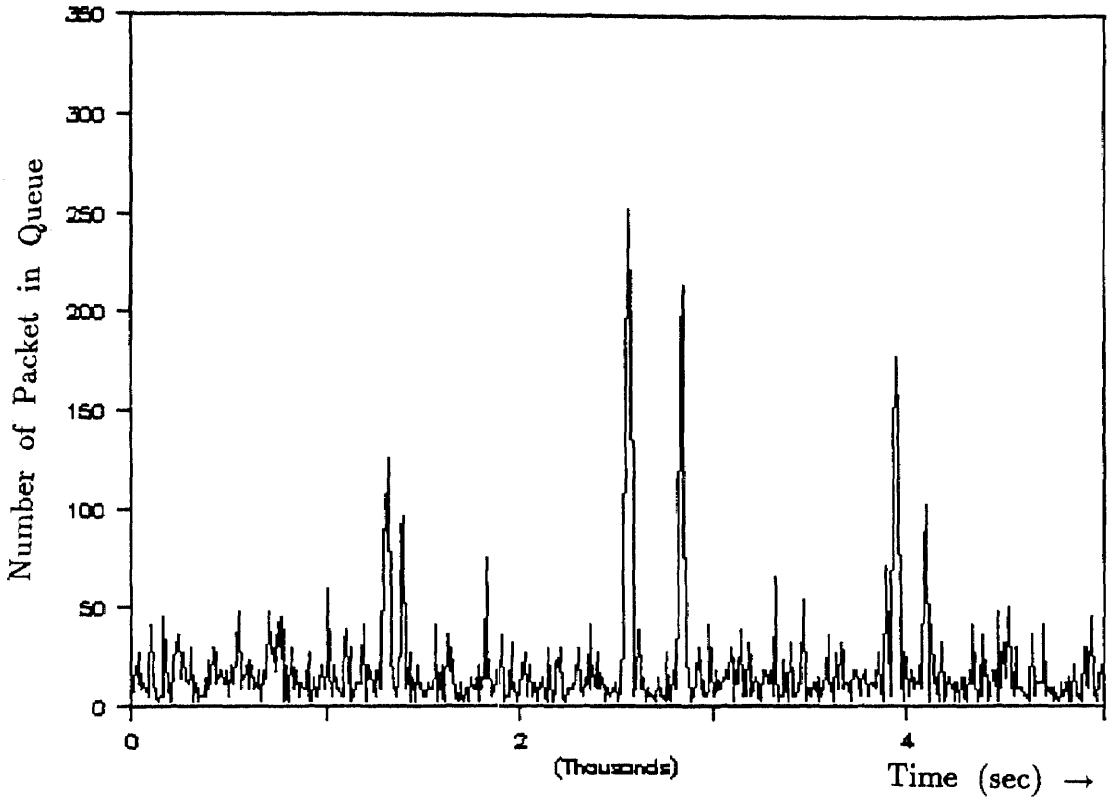


Fig. 4-7(d) Link 2 data queue length vs time

Fig. 4-8(a), (b), (c), and (d) show a real time plot of voice calls and the corresponding data queue length, as function of time, for decision-making scheme with ρ=3.5. Both of the queue length and time delay are much smaller than those in the movable boundary scheme and no overload situation ever occurred. As shown in Fig. 4-2, the attempt of minimizing time delay and queue length will cause the increase of blocking probability. Also from Fig. 4-8(a) and (c), intervals of time in which no call receives service can be found.

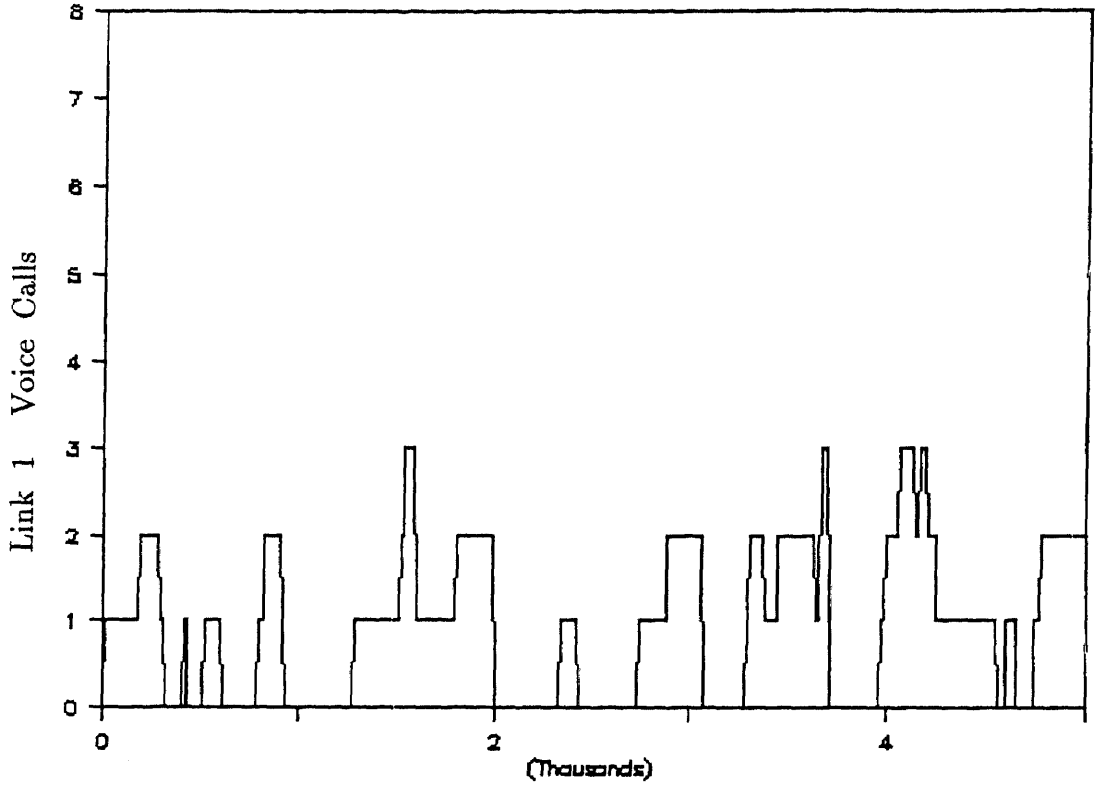


Fig. 4-8(a) Link 1 voice calls vs time

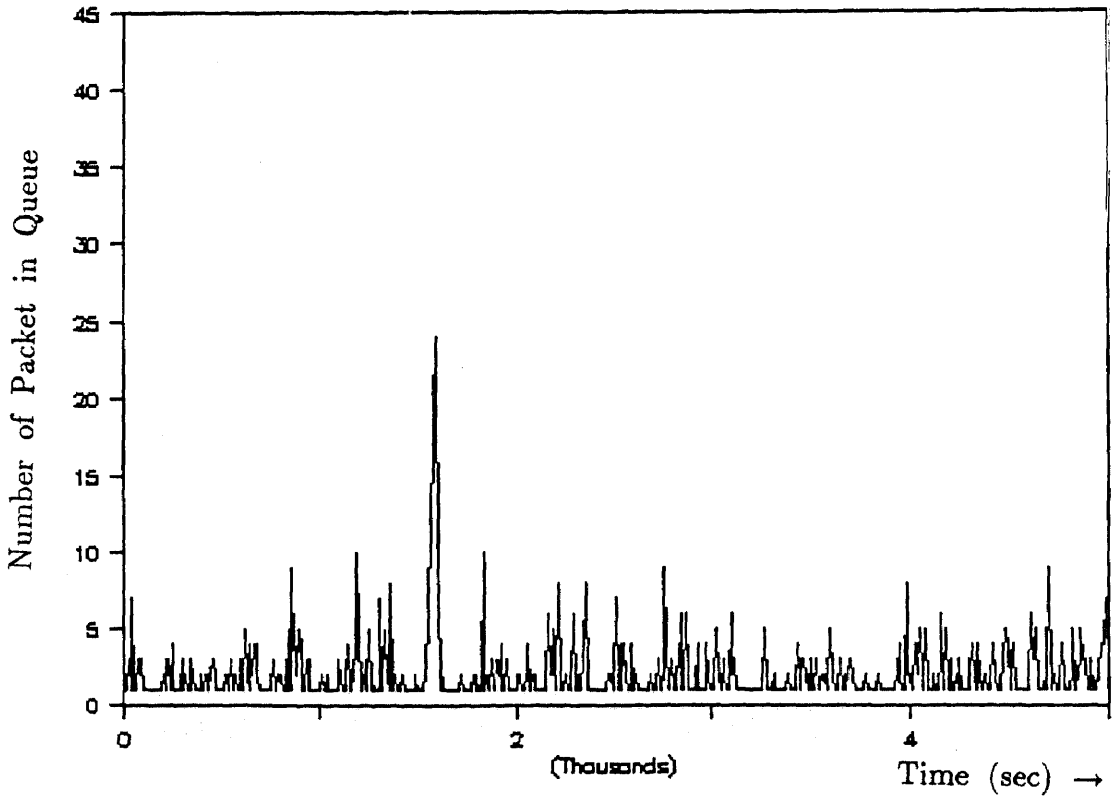


Fig. 4-8(b) Link 1 data queue length vs time

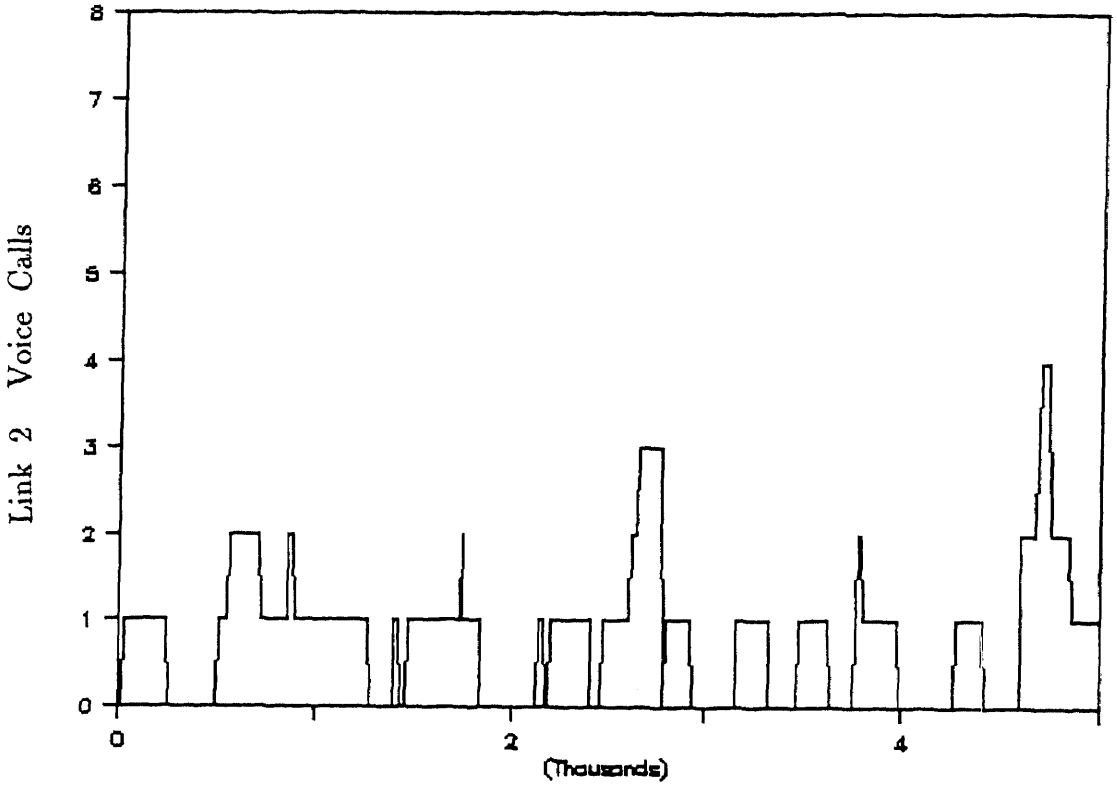


Fig. 4-8(c) Link 2 voice calls vs time

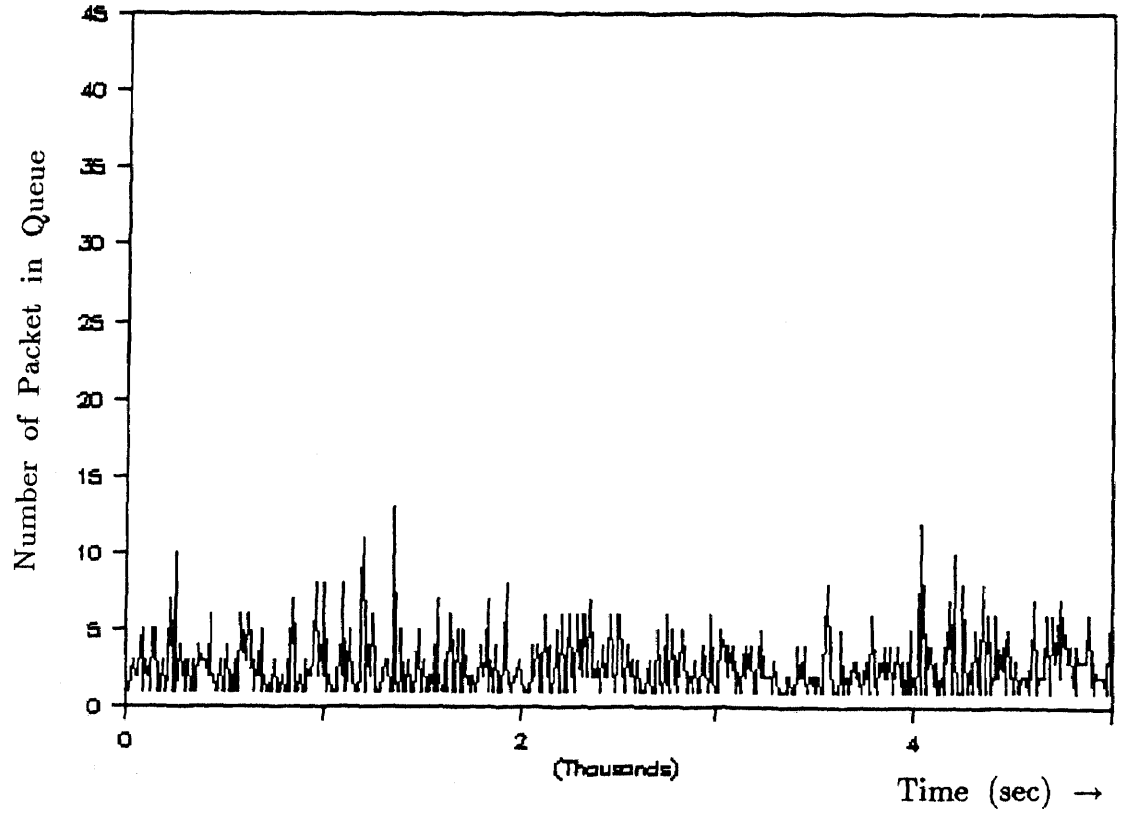


Fig. 4-8(d) Link 2 data queue length vs time

CHAPTER 5 CONCLUSION

We have studied the dynamic access control problem in simple ISDN networks. We also have adopted three different ways: the jumping mode, the ratio mode and the movable boundary mode, to simulate in the same tandem network, which enables us to compare their performance.

In the jumping mode or the ratio mode, we adopted cost function (3-7) to be a controller to decide whether to accept calls or not. We always have lower time delay, but higher blocking probability than that of the movable boundary mode at the same time.

The movable boundary mode, a totally different method to manage available channels, can have a lower blocking probability for voice calls, but has large time delay for data users. And it is also possible to let the system go into overload (unstable) region.

According to those simulation results in Chapter 4, we can say that it is a kind of "trade off." If we want to keep lower time delay and keep the system stable, we have to lose call's successful rate (high blocking probability). If we want to keep low blocking probability we may get more time delay.

REFERENCES

[1] G. Coviello and P. Vena, "Intergration of Circuit/ Packet-Switching by a SENET (Slotted Envelop Network)Concept," National Telecommunications Conference, New Orleans, Dec. 1975.

[2] M. Ross et al., "Design Approaches and Performance Criteria for Integrated Voice/Data Switching," Proc. IEEE, Sep. 1977.

[3] E. Port et al., "A Network Architecture for the Integrationof Circuit and Packet Switching," ICCC, Toronto, Aug. 1976.

[4] H. Rudin, "Studies in the Integration of the Circuit and Packet Switching," ICC, Toronto, June 1978.

[5] A. Leon-Garcia, R. H. Kwong, G. F. Williams, "Performance Evaluation Methods for an Integrated Voice-Data Link," IEEE Trans. on Comm., Aug. 1982.

[6] B. Hajek, "Optimal control of two interacting service stationa," IEEE Trans. on Automatic Control, June 1984.

[7] S. A. Lippman, "Semi-Markov decision processes with unbounded rewards," Management Science, vol. 19 1973.

APPENDIX

```
*************************************************************************
*                                                                       *
*Program One                                                            *
*Dynamic Programing Equation                                            *
*                                                                       *
*************************************************************************


#include (stdio.h)
#include (math.h)
float BETA=0.09, LAMDAc1=0.05, MUc1=0.01, ARFA1=28.5 ;
float J[5][10][10][10][10];
float   ARFA2=36.8, LAMDAc2=0.04, MUc2=0.01;
float   LAMDAm1=12.0, LAMDAm2=12.5;
float   MUm1=10.0, MUm2=12.5, DELTA=0.0, N1=9.0, N2=13.0;
int Z1(), Z2(), OP();

main()
{
int i1, i2, j1, j2, n, k1;
float AB=0.0, BA=0.0;
float   A=0.0, B=0.0, C=0.0, D=0.0, E=0.0, F=0.0, G=0.0, H=0.0, I=0.0;


printf("Please input BETA, ARFA1 and ARFA2.\n");
scanf("%f %f %f", &BETA, &ARFA1, &ARFA2);
for (i1=0; i1(=9; i1++)
      {
      for (i2=0; i2(=9; i2++)
            {
            for (j1=0; j1(=9; j1++)
                  {
                  for (j2=0; j2(=9; j2++)
                        {
                              J[0][i1][i2][j1][j2]=j1+j2;
                        }
                  }
            }
      }
```

```
for (n=0; n<=3; n++)
    {
    for (i1=0; i1<=6; i1++)
        {
        for (i2=0; i2<=6; i2++)
            {
            for (j1=0; j1<=8; j1++)
                {
                for (j2=0; j2<=8; j2++)
                    {

    AB=(1-Z1(n,i1,i2,j1,j2))*BETA*LAMDAc1*J[n][i1+1][i2][j1][j2];

    BA=(1-Z2(n,i1,i2,j1,j2))*BETA*LAMDAc2*J[n][i1][i2+1][j1][j2];

    k1=OP(n,i1,i2,j1,j2);

    DELTA=(N1-i1)*MUc1+(N2-i2)*MUc2+(2*i1+i2+k1)*MUm2+(N1-
k1)*MUm1;

    A=ARFA1*Z1(n,i1,i2,j1,j2)+BETA*LAMDAc1*Z1(n,i1,i2,j1,j2)*J[n][i1][i2][j
1][j2]+AB;

    B=ARFA2*Z2(n,i1,i2,j1,j2)+BETA*LAMDAc2*Z2(n,i1,i2,j1,j2)*J[n][i1][i2][j
1][j2]+BA;

    C=BETA*LAMDAm1*J[n][i1][i2][j1+1][j2];

    D=BETA*LAMDAm2*J[n][i1][i2][j1][j2+1];
    if (i1<=0)
        E=0.0;
    else
        E=BETA*i1*MUc1*J[n][i1-1][i2][j1][j2];
    if (i2<=0)
        F=0.0;
    else
        F=BETA*i2*MUc2*J[n][i1][i2-1][j1][j2];
    if (j1<=0)
        G=BETA*k1*MUm1*j1;
    else
        G=BETA*k1*MUm1*J[n][i1][i2][j1-1][j2+1];
    if (j2<=0)
        H=BETA*(N2-i2-i1-k1)*MUm2*j2;
    else
        H=BETA*(N2-i2-i1-k1)*MUm2*J[n][i1][i2][j1][j2-1];

    I=BETA*DELTA*J[n][i1][i2][j1][j2];

    J[n+1][i1][i2][j1][j2]=j1+j2+A+B+C+D+E+F+G+H+I;
                    }
                }
            }
        }
    }
```

```
printf("%d   ", Z1(n,0,0,0,0));
}
printf("j(0 10000=%3.2f   j(0 2000)=%3.2f \n", J[0][1][0][0][0], J[0][2][0][0][0]);
printf("j(1 10000=%3.2f   j(1 2000)=%3.2f \n", J[1][1][0][0][0], J[1][2][0][0][0]);
printf("j(2 10000=%3.2f   j(2 2000)=%3.2f \n", J[2][1][0][0][0], J[2][2][0][0][0]);
printf("j(3 10000=%3.2f   j(3 2000)=%3.2f \n\n\n", J[3][1][0][0][0],
J[3][2][0][0][0]);
for (i1=5; i1)=1; i1--)
        {
        for (j1=0; j1(=6; j1++)
                {
                printf("Z1=%d     ", Z1(1,i1,0,j1,0));
                }
        putchar('\n');
}
putchar('\n');
putchar('\n');
for (i1=5; i1)=1; i1--)
        {
        for (j1=0; j1(=6; j1++)
                {
                printf("Z1=%d     ", Z1(2,i1,0,j1,0));
                }
        putchar('\n');
        }
putchar('\n');
putchar('\n');

}

Z1(x,a,b,c,d)
int x,a,b,c,d;
{
int q=0;
if ((ARFA1-BETA*LAMDAc1*(J[x][a+1][b][c][d]-J[x][a][b][c][d]))(0)
    q=1;
else
    q=0;
return q;
}

Z2(xx,aa,bb,cc,dd)
int xx,aa,bb,cc,dd;
{
int q=0;
if ((ARFA2-BETA*LAMDAc2*(J[xx][aa][bb+1][cc][dd]-J[xx][aa][bb][cc][dd]))(0)
    q=1;
else
    q=0;
return q;
}


OP(a1,a2,a3,a4,a5)
```

```
int a1, a2, a3, a4, a5;
{
int k1=0;
float A, B, C;

A=MUm1*J[a1][a2][a3][a4-1][a5+1];
B=MUm2*J[a1][a2][a3][a4][a5-1];
C=(MUm2-MUm1)*J[a1][a2][a3][a4][a5];
if (A-B+C⟨0.0)
    k1=N1-2*a2-a3;
else
    k1=0;
return k1;
}
```

```
*************************************************************************
*                                                                       *
*Program Two                                                            *
*Simulation Program for Jumping Mode                                    *
*                                                                       *
*************************************************************************
```

```
# include <stdio.h>
# include <math.h>
int A1=1, A2=1, D1=0, D2=0;
int deltan1=2, deltan2=2, N1=9, N2=13;
int seed1, seed2;
float SCALE=65536.0;
float lamdam1=9.96, lamdam2=6.8, mum1=11.8, mum2=12.5;
float cka1[101], cka2[101];

float lamdac1, lamdac2, muc1, muc2;
float i1ck[20], i2ck[20];

int blockz1[6][6][8][8], blockz2[6][6][8][8];
FILE *in, *result;

main()
{
float TIMEOUT=0.0;
float infinite=500000.0, AD=0.0;
float ckd1, ckd2, ckaa2=2500.0, ckd22;
int counter=0, m1=0, m2=0, b2=0, mm2=0;
float ck0=0.0, AA, BB, AC, AE;
float RDD1(), RDD2(), RDD3(), RDD4();
int DEC();
extern float lamdam1, lamdam2, mum1, mum2;
extern int A1, D1, A2, D2, seed1, seed2;
extern int deltan1, deltan2, N1, N2;
float DELTAW1=0.0, DELTAW2=0.0, WAIT1=0.0, WAIT2=0.0;
int i1=0, i2=0, j1=0, j2=0;
float SER1=0.0, SER2=0.0, deltas1=0.0, deltas2=0.0;

extern float lamdac1, lamdac2, muc1, muc2;
float cki1, cki2;
int c1, c2, k1, k2;
int blocki1, blocki2, Z1, Z2, calli1, calli2;
float  RD1(), RD2();
float  prob1, prob2;
```

```
float p1, p2;

int v1, v2, d1, d2;


printf("How many seconds you want to simulate?\n");
scanf("%f", &TIMEOUT);
printf("Please input seed1 and seed2.\n");
scanf("%d  %d", &seed1, &seed2);
printf("Please input lamdam1, lamdam2, mum1 and mum2.\n");
scanf("%f %f %f %f", &lamdam1, &lamdam2, &mum1, &mum2);

printf("Please input lamdac1, lamdac2, muc1 and muc2.\n");
scanf("%f %f %f %f", &lamdac1, &lamdac2, &muc1, &muc2);
SCALE=pow(2.0, 31.0)-1;


/*** read switch curve Martix ***/

in=fopen("data.out","r");
result=fopen("final.out","w");


for (v1=0; v1<=5; v1++)
  {
     for (v2=0; v2<=5; v2++)
        {
           for (d1=0; d1<=6; d1++)
              {
                 for (d2=0; d2<=6; d2++)
                    fscanf(in,"%3d", &blockz1[v1][v2][d1][d2]);
                 fscanf(in,"\n");
              }
           fscanf(in,"\n");
        }
  }

fscanf(in,"\n");
fscanf(in,"\n");

for (v1=0; v1<=5; v1++)
  {
     for (v2=0; v2<=5; v2++)
        {
           for (d1=0; d1<=6; d1++)
              {
                 for (d2=0; d2<=6; d2++)
                    fscanf(in,"%3d", &blockz2[v1][v2][d1][d2]);
                 fscanf(in,"\n");
              }
           fscanf(in,"\n");
        }
  }
```

```
fclose(in);
fclose(result);


cka1[0]=0.0+RDD1(lamdam1);
ckd1=cka1[0]+RDD3(mum1);
ckd2=RDD2(lamdam2);
cka2[0]=0.0+ckd1;
cka2[1]=cka2[0]+RDD2(lamdam2);
ckd22=cka2[0]+RDD4(mum2);

cki1=0.0+RD1(lamdac1);
i1ck[0]=cki1+RD2(muc1);
cki2=0.0+RD1(lamdac2);
i2ck[0]=cki1+RD2(muc2);

printf("cka1[0]=%2.3e   ckd1=%2.3e .\n", cka1[0], ckd1);
do
   {                                              /** set clock ck0 **/
     ck0=ck0+0.001;
                                                  /*** call i1 ***/
if (cki1<= ck0)
{
       cki1=cki1+RD1(lamdac1);
       if (i1>=6 || i2>=6 || j1>=7|| j2>=7)
            Z1=1;
       else
            Z1=blockz1[i1][i2][j1][j2];
       if (2*i1>=N1+1)
            Z1=1;

       if (Z1==1)
           {
             blocki1=blocki1+1;
             cki1=cki1+RD1(lamdac1);
           }
       else
           {
             calli1=calli1+1;
             i1ck[c1]=ck0+RD2(muc1);
             c1=c1+1;
             if (c1>=20)
                 c1=0;
           }
       i1=0;
          for (k1=0; k1<=19; k1++)
         {
             if (i1ck[k1]>ck0)
                   i1=i1+1;
         }
}

if (cki2<=ck0)                          /*** call i2 ***/
```

```
{
        cki2=cki2+RD1(lamdac2);
        if (i1)=6 || i2)=6|| j1)=7|| j2)=7)
                Z2=1;
        else
                Z2=blockz2[i1][i2][j1][j2];
        if (2*i1+i2)=N2+1)
                Z2=1;

        if (Z2==1)
            {
               blocki2=blocki2+1;
               cki2=cki2+RD1(lamdac2);
            }
        else
            {
               calli2=calli2+1;
               i2ck[c2]=ck0+RD2(muc2);
               c2=c2+1;
               if (c2)=20)
                   c2=0;
            }
        i2=0;
        for (k2=0; k2(=19; k2++)
           {
               if (i2ck[k2])ck0)
                     i2=i2+1;
           }
   }


       /*** DATA1 ARRIVAL ***/

deltan2=N2-2*i1-i2-deltan1;
if (ck0(TIMEOUT)
  {
     if (ck0)=cka1[m1])
        {
         cka1[m1+1]=cka1[m1]+RDD1(lamdam1);
         A1=A1+1;
         m1=m1+1;
         if (m1)=100)
            {
             cka1[0]=cka1[100];
             m1=0;
            }
        }
   }


      /*** DATA1 DEPARTURE ***/
```

```
if (D1)=A1)
      ckaa2=infinite;
else
{
 if (ck0)=ckd1)
      {
       deltan1=DEC(i1, i2, j1, j2);      /*** DELTAN1=0 ? ***/
       deltan2=N2-2*i1-i2-deltan1;

/*** ckd1!=infinite ***/


            if (ckd1(=cka1[m2+1])
        {
              ckd1=cka1[m2+1];
              DELTAW1=0.0;
        }
          else
               DELTAW1=ckd1-cka1[m2+1];
p1=((float)deltan1)*mum1;

            deltas1=RDD3(p1);
            SER1=SER1+deltas1;
            ckd1=ckd1+deltas1;
                 ckaa2=ckd1;

            cka2[b2+1]=ckaa2;
            counter=counter+1;
            b2=b2+1;
            if (b2)=100)
                 {
                 cka2[0]=cka2[100];
                 b2=0;
                 }

            D1=D1+1;
            WAIT1=WAIT1+DELTAW1;
            m2=m2+1;
            if (m2)=100)
                m2=0;
     }
}

            /*** DATA2 ARRIVAL ***/

/*** kill ckd2=infinite ***/


if (ck0)=ckd2)
     {
       cka2[b2+1]=cka2[b2];
       cka2[b2]=ckd2;
       ckd2=ckd2+RDD2(lamdam2);
```

```
            A2=A2+1;
            b2=b2+1;
            if (b2>=100)
               {
                  cka2[0]=cka2[100];
                      b2=0;
               }
         }


    /*** DATA2 DEPARTURE ***/


    if (ck0>=ckd22 )
        {


 /*** ckd1 skip ***/



            if (ckd22<=cka2[mm2+1])
                {
                 ckd22=cka2[mm2+1];
                 DELTAW2=0.0;
               }
            else
                          DELTAW2=ckd22-cka2[mm2+1];
p2=(float)deltan2*mum2;
            deltas2=RDD4(p2);
            SER2=SER2+deltas2;
            ckd22=ckd22+deltas2;
                  D2=D2+1;
            WAIT2=WAIT2+DELTAW2;
                mm2=mm2+1;
            if (mm2>=100)
                mm2=0;
       }


    j1=A1-D1;                    /*** i1,i2 decide by call03.c ***/
    j2=A2+D1-D2;

} while (D2<A1+A2 || ck0<= TIMEOUT);
                                        /*** ck0 ***/


printf("A1=%d   D1=%d.\n", A1, D1);
printf("A2=%d   D2=%d.\n", A2, D2);
WAIT1=WAIT1-DELTAW1;
WAIT2=WAIT2-DELTAW2;
AC=(WAIT1+SER1)/D1;
AE=(WAIT2+SER2)/D2;
```

```
AD=(WAIT1+WAIT2+SER1+SER2)/(A1+A2);
printf("Total Data are %d and %d.\n", A1, A2);
printf("Total Waiting Time is %2.3e and %2.3e.\n"
, WAIT1, WAIT2);
printf("Total Service Time is %2.3e and %2.3e.\n"
, SER1, SER2);


printf("NODE1 Average Time Delay is %2.4e.\n", AC);
printf("NODE2 Average Time Delay is %2.4e.\n", AE);
printf("Average Time Delay is %2.4e.\n\n\n", AD);

/***AA=(1/mum1)/(1-lamdam1/mum1);***/
printf("NODE1 Average Time Delay Should be %2.4e\n", AA);

/***BB=(1/mum2)/(1-(lamdam2+lamdam1)/mum2);***/
printf("NODE2 Average Time Delay Should be %2.4e\n\n", BB);
printf("counter=%d.\n", counter);
printf("clock stop at %2.4e.\n", ck0);

prob1=(float)blocki1/((float)blocki1+(float)calli1);
prob2=(float)blocki2/((float)blocki2+(float)calli2);
printf("Trunk 1 blocking prob. is %2.3f.\n", prob1);
printf("Trunk 2 blocking prob. is %2.3f.\n", prob2);
printf(" current i1 is %d      i2 is %d.\n", i1,i2);
printf("Total call i1 is %d   i2 is %d \n", calli1, calli2);
printf("Total block i1 is %d   i2 is %d \n", blocki1, blocki2);

}                                                /*** main ***/



int DEC(a, b, c, d)
int a, b ,c, d;
{
int k1;
float kk;
extern float mum1, mum2;
extern int N1;

kk=(mum1*d)-(mum2*c);
if (kk)=0.0 || 2*a+b)=N1-2)
    k1=1;
else
    k1=N1-2*a-b;
return k1;
}



float RDD1(lamda)
float lamda;
{
float x, y;
```

```
extern seed1;

seed1=seed1+12;
srandom(seed1);
y=random()/SCALE;
x=(-log(y))/lamda;
return x;
}

float RDD2(mu)
float mu;
{
float x, y;
extern seed2;

seed2=seed2-3;
srandom(seed2);
y=random()/SCALE;
x=(-log(y))/mu;
return x;
}

float RDD3(lamda3)
float lamda3;
{
float x, y;
extern seed1;

seed1=seed1+42;
srandom(seed1);
y=random()/SCALE;
x=(-log(y))/lamda3;
return x;
}

float RDD4(mu4)
float mu4;
{
float x, y;
extern seed2;

seed2=seed2+23;
srandom(seed2);
y=random()/SCALE;
x=(-log(y))/mu4;
return x;
}

/*** delete blockz1() ***/

/*** delete blockz2() ***/
```

```
float  RD1(lam)
float  lam;
{
float  x,  y;
extern seed1;

seed1=seed1+22;
srandom(seed1);
y=random()/SCALE;
x=(-log(y))/lam;
return  x;
}


float  RD2(mu22)
float  mu22;
{
float  x,  y;
extern seed2;

seed2=seed2-1;
srandom(seed2);
y=random()/SCALE;
x=(-log(y))/mu22;
return  x;
}
```