THE CONTROL OF MULTIMEDIA COMMUNICATION PROCESSORS : FOR MESSAGES WITH TIME CONSTRAINTS

by

Jen-Lun Yuan

Thesis submitted to the faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1988

Advisory Committee :

Professor Dr. John S. Baras
Professor Dr. William S. Levine
Assistant Professor Dr. Pantelis G. Tsoucas

ABSTRACT

Title of Thesis : The Control of Multimedia Communication Processors for Messages with Time Constraints

Jen-Lun Yuan, Master of Science, 1988

Thesis Directed by : John S. Baras
Professor
Electrical Engineering Department
University of Maryland at College Park

Abstract — This thesis deals with the message switching problems in a multiple networks environment with time constraints for each message class. The sojourn time distribution for an overtake free path are first derived for two classes of routing algorithms, namely, the shortest path algorithms and the optimal routing algorithms, based on Kleinrock's assumptions for standard queueing systems. The estimates of 99% or other percentile of the delay distributions are derived. They are converted into sequences which represent the "reward" one gets by operating one of the subnetwork(media) with routing strategies specified. Based on this reward sequence we formulate the problem of message allocation into the famous "Multi-armed bandit problems" in which the optimal policy has been known to be operating the arm of the bandit with the "Gittins index". A simpler index is found by the non-increasing properties of our reward sequences. Our algorithms switch messages based on this index rule to the maximal-current- reward(MCR rule) network. Our algorithms are suitable for distributed implementation on a multimedia network environment. We show that the indices can be computed in time $O(n \log n)$ by employing the fast Fourier transform algorithms to compute sample points for delay distribution. The algorithms have been simulated by QNAP2 and the simulation results show that our algorithms give better delay

performance than most conventional routing algorithms do. It also reduces the portion of messages that are not able to meet the time constraints.

ACKNOWLEDGEMENT

I am indebted to my advisor Dr. John S. Baras for his patient guidance and enthusiastic advise during the preparation of this thesis. His kindly trust and timely encouragement motivated me in my academic study. I am also very grateful to Dr. William S. Levine and Dr. Pantelis G. Tsoucas for serving on the thesis committee. Special acknowledgement is given to National Science Foundation and CONTEL / Federal System Department for their support of this research. Finally and most importantly, I wish to acknowledge that this thesis could not have been accomplished without the encouragement of my parents, Shih-Chu Yuan and Chin-Lien H. Yuan.

TABLE OF CONTENTS

CHAPTER
ONE
INTRODUCTION

This thesis considers the problems for the control of communication processors which allocate incoming messages to an available set of transmission media. A major performance measure for this communication processor is to guarantee that a large percentile of messages, say 99%, must be delivered within a time bound. This problem arises in applications that involves time-critical message contents and in cases where the cost for the design and operation of a network is not a major concern[3].

In recent years, the communication technology of satellites, optical fibers and VHF/UHF radio are increasingly important in the design of a practical communication networks[21]. One may expect that networks with more than one transmission media will be available in the near future. On the other hand, the technique to store-and-forward messages in a computer communication network has received a lot of research effort since the pioneering works of ARPANET[1,3]. Therefore, it is of great interest and importance to investigate the problem for controlling the multimedia packet-switching communication networks[27,28,29].

In a practical multimedia network each communication processor(node) may

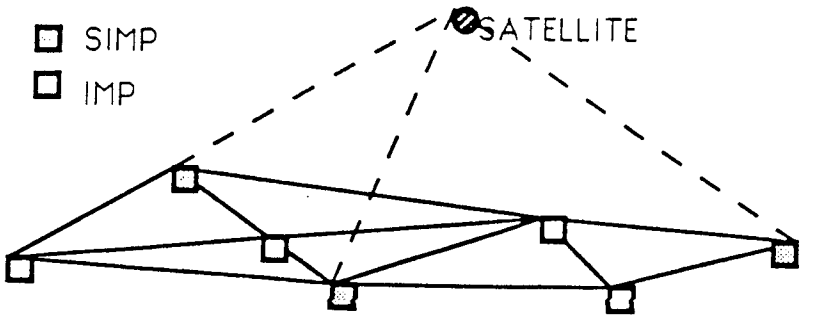


Figure 1.1: A multimedia network

have different set of media available for switching. For example, Consider the multimedia network of Fig.1.1 which consists of a ground(terrestrial) subnetwork and a satellite subnetwork. Each node is a interface message processor (IMP) or a satellite interface message processor(SIMP). For each SIMP, there are two media available but for each IMP, the message can only be routed to the terrestrial network. We are interested in designing the routing problems for the SIMP which switches messages between the ground subnetwork and the satellite subnetwork.

This problem is especially important when some of the subnetworks are existing networks[34]. In the design of such multimedia networks, we have no control of the messages once they are sent to those existing networks. The existing networks have their own protocols for routing. Therefore, our problem is to design a message allocation scheme(or a routing procedure) for the switching nodes(like SIMP's in Fig 1.1) which have the capability to utilize different media(subnetworks). The allocation scheme uses information (e.g. local queue length, utilization factors, mean service time, customer priorities , connectivity..) fedback from each media to make routing decisions. Usually, the the feedback information are different for different subnetworks and sometimes only part of these information are available. Our routing schemes have to make decisions based on the available infor-

mation[39].

This type of networks design problem have never been reported before in literature. Most previous research effort in the design of multimedia networks has been devoted to optimization of traffic flows and capacities assigned for each link subject to a set of delay constraints[1,27,28,29,30]. Their studies centered on networks utilizing all media available and assumed that we have control for every intermediate nodes(the IMP's in Fig.1.1) in every subnetwork. It is thus difficult to apply their result to our problems where some media may be an existing networks which already has its own protocols for routing. Therefore, our objective here is to propose a scheme(algorithm) which can be implemented on the communication processors working as message interfaces for some existing networks(media). Furthermore, our algorithms should be adaptive in nature, reponding to changes in the information provided by each subnetwork.

Besides the problems we outlined above, we have found several problems associated with currently existing routing and control strategies for a communication network:

- Up to the present, most studies and implementations have been centered on networks using solely point-to-point links like the Advanced Research Project Agency Network(ARPANET)[31,32,33] or multi-access network like ALOHANET[35]. Routing procedures for the point-to-point type network have been studied extensively[1,15,16,22,23]. The multi-access network also led to a rich field for the study of the random access protocols[1, 17,18,19,20]. However, very few has been done for the combined model where the point-to-point networks and the multi-access networks are considered at the same time[28,29].

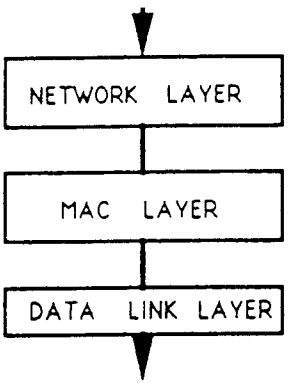


Figure 1.2: (a) MAC layer(1)

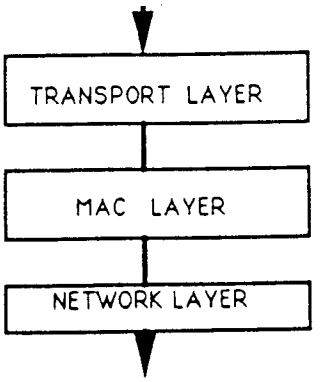


Fig.1.2(b) MAC layer(2)

- Most network design problems up to date fall into the category in which the location of nodes(topology), links and channel capacities are selected so as to minimizing some objective function under some constraints which are usually the traffic demands, estimates of some percentile of the delay distribution and reliability considerations[1,5,36,37,38]. None of them considered the design of network processors to control several existing networks in which the nodes, links and channel capacities are all fixed and the only free parameters are the feedback information from each media(subnetwork).

- The ISO standard seven layered architecture[8] which are adequate for most network designs[6,7] is not readily applicable to the design of multimedia networks. If we consider the control of different media as a sub-task performed by the network layer(ISO model), then we should introduce an additional layer between the network layer and the data link control layer. This layer is called the Medium Access Control(MAC) layer (see Fig.1.2(a)). On the other hand, if we consider each medium as a network , we are actually introducing the MAC layer as the protocol for media control which lies between the transport layer and the network layer(see Fig.1.2(b)).

- Routing strategies known today can be classified into two categories: one

is the shortest path routing[15,22,23] which usually consists of finding an estimated link delay(ELD) and assigning flows along the path with the smallest ELD. The other is known as the bifurcating routing[5],or optimal routing[4,16] or the best stochastic routing[39]. The approach optimizes the overall average packet delay given the set of link capacities and traffic requirements and finds the optimal flow on each link of the network. However no known implementations of this approach to a practical network has been known yet[5].

- The time constraint considered by present routing strategies are almost always the "socially optimal policy" which minimizes the average delay of all messages as described above. In some cases, we may need to consider the "individually optimal policy" which for each message minimizes its own delay[49]. Especially in time-critical cases where the performance measure is not just minimizing the average delay but keeping the sojourn time in the network below a time bound. In this case the "socially optimal policy" does not satisfy the problem since it may sacrifice one job if this job increase the overall delay for all jobs[49].

Based on the problem for current routing strategies, we proposed a control scheme with the following characteristics:

- A control scheme which utilizes multiple networks(media).
- Algorithms can be accommodated with existing networks.
- The routing decisions are made based only on partial information fed back from each media(subnetwork).

- Algorithms are heuristic and adaptive to the major changes of networks.
- Time constraints considered are "individual" as opposed to "socially" optimizing criteria. The purpose is to guarantee a large percentile of massages arrives within the time bound.

The organization of this thesis is as following: In Chapter 2, we first give some interesting models that are relevant to the control of multimedia networks. These models motivates our heuristic algorithms and the switching rule which is based on the computation of estimates of some percentile of delay distributions. The derivation of these estimates are given in Chapter 3. A simple model is proposed so that the analysis is tractable. In Chapter 4, we give the heuristic algorithm based on the "Multiarmed bandit problem"[54,55] and show that the index can be computed with time $O(n \log n)$ by the Fast Fourier Transform algorithms. Simulation results are given in Chapter 5. Finally, suggestions for implementation of our algorithms by VLSI chips are proposed in Chapter 6 where conclusions and comments are given.

CHAPTER
TWO

SOME RELATED MODELS AND BACKGROUND REVIEW

In this chapter, we give three queueing models that are related to our problem for the control of multimedia communication networks.

2.1 The Heterogeneous-Server Model

Consider the network in Fig 2.1: There are two media with the same point-to-point connections. Assume that each node in the network has the capabilities to switch messages to different media and each medium is characterized by a unique capacity C_i, $(i = 1, 2, C_1 > C_2)$.

To control the routing of messages for this two-media network, it suffices to propose a two level routing scheme and divide the whole problem into two portions:

- Global routing level
- Local routing level

By this approach, we try to combine the strength of centralized and distributed algorithms by an algorithm in two portions: In the centralized portion, a global de-

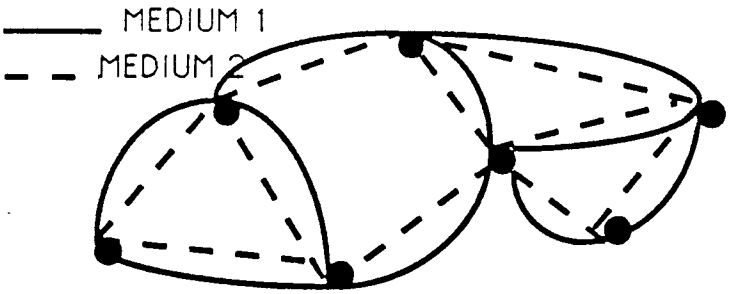
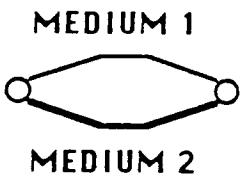


Figure 2.1: A two media network

cision is made as to which "logical path" to take among all the paths available for a source-destination pair and we can employ any of the existing routing algorithms which is optimal in the sense of smallest average delay[15,40]. In the distributed portion, each node makes its final routing decision about which medium to take based on its local information.

This two-level approach is like delta-routing[41] and R.P.Boorstyne and A.Livne proved that this two-level approach improves in time delay while maintaining good paths[42]. What we have done here is to introduce an additional layer between the network layer and the data link control layer like we did in Chapter 1(see Fig.1.2(a)).

Now, consider local routing for this network. Each link can be decomposed into two physical links(see Fig. 2.2(a)). If we model each physical link as a single server, we have a queueing model depicted in Fig. 2.2(b) where a queue is added to the head of each node as an input buffer. Messages coming to this node are queued in this input buffer waiting for service. Each server is characterized by a different service distribution according to the class of the customer. In standard queueing terminology this is a G/G/2 queueing system.



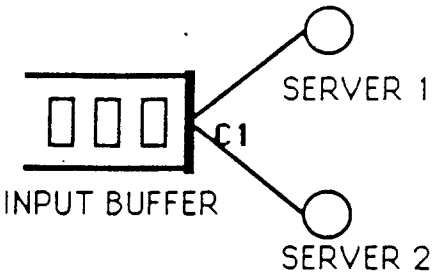


Figure 2.2: (a)A logical link. Fig.2.2(b) A G/G/2 queue.

The control problem of such a G/G/2 queues has received considerable attention in two respects: either seeking a individually optimal policy (see Chap.1) or a socially optimal policy which minimizes the average delay of all jobs.

In [43], Agrawala et.al. considered the problem of obtaining a socially optimal policy when there are no arrivals to the buffer and the service time distributions are exponential with mean $\mu^{-1}(i = 1, 2, \mu_1 \geq \mu_2)$. This is in fact the G/M/2 queueing system. They have shown that the socially optimal policy which minimizes the total delay of all existing jobs is given by:

Theorem 2.1 *The link with a smaller capacity(or the link associated with μ_2) should be utilized only when the number of customers waiting in the buffer is greater than T where T is given by:*

$$T = \mu_1/\mu_2 - 1 \tag{2.1}$$

The result can be generalized to the G/M/s case, with $s \geq 2$(see [49]). In [44], Larsen obtained the socially optimal policies for exponential servers when the arrivals form a Poisson process. He also conjectured that if arrivals are Poisson and service process are exponential, then the socially optimal policy is of threshold type. That is, there exists a number τ, such that the second media(link) should

be utilized only when the number of jobs in the buffer exceeds τ. This is proved by Lin and Kumar[66]. They also show that as τ increases and the rate λ of Poisson arrivals decreases τ will converge to (2.1) as $\lambda \rightarrow 0$. It is conjectured that this result can also be extended to the cases $s \geq 2$(see [49]).

Recently, Viniotis and Ephermides[45] extended the result to the case of arbitrary distribution of arrivals and with service time of unequal distributions(also arbitrary). Both average(socially) cost and discounted cost are considered[45].

In [39], Kumar and Walrand studied the individually optimal control of this queueing system also with general arrivals and service processes.

These results give us the dynamic routing rule for the local routing level. However, we would like to consider a more sophisticated model which is motivated by studying the node structure(Fig. 2.3(a)) in a real communication system[46].

2.2 The Multiple Output Queue Model

Consider the outgoing links of Fig.2.3(a), each link corresponds to a medium or a link in a sub-network. Taking the communication processor with the output links gives us the model of Fig.2.3(b).

The communication processors in Fig.2.3(b) operates as follows: Messages arrive from the incoming links, the communication processor fetches the message from the input buffers and decides which way to send it. The processor makes its routing decisions based on the following information:

1. The capacity C_1, C_2 of different media.

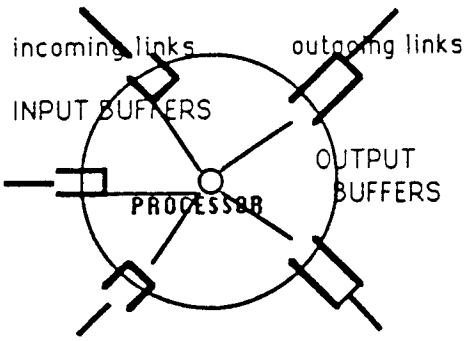


Fig. 2.3(a) The node structure for a communication network

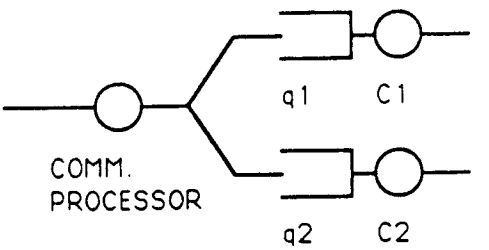


Fig.2.3(b) The output queue model

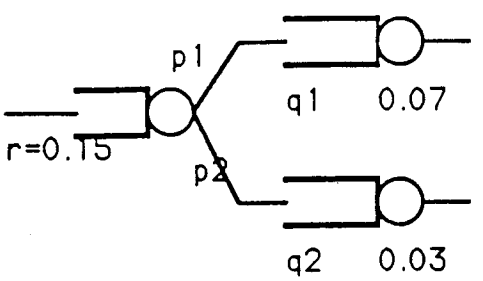
2. The message length distribution of each class of customers.

3. The "population" in q_1, q_2.

By class of customers, we mean each class is characterized by different message length distribution and possibly different paths(determined by global routing level). By "population" in each queue, we mean the information about the number of customers in the queue and the class each customer belonging to.

The control problem for switching messages to output buffers by this communication processor is a dynamic routing problem[47]. The difficulties involved in studying this problem stems from two sources: First, the information available at different nodes may be different. Second, dynamic routing strategies can lead to queue behavior whose statistical characteristics are not yet understood[47].

Because of the intrinsic difficulty outlined above, algorithms for dynamic routing are almost always heuristic[39,48,51]. Among these heuristics, it may be interesting to compare their performance with the bifurcating routing rule[22] which is optimal in the sense that it gives the minimal overall delay averaged for all source-destination pairs. With this routing rule, our problem simplifies to finding the optimal flow x_1, x_2 on q_1 and q_2.



	MEAN	BUSY PCT	NB
q1	0.2	0.490	512
q2	3.0	0.837	356

Fig. 2.4(a) A bifurcating problem Fig. 2.4(b) Simulation Data

The optimal flows are given by(see [4]):

$$\begin{cases} x_1 = r \\ x_2 = 0 \end{cases} \quad \text{for } r < C_1 - \sqrt{C_1 C_2}$$

$$\begin{cases} x_1 = \frac{\sqrt{C_1}}{\sqrt{C_1}+\sqrt{C_2}}(r - (C_1 + C_2)) + C_1 \\ x_2 = \frac{\sqrt{C_2}}{\sqrt{C_1}+\sqrt{C_2}}(r - (C_1 + C_2)) + C_2 \end{cases} \quad \text{for } r \geq C_1 - \sqrt{C_1 C_2} \tag{2.2}$$

We assign each incoming message to q_1 and q_2 according to probabilities p_1 and p_2 as in [39] where:

$$\begin{aligned} p_1 &= x_1/(x_1 + x_2) \\ p_2 &= x_2/(x_1 + x_2) \end{aligned} \tag{2.3}$$

One can easily find out the problems about the bifurcating routing rule. For example, consider the queueing system in Fig. 2.4(a): the routing probabilities p_1, p_2 are computed according to (2.3). However, one may find that the second queue may cumulate a lot of customers as in Fig. 2.4(a) while the first queue is empty. (It is also likely for queue one to cumulate customers as queue two.) One can readily see that resources are wasted and the performance will be decreased. This example also gives us the idea to "balance" the load between q_1 and q_2. It

also tells us that by adding some deterministic rules to the routing, it will reduce the overall delay.

The simplest balancing scheme is to send the message to the output queue with the shortest queue length. More balancing scheme can be found in [48]. For the join-the-shortest-queue rule(JSQ rule), Ephremides et. al. has proved that it is optimal under the constrain that $C_1 = C_2$ and that all messages have the same message length distributions[47].

However, in cases where multiple classes of customers are in the network, in order to get a better result, we need to use the information about the customers in each queue rather than just using the queue length for routing. For example, consider an arriving sequence, A,B,A,B with arrival time 0,1,2,3 respectively and no arrivals after that. Assume $C_1 = C_2$ for the queueing model in Fig. 2.3(b), if we use the JSQ rule to route the sequence, we find that the resulting output in each queue are as shown in Fig. 2.5(a) or Fig. 2.5(b). We also assume that the service time is deterministic for A and B with $\tau_A = 4, \tau_B = 10$. Now, the total delay for Fig. 2.5(a) and Fig. 2.5(b) can be computed:

$$T_1 = 4 + 10 + (2 + 4) + (8 + 10) = 38$$

$$T_2 = 4 + 10 + (4 + 8) + (10 + 1) = 37$$

However, one may easily verify that the optimal assignment for this sequence is as in Fig. 2.5(c), with total delay:

$$T_3 = 4 + 10 + (2 + 4) + (1 + 4 + 10) = 35$$

This example motivates the idea to make use of message length distribution to make a better decision. An algorithm has been proposed by Yuan[52] in which

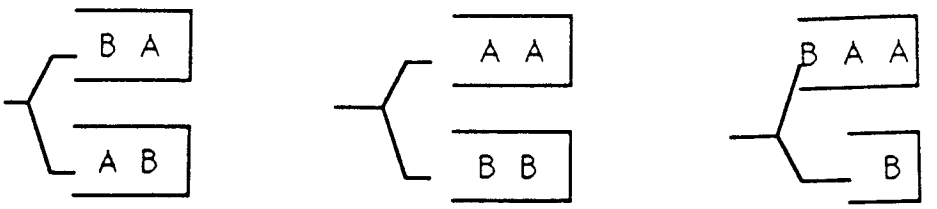


Figure 2.5: (a)JSQ rule(1). Fig.2.5(b)JSQ rule(2). Fig.2.5(c)Optimal assignment.

the effective queue length for queue j is given by:

$$\overline{L_j} = \sum_k L_j^k / C_j \tag{2.4}$$

where C_k is the capacity at link j (associated with queue j) and the $L_j{}^k$ denotes the message length for k_th message in queue j. This equation is the effective residual service time [4] of output queue j.

The simulation result given in [52] shows that their algorithms outperform the bifurcating routing[22] and the JSQ rule described above.

2.3 The Multiple-Subnetworks Model

In previous sections, we presented two queueing models, namely, the heterogeneous-server model and the multiple output queue model. In these two models, we assume that we have control over any single node for all media. This means that in order to implement the dynamic routing rules presented in Sec.2.1 and Sec.2.2, we have to control the processor C_1 in Fig.2.2(b) and the controller in Fig.2.3(b). Nevertheless, in practical networks, each medium may be an existing network(see

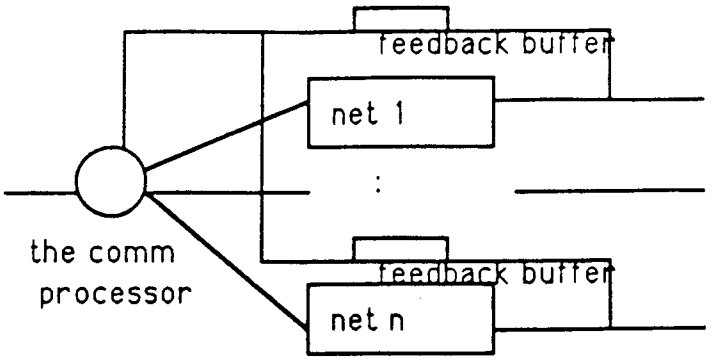


Figure 2.6: The Multiple Network Model

Chapter 1) in which we have no control over each single node so as to use they for the purpose of routing. It is thus possible to make routing decisions at certain "switching nodes" only and the decision is based on the information fed back by each media(subnetwork). In this case, we may consider each single medium network as a "black box". The switching nodes then choose one of these parallel boxes to send the message.

In this model, each subnetwork represents a single media, we will thus use "media" and "subnetwork" as synonymies.

Note that the only information one knows about each black box is from the feedback buffer which is refreshed periodically by the subnetwork. The feedback information may be different for different subnetworks. For example, one subnetwork may be an random access network like HF radio net, another one may be a terrestrial(ground) network. For the radio net, the feedback information may include the status or utilization of the collision channel[4] while the ground network may include load, message classes, connectivity, etc for the ground links. Since each network feeds back only part of the information about the network itself. The decision rules which utilizes these information are usually heuristic. It is the major purpose of this thesis to propose an efficient heuristic algorithm for this

multiple subnetwork model presented in this section.

Our heuristic algorithm switches messages according to an index associated with each subnetwork(see Chapter 4). The index is chosen according to the requirement of the specific problem. For example, if the requirement of the problem is to minimize the expected delay, we can choose the index to be the the average sojourn time for all messages throughout the network. If the problem is to guarantee that ninety-nine percent of the messages to arrive within the time bound(like what we did in Chapter 1), then we can choose the index to be the estimates of the 99% or other percentiles of the delay distributions. We will refer to these estimates collectively as delay bounds, by abuse of terms. We want to emphasize that they are different from average delay estimates.

The advantages of using this approach for message switching is the following:

- Flexibility - The index can be chosen to fit a particular problem.
- Modularity - The allocation algorithm is independent of the index computation.
- Portability - The algorithm can be easily accommodated with existing networks.
- Efficiency - The heuristics can be chosen so that the complexity of index computation is tractable.
- Adaptability - The routing rule is adaptive responding to the major changes of the network.

In the next chapter, the indices for message switching will be derived. We derive the index for computing the ninety-nine delay percentile since the index for computing the average delay can be easily found from the delay distribution. The

switching algorithms utilizing these indices will be given in Chapter 5 where we consider the switching problem to be the "Multi-armed bandit problem"[53] and the "Gittins index rules" are in fact the allocation rule we used for routing[54,55].

CHAPTER
THREE

DERIVATION OF INDICES FOR MESSAGE SWITCHING

In Chpater 2, we introduce three queueing models for the multimedia network. Our emphasis is on the multiple network model presented in Sec.2.3. The model overcomes some difficulties for the control of existing netowrks discussed in Chapter 1. In order to implement the allocation algorithms for this model we need to know the indices for switching. In this chapter the indices for delay bound are derived based on a simple mathematical model proposed in Sec.3.1.

Indices for evaluating most packet-switching networks are usually the following[1]:

- delay
- throughput
- cost
- reliability

For our problem, the most important index for performance evaluation is the delay experienced by messages throughout networks. Our main purpose is to keep the portion of messages whose sojourn time exceed the time constrain as small as

possible. This is the same as keeping the "tail" of the delay distribution which lies beyond the delay bound to a minimum. Therefore, we need to know the delay distribution which represents some good approximation of the sojourn time for a single message. The analysis is based on Kleinrock's model[2,3] with routing which satisfies the definitions of Marcov routing(Sec.3.1)[57]. It can be shown that the shortest path routing satisfies the definitions of our analytical model and its delay distribution can thus be obtained according to the product form results proposed in Sec.3.2. Our results on shortest path can be generalized to the case of multiple paths under the condition proposed in Sec.3.3. In fact, the results can be applied to practical routing(Sec.3.3) even though the constraints are violated[9]. It is thus reasonable for us to apply the indices we obtained in this chapter to practical network switching problems.

3.1 The Analytical Model

Consider the network of Fig.3.1 : each link in the network is a M/M/1 - First Come First Serve(FCFS) queue. Each node is a Marcov routing (to be defined later) device. We assume that all customers have the same message length distribution $exp(\mu)$. Furthermore, for each source-destination pair r, a fixed path is chosen so that all messages for this source-destination pair are routed through this path. We then define the class of these messages to be $C_r, r = 1, 2, \cdots, N(N-1)$. Note that the definition of class here are different from the class we discussed in Chapter 1. The processing delay and the propagation delay are neglected in this problem. This assumption implies that we can consider the Marcov routing devices in this model as a dummy queue in which the delay time is zero for all customers. We will see in Sec.3.3 that this is an important assumption for the non-passing condition to be satisfied in the case of shortest path routing. Some

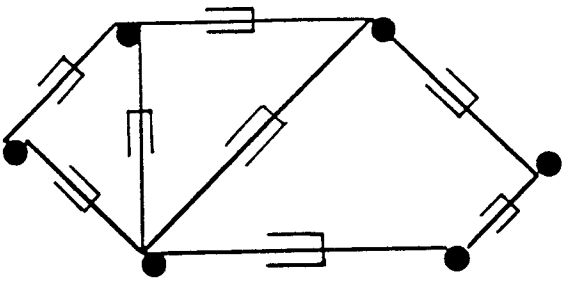


Figure 3.1: The model network

important quantities and notations we used for this model are listed below:

N = number of nodes in the network.

R = number of classes(paths).

M = number of links in the network.

C_i = capacity of link $i, i = 1, 2, \cdots, M$.

$(s_1, s_2, \cdots, s_M)$ = the state of the network.

$s_i = (n_{i1}, n_{i2}, \cdots, n_{iR}), i = 1, 2, \cdots, M$.

n_{ir} = number of customers of class r on link i.

$a(r) = \{i_{r1}, i_{r2}, \cdots, i_{rl}\}$: the set of links on path r with length l.

The reason we model each link to be an M/M/1-FCFS queue is that they satisfy the conditions which leads to product form networks[2,3,11]. This condition can be formally defined as follows:

Definition 3.1 (Quasi-reversibility) *Let $\{A_t\}_{t\geq 0}$ denote the arrival process to a queue i and $\{D_t\}_{t\geq 0}$ the departure process of a queue. A queue is quasi-reversible if :*

$$\{A_t\}_{t\geq s} \perp (n_{i1}, n_{i2}, \cdots, n_{iR}) \perp \{D_t\}_{0\leq t\leq s} \tag{3.1}$$

for all time s. where $\perp$ denotes the independence relations between two random processes.

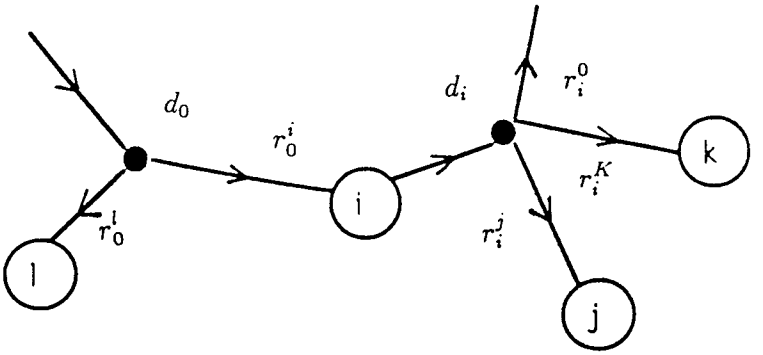


Figure 3.2: Marcov routing

One can see that the product form result is a direct consequence of the quasi-reversibility. There are several quasi-reversible queues which are encountered in practical applications[56,57]. The fact that the $M/M/1$-FCFS queue with the same service distribution for different customer classes is a quasi-reversible queue has been proved in[12] by Disney et. al. We restate their result as the following lemma:

Lemma 3.1 *A stationary $M/M/1$- FCFS queue with customer classes $C_1, \cdots, C_R$ which have the same service time distribution $exp(\mu)$ is quasi-reversible.*

In order to have the product form results, we need to define the routing mechanism used in our model. It is given as follows:

Definition 3.2 (Marcov Routing) *Consider Fig.3.2. The arrival process of class c into queue i is Poisson with rate r_{0i}^c. The customers leave queue i and move to queue j with probability r_{ij}^c. The probability that a customer leaves the network is :*

$$1 - \sum_j r_{ij}^c \tag{3.2}$$

The routing device d_i of Fig.3.2 with routing defined above is a Marcov routing device.

Intuitively, Marcov routing makes routing decisions depending only on the class of customers, not on the past evolution of the network. With the routing decisions described above, one can compute the flow λ_i for each link i of Fig.3.2, and $\{lambda_i, i = 1, 2, \ldots, M\}$ is the solution for the flow conservation equation[3]:

$$\lambda_i = \sum_c r_{0i}^c + \sum_c \sum_{j \neq i} \lambda_j r_{ji}^c \qquad i = 1, 2, \ldots, M. \tag{3.3}$$

One can also define the utilization factor of a link given $\{\lambda_i\}_{i=1,2,\ldots,M}$, $\{\mu_r^c\}_{r=1,2,\ldots,R}$ and $\{C_i\}_{i=1,2,\ldots,M}$ as :

$$\rho_{ir} = \lambda_i / \mu C_i \tag{3.4}$$

and the total utilization of queue i as:

$$\rho = \sum_{r=1}^{R} \rho_{ir} \tag{3.5}$$

Now, we have completed the description of our mathematical model for the network. The point is that such a network implies product form invariant distributions for state $(s_1, s_2, \ldots, s_M)$. This result is due to Baskett et al[11] and is given as follows:

Lemma 3.2 *The network of Fig.3.1 with $M/M/1$-FCFS queue which is quasi-reversible and Marcov routing devices defined in Def.3.2 has a product form invariant distribution:*

$$P(s_1, s_2, \ldots, s_M) = P_1(s_1) P_2(s_2) \cdots P_M(s_M) \tag{3.6}$$

where $P_i(s_i)$, $i = 1, \ldots, M$, is the marginal distribution of the queue length process $\{S_t^i\}_{t \geq 0}$.

3.2 Single Path Delay Distribution

Recall in Sec.2.3 that the indices for the switching algorithms are the delay bounds for a fixed percentile of messages or the average delay used by most current algorithms. We are thus interested in finding the delay distribution for messages through a single path. In order to make our analysis tractable, we need another assumption which gives us the product forms for the generation function of the delay distribution:

Assumption 3.1 (Non-passing condition) *Let link i, j denote any two links on path r, $r = 1, 2, \ldots, R$. Messages arriving after a tagged message at link i never overtake this tagged message at link j.*

Before deriving the delay distribution, we first give a combinatorial fact as a lemma which will be used for our later proof.

Lemma 3.3 *For all $x_i \in \mathcal{R}, i = 1, \ldots, M$, $N \in \mathcal{Z}^+$, the following equation always hols:*

$$(\sum_{i=1}^{M} x_i)^N = \sum_{\substack{(n_1, n_2, \ldots, n_M) \\ n_1+n_2+\cdots+n_M=N}} \frac{N!}{\prod_{i=1}^{M} n_i!} \prod_{i=1}^{M} x_i^{n_i} \tag{3.7}$$

where $n_i \in \{0, 1, 2, \ldots\}$, $i = 1, \ldots, M$. Proof: *see Appendix A.1*

Now, we are ready to derive the delay distribution for any path r defined in Sec.3.1. Let

$$n_r \triangleq \sum_{i=1}^{M} n_{ir} = \sum_{i \in a(r)} n_{ir} \tag{3.8}$$

Note that: $n_{ir} = 0, \quad \forall i \notin a(r)$. This defines the total number of class r messages in the network. Also define the the total number of messages on link i as:

$$n_i \triangleq \sum_{r=1}^{R} n_{ir} \tag{3.9}$$

The joint distribution for $s_i, n_i, \quad i = 1, \ldots, M$ can now be found since the classes of customers in a queue are assumed to be independent, the distribution is a product of utilization factors:

$$\begin{aligned} P(s_i, n_i) &= P\{n_{i1}, n_{i2}, \ldots, n_{iR}; \sum_{r=1}^{R} n_{ir} = n_i\} \\ &= A \frac{n_{i1} + n_{i2} + \ldots + n_{iR}}{n_{i1}! \cdots n_{iR}!} \rho_{i_1}^{n_{i1}} \rho_{i_2}^{n_{i2}} \cdots \rho_{i_R}^{n_{iR}} \end{aligned} \tag{3.10}$$

where A is a normalizing constant. A can be found by:

$$\begin{aligned} \sum_{n_i=0}^{\infty} \sum_{s_i} P(s_i, n_i) &= \sum_{n_i=0}^{\infty} \sum_{(n_{i1}, \ldots, n_{iR})} A \frac{n_i!}{\prod_{r=1}^{R} n_{ir}!} \prod_{r=1}^{R} \rho_{ir}^{n_{ir}} \\ &= A \sum_{n_i=0}^{\infty} \left(\sum_{r} \rho_{ir}\right)^{n_i} \\ &= A \sum_{n_i=0}^{\infty} {\rho_i}^{n_i} = 1 \end{aligned} \tag{3.11}$$

That implies $A = 1 - \rho_i$. After substituting A into (3.10), one gets:

$$P(s_i, n_i) = (1 - \rho_i)(\sum_{r=1}^{R} n_i r)! \prod_{r=1}^{R} \frac{\rho_{ir}^{n_{ir}}}{n_{ir}!} \tag{3.12}$$

From (3.12), we obtain:

$$P(n_i) = \sum_{s_i} P(s_i, n_i)$$

$$= \sum_{s_i}(1-\rho_i)(\sum_{r=1}^{R} n_i r)! \prod_{r=1}^{R} \frac{\rho_{ir}^{n_i r}}{n_{ir}!}$$

$$= (1-\rho_i)\sum_{s_i} \frac{n_i!}{\prod_{r=1}^{R} n_{ir}!} \prod_{r=1}^{R} \rho_{ir}^{n_i r}$$

$$= (1-\rho_i)\rho_i^{n_i} \tag{3.13}$$

Where the last equality follows from Lemma 3.3. Now, taking the Z-transform of (3.13), one finds:

$$N_i(z) = \sum_{n_i=0}^{\infty} P(n_i) z^{n_i}$$

$$= (1-\rho_i)\sum_{n_i=0}^{\infty} (\rho_i z)^{n_i} = \frac{1-\rho_i}{1-\rho_i z} \tag{3.14}$$

With this equation, we can find the Laplace transform by the relation between the Z-transform and Laplace transform evaluated at critical points:

$$N(z) = T^*(\lambda - \lambda z) \tag{3.15}$$

The derivation of this equality is due to Kleinrock[2]. Using (3.15), one may rewrite (3.14) as the Laplace transform of the delay distribution for link i:

$$T_i(s) = T_i(\lambda_i - \lambda_i z)$$

$$= N_i(z)$$

$$= N_i(1 - \frac{s}{\lambda_i})$$

$$= \frac{1-\rho_i}{1-\rho_i(1-\frac{s}{\lambda_i})} \tag{3.16}$$

Combining (3.4) and (3.16), we obtain:

$$T_i(s) = \frac{\mu C_i(1-\rho_i)}{\mu C_i(1-\rho_i)+s} \tag{3.17}$$

Define t_i as the delay time for a single message on link i, one sees that t_i is the inverse Laplace transform of (3.17) since:

$$T_i(s) = \int_0^\infty f(t_i) e^{-st_i} dt_i = E\{e^{-st_i}\} \tag{3.18}$$

Denote the path delay as:

$$t_r = \sum_{i \in a(r)} t_i \tag{3.19}$$

The Laplace transform of t_r is:

$$T_r(s) = E\{e^{-st_r}\} = E\{e^{-s\sum_{i \in a(r)} t_i}\} \tag{3.20}$$

With the non-passing assumption given in the beginning of this section, each t_i, $i \in a(r)$ is independent of one another and (3.20) becomes a product of marginal expectations. One then use (3.17) to get:

$$\begin{aligned} T_r(s) &= \prod_{i \in a(r)} E\{e^{-st_i}\} \\ &= \prod_{i \in a(r)} T_i(s) \\ &= \prod_{i \in a(r)} \frac{\mu C_i (1-\rho_i)}{s + \mu C_i (1-\rho_i)} \end{aligned} \tag{3.21}$$

We have thus obtained the Laplace transform for the path delay under the non-passing condition for the network model in Sec.3.1, This fact is stated as a theorem as follows:

Theorem 3.1 *The network model described in Sec.3.1 admits the delay distribution whose Laplace transform is given by:*

$$T_r(s) = \prod_{i \in a(r)} \frac{\mu C_i (1-\rho_i)}{s + \mu C_i (1-\rho_i)} \tag{3.22}$$

For a single path r, under the non-passing condition.

Our results here is similar to [58,59,60]. But our proof is much simpler and provides the intuitive interpretation of the model networks. In the next section, we shall apply this result to the shortest path routing and use it to approximate the delay distributions for optimal routing.

3.3 Delay Bounds for Two Classes of Routing Algorithms

The Laplace transform of t_r defined in Sec.3.2 provides us all the information one needs to derive the delay distribution of a single path. In this section, we apply these results to the shortest path routing and the optimal routing(static version). The Marcov routing is satisfied by these two classes of algorithms for most cases and we prove that the path delay derived in previous sections are applicable to the case of shortest path routing. For the case of optimal routing, the result still provides a good approximation due to the simulation result by[9]. This implies that the analysis here can be used to provide the indices in practical network switching problems.

We first prove the result for shortest path routing, after that, we generalize our result for a single path to the case of multiple paths and find the corresponding delay distribution for the optimal routing. Furthermore, the mean and delay for these two cases can be easily derived from the delay distribution.

The delay bound for the shortest path is given by the following theorem:

Theorem 3.2 (Delay bound for the shortest path algorithms) *Consider the network of Fig.3.3. Assume that the network parameters are defined as in Sec.3.1.*

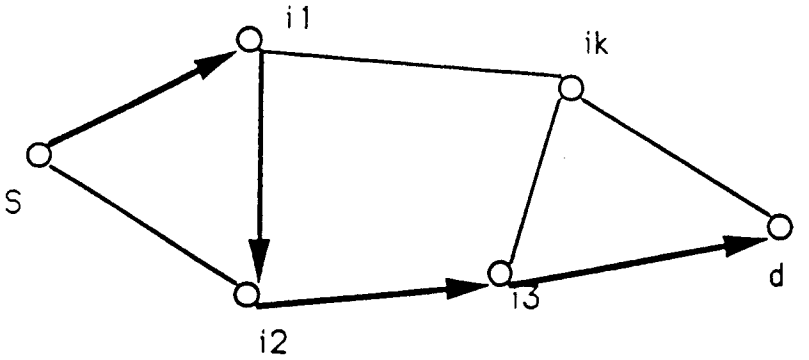


Figure 3.3: Network with Shortest path routing

Suppose we implement the shortest path routing on the routing devices defined in Def.3.2 where the algorithm is centralized, static and a unique path is determined for every source- destination pair. The delay distribution is then given by Theorem 3.1 and the delay bound B_0 for the 99% messages is given by solving the following equation:

$$\sum_{i \in a(r)} b_i exp(-\mu C_i(1-\rho_i)B_0) = c + 0.99 \tag{3.23}$$

where

$$c = \sum_{i \in a(r)} b_i$$

$$b_i = -\frac{a_i}{\mu C_i(1-\rho_i)}$$

$$a_i = \frac{\prod_{k \in a(r)} \mu C_k(1-\rho_k)}{\prod_{\substack{k \in a(r) \\ k \neq i}} (\mu C_k(1-\rho_k) - \mu C_i(1-\rho_i))}$$

Assume that $\mu C_s(1-\rho_s) \neq \mu C_t(1-\rho_t), \quad \forall s \neq t.$

Proof *We first prove that the non-passing conditions in Sec.3.2 are satisfied: Consider the network of Fig.3.3 where* $(s,d) = r$ *is a source-destination pair in the network. We denote the uniquely determined shortest path for* r *as* P_r.

$P_r = \{s, i_1, \ldots, i_{m-2}, d\}$, *where* m *is the path length of* r. *We claim that:*

$$\forall r' = (i,j), \quad i,j \in P_r \Longrightarrow a(r') \subset P_r. \tag{3.24}$$

$$\forall r'' = (k,l), \quad k,l \notin P_r \Longrightarrow a(r'') \not\subset P_r. \tag{3.25}$$

This means that the shortest path for any two points on P_r *lies on the shortest path found by* r. *It implies that the non-passing condition is satisfied without the intervention of nodes that do not belong to this path since there are no other path for messages belonging to* $(i,j) \in P_r$ *to overtake a tagged message in this path(Also note that the queue discipline is FCFS by Definition of the model network in Sec.3.1).*

Now, consider the nodes that do not belong to P_r. *If the non-passing condition is not true, by (3.25) one find that there exists a path* P_s , $P_s = \{j_p, j_{p+1}, \ldots, j_q\} \subset P_{(k,l)}$, *such that:*

$$j_p,\ j_q \in P_r$$

and the only way for one message in P_r *to be overtaken by some other messages is that the overtaking messages pass through another path like* P_s *and join the queue in* P_r *like Fig.3.4. Also note that the processing time is zero by assumption(see Sec.3.1) so that the only possibility for the non-passing condition to be violated is that when the overtaking messages arrive at* j_q, *the tagged message is still in path* $P_{(s,j_{q-1})}$. *That implies there exists two distinct shortest path* r, s *for a single source destination pair* (s,d) *but this contradicts our assumption that only one path is possible for every source-destination pair. Therefore, the non-passing condition is satisfied by the shortest path routing defined above.*

The shortest path routing only depends on the customer class for our case and the definition of Marcov routing(Def.3.2) is thus satisfied. One then conclude that the the condition in Theorem 3.2. are satisfied and the path delay is given by

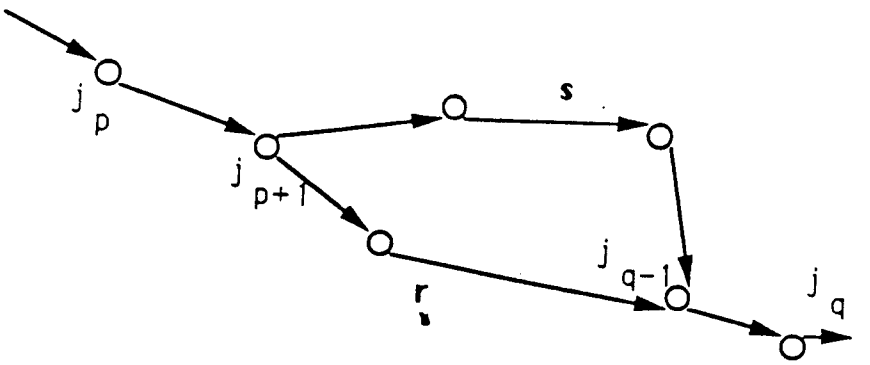


Figure 3.4: Two shortest paths

We can thus find the delay distribution by inverting the transform of (3.22) which is given by:

$$t_r(x) = (\prod_{i\in a(r)} \mu C_i(1-\rho_i))[\sum_{i\in a(r)} (\prod_{\substack{j\in a(r)\\ j\neq i}} \frac{1}{\mu C_j(1-\rho_j) - \mu C_i(1-\rho_i)}) \times$$

$$\times exp(-\mu C_i(1-\rho_i)x)] \tag{3.26}$$

Integrating (3.26) gives us the cumulative desity function(cdf) of t_r*:*

$$F_{t_r}(x) = \int_0^x t_r(y)dy$$

$$= \sum_{i\in a(r)} b_i exp(-\mu C_i(1-\rho_i)x) - c \tag{3.27}$$

where a_i, b_i, c *are defined as in (3.23). Setting* $F_{t_r}(B_0) = 0.99$ *gives (3.24). Q.E.D.*

This theorem shows that our model can be applied to the network utilizing the shortest path routing as long as the processing time at the routing devices are small compared with the queueing delay. However, this model is still too restrictive to be applied to practical network problem. In a practical network, there are

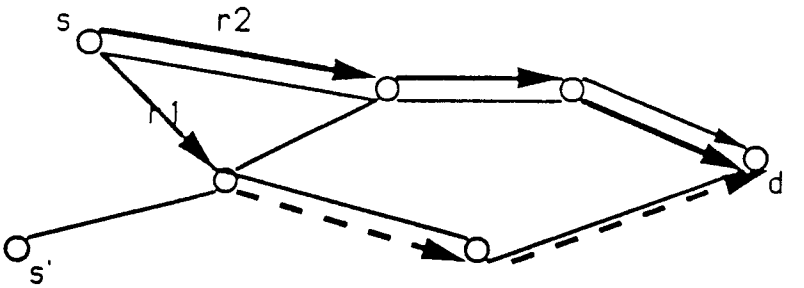


Figure 3.5: Multiple paths networks

usually several paths employed by a single source destination pair. Therefore, we would like to generalize the result we have obtained so far to the case of multiple paths case.

Consider the multiple paths case in Fig.3.5 where the source-destination pair $(s,d) = r$ (for class r) have several paths for routing:

$$r_1, r_2, \ldots, r_{K(s,d)}.$$

where $K(s,d)$ denotes the number of paths for j_th path for class r.Let

$a(r_j)$ = denote the set of links for j_th path for class r.

$q(r_j)$ = the probability that the j_th path is selected for class r.

The following theorem gives the Laplace transform for this multiple path model:

Theorem 3.3 *Let $r_j, \quad j = 1, 2, \ldots, K_{(s,d)}$ be one of the path for the source destination pair (s,d). Let $q(r_j)$ be the probability that p_j is chosen at s, then the Laplace transform of t_r is given by:*

$$T_r^*(s) = \sum_{j=1}^{K_{(s,d)}} q(r_j) \prod_{i \in a(r_j)} \frac{\mu C_i (1-\rho_i)}{s + \mu C_i (1-\rho_i)} \tag{3.28}$$

Proof

From (3.19), we have :

$$t_{r_j} = \sum_{i \in a(r_j)} t_i \tag{3.29}$$

Substituting (3.29) into (3.20), we get:

$$\begin{aligned}
E\{e^{-st_r} | r = r_j\} &= E\{e^{-st_{r_j}}\} \\
&= E\{e^{-\sum_{i \in a(r_j)} t_i}\} \\
&= \prod_{i \in a(r_j)} \frac{\mu C_i (1 - \rho_i)}{\mu C_i (1 - \rho_i) + s} \\
T_r^*(s) &= E\{E\{e^{st_r} | r\}\} \\
&= \sum_{j=1}^{K_r} q(r_j) E\{e^{-st_r} | r = r_j\} \\
&= \sum_{j=1}^{K_r} q(r_j) \prod_{i \in a(r_j)} \frac{\mu C_i (1 - \rho_i)}{\mu C_i (1 - \rho_i) + s}
\end{aligned} \tag{3.30}$$

Q.E.D.

Theorem 3.3 gives the multiple path delay for any (s, d) pair. Now we select one of the optimal routing algorithms(see Gallager's[40]) to show how we approximate the delay distributions for networks using optimal routing strategies.

The Gallager's algorithm[40] tries to minimize the expected delay over all links with respect to a set of routing variables $\{\phi_{ij}(k)\}$ with $i, j, k \in \{1, 2, \cdots, N\}$ which specifies the fraction of traffic leaving node i destined for node k via node j. The objective functions and the constraints for this mathematical programming

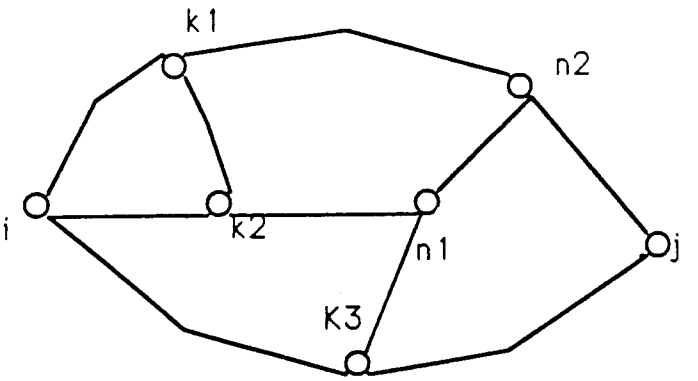


Figure 3.6: Network using Gallager's algorithms

problem are as following:

$$min[\ \sum_{i,k} D_{ik}(f_{ik})\ ] \tag{3.31}$$

constraints:

$$\begin{cases} f_{ik} = \sum_j s_i(j)\phi_{ik}(j)\,; \\ s_i(j) = t(r) + \sum_l s_l(j)\phi_{li}(j)\,; \\ \phi_{ik}(j) \geq 0, \qquad \forall i,j,k \end{cases} \tag{3.32}$$

where $t(r)$ is the traffic for source-destination pair (s,d), $s_i(j)$ is the inter-network traffic for (i,j) and f_{ik} is the flow on link (i,k). Their algorithms find the set $\{\phi_{ik}(j), \forall i,j,k\}$ by shifting the portion of traffic sent to the non-optimal links to the links that reduce the cost function. Since $\{\phi_{ik}(j), \forall i,j,k\}$ is totally specified by the algorithm, we can compute the flow for each possible paths for any source-destination pair (s,d). As an example, for a network in Fig.3.6, We can see that there are three nodes K_1, K_2, K_3 which are adjacent to node i. Taking node j as the destination node, we can enumerate all the paths starting from node i going to node j.(see Fig.3.7) Note that the tree in Fig.3.7 terminates with node j. This tree can be generated in finite time because Gallager's algorithm guarantee that no looping inside the network[40].

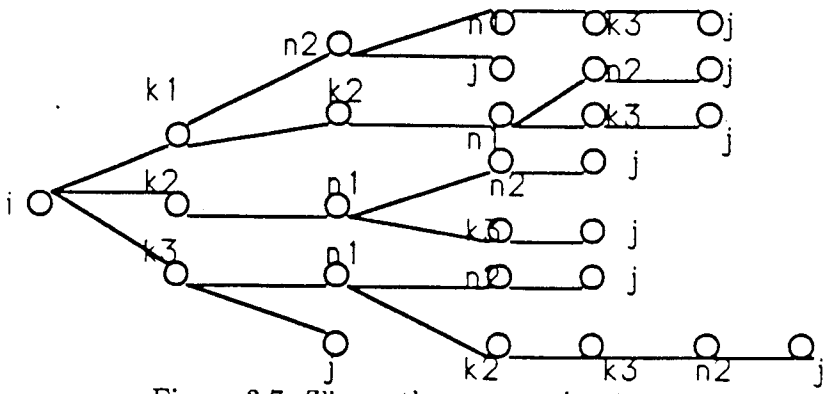


Figure 3.7: The paths enumeation tree

Now we denote the set $\{r_1, r_2, \cdots, r_M\}$ as the paths enumerated from the path tree for $r = (s, d)$. Each path r_i represents a sequence of nodes

$$(s, k_1^{(i)}, k_2^{(i)}, \cdots, k_{l_i-1}^{(i)}, d) \qquad i = 1, 2, \cdots, M$$

where l_i is the length of path r_i.

Suppose $t(r)$ is the traffic for (s, d), the portion of traffic for path r_i can be found by:

$$t(r_i) = t(r) \prod_{j=1}^{l_i} \phi_{k_{j-1}^{(i)} k_j^{(i)}}(d) \tag{3.33}$$

where we define:

$$s \triangleq k_0^{(i)} \quad \text{and } d \triangleq k_{l_i}^{(i)}$$

Let,

$$q_i(r) \triangleq \prod_{j=1}^{l_i} \phi_{k_{j-1}^{(i)} k_j^{(i)}}(d)$$

Using (3.28), we can find the Laplace transform of the delay distribution to be

the average of delay for each path in P_{r_i} for all i:

$$T_r^*(s) = \sum_{i=1}^{m} q_i(r) \prod_{j \in P_{r_i}} \frac{\mu C_j(1-\rho_j)}{s + \mu C_j(1-\rho_j)} \tag{3.34}$$

Where $q_i(r), P_{r_i}, \mu, C_j, \rho_j (j = 1, 2, \cdots, l_{r_i})$ are as specified before.

Taking the inverse transform of (3.34), one gets:

$$\begin{aligned} t_r^*(x) &= \sum_{i=1}^{m} q_i(r)(\prod_{l=1}^{M_i} \mu C_l^{(i)}(1-\rho_l^{(i)})) \times \\ &\quad \times(\sum_{j=1}^{M_i}(\prod_{\substack{k=1 \\ k\neq j}}^{M_i} \frac{exp(-\mu C_j^{(i)}(1-\rho_j^{(i)})x)}{\mu C_k^{(i)}(1-\rho_k^{(i)}) - \mu C_j^{(i)}(1-\rho_j^{(i)})}) \end{aligned} \tag{3.35}$$

Rewritting (3.35), we obtain:

$$t_r^*(x) = \sum_{i=1}^{m} \sum_{j=1}^{M_i} b_{ij} exp(-\mu C_j^{(i)}(1-\rho_j^{(i)})x) \tag{3.36}$$

Where,

$$a_i = \prod_{l=1}^{M_i} \mu C_l^{(i)}(1-\rho_l^{(i)})$$

$$b_{ij} = q_i(r) a_i (\prod_{\substack{k=1 \\ k\neq j}}^{M_i} [\mu C_k^{(i)}(1-\rho_k^{(i)}) - \mu C_j^{(i)}(1-\rho_j^{(i)})]^{-1})$$

where M_i is the number of links on the source-destination pair r.

Now, the cumulative distribution function(cdf) of t_r^* can be found :

$$F_{t_r}^*(x) = \sum_{i=1}^{m} \sum_{j=1}^{M_i} [b_{ij} exp(-\mu C_j^{(i)}(1-\rho_j^{(i)})x)] - c \tag{3.37}$$

with

$$c = \sum_{i=1}^{m} \sum_{j=1}^{M_i} b_{ij}$$

By solving

$$F^*_{t_r}(B^*_s) = 0.99$$

which is:

$$\sum_{i=1}^{m} \sum_{j=1}^{M_i} b_{ij} exp(-\mu C_j^{(i)}(1 - \rho_j^{(i)}) B^*_s) = c + 0.99 \tag{3.38}$$

This equation gives us the delay bound for ninety-nine percent of the messages.

Note that for the optimal routing strategies like [40], the non-passing conditions are violated and it seems that the approximation of delay distribution based on the assumption of Theorem 3.3 will be a poor one. However, Wong and Lam[6] have shown that the dependence on the non-passing condition is very mild. They also showed by simulation that even though the non-passing condition and Marcov routing is not satisfied, the delay distribution of (3.22) is still a good approximation.

From Theorem 3.3 and (3.38) we found the delay bound for ninety-nine percent of the messages by B_0 and B^*_0. They are the indices we want since our problem has time constrains for message sojourn times (Chapter 1). However, one may interested in finding other indices like mean and the variance of the delay mentioned before. With (3.22) and (3.28), one can easily find the following theorem:

Theorem 3.4

(1) The mean and variance for a single path model defined in Sec 3.1 are given

by:

$$\overline{t_r} = \sum_{i \in P_r} \frac{1}{\lambda_i} \frac{\rho_i}{1-\rho_i} \tag{3.39}$$

$$\sigma_{t_r}^2 = \sum_{i \in P_r} \frac{1}{\lambda_i^2} \left(\frac{\rho_i}{1-\rho_i}\right)^2 \tag{3.40}$$

(2) The mean and variance for a multiple path model defined in Sec.3.3 are given by:

$$\overline{t_r^*} = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i} \frac{\rho_i}{1-\rho_i} \tag{3.41}$$

$$\sigma_{t_r^*}^2 = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i^2} \left(\frac{\rho_i}{1-\rho_i}\right)^2 \tag{3.42}$$

Proof

By (3.20), we have:

$$T_r(s) = E\{e^{-st_r}\} \tag{3.43}$$

Differentiating (3.43) with respect to s on both sides and setting $s = 0$, we obtain:

$$\frac{\partial}{\partial s} T_r(s)|_{s=0} = E\{-t_r e^{-st_r}\}|_{s=0} = E\{-t_r\} \tag{3.44}$$

$$\frac{\partial}{\partial s^2} T_r^2(s)|_{s=0} = E\{t_r{}^2 e^{-st_r}\}|_{s=0} = E\{t_r^2\} \tag{3.45}$$

From (3.22), one finds:

$$\frac{\partial}{\partial s} T_r(s) = \sum_{i \in a(r)} \frac{-\mu C_i (1-\rho_i)}{(s + \mu C_i (1-\rho_i))^2} \tag{3.46}$$

$$\frac{\partial^2}{\partial s^2} T_r(s) = \sum_{i \in a(r)} \frac{2\mu C_i (1-\rho_i)}{(\mu C_i (1-\rho_i))^2} \tag{3.47}$$

From the equation:

$$\sigma_{t_r}^2 = E\{t_r^2\} - E^2\{t_r\}. \tag{3.48}$$

and (3.46),(3.47), and after simplifications one gets:

$$E\{t_r^*\} = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i} \frac{\rho_i}{1-\rho_i} \tag{3.49}$$

$$E\{t_r^2\} = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i^2} (\frac{\rho_i}{1-\rho_i})^2 \tag{3.50}$$

The proof for the multiple paths case is similar. *Q.E.D.*

It can be easily verified that (3.39) and (3.41) are consistent with Little's result[61]. We have now concluded the derivation of some delay bounds which can be used as indices for switching in the allocation algorithms of Chapter 4. The shortest path routing algorithms are used for simulation in Chapter 5.

CHAPTER

FOUR

COMPUTATION OF INDICES AND THE SWITCHING ALGORITHMS

In Chapter 3, the delay distribution are derived in the forms of the Laplace transform of path sojourn time. From Theorem 3.2 , we know that the delay bounds are given by solving the non-linear equation of (3.23). One then applies the numerical algorithms like Newton iteration method[62]. to compute the root of that equation. The monotonicity of the probability cumulative density function guarantees the convergence of iterations in finite time[62]. However, we give another approach for the computation of indices which is based on the fast algorithms of Fourier transform[63]. By this approach, we show that the delay bound can be derived similar to many signal processing problems using discrete Fourier transform.

The $O(n \log n)$ time complexity for FFT algorithm provides us an alternative to implement our index rules in which the computation time is crucial for practical use. In the following sections, we first outline the FFT algorithm for indices computation in Sec.4.1. After that, we consider the switching algorithms which are based on the "Multi-armed Bandit problem"[53,54,55]. Using the optimal policy for this model, we show that the switching algorithm is simply the switching to maximal-current-reward(MCR) rule. These results are given in Sec.4.2.

4.1 Computation of Idices

We first show that our result for delay distributions in Sec.3.2. which are given in the form of Laplace transform can be converted into the Fourier transform. This fact is based on the following lemma[64]:

Lemma 4.1 *Let $f(t) = 0$, for $t < 0$, We denote the Laplace transform and the Fourier transform of $f(t)$ as $\mathcal{L}[f](s)$ and $\Phi[f](\omega)$ respectively, where $s = \sigma + j\omega$. If $\mathcal{L}[f]$ is absolute convergent, i.e. $\int_0^\infty |f(t)|dt < +\infty$, then*

$$\Phi[f] = \mathcal{L}[f]|_{\sigma=0} \tag{4.1}$$

Proof *Rewrite the definition of Laplace transform, one has:*

$$\begin{aligned} \mathcal{L}[f] = \int_0^\infty f(t)e^{-st}dt &= \int_0^\infty f(t)e^{-\sigma t - j\omega t}dt \\ &= \int_0^\infty (f(t)e^{\sigma t})e^{-j\omega t}dt \\ &= \Phi[f(t)e^{-\sigma t}] \end{aligned} \tag{4.2}$$

Under the condition for absolute convergence of $\mathcal{L}[f]$ and setting $\sigma = 0$ gives the result.

Using (4.1), we may write the result of Theorem 3.1 as the Fourier transform of the delay distribution:

$$\Phi(\omega) = \prod_{i=1}^{m} \frac{\mu C_i(1-\rho_i)}{j\omega + \mu C_i(1-\rho_i)} \tag{4.3}$$

where m is the number of links in $a(r)$.

Following the proof of Theorem 3.2, we know that in order to compute the delay bound B_0, one needs to find the inverse transform of (4.3). However, instead of finding the inverse transform directly and then solving the non-linear equations, we can use the FFT algorithms to find the discrete points for inverse transforms and then compute the cumulative density function,

To do this, we first define the discrete Fourier transform:

Definition 4.1 (Discrete Fourier Transform) *Given* $\underline{x} = [x_0, x_1, \ldots, x_{n-1}] \in C^n$, *the n sample points of* $x(t)$, *the discrete Fourier transform of* $\underline{x}$ *is given by:*

$$\underline{X} \triangleq [X_0, X_1, \ldots, X_{n-1}]$$

where

$$x_i = \frac{1}{n} \sum_{k=0}^{n-1} x_k e^{j(\frac{2\pi i}{n})k} \tag{4.4}$$

and the inverse transform is defined by:

$$x_k = \sum_{i=0}^{n-1} X_i e^{-j(\frac{2\pi k}{n})i} \tag{4.5}$$

In our case, $\underline{X}$ is obtained by sampling the delay distribution of (4.3). Note that each sample point X_i is obtained by a product of m terms, the addition and multiplication inside each term of the product is assumed to cost a constant time. Hence, the product costs $O(m)$ time. In practical situations, it is always the case that $m \ll n$ since m is the number of links and is bounded by the path length of the network but n is the number of sample points which is usually much larger than m. Therefore we may well assume that the sampling step takes $O(mn) \simeq O(n)$ computation time.

Now, our problem is to compute (4.5) in time $O(n \log n)$ which is the time complexity for current FFT algorithms. We first note that (4.5) is a polynomial of w^k, if

$$w^k = e^{-j(\frac{2\pi k}{n})}$$

with coefficients $x_0, x_1, \ldots, x_{n-1}$. Furthermore, w^k is a primitive n_th root of unity which is defined by:

Definition 4.2 (Primitive n_th root of unity) *$\beta \in C$ is a primitive n_th root of unity if $\beta \neq 1$ satisfies:*

(1) $\beta^n = 1$

(2) $\sum_{k=0}^{n-1} \beta^{jk} = 0 \qquad , j = 1, 2, \ldots, n-1$

With Definition 4.2 one may easily verify that $w = e^{-j\frac{2\pi}{n}}$ is a primitive n_th root of unity. Furthermore, the following theorem which will be used to construct the recursive relations for computing x_k can be proved:

Theorem 4.1

(a) Let $n = 2^p$ and assume w is a primitive n_th root of unity then w^2 is a primitive $n/2$_th root of unity.

(b) $w^{k+\frac{n}{2}} = -w^k$.

Proof *see Appendix A.2.*

Now, we can write (4.5) as:

$$x_k = \sum_{i=0}^{n-1} X_i w^{ki}, \qquad k = 0, 1, \ldots, n-1. \tag{4.6}$$

Assume $n = 2^p$, then (4.6) can be expanded into

$$\begin{aligned} x_k &= X_0 + X_1 w^k + X_2 w^{k2} + \cdots + X_{n-1} w^{k(2^p-1)} \\ &= (X_0 + X_2 w^{k2} + X_4 w^{k4} + \cdots + X_{n-2} w^{k(2^p-2)}) + \\ &\quad +(X_1 w^k + X_3 w^{k3} + \cdots + X_{n-1} w^{k(2^p-1)}) \\ &= (X_0 + X_2 (w^2)^k + \cdots + X_{n-2} (w^2)^{k(2^{p-1}-1)}) + \\ &\quad +(X_1 + X_3 (w^2)^k + \cdots + X_{n-1} (w^2)^{k(2^{p-1}-1)}) w^k \end{aligned} \tag{4.7}$$

From Theorem 4.1(a), one knows that w^2 is a primitive $\frac{n}{2}$_th root of unity. Hence, we find that (4.7) can be divided into two sequence each one is a discrete Fourier transform with length $\frac{n}{2}$. Define

$$y_k \triangleq \sum_{i=0}^{\frac{n}{2}-1} Y_i w'^{ki} \tag{4.8}$$

$$z_k \triangleq \sum_{i=0}^{\frac{n}{2}-1} Z_i w'^{ki} \tag{4.9}$$

$$\tag{4.10}$$

where $Y_i \triangleq X_{2i}$, $Z_i \triangleq X_{2i+1}$, $i = 0, 1, \ldots, \frac{n}{2} - 1$ and (4.7) can be written as

$$x_k = j_k + z_k w^k, \qquad k = 0, 1, \ldots, \frac{n}{2} - 1. \tag{4.11}$$

Similarly, one can expand (4.5) using Theorem 4.1(b) into:

$$x_k = \sum_{i=0}^{n-1} X_i w^{ki}$$

$$
\begin{aligned}
&= \sum_{i=0}^{n-1} X_i(-w^{k-\frac{n}{2}})^i, \qquad , k = \frac{n}{2}, \frac{n}{2}+1, \ldots, n. \\
&= \sum_{i=0}^{n-1} X_1(-w^{k'})^i, \qquad , k' = 0, 1, \ldots, \frac{n}{2}-1. \\
&= X_0 - X_1 w^{k'} + X_2 w^{k'2} - \cdots - X_{n-1} w^{k(2^p-1)} \\
&= [X_0 + X_2 w^{k'2} + \cdots + X_{n-2} w^{k'(2^p-2)}] \\
&\quad -[X_1 + X_3 w^{k'3} + \cdots + X_{n-1} w^{k'(2^p-1)}] \\
&= [X_0 + X_2 (w^2)^{k'} + \cdots + X_{n-2} w^{2k'(2^{p-1}-1)}] \\
&\quad -[X_1 + X_3 (w^2)^{k'} + \cdots + X_{n-1} w^{2k'(2^{p-1}-1)}] w^k \\
&= y_k - z_k w^k \qquad , k = \frac{n}{2}, \ldots, n-1 \qquad (4.12)
\end{aligned}
$$

Combining (4.11) and (4.12), we can decompose the problem of finding the inverse discrete Fourier transform of size n into two subproblems, each with size $\frac{n}{2}$. Solving these two equations recursively gives us the points$[x_0, x_1, \ldots, x_{n-1}]$.

The recursive algorithms based on (4.11) and (4.12) are listed below which is due to Horowitze[65]:

Algorithm IDFT($\underline{X}, w, n$);

begin

if $n = 1$ then $x_0 \leftarrow X_0$;

else

begin

$\underline{Y} \leftarrow [X_0, X_2, \ldots, X_{n-2}]$;

$\underline{Z} \leftarrow [X_1, X_3, \ldots, X_{n-1}]$;

$\underline{y} \leftarrow \text{IDFT}(\underline{Y}, w^2, n/2))$;

$\underline{z} \leftarrow \text{IDFT}(\underline{Z}, w^2, n/2))$;

$x_k \leftarrow y_k + z_k w^k, \qquad k = 0, 1, \ldots, 2^{p-1}$;

$x_k \leftarrow y_k - z_k w^k, \qquad k = 2^{p-1}, \ldots, 2^p$;

end;

end;

Let $T(n)$ denote the time complexity of IDFT with n sample points $X_0, X_1, \ldots, X_{n-1}$, then $T(n)$ can be computed by the following recursive equation:

$$\begin{aligned} T(n) = T(2^p) &= 2T(2^{p-1}) + cn \\ &= 2[2T(2^{p-2}) + c\frac{n}{2}] + cn \\ &\vdots \\ &= 2^p T(1) + \underbrace{cn + \cdots + cn}_{p} \\ &= nT(1) + cnp = c'n + cn \log n \end{aligned} \tag{4.13}$$

After we have obtained $[x_0, x_1, \ldots, x_{n-1}]$, the cumulative density function of t_r can be written as:

$$F_{tr}(k) = \sum_{i=0}^{k} x_i \tag{4.14}$$

Setting $F_k = 0.99$ and finding the index which achieves this value gives the delay bound B_0. The overall computation time for our indices computation is seen to

be the sum of sampling times plus the computation time for IDFT algorithm and finally the time for computing (4.14):

$$O(mn) + O(n \log n) + O(n) \tag{4.15}$$

where

$$\frac{O(f(n))}{f(n)} \rightarrow c, c \in \Re$$

If $m \ll n$, then (4.15) simplifies to $O(n \log n)$, we thus concluded that we have found a procedure for computing the delay bound for a path delay defined in previous chapters with time complexity $O(n \log n)$, This result make it appealing for practical implementation. We conclude this section with the following theorem:

Theorem 4.2 *The delay bound for a single path with length m defined in Chapter 3 can be computed in time $O(n \log n)$ under the assumption : $m \ll n$, and using the IDFT algorithms and the sampling procedure which gives spectrum points $[X_0, \ldots, X_1]$ in time $O(mn)$.*

4.2 The Switching Algorithms

In previous sections, the delay bound for the 99% messages are derived and an $O(n \log n)$ procedure using IDFT Algorithm for the the inverse Fourier Transform have been proposed. The bounds computed for each subnetwork can be look as a "penalty" or "cost" for operating the subnetworks. It is non-decreasing with

respect to the number of messages sent to the network per unit time[1,3]. In fact, one can approximate this sequence by:

$$B_n^i = B_0^i + \frac{bn}{a-n} \qquad n = 0, 1, \cdots \tag{4.16}$$

where n is the number of messages(of the same class) sent to the network per unit time, a, b are constants. This follows from the delay characteristics of an M/M/1 queue[1,3]. This sequence tells us that as more and more messages sent to the network, the 99% bounds shift to a higher value and increase without bound as $n \rightarrow \infty$.

Now, we define the reward sequence based on the above bound sequence:

Definition 4.3 (Reward sequence) *A reward sequence $\{X_n^i, n \geq 0\}$ for network i is a sequence of non-negative numbers where each X_n^i is defined by:*

$$X_n^i = [L - B_n^i]^+ \tag{4.17}$$

where $[a]^+ = \max(a, 0)$, $\forall a \in \Re$, L is the time-bound specified by users. B_n^i is the bound sequence given in (4.16).

For our problem, we have several networks, each network is characterized by a reward sequence $\{X_n^i, n \geq 0\}$ defined above. Messages arrive at an input queue according to Poisson distribution with rate λ. At any instant t (which is an integer), the input queue is either empty or more than one customers are waiting for service(Fig.4.1).

If the input queue is empty the server enters "reset state" and the reward sequence is re-evaluated according to (4.16) and (4.17). If there are more than one customers in the waiting room the server enters "busy state", in this state the

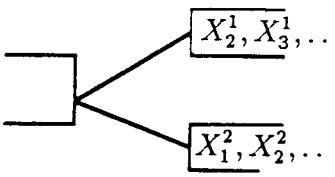


Figure 4.1: (a) the "busy" state

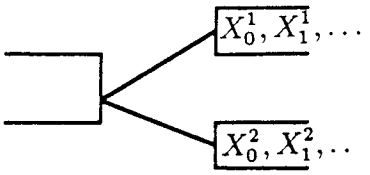


Figure 4.1:(b) the "reset" state

server takes one message for every time unit and allocates it to one of the subnetworks for transmission(That is like one arm of the bandit has been pulled.). Now, we formulate the switching problem as finding the sequence of the subnetworks to assign those waiting customers such that the total discounted reward is maximized:

Formally, we want:

$$\max[\ \sum_{t=1}^{\infty} \alpha^t r(t)\ ] \tag{4.18}$$

where $\alpha \in (0,1)$ is the discount factor and $r(t)$ is the reward by operating one of the networks.

We assume that $\sum_{t=0}^{\infty} \alpha^t X_{n_t}^i < +\infty, \quad \forall i = 1,2,..,M$ where n_t denotes the n_t th times the network has been operated (at time t).

This problem has been known as the "Multiarmed bandit problem"[54],[56]. The optimal policy for this problem has been shown to be one that operates the arm with the largest index[56]. This is called the index rule and the index was shown in [55], [56] to be:

$$r_t^i = sup_{\tau \geq t}[\ \frac{\sum_{m=t}^{\tau-1} \alpha^m X_m^i}{\sum_{m=t}^{\tau-1} \alpha^m}\ ] \tag{4.19}$$

However, the calculation of this index takes considerable computation time. We can simplify this calculation by the properties of monotonicity of our reward sequences.

From (4.16), it is easy to verify that:

$$B_m^i \geq B_n^i, \qquad \forall m \geq n \tag{4.20}$$

Since B_n^i is a random variable, we formally define this monotonicity as follows:

Definition 4.4 (Non-increasing random reward) *$\{X_n^i, n \geq 0\}$ is a reward sequence, it is non-increasing if*

$$P\{X_{n+1}^i \leq X_n^i, n \geq 0, 1 \leq i < \infty\} = 1 \tag{4.21}$$

The following theorem gives us the index rule which utilizes the non-increasing properties of our reward sequence so that one only needs to send the message to the subnetwork with maximal current index. The proof is based on the interchanging argument:

Theorem 4.3 (Index rules for message assignment) *If $\{X_n^i\}$ is a non-increasing sequence defined above,*
Let X_t^i denote one of the $\{X_n^i, n = 0, 1, \ldots\}$

$$\delta(t) \triangleq arg[\max_{i=1..M}\{X_{n_t}^i\}] \tag{4.22}$$

where t is the current time, then the optimal policy is to operate $\delta(t)$ at time t.

Proof

Let $C(i_1, i_2, .., i_N), \forall N$ denote the cost of the objective function:

$$\sum_{t=1}^{\infty} \alpha^t r(t)$$

where $(i_1, i_2, .., i_N)$ is the sequence for operating networks $\{X_n^{i_t}, \forall n = 0, 1, ..N\}$ at

time $t = 1, 2, .., N$ *Assume there is some* s *such that*

$$Pr\{X_{n_s}^{i_s} \leq X_{n_{s+1}}^{i_{s+1}}, n_{s+1} > n_s\} = 1 \tag{4.23}$$

thus,

$$\begin{aligned} & R(i_1, i_2, \cdots, i_s, i_{s+1}, \cdots, i_N) - R(i_1, i_2, \cdot, i_{s+1}, i_s, \cdots, i_N) \\ = & \sum_{i=1}^{\infty} \alpha^n r_1(t) - \sum_{t=1}^{\infty} \alpha^n r_2(t) \end{aligned} \tag{4.24}$$

where $r_1(t), r_2(t)$ *are given by:*

$$r_1(t) = \begin{cases} r_2(t), & \forall t \neq s, s+1; \\ \alpha^s X_{n_s}^{i_s}, & t = s; \\ \alpha^{s+1} X_{n_{s+1}}^{i_{s+1}}, & t = s+1. \end{cases} \tag{4.25}$$

$$r_2(t) = \begin{cases} r_1(t), & \forall t \neq s, s+1; \\ \alpha^s X_{n_{s+1}}^{i_{s+1}}, & t = s; \\ \alpha^{s+1} X_{n_s}^{i_s}, & t = s+1. \end{cases} \tag{4.26}$$

Substitute (4.25),(4.26) into (4.24), we have:

$$\begin{aligned} & \alpha^s X_{n_s}^{i_s} + \alpha^{s+1} X_{n_{s+1}}^{i_{s+1}} - \alpha^s X_{n_{s+1}}^{i_{s+1}} - \alpha^{s+1} X_{n_s}^{i_s} \\ = & \alpha^s (X_{n_s}^{i_s} - X_{n_{s+1}}^{i_{s+1}}) + \alpha^{s+1} (X_{n_{s+1}}^{i_{s+1}} - X_{n_s}^{i_s}) \\ = & \alpha^s (1 - \alpha)(X_{n_s}^{i_s} - X_{n_{s+1}}^{i_{s+1}}) \end{aligned} \tag{4.27}$$

Combining (4.27) and (4.23) we get:

$$Pr\{(R(i_1, \cdots, i_s, i_{s+1}, \cdots, i_N) - R(i_1, \cdots, i_{s+1}, i_s, \cdots, i_N)) \leq 0\} = 1 \tag{4.28}$$

Which tells us that there exists another sequence $(i_1, \cdots, i_{s+1}, i_s, \cdots, i_N)$ *which gets more reward by interchanging two arguments of the sequence of operation. Therefore, the sequence* $(i_1, \cdots, i_s, i_{s+1}, \cdots, i_N)$ *is not optimal. The optimal policy is to operate the networks with index* $\delta(n)$ *which chooses the network with the maximal current reward.* *QED*

This theorem gives us a simple index rule for message routing. Since the number of subnetworks are usually small. We do not consider the time complexity of this switching algorithm here. A linear search algorithm for the maximal index suffices for this problem.

CHAPTER
FIVE
SIMULATION RESULTS

In previous chapters, we have shown that one can approximate the delay distribution by the network model presented in Chapter 2. To control the multiple subnetwork model(see Sec.2.3), we only need the utilization factors for each communication links provided by each subnetwork(under the assumption that the arrival processes are stationary and the channel capacities are given parameters.). The dynamic switching algorithms we derived are based on the "Gittins index"(with objective function which minimizes the discounted cost). The indices can be implemented with computation time $O(n \log n)$ (see Sec.4.1) which makes it practical for applications in real systems.

In this chapter, we simulate a simple two-media network and demonstrate our simulation results for our switching algorithms.

5.1 The Simulation Scenario

In order to test our switching algorithms, we simulate a simple network given in Fig.5.1(a). In this network, there are two subnetworks, each one uses a different medium for transmission. The first subnetwork(Fig.5.1(b)) uses medium A

interarrival time(mean)	service time mean		node proc. time	simulation time	data files
	medium A	medium B			
0.5 sec	0.1 sec	0.2 sec	0.05 sec	2050 sec	e11,e12 e13,e14
0.4 sec	0.1 sec	0.2 sec	0.05 sec	1640 sec	e21,e22 e23,e24
0.37 sec	0.1 sec	0.2 sec	0.05 sec	1537.5 sec	e41,e42 e43,e44
0.3 sec	0.1 sec	0.2 sec	0.05 sec	1230 sec	e31,e33,e34

Table 5.1: data set I

interarrival time(mean)	service time mean		node proc. time	simulation time	data files
	medium A	medium B			
0.5 sec	0.12 sec	0.15 sec	0.01 sec	2050 sec	e51,e52 e53,e54
0.4 sec	0.12 sec	0.15 sec	0.01 sec	1640 sec	e61,e62 e63,e64
0.3 sec	0.12 sec	0.15 sec	0.01 sec	1230 sec	e71,e73,e74

Table 5.2: data set II

which may be considered as a "terrestrial" network mentioned in Chapter 1. The second subnetwork uses medium B for transmission which may be considered as the satellite channel(also see Chapter 1).

The performance of this network for our switching strategies are evaluated for three set of testing data. Each set of these data consists of the network parameters for interarrival time, service time and processing time for each node. The three data sets we used for simulation are listed in Table 5.1–Table.5.3.

For each data set, we use four routing algorithms, denoted by A1,A2,A3 and A4(see Table 5.4), where A1 and A2 uses conventional shortest path routing algorithms for subnet 1 and subnet 2 respectively while A3 and A4 uses index rules

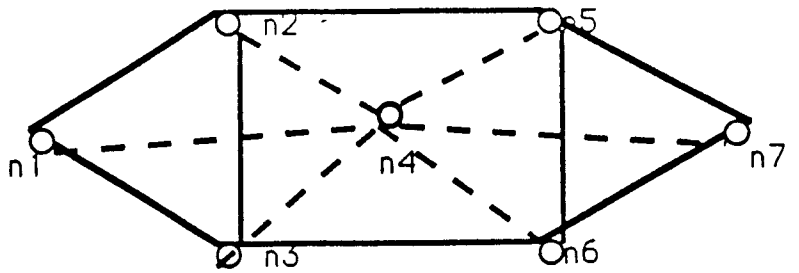


Figure 5.1(a) The overall network

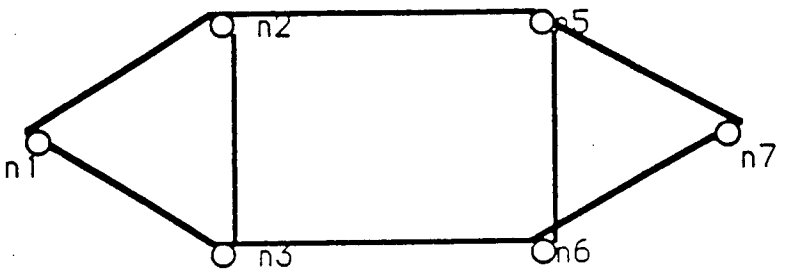


Figure 5.1(b) Subnet 1 (Medium A)

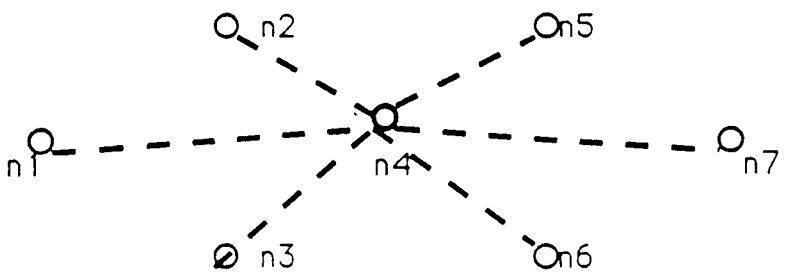


Figure 5.1(c) Subnet 2 (Medium B)

interarrival time(mean)	service time (mean)		node proc. time	simulation time	data files
	medium A	medium B			
0.5 sec	0.12 sec	0.17 sec	0.05 sec	2050 sec	e11,e12 e13,e14
0.4 sec	0.12 sec	0.17 sec	0.05 sec	1640 sec	e21,e22 e23,e24
0.35 sec	0.12 sec	0.17 sec	0.05 sec	1230 sec	e51,e53,e54

Table 5.3: data set III

Algorithms	Descriptions
A1	Shortest path algorithms for subnet 1
A2	Shortest path algorithms for subnet 2
A3	Index switching(using expected delay)
A4	Index switching(using delay bound)

Table 5.4: The switching algorithms for simulation

for switching. The indices we used are:

- the expected delay
- the 99% delay bound

The performance of algorithm A3 and A4 are compared with the performance of A1 and A2. For algorithm A1 and A2, we assume that the two subnetworks have their own routing rules which are static, centralized Bellman-Ford shortest path algorithms[4]. We also assume that the shortest paths they select initially are the shortest paths during the whole simulation process. That is, The traffic is static and the testing messages we sent into the network do not change the load of the network to a significant level so that the shortest path will change to another one. Under this assumption, we found two shortest paths for the two subnetwork by algorithm A1 and A2 respectively. For subnet 1, the shortest path found by A1 for the source destination pair (n1,n7) is the path from n1 to n2, n2 to n3, n3

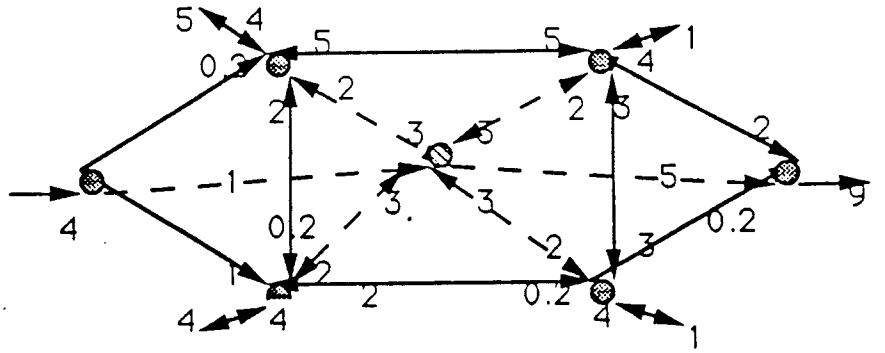


Figure 5.2: the "relative" flow for each link

to n6 and n6 to n7(see Fig.5.2). For subnet 2, the shortest path is the only path from n1 to n7(via n4).

The traffic for the network of Fig.5.1(a) are given in Fig.5.2, the number associated with each link denotes the relative flow for that link, for example the arrival rate at link (n2,n5) is computed by:

$$(arrival\_rate\_for\_n2) \times (\frac{5}{5+3+0.2})$$

where the $arrival\_rate\_for\_n2$ is computed from the flow conservation equation (see Sec.3.1).

In the next two sections, simulation results for algorithms A3 and A4 are presented where we compare the performance of A1, A2 and A3 under the simulation scenario described in this section in Sec.5.2. and use the same data to simulate A4(in Sec.5.3) in order to compare its performance with A1,A2 and A3.

5.2 Simulations for the Expected Delay Index Rules

Using the switching algorithms we proposed in Chapter 4, we implemented it with the expected delay indices on the processing node n1 of Fig.5.1 which

is the interface for subnet 1(medium A) and subnet 2(medium B). The shortest path algorithms have been implemented for subnet 1 which choose the path $1- > 2- > 3- > 6- > 7$ as described in Sec.5.1.

We simulate A1, A2 and A3 with data set I, II, III for each one of them. The delay distributions obtained for our testing messages sent to the network for data set I, II and III are given in Fig.5.3, Fig.5.4 and Fig.5.5 respectively. The solid lines in these figures are the histograms (or distributions) for the sojourn time of testing messages whereas the dotted lines are the delay histograms for the shortest paths algorithms.

The source codes in QNAP2 and the statistics generated by the "simulation solvers" of QNAP2[67] are listed in the Appendix.

From Fig.5.3, we found that the expected delay index used by algorithm A3 reduces the average delay for most cases. However, the "tail" of the delay distribution may be greater than those using the shortest paths algorithms A1, A2(see Fig.5.3(a1), Fig.5.4(c1), for example).

If the second subnet has a capacity approximately equal to to subnet 1 (see Table 5.2, for example), then one finds that the delay distributions for the shortest paths algorithms have a smaller average delay than those using indices for switching. This is true especially for light traffic conditions(see Fig.5.4(a1), Fig.5.4(a2)). The reason for this is that subnet 2 only uses two links to transmit data from n1 to n7 and has thus less chances of getting congested with other messages from nodes n2, n3, n5 and n6.

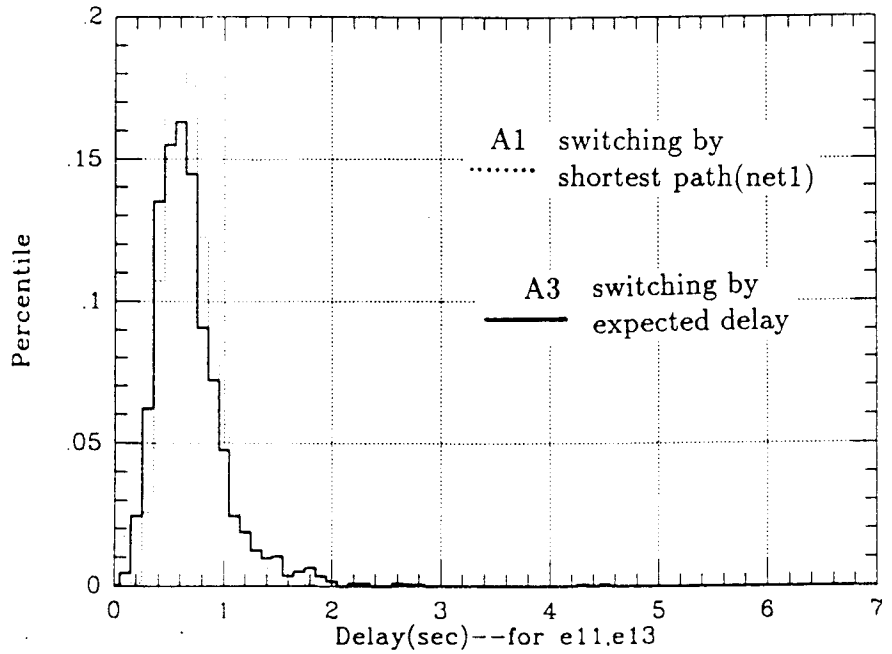


Figure 5.3(a1) Delay distributions for algorithms A1,A3 with data set I(i.a. time: 0.5 sec).

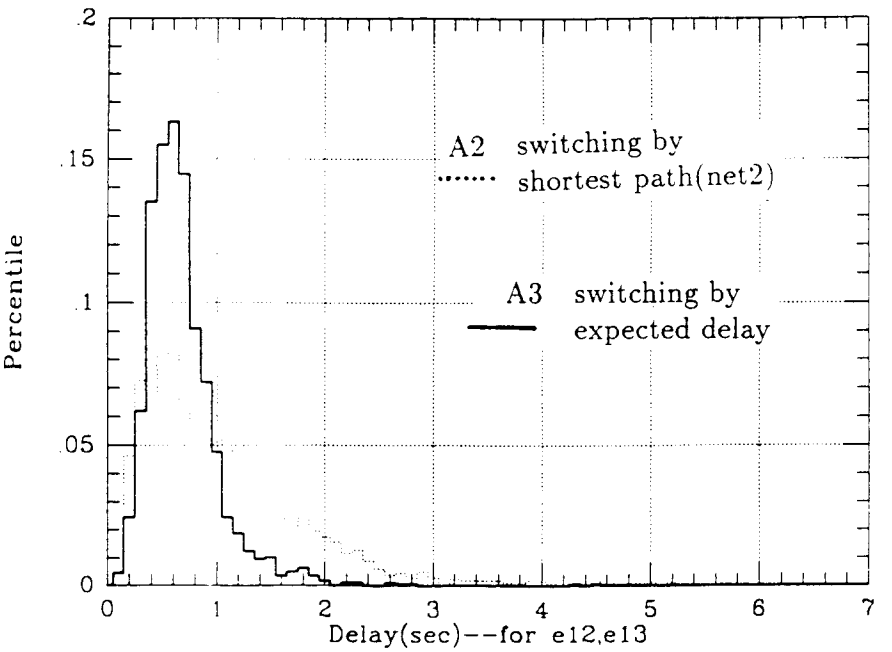


Figure 5.3(a2) Delay distributions for algorithms A2,A3 with data set I(i.a. time: 0.5 sec).

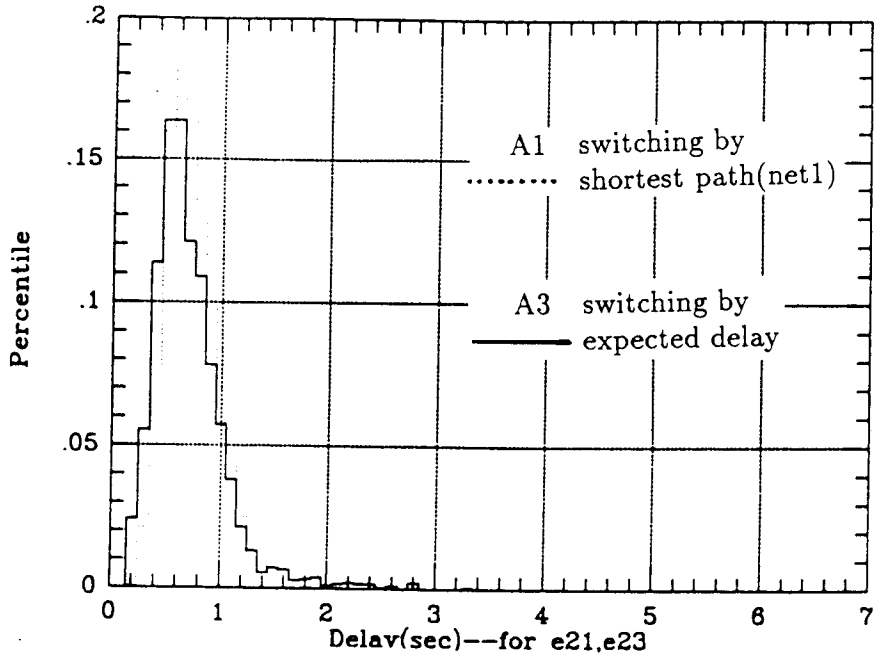


Figure 5.3(b1) Delay distributions for algorithms A1,A3 with data set I(i.a. time: 0.4 sec).

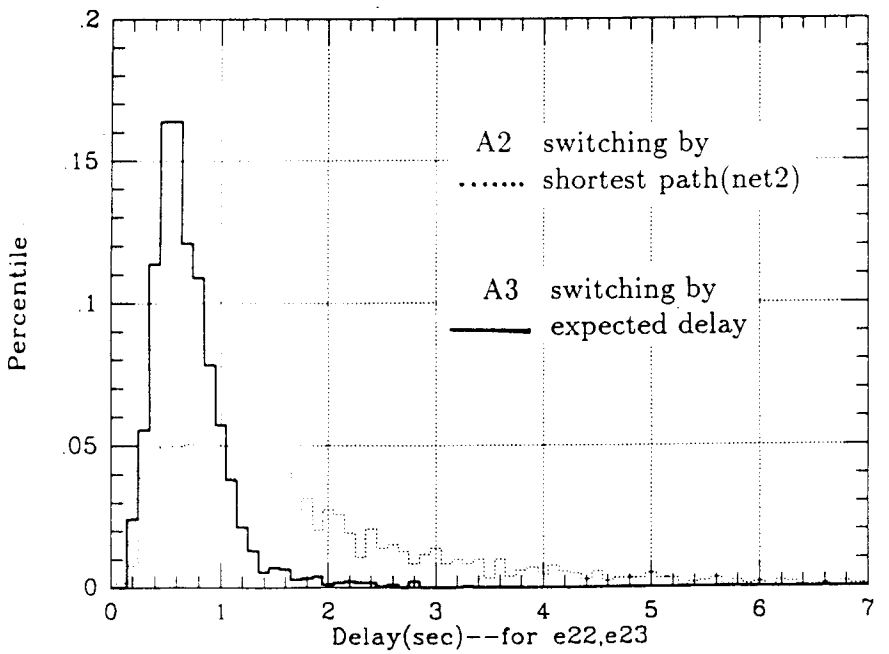


Figure 5.3(b2) Delay distributions for algorithms A2,A3 with data set I (i.a. time: 0.4 sec).

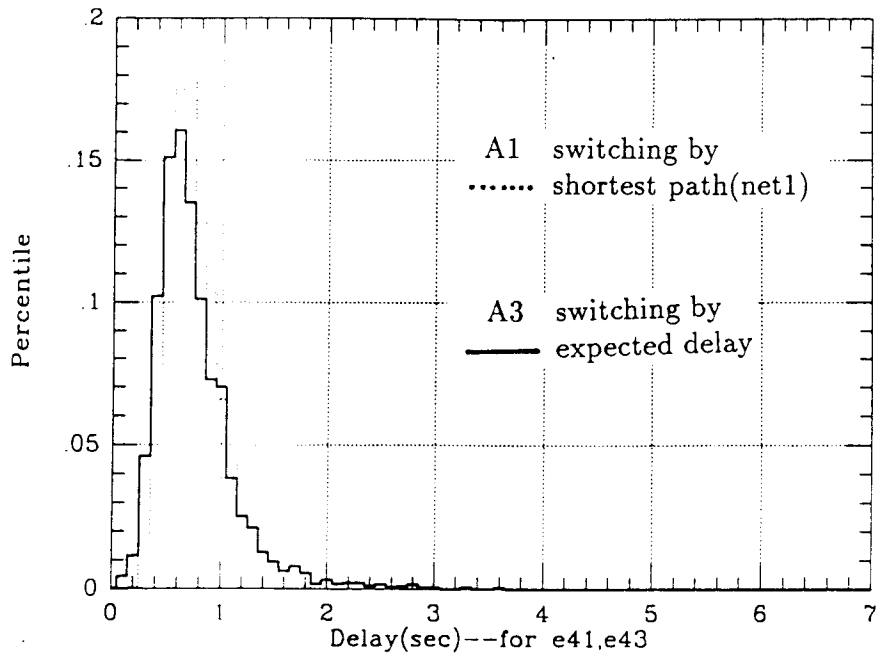


Figure 5.3(c1) Delay distributions for algorithms A1,A3 with data set I (i.a. time: 0.37 sec).

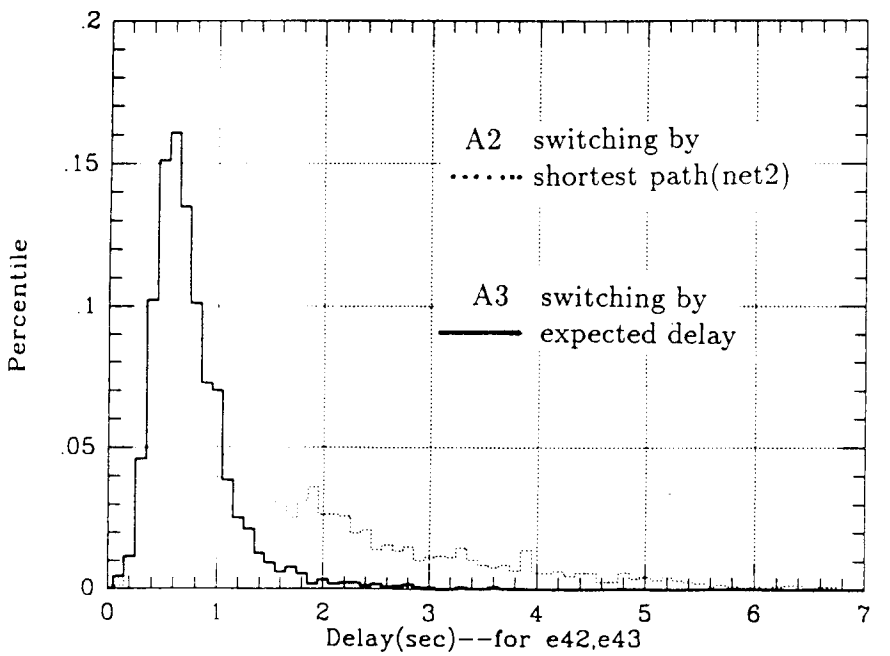


Figure 5.3(c2) Delay distributions for algorithms A2,A3 with data set I (i.a. time: 0.37 sec).

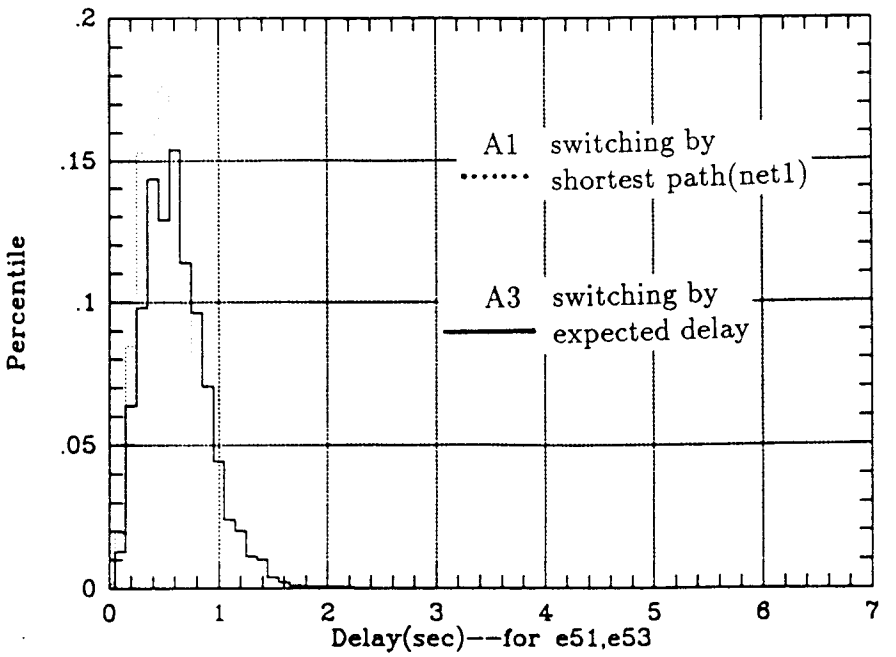


Figure 5.4(a1) Delay distributions for algorithms A1,A3 with data set II (i.a. time: 0.5 sec).

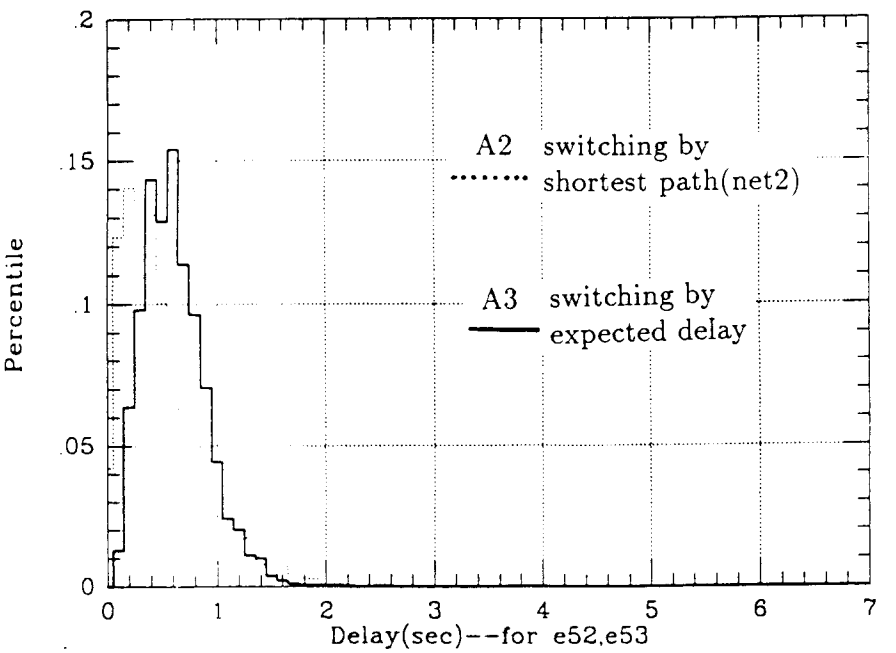


Figure 5.4(a2) Delay distributions for algorithms A2,A3 with data set II (i.a. time: 0.5 sec).

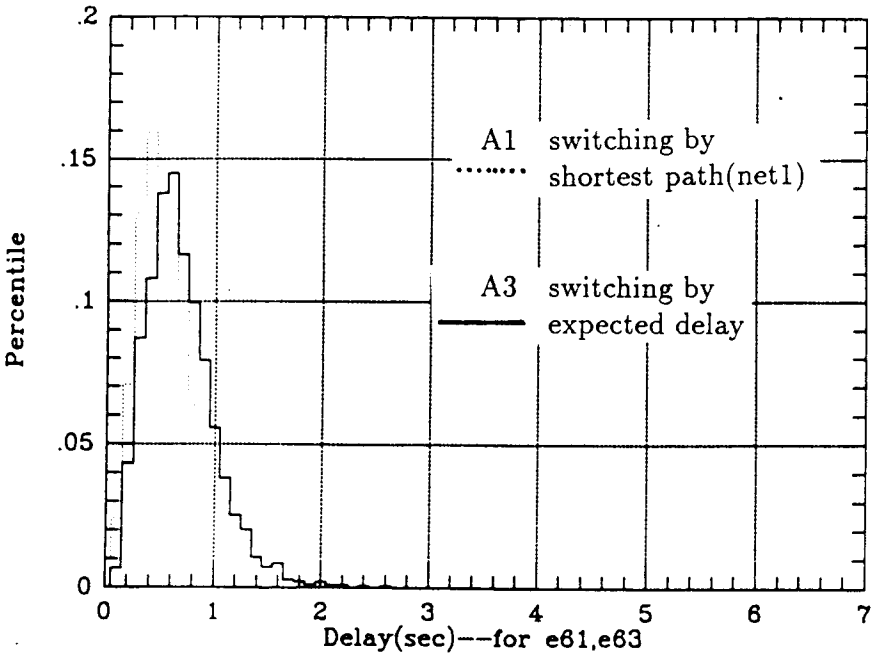


Figure 5.4(b1) Delay distributions for algorithms A1,A3 with data set II (i.a. time: 0.4 sec).

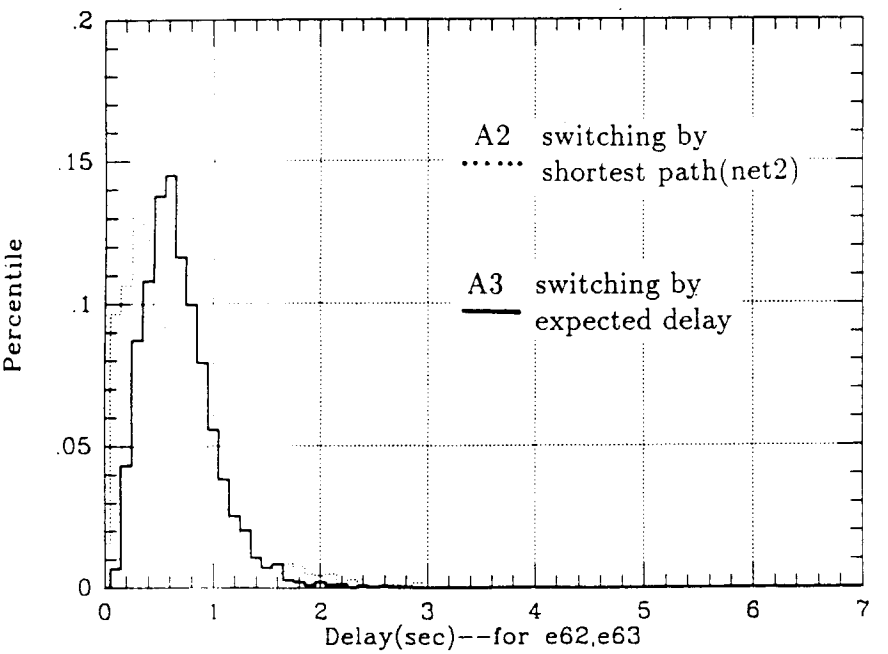


Figure 5.4(b2) Delay distributions for algorithms A2,A3 with data set II (i.a. time: 0.4 sec).

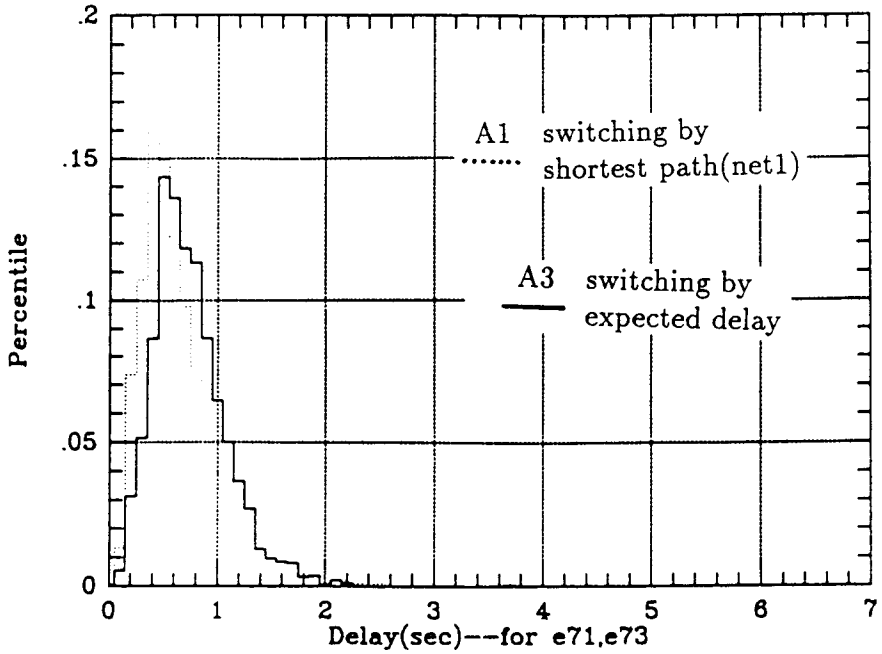


Figure 5.4(c1) Delay distributions for algorithms A1,A3 with data set II (i.a. time: 0.3 sec).

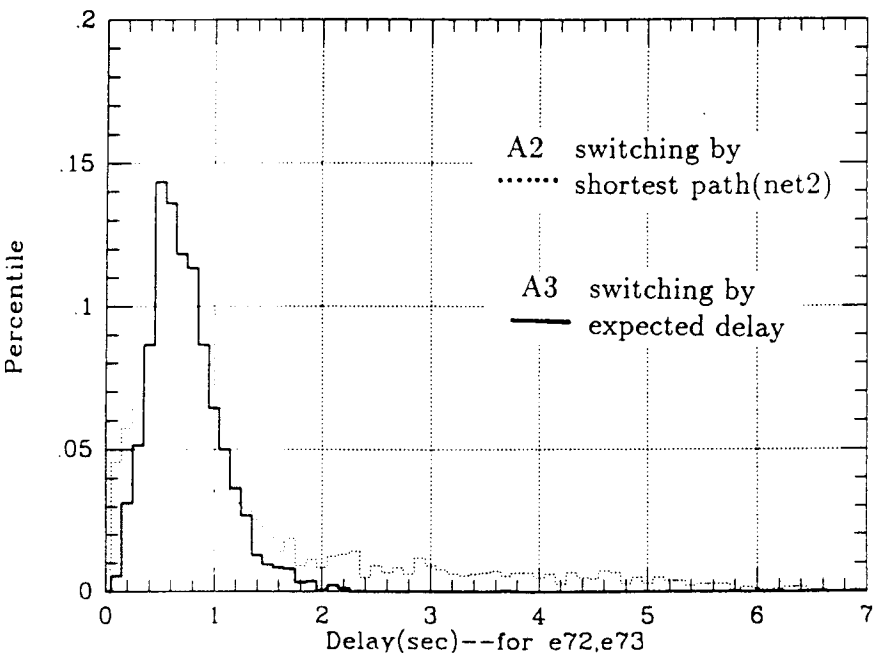


Figure 5.4(c2) Delay distributions for algorithms A2,A3 with data set II (i.a. time: 0.3 sec).

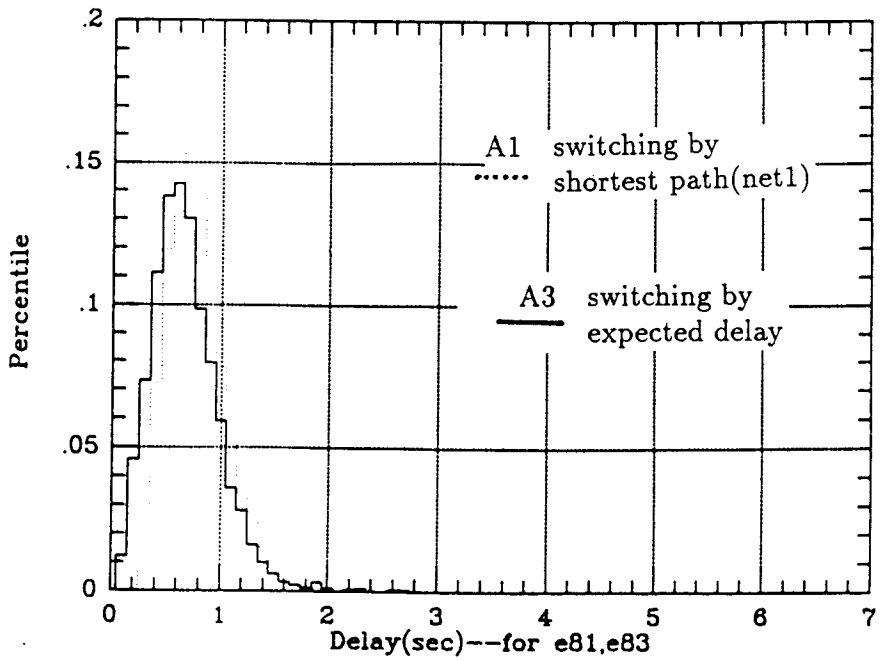


Figure 5.5(a1) Delay distributions for algorithms A1,A3 with data set III (i.a. time: 0.5 sec).

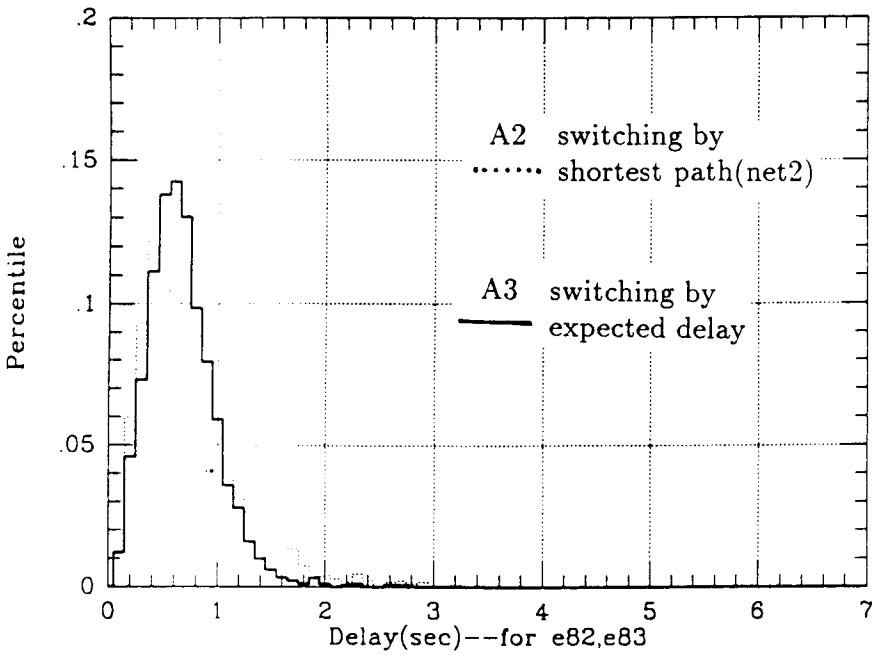


Figure 5.5(a2) Delay distributions for algorithms A2,A3 with data set III (i.a. time: 0.5 sec).

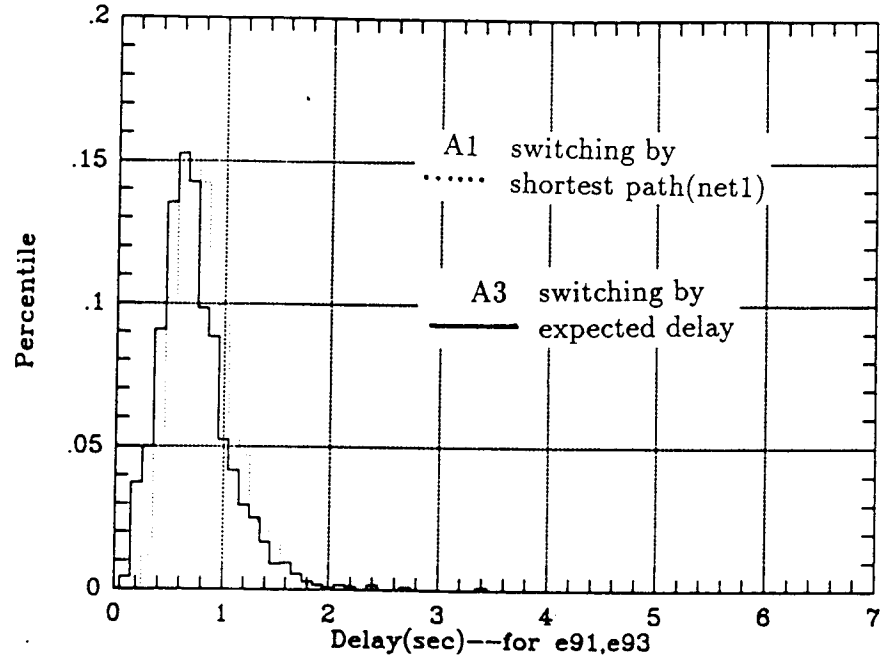


Figure 5.5(b1) Delay distributions for algorithms A1,A3 with data set III (i.a. time: 0.4 sec).

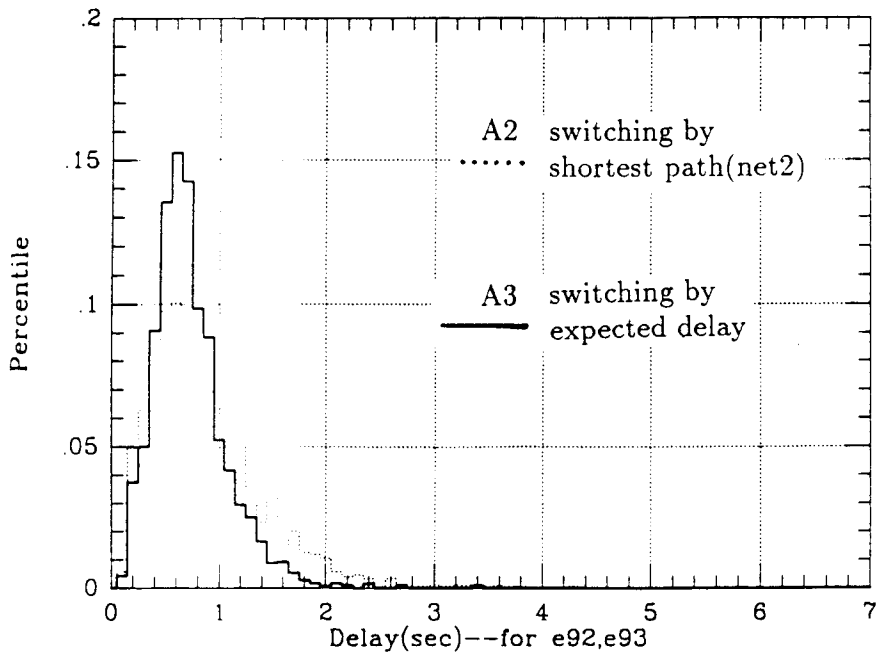


Figure 5.5(b2) Delay distributions for algorithms A2,A3 with data set III (i.a. time: 0.4 sec).

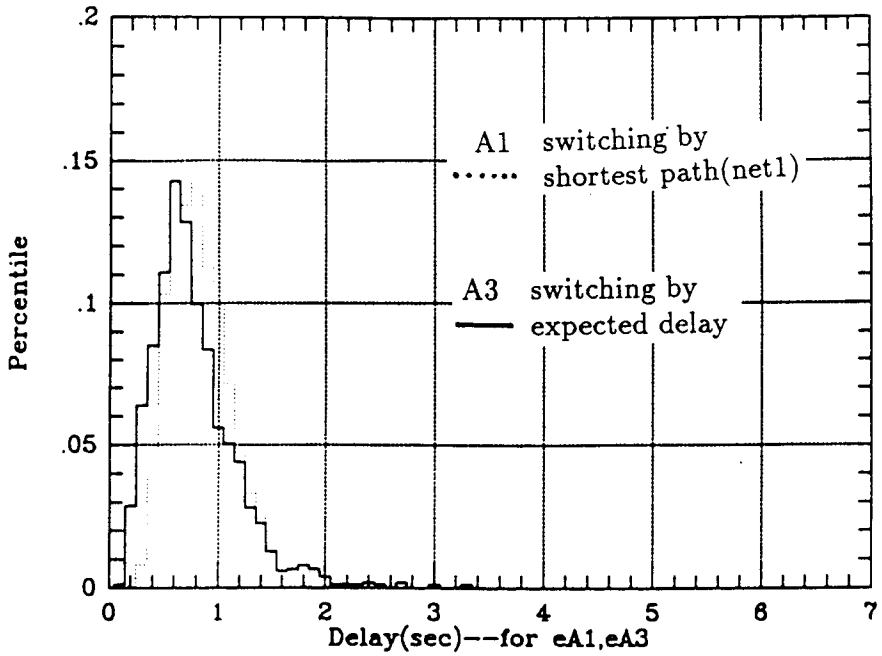


Figure 5.5(c1) Delay distributions for algorithms A1,A3 with data set III (i.a. time: 0.35 sec).

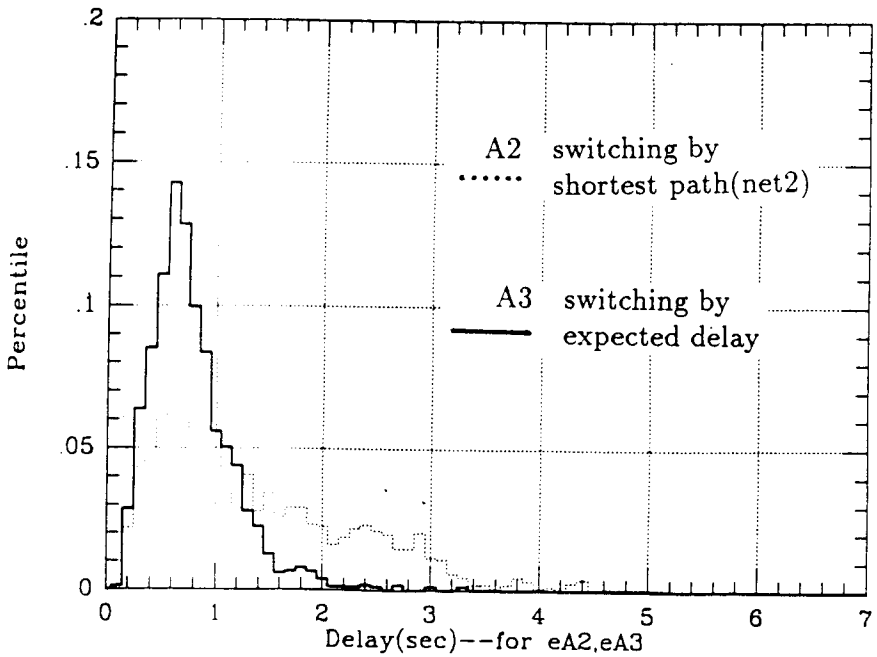


Figure 5.5(c2) Delay distributions for algorithms A2,A3 with data set III (i.a. time: 0.35 sec).

The disadvantages for algorithms A3 can also be found when the capacities for the two subnets are approximately equal and the processing time is small(see Figure 5.4). One finds that the best strategies for routing is to choose one of the shortest path from the two subnets since the delay distribution doest not give better performance in this case. They may have a large delay than the shortest path algorithms do and may even have a greater "tail".

We concluded that in order to have a better delay performance for either the expected delay or for the behavior of the "tail", we cannot solely rely on the index using expected delay. In next section, index for the delay bound are used for this purpose. The simulation results shows that they overcome most problems we encountered by using the expected delay as an index.

5.3 Simulations for Delay Bound Index Rule

From the simulation results of Sec.5.2, we find that algorithms using the expected delay as an index for switching may not suffice the problem for messages with critical time constraints. Therefore, we are interested in using the index we derived in Chapter 3 which are concerned with the critical time constraints.

We simulate algorithms A4 which uses this index for switching for data set I, II and III. The delay distribution we got for these three data sets are given in Fig.5.6, Fig.5.7 and Fig.5.8 where the delay distribution for A4 (the solid lines) are compared with the distributions obtained from last section(the dotted lines). We only compare A4 with A1 and A3 since the delay performance for A2 is "worse" than A1 and A3 for most cases in Sec.5.2. The source codes for this algorithms are also listed in the Appendix.

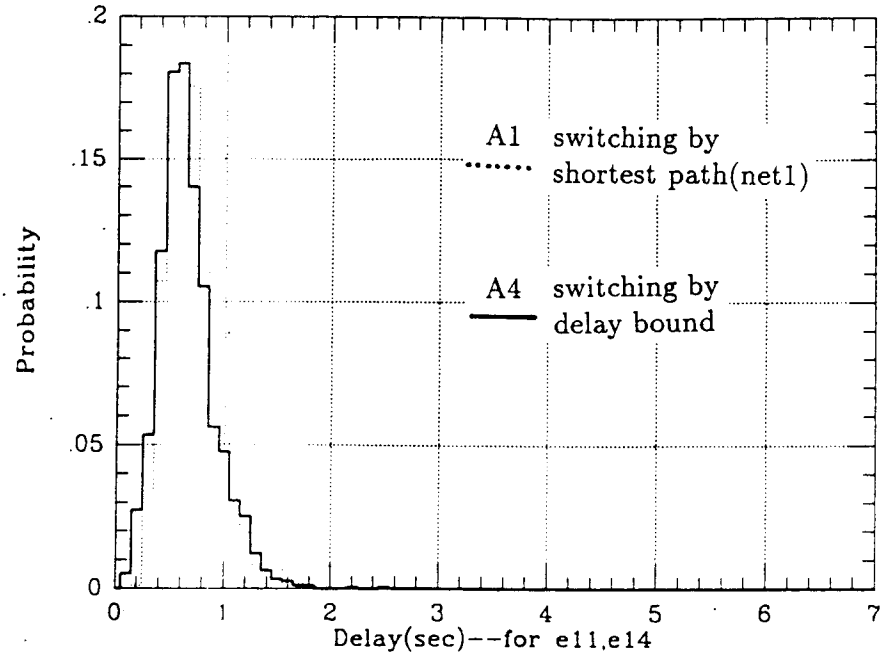


Figure 5.6(a1) Delay distributions for algorithms A1,A4 with data set I(i.a. time: 0.5 sec).

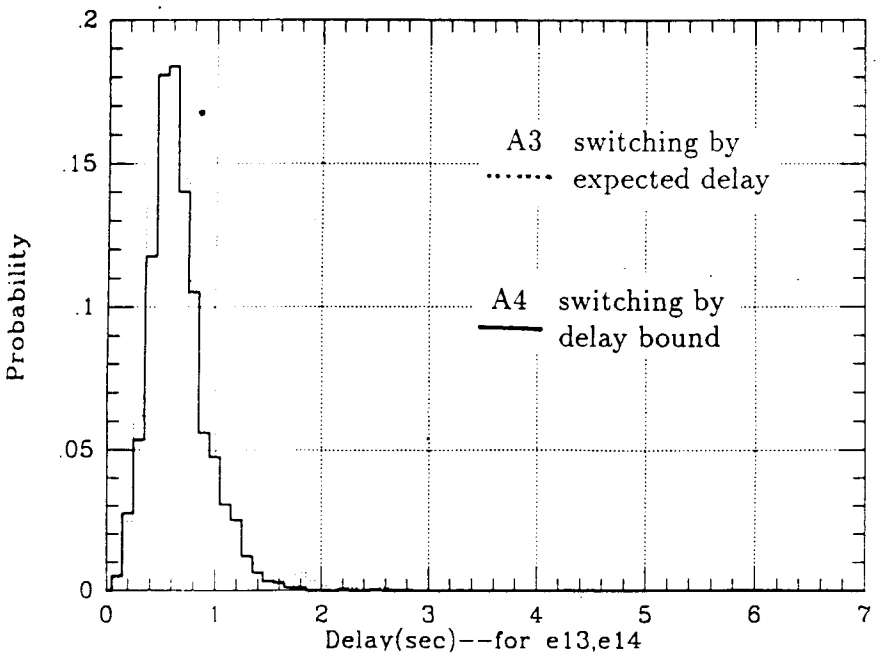


Figure 5.6(a2) Delay distributions for algorithms A3,A4 with data set I(i.a. time: 0.5 sec).

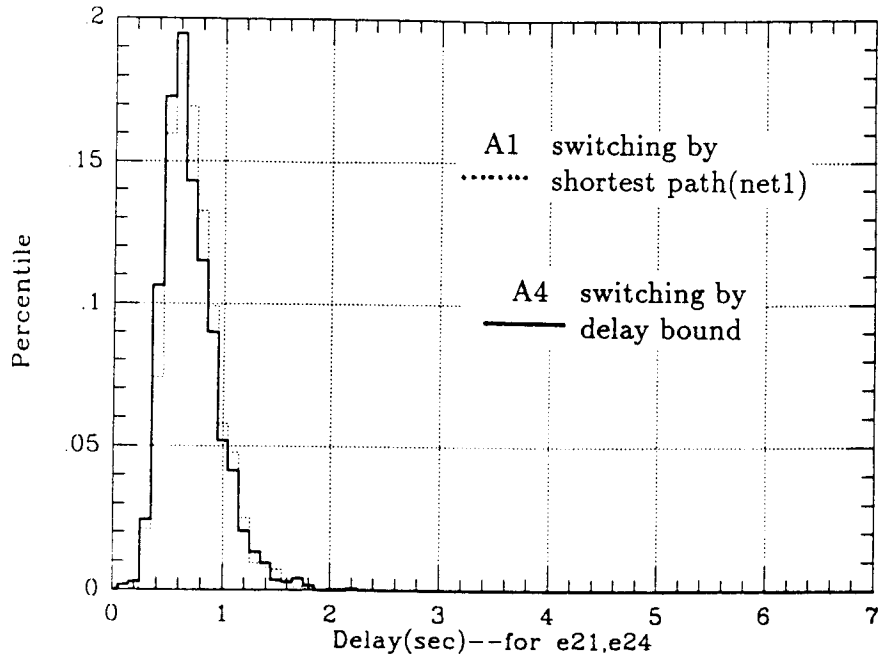


Figure 5.6(b1) Delay distributions for algorithms A1,A4 with data set I(i.a. time: 0.4 sec).

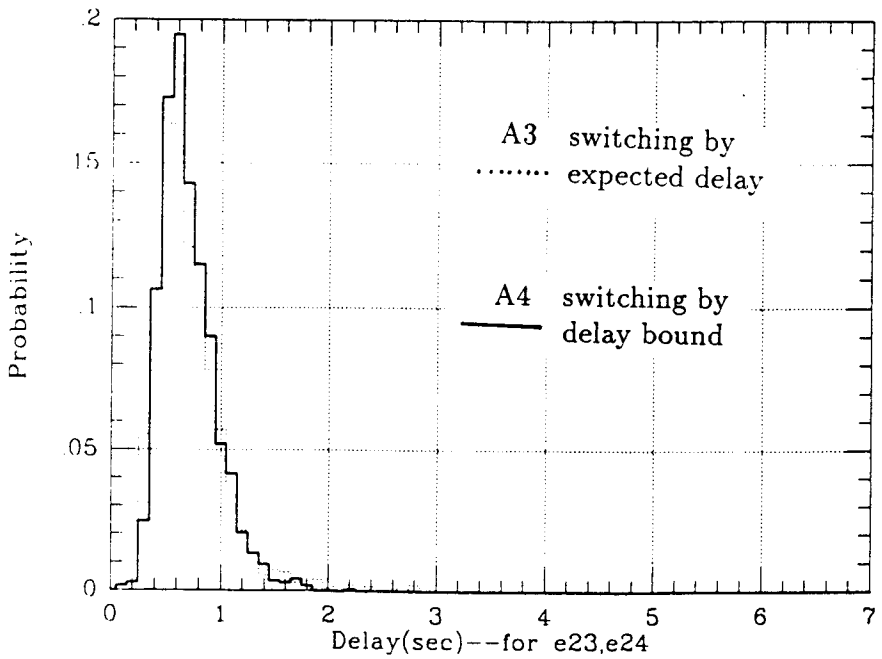


Figure 5.6(b2) Delay distributions for algorithms A3,A4 with data set I (i.a. time: 0.4 sec).

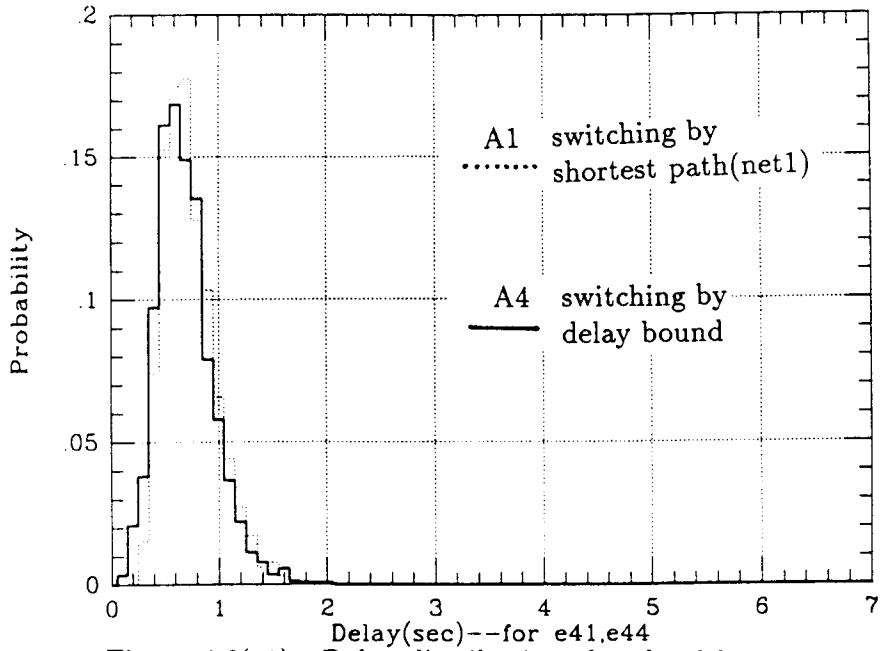


Figure 5.6(c1) Delay distributions for algorithms A1,A4 with data set I (i.a. time: 0.37 sec).

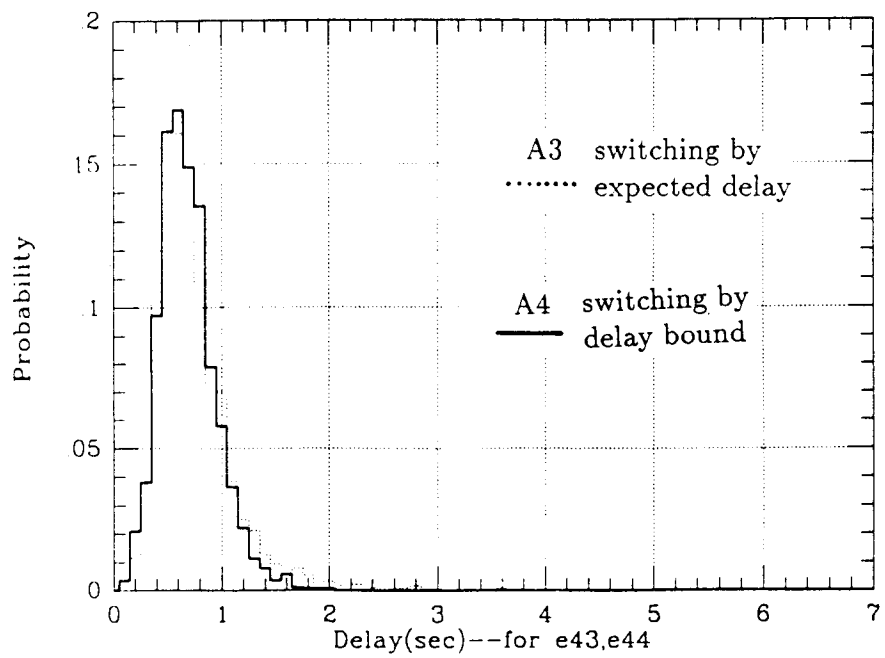


Figure 5.6(c2) Delay distributions for algorithms A3,A4 with data set I (i.a. time: 0.37 sec).

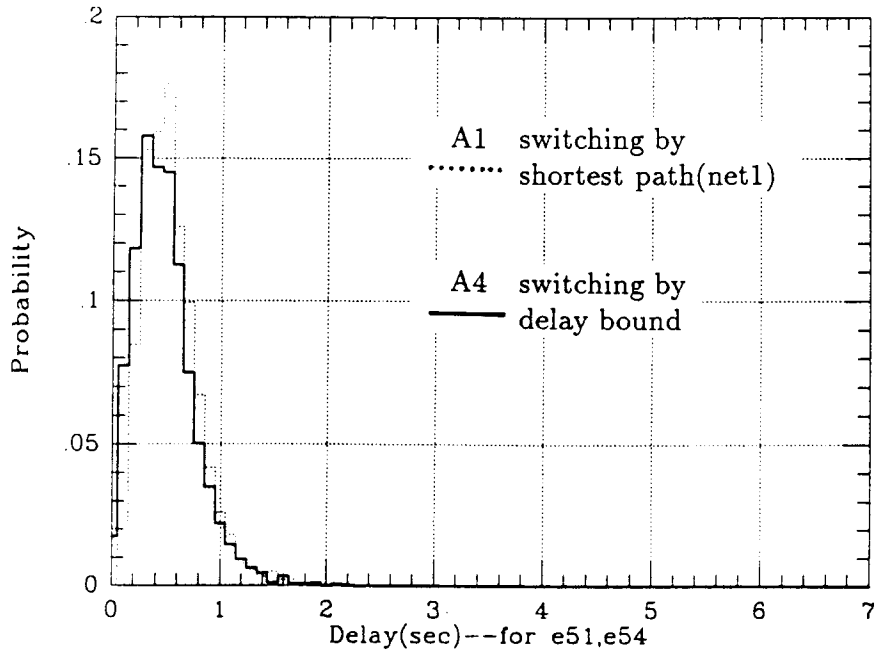


Figure 5.7(a1) Delay distributions for algorithms A1,A4 with data set II (i.a. time: 0.5 sec).

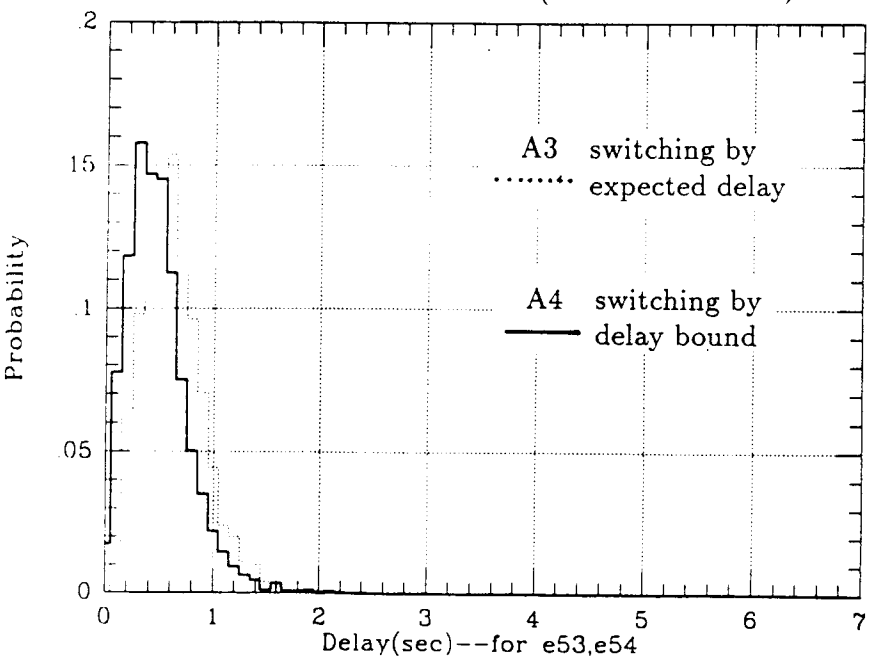


Figure 5.7(a2) Delay distributions for algorithms A3,A4 with data set II (i.a. time: 0.5 sec).

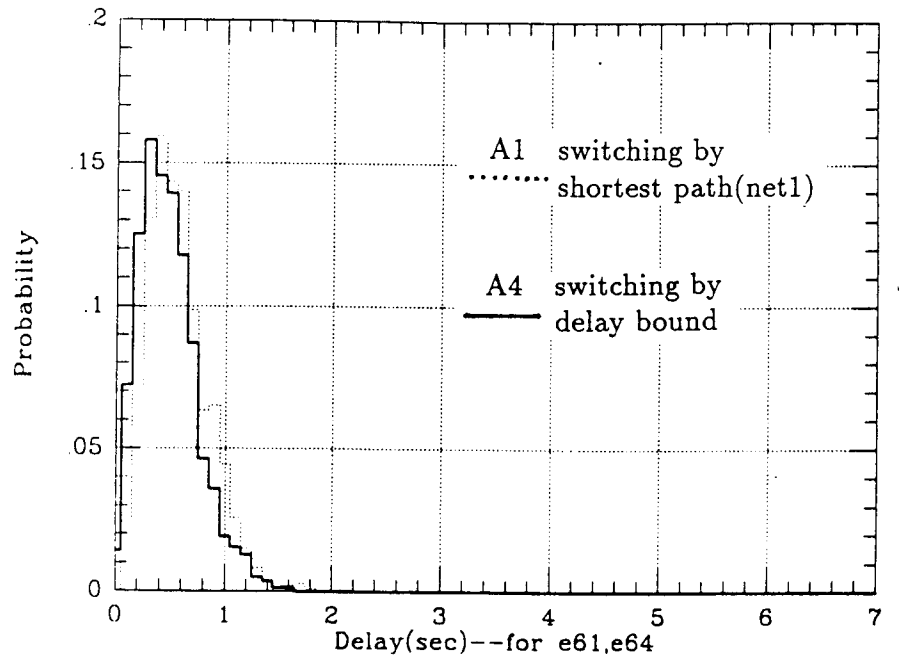


Figure 5.7(b1) Delay distributions for algorithms A1,A4 with data set II (i.a. time: 0.4 sec).

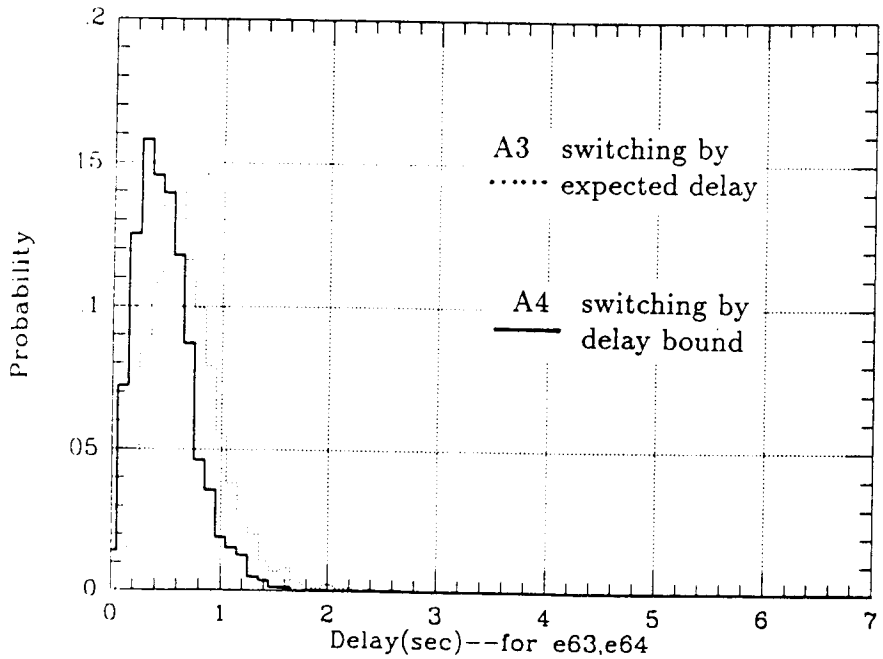


Figure 5.7(b2) Delay distributions for algorithms A3,A4 with data set II (i.a. time: 0.4 sec).

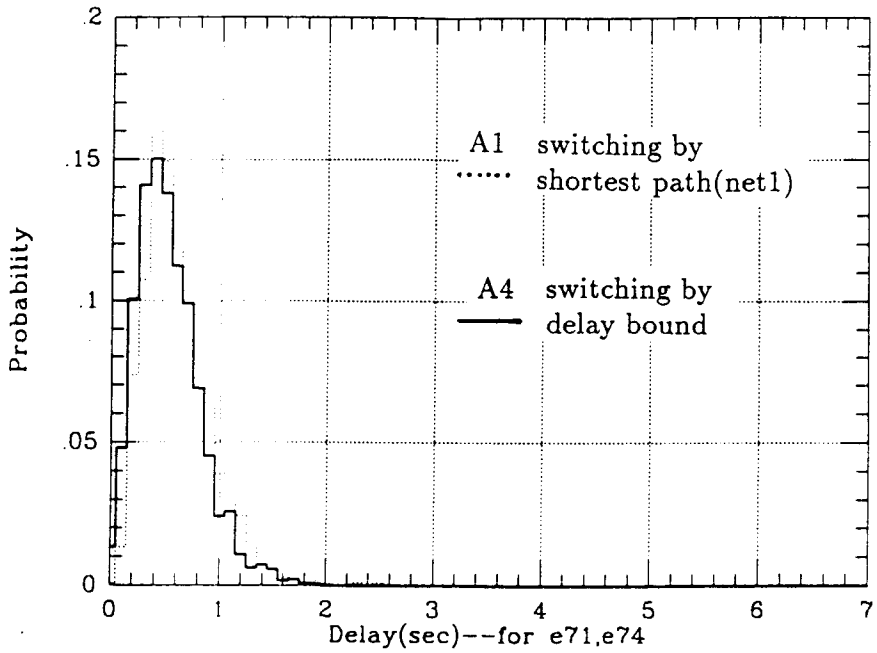


Figure 5.7(c1) Delay distributions for algorithms A1,A4 with data set II (i.a. time: 0.3 sec).

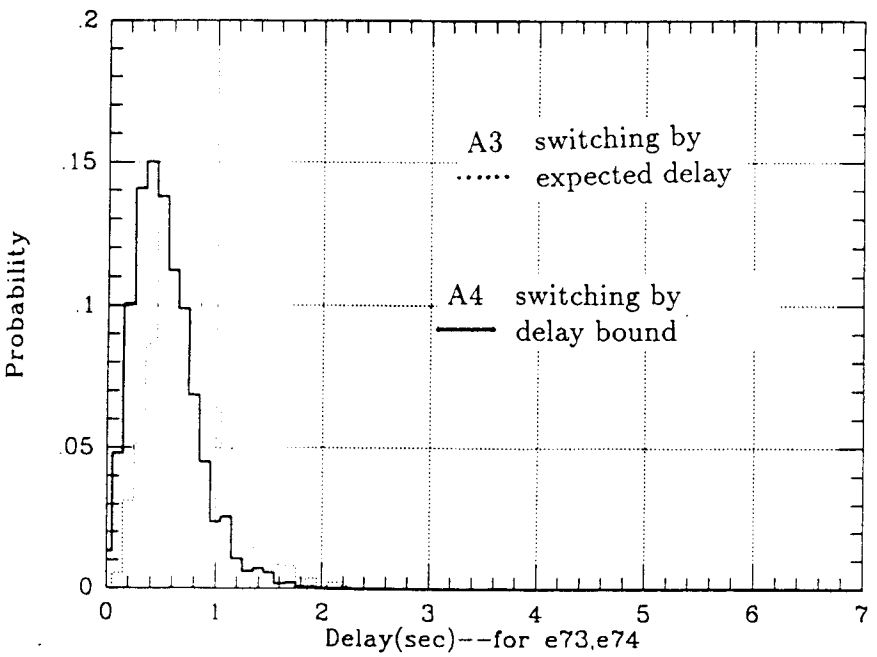


Figure 5.7(c2) Delay distributions for algorithms A3,A4 with data set II (i.a. time: 0.3 sec).

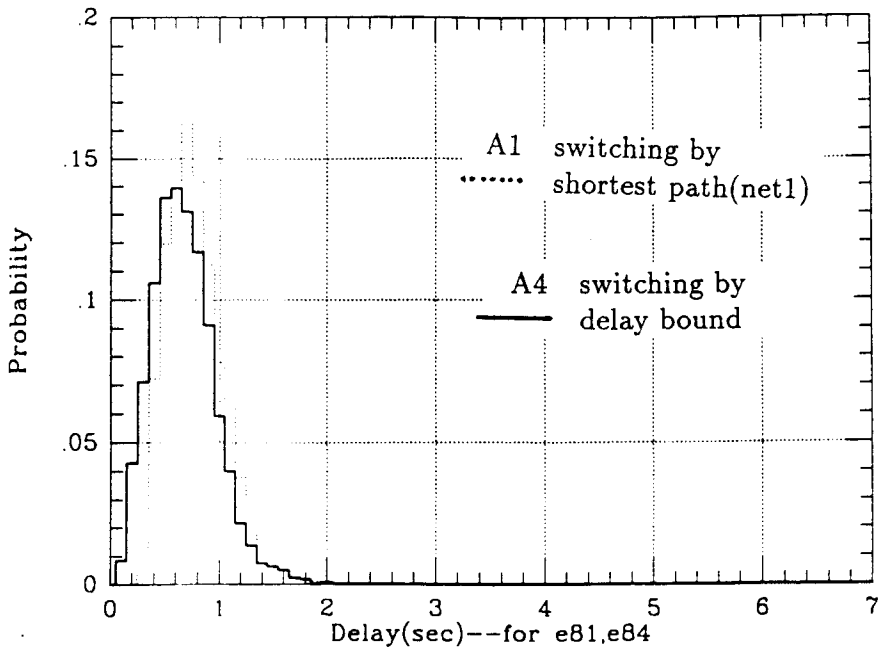


Figure 5.8(a1) Delay distributions for algorithms A1,A4 with data set III (i.a. time: 0.5 sec).

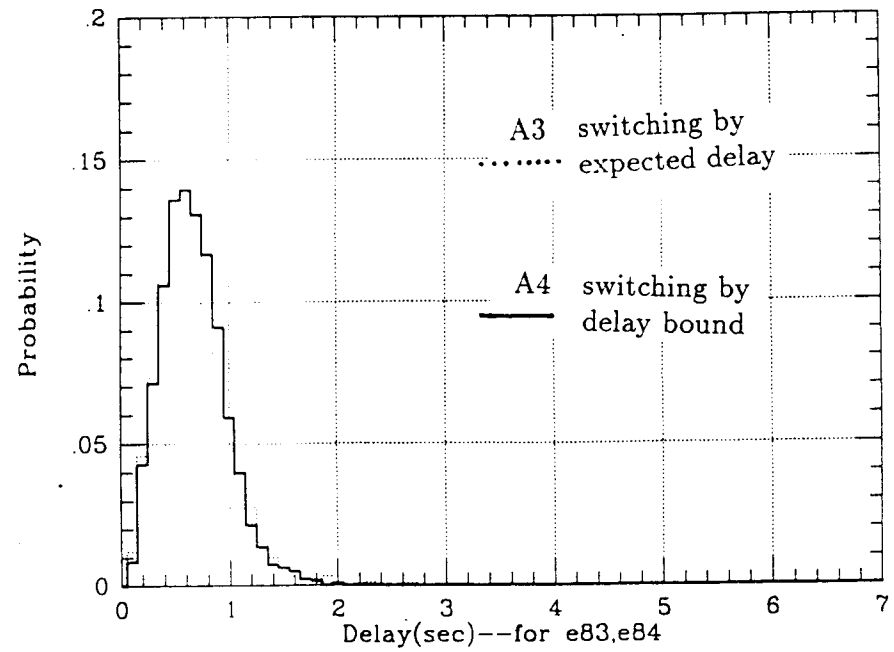


Figure 5.8(a2) Delay distributions for algorithms A3,A4 with data set III (i.a. time: 0.5 sec).

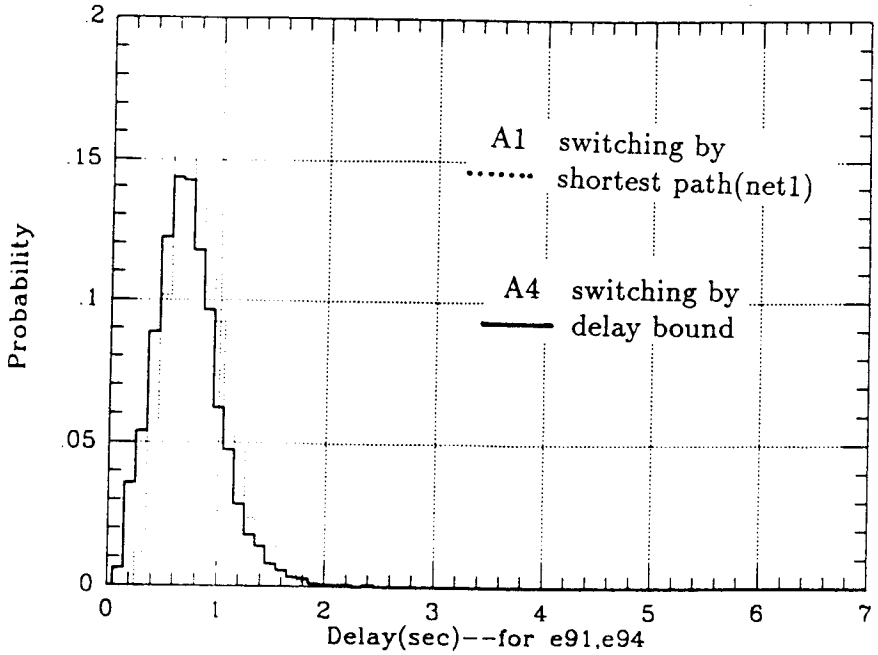


Figure 5.8(b1) Delay distributions for algorithms A1,A4 with data set III (i.a. time: 0.4 sec).

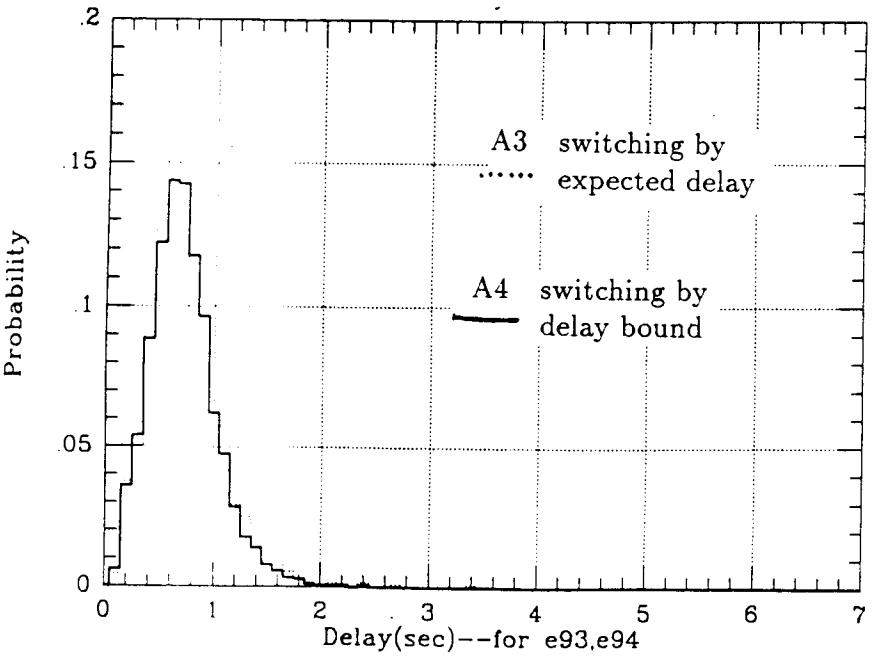


Figure 5.8(b2) Delay distributions for algorithms A3,A4 with data set III (i.a. time: 0.4 sec).

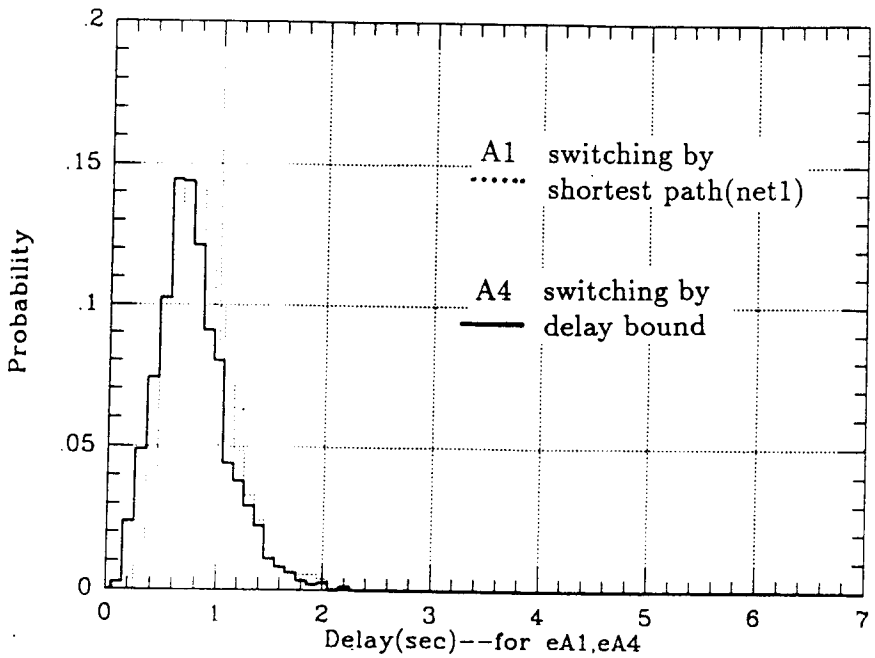


Figure 5.8(c1) Delay distributions for algorithms A1,A4 with data set III (i.a. time: 0.35 sec).

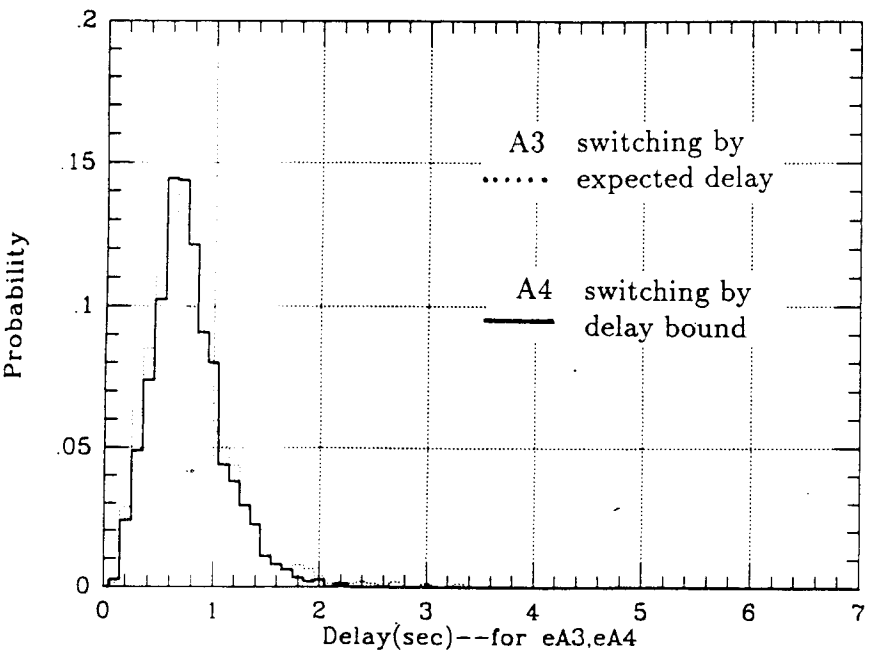


Figure 5.8(c2) Delay distributions for algorithms A3,A4 with data set III (i.a. time: 0.35 sec).

From Fig.5.6, Fig.5.7 and Fig.5.8, we find that the portion of total messages that exceeds the time bound(for example, 1.8 sec). are greatly reduced compared the shortest paths algorithms(A1). Their average delay may be greater than those obtained from algorithm A3 but one sees that the "tail" of the delay distribution for A4 "drops" rapidly for several cases(see Fig.5.6(c2), for example).

Surprisingly, we find that in some cases algorithm A4 not only reduces the delay bound but also the average delay(see Fig.5.7). This means that we may use A4 instead of A3 to reduce both the average delay and the delay bounds.

The mean, variance and the delay bound for our previous data are shown for the data set III in Fig.5.9, Fig.5.10 and Fig.5.11 respectively. From these figures, we conclude that the delay bounds of algorithm A4 are lower for most cases than algorithms A1,A3. The variance of A4 is larger than the variance of A1 but it is lower than that of A3. The mean of the delay are lower than A1 and is very close to that of A3.

Our conclusion is that algorithm A4 which uses the delay bound as an index for switching is the best strategy we need for our problem where the time constraints are considered. We do not lose much by using the delay bound as the indices since its expected delay is very close to those obtained from algorithms employing the expected delay indices.

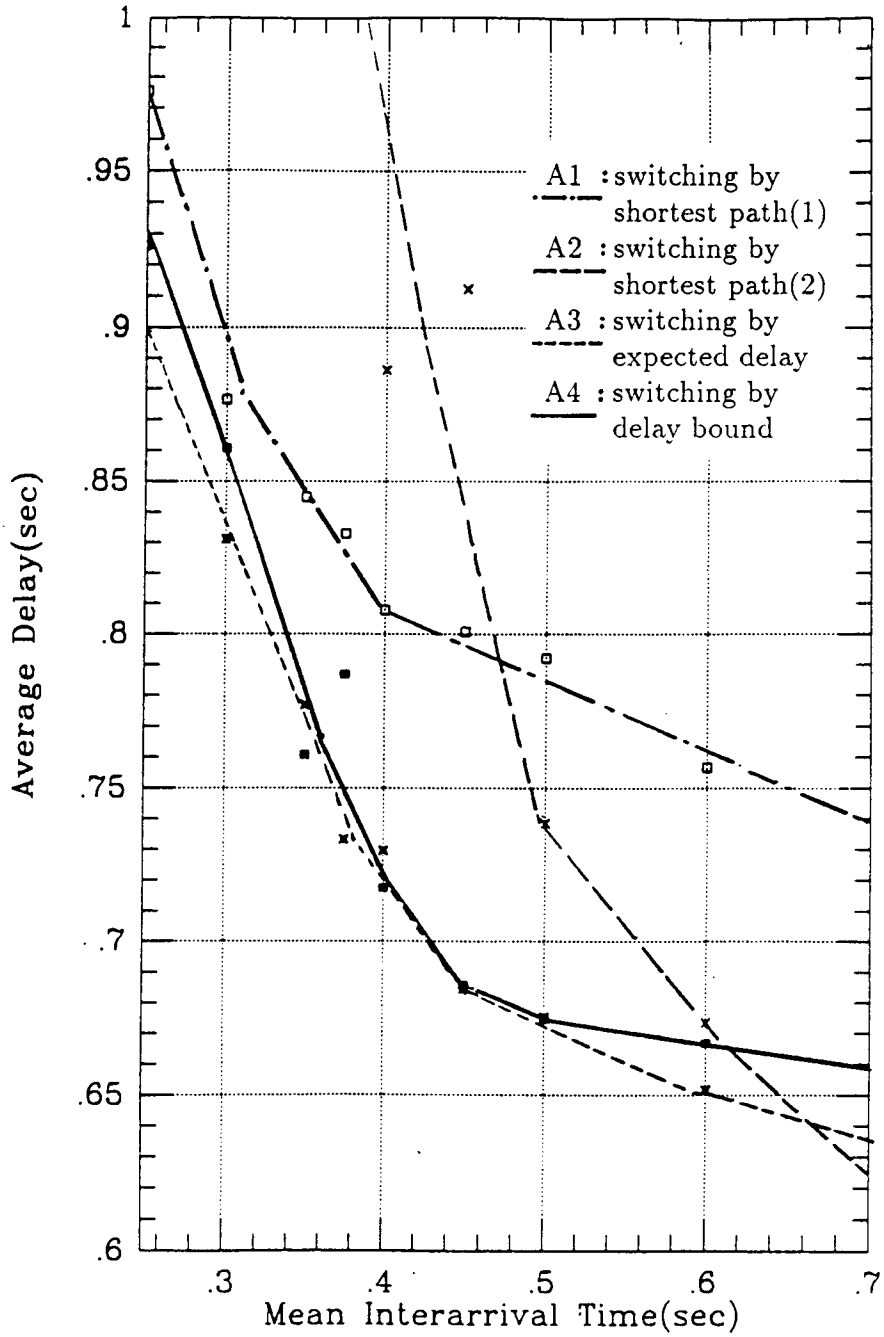


Figure 5.9: the average delay for the delay distributions

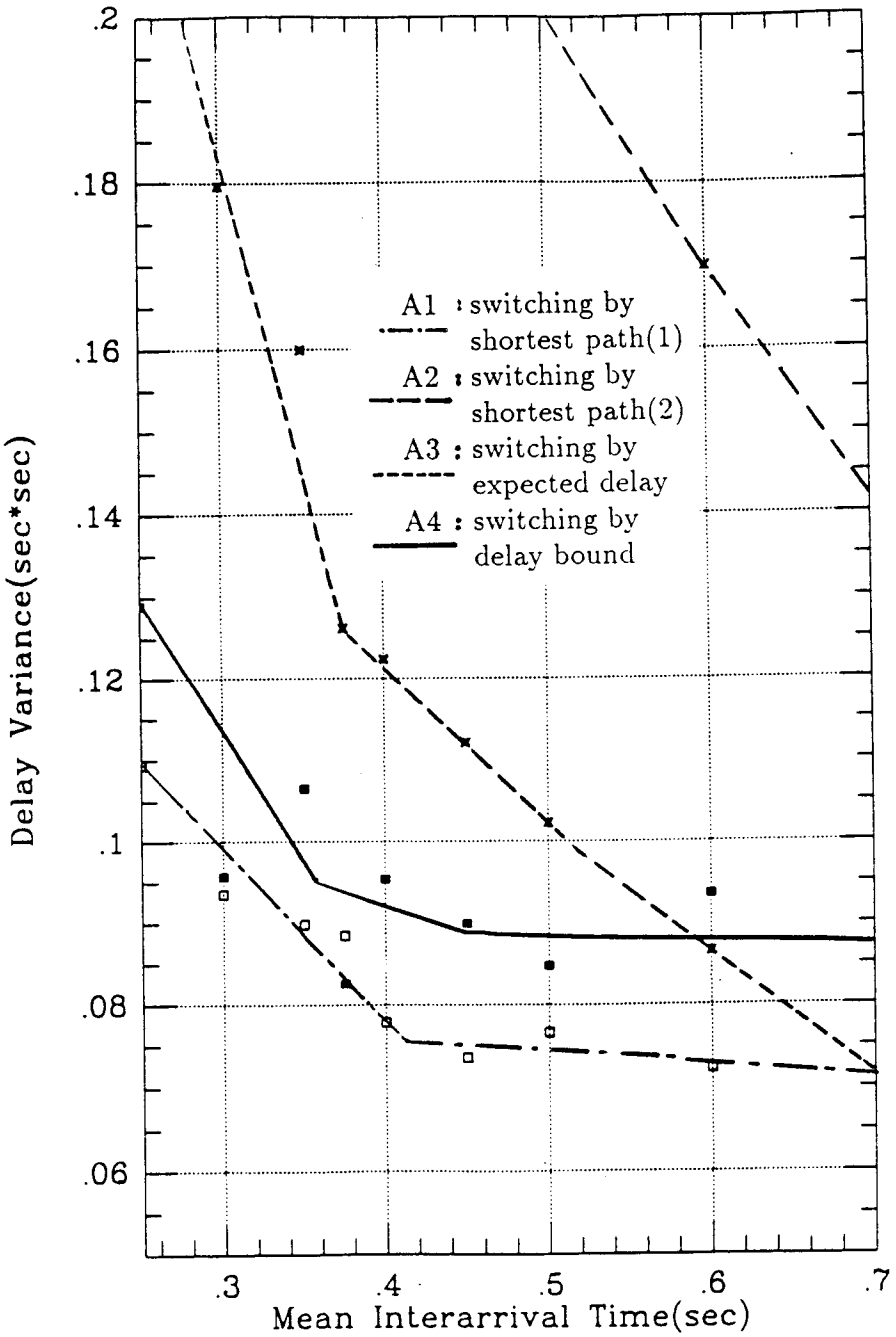


Figure 5.10: the variance for the delay distributions

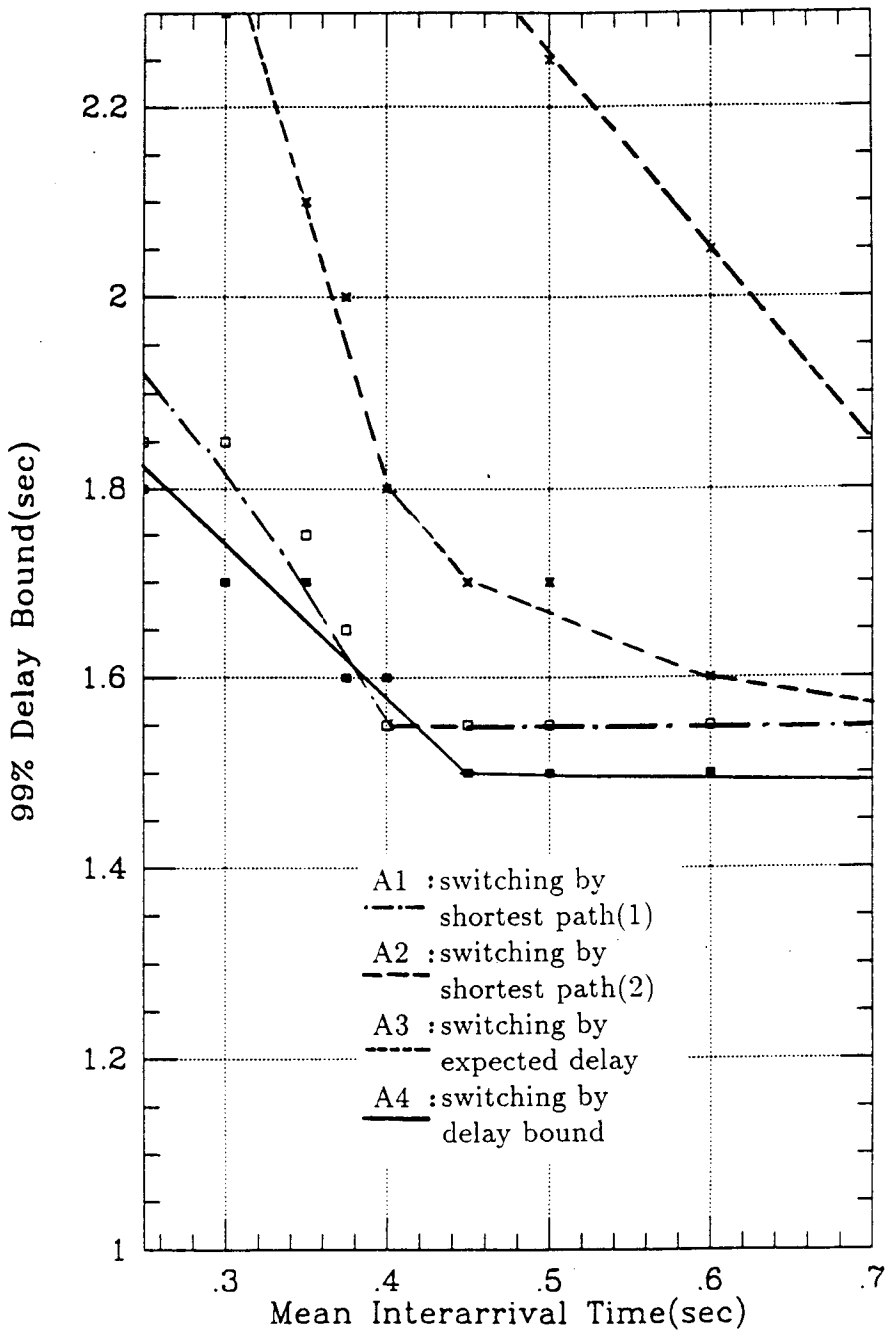


Figure 5.11: the 99% delay bounds for delay distributions

CHAPTER
SIX
CONCLUSIONS

In this thesis, we propose a switching algorithm for the control of multimedia networks or more specifically, the multi-subnetworks with objective which minimizes the portion of delay distributions that lies beyond the time constraints. We started from the analytical model presented in Sec.3.1 and end up with the delay bound index which was shown to give better "worst case" delay performance.

By simulation, we have shown that our algorithms outperform the conventional shortest path switching algorithms and the switching algorithms which use the expected delay as an objective function for optimizations. Those algorithms are based on shifting messages to links with smaller ELD(expected link delay)[15,22,23]. We have thus find a new direction for network optimization problems in which the objective functions are not solely the ELD's.

We also showed that the computation time for our indices can be computed in time $O(n \log n)$ with the IDFT(Inverse Discrete Fourier Transform) algorithm proposed in Chapter 4. This result show that we can implement our switching algorithms in a real time environment where the network parameters are changing rapidly.

With recent VLSI technology and the highly regular structure like systolic array[70]. One can even reduces the complexity of our problem to an order for $T = O(\log n)$ and $A = O(n \log n)$ (see [68,69]), where A denotes the chip area required for that algorithm.

From our subnetwork model(Chapter 2), we find that our indices can be easily changed to meet different requirements of specific users. For example, we can incorporate new heuristic rules into the index as long as the the computation time is tolerable. By this approach, we can uses the knowledges or the "rules of thumbs" to make a more intelligent decision. This is the goal of recent research for AI(Artificial Intelligence) approach and expert system applications for telecommunications[71,72]. As the technology in AI and the capabilities of the networks continue to grow, we expect that the ultimate communication processor will be an "intelligent element" which can monitor or control many transmission media and interact with many heterogeneous networks of future telecommunication systems.

APPENDIX

A.1 Proofs of Lemma 3.3

Proof: By induction, we assume that (3.2-7) is true for $M = k$ and $\forall N \in \mathcal{Z}^+$, then the induction step for $M = k + 1$ is:

$$
\begin{aligned}
(\sum_{i=1}^{k} x_i + x_{k+1})^N &= \sum_{j=0}^{N} \binom{N}{j} (\sum_{i=1}^{k} x_i)^j x_{k+1}^{N-j} \\
&= \sum_{j=0}^{N} \binom{N}{j} \Big(\sum_{\substack{(n_1,n_2,\ldots,n_k) \\ n_1+n_2+\cdots+n_k=j}} \frac{j!}{\prod_{i=1}^{k} n_i!} \prod_{i=1}^{k} x_i^{n_i} \Big) x_{k+1}^{N-j} \\
&= \sum_{j=0}^{N} \sum_{\substack{(n_1,\ldots,n_k) \\ n_1+\cdots+n_k=j}} \frac{N!}{j!(N-j)!} \frac{j!}{\prod_{i=1}^{k} n_i!} \prod_{i=1}^{k} x_i^{n_i} x_{k+1}^{N-j} \\
&= \sum_{\substack{n_1,\ldots,n_k,n_{k+1}=N-j \\ \sum_{i=1}^{k+1} n_i = N}} \frac{N!}{\prod_{i=1}^{k+1} n_i!} \prod_{i=1}^{k+1} x_i^{n_i}
\end{aligned}
\tag{1.1}
$$

Therefore, (3.2-7) is also true for $M = k + 1$. Q.E.D.

A.2 Proofs of Theorem 4.1

Proof:

(a) Let w be a primitive n_th root of unity, $n = 2^p$, then

$$w^n = w^{2^p} = (w^2)^{2^{p-1}} = (w^2)^{\frac{n}{2}} = 1$$

One sees that item (1) of Definition 4.2 is thus satisfied. Now, from Definition 4.2, we know w satisfies the item (2) of Definition 4.2:

$$\sum_{k=0}^{n-1} w^k = 0$$

which is equal to:

$$\begin{aligned} & 1 + w + w^2 + \cdots + w^{2^p-1} \\ = \; & (1 + w^2 + \cdots + w^{2^p-2}) + (w + w^3 + \cdots + w^{2^p-1}) \\ = \; & (1 + w^2 + \cdots + w^{2^p-2} + w(1 + w^2 + \cdots + w^{2^p-2}) \\ = \; & (1 + w)(1 + w^2 + \cdots + w^{2^p-2}) = 0 \end{aligned} \tag{1.2}$$

That implies:

$$\sum_{k=0}^{2^{p-1}-1} (w^2)^k = 0$$

since $w \neq -1$. Therefore, (2) of Definition 4.2 is also satisfied and w^2 is a primitive $n/2$_th root of unity.

(b)

$$\begin{aligned} & (w^{k+\frac{n}{2}} - w^k)(w^{k+\frac{n}{2}} + w^k) \\ = \; & (w^{k+\frac{n}{2}})^2 - (w^k)^2 \\ = \; & w^{2k+n} - w^{2k} \\ = \; & w^{2k}w^n - w^{2k} = 0 \end{aligned} \tag{1.3}$$

Note that $1, w, w^2, \ldots, w^{n-1}$ are all distinct, $w^{k+\frac{n}{2}} \neq w^k$, so we have:

$$w^{k+\frac{n}{2}} = -w^k$$

Q.E.D.

A.3 QNAP2 Simualtion Programs :

the Expected Delay Case

```
SIMULOG    ***  QNAP2  ***   ( 15-FEB-88 ) V 5.0
(C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986

 1 &
 2 & Network model b3
 3 & Routing: Use switching algorithms at nodel for class X
 4 & Index for switching: The estimates for the expexted delays
 5 &
 6 /DECLARE/
 7     INTEGER N=7;                          & number of processing nodes
 8     INTEGER S=4;                          & Backqround traffic sources
 9     INTEGER Ix=2000;                      & number of points for delay distr.
10     INTEGER ix_max=200;                   & printing limit for delay distr.
11     INTEGER nb_max;                       & nb of messages that exceed ix_max
12     INTEGER Nx;                           & number of X messag sent to netwk
13     INTEGER nb_msg;                       & number of X messag received
14     INTEGER ix,jx;
15     INTEGER i,j;
16     INTEGER nb;
17     REAL xxx=999.99;                      & a large cost for disconn. links
18     REAL warm_up=50.0;                    & system warm-up time
19     REAL t1,t2;                           & indexes for switching
20     REAL lambda;                          & temp var for thruput
21     REAL ct;                              & storage for message delay
22     REAL fx(Ix);                          & the delay distribution for X
23     REAL ia_time;                         & sources inter_arrival time
24     REAL sva_time;                        & links service time for net 1
25     REAL svb_time;                        & links service time for net 2
26     REAL pc_time;                         & nodes processing time
27     REAL tmax;                            & simulation time
28     REAL jack_pr(N,0:N);                  & the Jacksonial routage array
29     REAL temp;
30     REAL d_mean,d_var,d_bound;            & delay statistices
31 &
32                                           & attributes declaration:
33     QUEUE REAL d_time;                    & nb/lambda for queueing delay
34     CUSTOMER REAL s_time;                 & starting time for a message
35     CUSTOMER INTEGER s_nb;                & sequential number for a message
36 &
37     QUEUE n(N);                           & processing nodes
38     QUEUE s(N);                           & background sources
39     QUEUE l(N,N);                         & communication links
40 &
41     REF CUSTOMER cx;                      & an X message
42 &
43     CLASS X,Y;                            & two classes of traffic
44                                           & X:messages ; Y:background
45 &
46 & ------------------------------------------------------------
47 & Followings are the variabels for shortest path algorithms
48 & ------------------------------------------------------------
49     REAL graph(N,0:N)=( xxx, 0.0, 0.2, 1.0, 1.0, xxx, xxx, xxx,
50                         5.0, xxx, 0.0, 0.2, 3.0, 5.0, xxx, xxx,
51                         4.0, xxx, 2.0, 0.0, 3.0, xxx, 0.2, xxx,
52                         xxx, xxx, 2.0, 2.0, 0.0, 2.0, 2.0, 5.0,
53                         1.0, xxx, 5.0, xxx, 3.0, 0.0, 3.0, 2.0,
54                         1.0, xxx, xxx, 2.0, 3.0, 3.0, 0.0, 0.2,
55                         9.0, xxx, xxx, xxx, xxx, xxx, xxx, xxx);
```

```
56                                      & the connectivity and the
57                                      & cost for each links
58                                      & row 0 denotes the out flow
59      REAL rout_pr(N,N)=
60             (1.0,0.0,0.0,0.0,0.0,0.0,
61              0.0,1.0,0.0,0.0,0.0,0.0,
62              0.0,0.0,1.0,0.0,0.0,0.0,
63              0.0,0.0,0.0,1.0,0.0,0.0,
64              0.0,0.0,0.0,0.0,1.0,0.0,
65              0.0,0.0,0.0,0.0,0.0,1.0,
66              0.0,0.0,0.0,0.0,0.0,0.0,1.0);
67                                      & routing probability (i,j) :
68                                      & node i destined for node j
69                                      & for class X messages
70      INTEGER path(N);                & path array for sp
71      REAL d(N,N);                    & evolution array for min cost
72
73  &-------------------------------------------------------------
74  &
75  & The BELLMAN-FORD algorithm for computing
76  & the shortest path
77  &
78      PROCEDURE spl(k1,k2);
79          INTEGER k1,k2;              & the formal parameters
80                                      & k:source, l:destination
81          REAL m;
82          INTEGER i1,j1,n1,h;         & computation variables
83      BEGIN
84          FOR j1:=1 STEP 1 UNTIL N DO
85              BEGIN
86                  d(1,j1):=graph(k1,j1);
87                  path(j1):= IF graph(k1,j1)=xxx THEN 0 ELSE k1;
88              END;
89          FOR h:=2 STEP 1 UNTIL N DO
90              FOR j1:=1 STEP 1 UNTIL N DO
91                  BEGIN
92                      m:=d(h-1,j1);
93                      FOR n1:=1 STEP 1 UNTIL N DO
94                          IF (d(h-1,n1)+graph(n1,j1))<m THEN
95                              BEGIN
96                                  m:=d(h-1,n1)+graph(n1,j1);
97                                  path(j1):=n1;
98                              END;
99                      d(h,j1):=m;
100                 END;
101         i1:=k2;
102         WHILE (path(i1)<>k1) AND (path(i1)<>0) DO
103             BEGIN
104                 FOR j1:=1 STEP 1 UNTIL N DO
105                     rout_pr(path(i1),j1):=0;  & clear the row
106                 rout_pr(path(i1),i1):=1;
107                 i1:=path(i1);
108             END;
109         IF path(i1)=k1 THEN BEGIN
110             FOR j1:=1 STEP 1 UNTIL N DO
111                 rout_pr(k1,j1):=0;   & clear the row
112             rout_pr(k1,i1):=1;
113         END;
114
115         PRINT("* * * The Shortest Path Matrix (1) * * *");
```

```
116        PRINT("1) The path array:");
117        PRINT(path(1 STEP 1 UNTIL N));
118        PRINT("2) The cost matrix");
119        FOR j1:=1 STEP 1 UNTIL N DO
120           PRINT(d(j1, 1 STEP 1 UNTIL N));
121        PRINT("3) The routing probability matrix:");
122        FOR j1:=1 STEP 1 UNTIL N DO
123           PRINT(rout_pr(j1, 1 STEP 1 UNTIL N));
124
125     END;
126  &
127  &-----------------------------------------------------------------
128  & The shortest path for the second network
129  &
130     PROCEDURE sp2(k1,k2);
131        INTEGER k1,k2;
132        INTEGER i1,j1;
133     BEGIN
134        IF k1=1 THEN
135           BEGIN
136              FOR j1:=1 STEP 1 UNTIL N DO
137                 rout_pr(1,j1):=0;
138              rout_pr(1,4):=1.0;
139           END ELSE
140        IF k1=7 THEN
141           BEGIN
142              rout_pr(7,7):=1.0;
143           END ELSE
144        IF k1=4 THEN
145           BEGIN
146              FOR j1:=1 STEP 1 UNTIL N DO
147                 rout_pr(4,j1):=0;
148              rout_pr(4,7):=1.0;
149           END
150        ELSE
151           BEGIN
152              FOR i1:=(2,3,5,6) DO
153                 BEGIN
154                    FOR j1:=1 STEP 1 UNTIL N DO
155                       rout_pr(i1,j1):=0;
156                    rout_pr(i1,4):=1;
157                 END;
158           END;
159
160        PRINT("* * * The Shortest Path Matrix (II)* * *");
161        PRINT("1) The path array:");
162        PRINT(path(1 STEP 1 UNTIL N));
163        PRINT("2) The cost matrix");
164        FOR i1:=1 STEP 1 UNTIL N DO
165           PRINT(d(i1, 1 STEP 1 UNTIL N));
166        PRINT("3) The routing probability matrix:");
167        FOR i1:=1 STEP 1 UNTIL N DO
168           PRINT(rout_pr(i1,1 STEP 1 UNTIL N));
169     END;
170
171
172  &-----------------------------------------------------------------
173  & Find the Jacksonian routing probability
174  &
175  &
```

```
176     PROCEDURE Jackson;
177
178     REAL r_sum;                          & sum of one row in jack_pr
179     INTEGER i1,j1;
180     BEGIN
181        FOR i1:=1 STEP 1 UNTIL N DO
182        BEGIN
183           r_sum:=0;
184           FOR j1:=0 STEP 1 UNTIL N DO
185              r_sum:=r_sum+(IF graph(i1,j1)<>xxx THEN graph(i1,j1)
186                                                 ELSE 0.0);
187           FOR j1:=0 STEP 1 UNTIL N DO
188              jack_pr(i1,j1):=IF graph(i1,j1)<>xxx THEN graph(i1,j1)/r_sum
189                                                   ELSE 0.0;
190        END;
191        PRINT("* * * The Jacksonian Matrix * * *");
192        FOR i1:=1 STEP 1 UNTIL N DO
193        PRINT(jack_pr(i1,0 STEP 1 UNTIL N));
194     END;
195
196
197
198 &
199 &
200 & processing nodes
201 &
202 /STATION/
203     NAME=n(1);
204     SERVICE(X)=
205     BEGIN
206        CST(pc_time);
207        cx:=n(1).FIRST;
208        $MACRO l_delay(node1,node2)
209           nb:=l(node1,node2).NB;
210           lambda:=REALINT(l(node1,node2).NBOUT)/TIME;
211           l(node1,node2).d_time:=IF lambda<>0 THEN REALINT(nb)/lambda
212                                              ELSE 0.0;
213        $END
214        $MACRO n_delay(node)
215           nb:=n(node).NB;
216           lambda:=REALINT(n(node).NBOUT)/TIME;
217           n(node).d_time:=IF lambda<>0 THEN REALINT(nb)/lambda
218                                        ELSE 0.0;
219        $END
220        $l_delay(1,2) $n_delay(2) $l_delay(2,3)
221        $n_delay(3) $l_delay(3,6) $n_delay(6) $l_delay(6,7) $n_delay(7)
222        $l_delay(1,4) $n_delay(4) $l_delay(4,7)
223        t1:=l(1,2).d_time+n(2).d_time+l(2,3).d_time+n(3).d_time+l(3,6).d_t
 ==>  ime
224             +n(6).d_time+l(6,7).d_time;
225        t2:=l(1,4).d_time+n(4).d_time+l(4,7).d_time;
226        IF t1<=t2 THEN
227                   BEGIN
228                   &  PRINT(TIME, t1,"|",t2,"MESSAGE:",cx.s_nb,
229                   &        " sent to network1");
230                      TRANSIT(l(1,2));
231                   END
232                  ELSE
233                   BEGIN
234                   &  PRINT(TIME, t1,"|",t2,"MESSAGE:",cx.s_nb,
```

```
235                     &       " sent to network2");
236                        TRANSIT(1(1,4));
237                     END;
238     END;
239     SERVICE(Y)= CST(pc_time);
240     TRANSIT(Y)=1(1,1 STEP 1 UNTIL N), jack_pr(1, 1 STEP 1 UNTIL N),OUT;
241  /STATION/
242     NAME=n(2);
243     SERVICE=CST(pc_time);
244     TRANSIT(X)=1(2,1 STEP 1 UNTIL N),rout_pr(2,1 STEP 1 UNTIL N);
245     TRANSIT(Y)=1(2,1 STEP 1 UNTIL N),jack_pr(2,1 STEP 1 UNTIL N),OUT;
246  /STATION/
247     NAME=n(3);
248     SERVICE=CST(pc_time);
249     TRANSIT(X)=1(3,1 STEP 1 UNTIL N),rout_pr(3,1 STEP 1 UNTIL N);
250     TRANSIT(Y)=1(3,1 STEP 1 UNTIL N),jack_pr(3,1 STEP 1 UNTIL N),OUT;
251  /STATION/
252     NAME=n(4);
253     SERVICE=CST(pc_time);
254     TRANSIT(X)=1(4,1 STEP 1 UNTIL N),rout_pr(4,1 STEP 1 UNTIL N);
255     TRANSIT(Y)=1(4,1 STEP 1 UNTIL N),jack_pr(4,1 STEP 1 UNTIL N); & no ou
 ==>  t
256  /STATION/
257     NAME=n(5);
258     SERVICE=CST(pc_time);
259     TRANSIT(X)=1(5,1 STEP 1 UNTIL N),rout_pr(5,1 STEP 1 UNTIL N);
260     TRANSIT(Y)=1(5,1 STEP 1 UNTIL N),jack_pr(5,1 STEP 1 UNTIL N),OUT;
261  /STATION/
262     NAME=n(6);
263     SERVICE=CST(pc_time);
264     TRANSIT(X)=1(6,1 STEP 1 UNTIL N),rout_pr(6,1 STEP 1 UNTIL N);
265     TRANSIT(Y)=1(6,1 STEP 1 UNTIL N),jack_pr(6,1 STEP 1 UNTIL N),OUT;
266  /STATION/
267     NAME=n(7);
268     SERVICE=
269     BEGIN
270        CST(pc_time);
271        cx:=n(7).FIRST;
272        IF cx.CCLASS=X THEN
273        BEGIN
274           ct:=TIME-cx.s_time;
275        &  PRINT("MESSAGE:",cx.s_nb,"delay=",ct);
276           ix:=INTREAL(ct*10)*2+1;
277           IF ix>Ix-1 THEN ix:=Ix-1;  & (fx(1999),fx(2000)) is the last pa
 ==>  ir
278           IF ix<1 THEN ix:=1;
279           IF ix>=ix_max-1 THEN nb_max:=nb_max+1; & the delay that exceeds
 ==>   ix_max
280           nb_msg:=nb_msg+1;
281           fx(ix+1):=fx(ix+1)+1;
282        END;
283     END;
284     TRANSIT=OUT;
285  &
286  & background source nodes
287  &
288  /STATION/
289     NAME=s(1);
290     TYPE=SOURCE;
291     SERVICE=
```

```
292     BEGIN
293        EXP(ia_time);
294        & X is created only when system is warm-up
295        IF (TIME>warm_up) AND DRAW(0.45) THEN
296        BEGIN
297           cx:=s(1).FIRST;
298           cx.s_time:=TIME;
299           cx.s_nb:=Nx;
300           Nx:=Nx+1;
301           TRANSIT(n(1),X);
302        END
303        ELSE TRANSIT(n(1),Y);
304     END;
305  /STATION/
306     NAME=s(2);
307     TYPE=SOURCE;
308     SERVICE=EXP(ia_time);
309     TRANSIT=n(2),Y,1;
310  /STATION/
311     NAME=s(3);
312     TYPE=SOURCE;
313     SERVICE=EXP(ia_time);
314     TRANSIT=n(3),Y,1;
315  /STATION/
316     NAME=s(5);
317     TYPE=SOURCE;
318     SERVICE=EXP(ia_time);
319     TRANSIT=n(5),Y,1;
320  /STATION/
321     NAME=s(6);
322     TYPE=SOURCE;
323     SERVICE=EXP(ia_time);
324     TRANSIT=n(6),Y,1;
325  &
326  & comm. links
327  &
328     $MACRO link(s_node,d_node,s_time) & definition for communnication
329     /STATION/ NAME=l(s_node,d_node);  & link stations
330               SERVICE=EXP(s_time);
331               TRANSIT=n(d_node),1;
332     $END
333  &
334  $link(1,2,sva_time)   $link(1,3,sva_time)
335  $link(2,3,sva_time)   $link(2,5,sva_time)
336  $link(3,2,sva_time)   $link(3,6,sva_time)
337  $link(5,2,sva_time)   $link(5,6,sva_time)   $link(5,7,sva_time)
338  $link(6,3,sva_time)   $link(6,5,sva_time)   $link(6,7,sva_time)
339  &
340  $link(1,4,svb_time)   $link(4,7,svb_time)
341  $link(2,4,svb_time)   $link(3,4,svb_time)   $link(5,4,svb_time)
342  $link(6,4,svb_time)
343  $link(4,2,svb_time)   $link(4,3,svb_time)   $link(4,5,svb_time)
344  $link(4,6,svb_time)
345  &
346  /CONTROL/
347     CLASS=ALL QUEUE;
348     ENTRY=BEGIN
349        PRINT("Inter-arrival time?"); ia_time:=GET(REAL);
350        PRINT("Service time for net 1?"); sva_time:=GET(REAL);
351        PRINT("Service time for net 2?"); svb_time:=GET(REAL);
```

```
352       PRINT("Processing time?");      pc_time:=GET(REAL);
353       PRINT("Simulation time?");      tmax:=GET(REAL);
354       FOR ix:=1 STEP 2 UNTIL Ix-1 DO  & initializing the delay distr. ar
 ==>  ray
355           fx(ix):=REALINT(ix-1)/20.0;
356    END;
357    TMAX=tmax;
358    EXIT=BEGIN     & print the delay distribution
359       PRINT("nb of msg=",nb_msg);
360       PRINT("* * * Delay Distribution:");
361       FOR ix:=1 STEP 2 UNTIL ix_max-1 DO BEGIN
362          fx(ix+1):=fx(ix+1)/REALINT(nb_msg);
363          PRINT(fx(ix),fx(ix+1),CURVE(fx,fx(ix)));
364          d_mean:=d_mean+fx(ix)*fx(ix+1);
365          d_var:=d_var+fx(ix)*fx(ix)*fx(ix+1);
366       END;
367       d_var:=d_var - d_mean*d_mean;
368       temp:=REALINT(nb_max)/REALINT(nb_msg);
369       jx:=ix_max;
370       WHILE (temp<0.01) AND (jx>=1) DO
371       BEGIN
372          temp:=temp+fx(jx);
373          jx:=jx-2;
374       END;
375       d_bound:=REALINT(jx)/20.0;
376       PRINT("* nb_max=",nb_max);
377       PRINT("* * * Expected Delay :",d_mean);
378       PRINT("* * * Variance :",d_var);
379       PRINT("* * * Delay Bound:",d_bound);
380    END;
381 /EXEC/
382 BEGIN
383    sp1(1,7);      & find the shortest path in network 1
384    sp2(4,7);      & find the shortest path in network 2
385    Jackson;       & find the routing prob. for Y
386
387    PRINT("* * * Simulation result for single SHORTEST PATH * * *");
388    SIMUL;
389 END;
```

```
* * * The Shortest Path Matrix (1) * * *
1) The path array:
      1        1        2        1        4        3        6
2) The cost matrix
     0.      0.2000    1.000    1.000    1000.0   1000.0   1000.0
     0.      0.2000   0.4000    1.000     3.000    1.200    6.000
     0.      0.2000   0.4000    1.000     3.000   0.6000    1.400
     0.      0.2000   0.4000    1.000     3.000   0.6000   0.8000
     0.      0.2000   0.4000    1.000     3.000   0.6000   0.8000
     0.      0.2000   0.4000    1.000     3.000   0.6000   0.8000
     0.      0.2000   0.4000    1.000     3.000   0.6000   0.8000
3) The routing probability matrix:
     0.      1.000      0.        0.        0.        0.        0.
     0.        0.     1.000       0.        0.        0.        0.
     0.        0.       0.        0.        0.      1.000       0.
     0.        0.       0.      1.000       0.        0.        0.
     0.        0.       0.        0.      1.000       0.        0.
     0.        0.       0.        0.        0.        0.      1.000
     0.        0.       0.        0.        0.        0.      1.000
* * * The Shortest Path Matrix (II)* * *
```

```
1) The path array:
      1         1         2         1         4         3         6
2) The cost matrix
      0.      0.2000     1.000     1.000     1000.0    1000.0    1000.0
      0.      0.2000    0.4000     1.000      3.000     1.200     6.000
      0.      0.2000    0.4000     1.000      3.000    0.6000     1.400
      0.      0.2000    0.4000     1.000      3.000    0.6000    0.8000
      0.      0.2000    0.4000     1.000      3.000    0.6000    0.8000
      0.      0.2000    0.4000     1.000      3.000    0.6000    0.8000
      0.      0.2000    0.4000     1.000      3.000    0.6000    0.8000
3) The routing probability matrix:
      0.      1.000       0.         0.         0.        0.        0.
      0.        0.      1.000        0.         0.        0.        0.
      0.        0.        0.         0.         0.      1.000       0.
      0.        0.        0.         0.         0.        0.      1.000
      0.        0.        0.         0.       1.000       0.        0.
      0.        0.        0.         0.         0.        0.      1.000
      0.        0.        0.         0.         0.        0.      1.000
* * * The Jacksonian Matrix * * *
      0.        0.     0.9091e-01  0.4545     0.4545      0.        0.
  0.3788        0.        0.       0.1515e-01 0.2273    0.3788      0.
  0.4348        0.     0.2174        0.       0.3261      0.     0.2174e
      0.        0.     0.1538     0.1538        0.       0.1538    0.1538
  0.7143e-01    0.     0.3571        0.       0.2143      0.      0.2143
  0.1087        0.        0.      0.2174     0.3261    0.3261      0.
   1.000        0.        0.         0.         0.        0.        0.
* * * Simulation result for single SHORTEST PATH * * *

Inter-arrival time?
     1 0.5
Service time for net 1?
     2 0.12
Service time for net 2?
     3 0.17
Processing time?
     4 0.05
Simulation time?
     5 2050

*** SIMULATION ***
... TIME =      2050.00
**************************************************************************
*   NAME    * SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
**************************************************************************
*           *         *          *          *           *          *
* n       1 *0.5002e-01*0.9816e-01*0.1036    *0.5281e-01*      4023*
*(X        )*0.5002e-01*0.4370e-01*0.4636e-01*0.5306e-01*      1791*
*(Y        )*0.5002e-01*0.5446e-01*0.5728e-01*0.5261e-01*      2232*
*           *         *          *          *           *          *
* n       2 *0.5002e-01*0.3520    *0.4485    *0.6372e-01*     14427*
*(X        )*0.5002e-01*0.2345e-01*0.2939e-01*0.6269e-01*       961*
*(Y        )*0.5002e-01*0.3286    *0.4191    *0.6380e-01*     13466*
*           *         *          *          *           *          *
* n       3 *0.5002e-01*0.2516    *0.2919    *0.5803e-01*     10311*
*(X        )*0.5002e-01*0.2345e-01*0.2662e-01*0.5678e-01*       961*
*(Y        )*0.5002e-01*0.2281    *0.2653    *0.5816e-01*      9350*
*           *         *          *          *           *          *
* n       4 *0.5002e-01*0.3458    *0.4341    *0.6280e-01*     14170*
*(X        )*0.5002e-01*0.2025e-01*0.2503e-01*0.6182e-01*       830*
*(Y        )*0.5002e-01*0.3255    *0.4091    *0.6286e-01*     13340*
```

*	*	*	*	*	*	*
* n	5	*0.5002e-01	*0.3469	*0.4427	*0.6384e-01	14216*
*(Y		)*0.5002e-01	*0.3469	*0.4427	*0.6384e-01	14216*
*	*	*	*	*	*	*
* n	6	*0.5002e-01	*0.2537	*0.2949	*0.5814e-01	10398*
*(X		)*0.5002e-01	*0.2345e-01	*0.2685e-01	*0.5727e-01	961*
*(Y		)*0.5002e-01	*0.2303	*0.2681	*0.5823e-01	9437*
*	*	*	*	*	*	*
* n	7	*0.5002e-01	*0.2249	*0.2572	*0.5718e-01	9219*
*(X		)*0.5002e-01	*0.4368e-01	*0.4927e-01	*0.5642e-01	1790*
*(Y		)*0.5002e-01	*0.1813	*0.2079	*0.5737e-01	7429*
*	*	*	*	*	*	*
* s	1	*0.5094	* 1.000	* 1.000	*0.5094	* 4023*
*	*	*	*	*	*	*
* s	2	*0.4960	* 1.000	* 1.000	*0.4960	* 4133*
*	*	*	*	*	*	*
* s	3	*0.5044	* 1.000	* 1.000	*0.5044	* 4064*
*	*	*	*	*	*	*
* s	5	*0.4980	* 1.000	* 1.000	*0.4980	* 4116*
*	*	*	*	*	*	*
* s	6	*0.5024	* 1.000	* 1.000	*0.5024	* 4079*
*	*	*	*	*	*	*
* l	2	*0.1191	*0.6669e-01	*0.6840e-01	*0.1221	* 1148*
*(X		)*0.1166	*0.5465e-01	*0.5522e-01	*0.1178	* 961*
*(Y		)*0.1320	*0.1204e-01	*0.1318e-01	*0.1445	* 187*
*	*	*	*	*	*	*
* l	3	*0.1243	*0.5973e-01	*0.6324e-01	*0.1316	* 985*
*(Y		)*0.1243	*0.5973e-01	*0.6324e-01	*0.1316	* 985*
*	*	*	*	*	*	*
* l	4	*0.1736	*0.1601	*0.1816	*0.1970	* 1890*
*(X		)*0.1790	*0.7246e-01	*0.7935e-01	*0.1960	* 830*
*(Y		)*0.1694	*0.8760e-01	*0.1023	*0.1978	* 1060*
*	*	*	*	*	*	*
* l	10	*0.1253	*0.7113e-01	*0.7421e-01	*0.1307	* 1164*
*(X		)*0.1238	*0.5804e-01	*0.6005e-01	*0.1281	* 961*
*(Y		)*0.1322	*0.1309e-01	*0.1416e-01	*0.1430	* 203*
*	*	*	*	*	*	*
* l	11	*0.1737	*0.2655	*0.3619	*0.2367	* 3134*
*(Y		)*0.1737	*0.2655	*0.3619	*0.2367	* 3134*
*	*	*	*	*	*	*
* l	12	*0.1225	*0.3018	*0.4329	*0.1758	* 5049*
*(Y		)*0.1225	*0.3018	*0.4329	*0.1758	* 5049*
*	*	*	*	*	*	*
* l	16	*0.1228	*0.1240	*0.1420	*0.1407	* 2070*
*(Y		)*0.1228	*0.1240	*0.1420	*0.1407	* 2070*
*	*	*	*	*	*	*
* l	18	*0.1733	*0.2567	*0.3404	*0.2297	* 3037*
*(Y		)*0.1733	*0.2567	*0.3404	*0.2297	* 3037*
*	*	*	*	*	*	*
* l	20	*0.1217	*0.6816e-01	*0.7016e-01	*0.1253	* 1148*
*(X		)*0.1217	*0.5706e-01	*0.5825e-01	*0.1243	* 961*
*(Y		)*0.1216	*0.1109e-01	*0.1191e-01	*0.1306	* 187*
*	*	*	*	*	*	*
* l	23	*0.1698	*0.1652	*0.1940	*0.1994	* 1994*
*(Y		)*0.1698	*0.1652	*0.1940	*0.1994	* 1994*
*	*	*	*	*	*	*
* l	24	*0.1771	*0.1732	*0.2117	*0.2165	* 2004*
*(Y		)*0.1771	*0.1732	*0.2117	*0.2165	* 2004*
*	*	*	*	*	*	*
* l	26	*0.1747	*0.1739	*0.2066	*0.2076	* 2039*

```
*(Y       )*0.1747     *0.1739     *0.2066     *0.2076     *       2039*
*             *           *           *           *           *          *
* 1     27 *0.1760     *0.1784     *0.2138     *0.2109     *       2078*
*(Y       )*0.1760     *0.1784     *0.2138     *0.2109     *       2078*
*             *           *           *           *           *          *
* 1     28 *0.1717     *0.5068     * 1.012     *0.3429     *       6052*
*(X       )*0.1697     *0.6868e-01*0.1206      *0.2981     *        829*
*(Y       )*0.1720     *0.4381     *0.8917     *0.3500     *       5223*
*             *           *           *           *           *          *
* 1     30 *0.1183     *0.2933     *0.4131     *0.1666     *       5082*
*(Y       )*0.1183     *0.2933     *0.4131     *0.1666     *       5082*
*             *           *           *           *           *          *
* 1     32 *0.1706     *0.2476     *0.3352     *0.2309     *       2976*
*(Y       )*0.1706     *0.2476     *0.3352     *0.2309     *       2976*
*             *           *           *           *           *          *
* 1     34 *0.1230     *0.1857     *0.2293     *0.1519     *       3094*
*(Y       )*0.1230     *0.1857     *0.2293     *0.1519     *       3094*
*             *           *           *           *           *          *
* 1     35 *0.1206     *0.1189     *0.1378     *0.1397     *       2021*
*(Y       )*0.1206     *0.1189     *0.1378     *0.1397     *       2021*
*             *           *           *           *           *          *
* 1     38 *0.1223     *0.1249     *0.1410     *0.1381     *       2094*
*(Y       )*0.1223     *0.1249     *0.1410     *0.1381     *       2094*
*             *           *           *           *           *          *
* 1     39 *0.1730     *0.2645     *0.3472     *0.2271     *       3135*
*(Y       )*0.1730     *0.2645     *0.3472     *0.2271     *       3135*
*             *           *           *           *           *          *
* 1     40 *0.1210     *0.1778     *0.2112     *0.1437     *       3012*
*(Y       )*0.1210     *0.1778     *0.2112     *0.1437     *       3012*
*             *           *           *           *           *          *
* 1     42 *0.1183     *0.6613e-01*0.6781e-01*0.1213       *       1146*
*(X       )*0.1193     *0.5592e-01*0.5730e-01*0.1222       *        961*
*(Y       )*0.1132     *0.1022e-01*0.1051e-01*0.1165       *        185*
*             *           *           *           *           *          *
*****************************************************************************
... END OF SIMULATION ...

              MEMORY USED:      25072 WORDS OF 4 BYTES
               ( 10.03  % OF TOTAL MEMORY)
nb of msg=      1790
* * * Delay Distribution:
       0.           0.           0.
   0.1000       0.1229e-01   0.1229e-01
   0.2000       0.4581e-01   0.4581e-01
   0.3000       0.7318e-01   0.7318e-01
   0.4000       0.1112       0.1112
   0.5000       0.1380       0.1380
   0.6000       0.1425       0.1425
   0.7000       0.1302       0.1302
   0.8000       0.9832e-01   0.9832e-01
   0.9000       0.7933e-01   0.7933e-01
    1.000       0.5922e-01   0.5922e-01
    1.100       0.3575e-01   0.3575e-01
    1.200       0.2793e-01   0.2793e-01
    1.300       0.1620e-01   0.1620e-01
    1.400       0.1006e-01   0.1006e-01
    1.500       0.6145e-02   0.6145e-02
    1.600       0.3352e-02   0.3352e-02
    1.700       0.2235e-02   0.2235e-02
```

1.800	0.1117e-02	0.1117e-02
1.900	0.3352e-02	0.3352e-02
2.000	0.1117e-02	0.1117e-02
2.100	0.	0.
2.200	0.5587e-03	0.5587e-03
2.300	0.1117e-02	0.1117e-02
2.400	0.	0.
2.500	0.	0.
2.600	0.5587e-03	0.5587e-03
2.700	0.5587e-03	0.5587e-03
2.800	0.	0.
2.900	0.	0.
3.000	0.	0.
3.100	0.	0.
3.200	0.	0.
3.300	0.	0.
3.400	0.	0.
3.500	0.	0.
3.600	0.	0.
3.700	0.	0.
3.800	0.	0.
3.900	0.	0.
4.000	0.	0.
4.100	0.	0.
4.200	0.	0.
4.300	0.	0.
4.400	0.	0.
4.500	0.	0.
4.600	0.	0.
4.700	0.	0.
4.800	0.	0.
4.900	0.	0.
5.000	0.	0.
5.100	0.	0.
5.200	0.	0.
5.300	0.	0.
5.400	0.	0.
5.500	0.	0.
5.600	0.	0.
5.700	0.	0.
5.800	0.	0.
5.900	0.	0.
6.000	0.	0.
6.100	0.	0.
6.200	0.	0.
6.300	0.	0.
6.400	0.	0.
6.500	0.	0.
6.600	0.	0.
6.700	0.	0.
6.800	0.	0.
6.900	0.	0.
7.000	0.	0.
7.100	0.	0.
7.200	0.	0.
7.300	0.	0.
7.400	0.	0.
7.500	0.	0.
7.600	0.	0.
7.700	0.	0.

```
   7.800          0.          0.
   7.900          0.          0.
   8.000          0.          0.
   8.100          0.          0.
   8.200          0.          0.
   8.300          0.          0.
   8.400          0.          0.
   8.500          0.          0.
   8.600          0.          0.
   8.700          0.          0.
   8.800          0.          0.
   8.900          0.          0.
   9.000          0.          0.
   9.100          0.          0.
   9.200          0.          0.
   9.300          0.          0.
   9.400          0.          0.
   9.500          0.          0.
   9.600          0.          0.
   9.700          0.          0.
   9.800          0.          0.
   9.900          0.          0.
* nb_max=         0
* * * Expected Delay :  0.6754
* * * Variance :  0.1022
* * * Delay Bound:   1.700

   390
   391 /END/
```

A.4 QNAP2 Simulation Programs: the Delay-Bound Case

```
SIMULOG    ***  QNAP2  ***   ( 15-FEB-88 )  V 5.0
(C)  COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1986


 1 &
 2 & Network model B4
 3 & Routing: Use switching algorithms at node1 for class X
 4 & Index for switching: Delay bounds setup for each subnetwork
 5 &
 6 /DECLARE/
 7     INTEGER N=7;                          & number of processing nodes
 8     INTEGER S=4;                          & Backqround traffic sources
 9     INTEGER Ix=2000;                      & number of points for delay distr.
10     INTEGER ix_max=200;                   & printing limit for delay distr.
11     INTEGER nb_max;                       & nb of messages that exceed ix_max
12     INTEGER nb1_max;                      & nb of msgs that exceeds ix_max in
==>    net1
13     INTEGER nb2_max;                      & nb of msgs that exceeds ix_max in
==>    net2
14     INTEGER Nx;                           & number of X messag sent to netwk
15     INTEGER nb_msg;                       & number of X messag received
16     INTEGER nb1_msg;                      & number of X msgs sent to net 1
17     INTEGER nb2_msg;                      & number of X msgs sent to net 2
18     INTEGER ix,jx;
19     INTEGER i,j;
20     INTEGER nb,n_id;
21     REAL xxx=999.99;                      & a large cost for disconn. links
22     REAL warm_up=50.0;                    & system warm-up time
23     REAL t1,t2;                           & indexes for switching
24     REAL lambda;                          & temp var for thruput
25     REAL ct;                              & storage for message delay
26     REAL fx(Ix);                          & the delay distribution for X
27     REAL fx1(Ix);                         & the delay distribution for net1
28     REAL fx2(Ix);                         & the delay distribution for net2
29     REAL ia_time;                         & sources inter_arrival time
30     REAL sva_time;                        & links service time for net 1
31     REAL svb_time;                        & links service time for net 2
32     REAL pc_time;                         & nodes processing time
33     REAL tmax;                            & simulation time
34     REAL jack_pr(N,0:N);                  & the Jacksonial routage array
35     REAL B;                               & the delay bound
36     REAL B_pct=0.99;                      & 99 percentile
37     REAL b1,b2;                           & indices for delay bounds
38     REAL tail,temp;
39     REAL d_mean,d_var,d_bound;            & delay statistics
40 &
41                                           & attributes declaration:
42     QUEUE REAL d_time;                    & nb/lambda for queueing delay
43     CUSTOMER REAL s_time;                 & starting time for a message
44     CUSTOMER INTEGER s_nb;                & sequential number for a message
45     CUSTOMER INTEGER net_id;              & subnet identifier
46 &
47     QUEUE n(N);                           & processing nodes
48     QUEUE s(N);                           & background sources
49     QUEUE l(N,N);                         & communication links
50 &
51     REF CUSTOMER cx;                      & an X message
52 &
53     CLASS X,Y;                            & two classes of traffic
```

```
                                        & X:messages ; Y:background
&
& ------------------------------------------------------------
& Followings are the variabels for shortest path algorithms
& ------------------------------------------------------------
   REAL graph(N,0:N)=( xxx, 0.0; 0.2, 1.0, 1.0, xxx, xxx, xxx,
                       5.0, xxx, 0.0, 0.2, 3.0, 5.0, xxx, xxx,
                       4.0, xxx, 2.0, 0.0, 3.0, xxx, 0.2, xxx,
                       xxx, xxx, 2.0, 2.0, 0.0, 2.0, 2.0, 5.0,
                       1.0, xxx, 5.0, xxx, 3.0, 0.0, 3.0, 2.0,
                       1.0, xxx, xxx, 2.0, 3.0, 3.0, 0.0, 0.2,
                       9.0, xxx, xxx, xxx, xxx, xxx, xxx, xxx);
                                        & the connectivity and the
                                        & cost for each links
                                        & row 0 denotes the out flow

   REAL rout_pr(N,N)=
          (1.0,0.0,0.0,0.0,0.0,0.0,0.0,
           0.0,1.0,0.0,0.0,0.0,0.0,0.0,
           0.0,0.0,1.0,0.0,0.0,0.0,0.0,
           0.0,0.0,0.0,1.0,0.0,0.0,0.0,
           0.0,0.0,0.0,0.0,1.0,0.0,0.0,
           0.0,0.0,0.0,0.0,0.0,1.0,0.0,
           0.0,0.0,0.0,0.0,0.0,0.0,1.0);
                                        & routing probability (i,j) :
                                        & node i destined for node j
                                        & for class X messages
   INTEGER path(N);                     & path array for sp
   REAL d(N,N);                         & evolution array for min cost

&------------------------------------------------------------
& The BELLMAN-FORD algorithm for computing
& the shortest path
&
   PROCEDURE sp1(k1,k2);
      INTEGER k1,k2;                    & the formal parameters
                                        & k:source, l:destination
      REAL m;
      INTEGER i1,j1,n1,h;               & computation variables
   BEGIN
      FOR j1:=1 STEP 1 UNTIL N DO
         BEGIN
            d(1,j1):=graph(k1,j1);
            path(j1):= IF graph(k1,j1)=xxx THEN 0 ELSE k1;
         END;
      FOR h:=2 STEP 1 UNTIL N DO
         FOR j1:=1 STEP 1 UNTIL N DO
            BEGIN
               m:=d(h-1,j1);
               FOR n1:=1 STEP 1 UNTIL N DO
                  IF (d(h-1,n1)+graph(n1,j1))<m THEN
                     BEGIN
                        m:=d(h-1,n1)+graph(n1,j1);
                        path(j1):=n1;
                     END;
               d(h,j1):=m;
            END;
      i1:=k2;
      WHILE (path(i1)<>k1) AND (path(i1)<>0) DO
         BEGIN
```

```
114             FOR j1:=1 STEP 1 UNTIL N DO
115                rout_pr(path(i1),j1):=0;  & clear the row
116             rout_pr(path(i1),i1):=1;
117             i1:=path(i1);
118          END;
119       IF path(i1)=k1 THEN BEGIN
120          FOR j1:=1 STEP 1 UNTIL N DO
121             rout_pr(k1,j1):=0;    & clear the row
122          rout_pr(k1,i1):=1;
123       END;
124
125       PRINT("* * * The Shortest Path Matrix (1) * * *");
126       PRINT("1) The path array:");
127       PRINT(path(1 STEP 1 UNTIL N));
128       PRINT("2) The cost matrix");
129       FOR j1:=1 STEP 1 UNTIL N DO
130          PRINT(d(j1, 1 STEP 1 UNTIL N));
131       PRINT("3) The routing probability matrix:");
132       FOR j1:=1 STEP 1 UNTIL N DO
133          PRINT(rout_pr(j1, 1 STEP 1 UNTIL N));
134
135    END;
136 &
137 &-----------------------------------------------------------------
138 & The shortest path for the second network
139 &
140    PROCEDURE sp2(k1,k2);
141       INTEGER k1,k2;
142       INTEGER i1,j1;
143    BEGIN
144       IF k1=1 THEN
145          BEGIN
146             FOR j1:=1 STEP 1 UNTIL N DO
147                rout_pr(1,j1):=0;
148             rout_pr(1,4):=1.0;
149          END ELSE
150       IF k1=7 THEN
151          BEGIN
152             rout_pr(7,7):=1.0;
153          END ELSE
154       IF k1=4 THEN
155          BEGIN
156             FOR j1:=1 STEP 1 UNTIL N DO
157                rout_pr(4,j1):=0;
158             rout_pr(4,7):=1.0;
159          END
160       ELSE
161          BEGIN
162             FOR i1:=(2,3,5,6) DO
163                BEGIN
164                   FOR j1:=1 STEP 1 UNTIL N DO
165                      rout_pr(i1,j1):=0;
166                   rout_pr(i1,4):=1;
167                END;
168          END;
169
170       PRINT("* * * The Shortest Path Matrix (II)* * *");
171       PRINT("1) The path array:");
172       PRINT(path(1 STEP 1 UNTIL N));
173       PRINT("2) The cost matrix");
```

```
174         FOR i1:=1 STEP 1 UNTIL N DO
175            PRINT(d(i1, 1 STEP 1 UNTIL N));
176         PRINT("3) The routing probability matrix:");
177         FOR i1:=1 STEP 1 UNTIL N DO
178            PRINT(rout_pr(i1,1 STEP 1 UNTIL N));
179      END;
180
181
182  &-----------------------------------------------------------------
183  & Find the Jacksonian routing probability
184  &
185  &
186      PROCEDURE Jackson;
187
188      REAL r_sum;                          & sum of one row in jack_pr
189      INTEGER i1,j1;
190      BEGIN
191         FOR i1:=1 STEP 1 UNTIL N DO
192         BEGIN
193            r_sum:=0;
194            FOR j1:=0 STEP 1 UNTIL N DO
195               r_sum:=r_sum+(IF graph(i1,j1)<>xxx THEN graph(i1,j1)
196                                                  ELSE 0.0);
197            FOR j1:=0 STEP 1 UNTIL N DO
198               jack_pr(i1,j1):=IF graph(i1,j1)<>xxx THEN graph(i1,j1)/r_sum
199                                                    ELSE 0.0;
200         END;
201         PRINT("* * * The Jacksonian Matrix * * *");
202         FOR i1:=1 STEP 1 UNTIL N DO
203         PRINT(jack_pr(i1,0 STEP 1 UNTIL N));
204      END;
205
206
207
208  &
209  &
210  & processing nodes
211  &
212  /STATION/
213      NAME=n(1);
214      SERVICE(X)=
215      BEGIN
216         CST(pc_time);
217         cx:=n(1).FIRST;
218         $MACRO l_delay(node1,node2,sv_time)
219            nb:=l(node1,node2).NB;
220            l(node1,node2).d_time:=nb*sv_time;
221         $END
222         $MACRO n_delay(node)
223            nb:=n(node).NB;
224            n(node).d_time:=nb*pc_time;
225         $END
226         $l_delay(1,2,sva_time) $n_delay(2) $l_delay(2,3,sva_time)
227         $n_delay(3) $l_delay(3,6,sva_time) $n_delay(6)
228         $l_delay(6,7,sva_time) $n_delay(7)
229         $l_delay(1,4,svb_time) $n_delay(4) $l_delay(4,7,svb_time)
230         t1:=l(1,2).d_time+n(2).d_time+l(2,3).d_time+n(3).d_time+l(3,6).d_t
 ==>   ime
231               +n(6).d_time+l(6,7).d_time;
232         t2:=l(1,4).d_time+n(4).d_time+l(4,7).d_time;
```

```
233     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
234     & New codes for A4 algorithm &
235     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
236     IF (b1<=B) AND (b2<=B) THEN
237        IF t1<=t2 THEN
238           BEGIN
239              cx.net_id:=1;
240              TRANSIT(1(1,2));
241           END
242        ELSE
243           BEGIN
244              cx.net_id:=2;
245              TRANSIT(1(1,4));
246           END;
247     IF (b1<=B) AND (b2>B) THEN
248           BEGIN
249              cx.net_id:=1;
250              TRANSIT(1(1,2));
251           END;
252     IF (b1>B) AND (b2<=B) THEN
253           BEGIN
254              cx.net_id:=2;
255              TRANSIT(1(1,4));
256           END;
257     IF (b1>B) AND (b2>B)  THEN
258        BEGIN
259        IF b1 <= b2 THEN
260           BEGIN
261              cx.net_id:=1;
262              TRANSIT(1(1,2));
263           END;
264        IF b1 > b2 THEN
265           BEGIN
266              cx.net_id:=2;
267              TRANSIT(1(1,4));
268           END;
269        END;
270     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
271     END;
272     SERVICE(Y)= CST(pc_time);
273     TRANSIT(Y)=1(1,1 STEP 1 UNTIL N), jack_pr(1, 1 STEP 1 UNTIL N),OUT;
274  /STATION/
275     NAME=n(2);
276     SERVICE=CST(pc_time);
277     TRANSIT(X)=1(2,1 STEP 1 UNTIL N),rout_pr(2,1 STEP 1 UNTIL N);
278     TRANSIT(Y)=1(2,1 STEP 1 UNTIL N),jack_pr(2,1 STEP 1 UNTIL N),OUT;
279  /STATION/
280     NAME=n(3);
281     SERVICE=CST(pc_time);
282     TRANSIT(X)=1(3,1 STEP 1 UNTIL N),rout_pr(3,1 STEP 1 UNTIL N);
283     TRANSIT(Y)=1(3,1 STEP 1 UNTIL N),jack_pr(3,1 STEP 1 UNTIL N),OUT;
284  /STATION/
285     NAME=n(4);
286     SERVICE=CST(pc_time);
287     TRANSIT(X)=1(4,1 STEP 1 UNTIL N),rout_pr(4,1 STEP 1 UNTIL N);
288     TRANSIT(Y)=1(4,1 STEP 1 UNTIL N),jack_pr(4,1 STEP 1 UNTIL N); & no ou
 ==>  t
289  /STATION/
290     NAME=n(5);
291     SERVICE=CST(pc_time);
```

```
292     TRANSIT(X)=1(5,1 STEP 1 UNTIL N),rout_pr(5,1 STEP 1 UNTIL N);
293     TRANSIT(Y)=1(5,1 STEP 1 UNTIL N),jack_pr(5,1 STEP 1 UNTIL N),OUT;
294  /STATION/
295     NAME=n(6);
296     SERVICE=CST(pc_time);
297     TRANSIT(X)=1(6,1 STEP 1 UNTIL N),rout_pr(6,1 STEP 1 UNTIL N);
298     TRANSIT(Y)=1(6,1 STEP 1 UNTIL N),jack_pr(6,1 STEP 1 UNTIL N),OUT;
299  /STATION/
300     NAME=n(7);
301     SERVICE=
302     BEGIN
303        CST(pc_time);
304        cx:=n(7).FIRST;
305        IF cx.CCLASS=X THEN
306        BEGIN
307           ct:=TIME-cx.s_time;
308        &  PRINT("MESSAGE:",cx.s_nb,"delay=",ct);
309           ix:=INTREAL(ct*10)*2+1;
310           IF ix>Ix-1 THEN ix:=Ix-1;  & (fx(1999),fx(2000)) is the last pa
 ==>  ir
311           IF ix<1 THEN ix:=1;
312           IF ix>=ix_max-1 THEN nb_max:=nb_max+1; & the delay that exceeds
 ==>   ix_max
313           nb_msg:=nb_msg+1;
314           fx(ix+1):=fx(ix+1)+1;
315           &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
316           & New codes for A4 algorithm &
317           &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
318           n_id:=cx.net_id;
319           IF n_id=1 THEN
320           BEGIN
321              nb1_msg:=nb1_msg+1;
322              fx1(ix+1):=fx1(ix+1)+1;
323              IF ix>=ix_max-1 THEN nb1_max:=nb1_max+1;
324              tail:=REALINT(nb1_msg)*(1-B_pct);
325              temp:=nb1_max;
326              jx:=ix_max;
327              WHILE (temp<tail) AND (jx>=1) DO
328              BEGIN
329                 temp:=temp+fx1(jx);
330                 jx:=jx-2;
331              END;
332              b1:=REALINT(jx)/20.0;
333           END;
334           IF n_id=2 THEN
335           BEGIN
336              nb2_msg:=nb2_msg+1;
337              fx2(ix+1):=fx2(ix+1)+1;
338              IF ix>=ix_max-1 THEN nb2_max:=nb2_max+1;
339              tail:=REALINT(nb2_msg)*(1-B_pct);
340              temp:=nb2_max;
341              jx:=ix_max;
342              WHILE (temp<tail) AND (jx>=1) DO
343              BEGIN
344                 temp:=temp+fx2(jx);
345                 jx:=jx-2;
346              END;
347              b2:=REALINT(jx)/20.0;
348           END;
349        &  PRINT("* * * message:",cx.s_nb,"b1=",b1,"b2=",b2);
```

```
350         &  PRINT("received from net",n_id);
351         &  PRINT("tail:",tail,"temp=",temp);
352            &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
353         END;
354      END;
355      TRANSIT=OUT;
356 &
357 & background source nodes
358 &
359 /STATION/
360      NAME=s(1);
361      TYPE=SOURCE;
362      SERVICE=
363      BEGIN
364         EXP(ia_time);
365         & X is created only when system is warm-up
366         IF (TIME>warm_up) AND DRAW(0.45) THEN
367         BEGIN
368            cx:=s(1).FIRST;
369            cx.s_time:=TIME;
370            cx.s_nb:=Nx;
371            Nx:=Nx+1;
372            TRANSIT(n(1),X);
373         END
374         ELSE TRANSIT(n(1),Y);
375      END;
376 /STATION/
377      NAME=s(2);
378      TYPE=SOURCE;
379      SERVICE=EXP(ia_time);
380      TRANSIT=n(2),Y,1;
381 /STATION/
382      NAME=s(3);
383      TYPE=SOURCE;
384      SERVICE=EXP(ia_time);
385      TRANSIT=n(3),Y,1;
386 /STATION/
387      NAME=s(5);
388      TYPE=SOURCE;
389      SERVICE=EXP(ia_time);
390      TRANSIT=n(5),Y,1;
391 /STATION/
392      NAME=s(6);
393      TYPE=SOURCE;
394      SERVICE=EXP(ia_time);
395      TRANSIT=n(6),Y,1;
396 &
397 & comm. links
398 &
399      $MACRO link(s_node,d_node,s_time) & definition for communnication
400      /STATION/ NAME=l(s_node,d_node);  & link stations
401                SERVICE=EXP(s_time);
402                TRANSIT=n(d_node),1;
403      $END
404 &
405 $link(1,2,sva_time)   $link(1,3,sva_time)
406 $link(2,3,sva_time)   $link(2,5,sva_time)
407 $link(3,2,sva_time)   $link(3,6,sva_time)
408 $link(5,2,sva_time)   $link(5,6,sva_time)   $link(5,7,sva_time)
409 $link(6,3,sva_time)   $link(6,5,sva_time)   $link(6,7,sva_time)
```

```
410 &
411 $link(1,4,svb_time)   $link(4,7,svb_time)
412 $link(2,4,svb_time)   $link(3,4,svb_time)   $link(5,4,svb_time)
413 $link(6,4,svb_time)
414 $link(4,2,svb_time)   $link(4,3,svb_time)   $link(4,5,svb_time)
415 $link(4,6,svb_time)
416 &
417 /CONTROL/
418    CLASS=ALL QUEUE;
419    ENTRY=BEGIN
420       PRINT("Inter-arrival time?"); ia_time:=GET(REAL);
421       PRINT("Service time for net 1?"); sva_time:=GET(REAL);
422       PRINT("Service time for net 2?"); svb_time:=GET(REAL);
423       PRINT("Processing time?");     pc_time:=GET(REAL);
424       PRINT("Delay bound?"); B:=GET(REAL);
425       PRINT("Simulation time?");     tmax:=GET(REAL);
426       FOR ix:=1 STEP 2 UNTIL Ix-1 DO  & initializing the delay distr. ar
 ==>  ray
427          fx(ix):=REALINT(ix-1)/20.0;
428    END;
429    TMAX=tmax;
430    EXIT=BEGIN    & print the delay distribution
431       PRINT("nb of msg=",nb_msg);
432       PRINT("* * * Delay Distribution:");
433       FOR ix:=1 STEP 2 UNTIL ix_max-1 DO BEGIN
434          fx(ix+1):=fx(ix+1)/REALINT(nb_msg);
435          PRINT(fx(ix),fx(ix+1),CURVE(fx,fx(ix)));
436          d_mean:=d_mean+fx(ix)*fx(ix+1);
437          d_var:=d_var+fx(ix)*fx(ix)*fx(ix+1);
438       END;
439       d_var:=d_var - d_mean*d_mean;
440       temp:=REALINT(nb_max)/REALINT(nb_msg);
441       jx:=ix_max;
442       WHILE (temp<0.01)*AND (jx>=1) DO
443       BEGIN
444          temp:=temp+fx(jx);
445          jx:=jx-2;
446       END;
447       d_bound:=REALINT(jx)/20.0;
448       PRINT("* nb_max=",nb_max);
449       PRINT("* * * Expected Delay :",d_mean);
450       PRINT("* * * Variance :",d_var);
451       PRINT("* * * Delay Bound:",d_bound);
452    END;
453 /EXEC/
454 BEGIN
455    sp1(1,7);      & find the shortest path in network 1
456    sp2(4,7);      & find the shortest path in network 2
457    Jackson;       & find the routing prob. for Y
458
459    PRINT("* * * Simulation result for single SHORTEST PATH * * *");
460    SIMUL;
461 END;

* * * The Shortest Path Matrix (1) * * *
1) The path array:
      1         1         2         1         4         3         6
2) The cost matrix
     0.      0.2000     1.000     1.000     1000.0    1000.0    1000.0
     0.      0.2000     0.4000    1.000     3.000     1.200     6.000
```

```
      0.       0.2000       0.4000       1.000        3.000        0.6000       1.400
      0.       0.2000       0.4000       1.000        3.000        0.6000       0.8000
      0.       0.2000       0.4000       1.000        3.000        0.6000       0.8000
      0.       0.2000       0.4000       1.000        3.000        0.6000       0.8000
      0.       0.2000       0.4000       1.000        3.000        0.6000       0.8000
3) The routing probability matrix:
      0.       1.000          0.           0.           0.           0.           0.
      0.         0.         1.000          0.           0.           0.           0.
      0.         0.           0.           0.           0.         1.000          0.
      0.         0.           0.         1.000          0.           0.           0.
      0.         0.           0.           0.         1.000          0.           0.
      0.         0.           0.           0.           0.           0.         1.000
      0.         0.           0.           0.           0.           0.         1.000
* * * The Shortest Path Matrix (II)* * *
1) The path array:
       1          1          2          1          4          3          6
2) The cost matrix
      0.       0.2000        1.000        1.000       1000.0       1000.0       1000.0
      0.       0.2000       0.4000        1.000        3.000        1.200        6.000
      0.       0.2000       0.4000        1.000        3.000       0.6000        1.400
      0.       0.2000       0.4000        1.000        3.000       0.6000       0.8000
      0.       0.2000       0.4000        1.000        3.000       0.6000       0.8000
      0.       0.2000       0.4000        1.000        3.000       0.6000       0.8000
      0.       0.2000       0.4000        1.000        3.000       0.6000       0.8000
3) The routing probability matrix:
      0.       1.000          0.           0.           0.           0.           0.
      0.         0.         1.000          0.           0.           0.           0.
      0.         0.           0.           0.           0.         1.000          0.
      0.         0.           0.           0.           0.           0.         1.000
      0.         0.           0.           0.         1.000          0.           0.
      0.         0.           0.           0.           0.           0.         1.000
      0.         0.           0.           0.           0.           0.         1.000
* * * The Jacksonian Matrix * * *
      0.         0.       0.9091e-01    0.4545       0.4545         0.           0.
  0.3788         0.         0.          0.1515e-01    0.2273       0.3788         0.
  0.4348         0.       0.2174          0.          0.3261         0.         0.2174e
      0.         0.       0.1538        0.1538          0.         0.1538       0.1538
  0.7143e-01     0.       0.3571          0.          0.2143         0.         0.2143
  0.1087         0.         0.          0.2174        0.3261       0.3261         0.
   1.000         0.         0.            0.            0.           0.           0.
* * * Simulation result for single SHORTEST PATH * * *

Inter-arrival time?
     1 0.5
Service time for net 1?
     2 0.12
Service time for net 2?
     3 0.17
Processing time?
     4 0.05
Delay bound?
     5 2.0
Simulation time?
     6 2050

*** SIMULATION ***
... TIME =      2050.00
************************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
************************************************************************
```

*		*	*	*	*	*
* n	1	*0.5002e-01	*0.9690e-01	*0.1018	*0.5257e-01	3971*
*(X		)*0.5002e-01	*0.4288e-01	*0.4496e-01	*0.5245e-01	1757*
*(Y		)*0.5002e-01	*0.5402e-01	*0.5688e-01	*0.5267e-01	2214*
*		*	*	*	*	*
* n	2	*0.5002e-01	*0.3567	*0.4516	*0.6332e-01	14619*
*(X		)*0.5002e-01	*0.2926e-01	*0.3652e-01	*0.6243e-01	1199*
*(Y		)*0.5002e-01	*0.3275	*0.4150	*0.6340e-01	13420*
*		*	*	*	*	*
* n	3	*0.5002e-01	*0.2630	*0.3092	*0.5881e-01	10779*
*(X		)*0.5002e-01	*0.2926e-01	*0.3437e-01	*0.5876e-01	1199*
*(Y		)*0.5002e-01	*0.2338	*0.2748	*0.5881e-01	9580*
*		*	*	*	*	*
* n	4	*0.5002e-01	*0.3406	*0.4257	*0.6252e-01	13957*
*(X		)*0.5002e-01	*0.1362e-01	*0.1649e-01	*0.6057e-01	558*
*(Y		)*0.5002e-01	*0.3270	*0.4092	*0.6260e-01	13399*
*		*	*	*	*	*
* n	5	*0.5002e-01	*0.3435	*0.4331	*0.6306e-01	14079*
*(Y		)*0.5002e-01	*0.3435	*0.4331	*0.6306e-01	14079*
*		*	*	*	*	*
* n	6	*0.5002e-01	*0.2577	*0.3034	*0.5887e-01	10562*
*(X		)*0.5002e-01	*0.2926e-01	*0.3469e-01	*0.5931e-01	1199*
*(Y		)*0.5002e-01	*0.2285	*0.2687	*0.5882e-01	9363*
*		*	*	*	*	*
* n	7	*0.5002e-01	*0.2223	*0.2554	*0.5747e-01	9112*
*(X		)*0.5002e-01	*0.4287e-01	*0.4948e-01	*0.5774e-01	1757*
*(Y		)*0.5002e-01	*0.1795	*0.2059	*0.5740e-01	7355*
*		*	*	*	*	*
* s	1	*0.5161	* 1.000	* 1.000	*0.5161	* 3972*
*		*	*	*	*	*
* s	2	*0.5118	* 1.000	* 1.000	*0.5118	* 4005*
*		*	*	*	*	*
* s	3	*0.4875	* 1.000	* 1.000	*0.4875	* 4204*
*		*	*	*	*	*
* s	5	*0.5120	* 1.000	* 1.000	*0.5120	* 4004*
*		*	*	*	*	*
* s	6	*0.5004	* 1.000	* 1.000	*0.5004	* 4097*
*		*	*	*	*	*
* l	2	*0.1220	*0.8267e-01	*0.8983e-01	*0.1326	* 1389*
*(X		)*0.1220	*0.7135e-01	*0.7753e-01	*0.1326	* 1199*
*(Y		)*0.1221	*0.1131e-01	*0.1229e-01	*0.1327	* 190*
*		*	*	*	*	*
* l	3	*0.1255	*0.6344e-01	*0.6686e-01	*0.1323	* 1036*
*(Y		)*0.1255	*0.6344e-01	*0.6686e-01	*0.1323	* 1036*
*		*	*	*	*	*
* l	4	*0.1661	*0.1254	*0.1374	*0.1821	* 1545*
*(X		)*0.1611	*0.4385e-01	*0.4463e-01	*0.1640	* 558*
*(Y		)*0.1690	*0.8154e-01	*0.9276e-01	*0.1923	* 987*
*		*	*	*	*	*
* l	10	*0.1273	*0.8716e-01	*0.9386e-01	*0.1370	* 1404*
*(X		)*0.1260	*0.7368e-01	*0.7932e-01	*0.1356	* 1199*
*(Y		)*0.1348	*0.1348e-01	*0.1454e-01	*0.1454	* 205*
*		*	*	*	*	*
* l	11	*0.1697	*0.2640	*0.3598	*0.2312	* 3190*
*(Y		)*0.1697	*0.2640	*0.3598	*0.2312	* 3190*
*		*	*	*	*	*
* l	12	*0.1188	*0.2907	*0.3978	*0.1626	* 5016*
*(Y		)*0.1188	*0.2907	*0.3978	*0.1626	* 5016*
*		*	*	*	*	*
* l	16	*0.1267	*0.1276	*0.1443	*0.1432	* 2065*

```
*(Y       )*0.1267    *0.1276    *0.1443    *0.1432    *       2065*
*            *          *          *          *          *          *
* 1     18 *0.1731    *0.2722    *0.3601    *0.2290    *       3224*
*(Y       )*0.1731    *0.2722    *0.3601    *0.2290    *       3224*
*            *          *          *          *          *          *
* 1     20 *0.1132    *0.7844e-01*0.8409e-01*0.1214    *       1420*
*(X       )*0.1138    *0.6657e-01*0.7171e-01*0.1226    *       1199*
*(Y       )*0.1101    *0.1187e-01*0.1238e-01*0.1149    *        221*
*            *          *          *          *          *          *
* 1     23 *0.1705    *0.1715    *0.2140    *0.2128    *       2062*
*(Y       )*0.1705    *0.1715    *0.2140    *0.2128    *       2062*
*            *          *          *          *          *          *
* 1     24 *0.1731    *0.1751    *0.2053    *0.2030    *       2074*
*(Y       )*0.1731    *0.1751    *0.2053    *0.2030    *       2074*
*            *          *          *          *          *          *
* 1     26 *0.1741    *0.1715    *0.2019    *0.2050    *       2018*
*(Y       )*0.1741    *0.1715    *0.2019    *0.2050    *       2018*
*            *          *          *          *          *          *
* 1     27 *0.1747    *0.1759    *0.2155    *0.2141    *       2063*
*(Y       )*0.1747    *0.1759    *0.2155    *0.2141    *       2063*
*            *          *          *          *          *          *
* 1     28 *0.1671    *0.4678    *0.8052    *0.2876    *       5739*
*(X       )*0.1602    *0.4361e-01*0.5613e-01*0.2062    *        558*
*(Y       )*0.1678    *0.4242    *0.7491    *0.2964    *       5181*
*            *          *          *          *          *          *
* 1     30 *0.1213    *0.3016    *0.4231    *0.1701    *       5098*
*(Y       )*0.1213    *0.3016    *0.4231    *0.1701    *       5098*
*            *          *          *          *          *          *
* 1     32 *0.1688    *0.2435    *0.3140    *0.2176    *       2958*
*(Y       )*0.1688    *0.2435    *0.3140    *0.2176    *       2958*
*            *          *          *          *          *          *
* 1     34 *0.1218    *0.1773    *0.2190    *0.1504    *       2984*
*(Y       )*0.1218    *0.1773    *0.2190    *0.1504    *       2984*
*            *          *          *          *          *          *
* 1     35 *0.1167    *0.1136    *0.1284    *0.1319    *       1993*
*(Y       )*0.1167    *0.1136    *0.1284    *0.1319    *       1993*
*            *          *          *          *          *          *
* 1     38 *0.1199    *0.1206    *0.1368    *0.1361    *       2061*
*(Y       )*0.1199    *0.1206    *0.1368    *0.1361    *       2061*
*            *          *          *          *          *          *
* 1     39 *0.1722    *0.2556    *0.3371    *0.2270    *       3041*
*(Y       )*0.1722    *0.2556    *0.3371    *0.2270    *       3041*
*            *          *          *          *          *          *
* 1     40 *0.1202    *0.1783    *0.2153    *0.1451    *       3041*
*(Y       )*0.1202    *0.1783    *0.2153    *0.1451    *       3041*
*            *          *          *          *          *          *
* 1     42 *0.1192    *0.8023e-01*0.8467e-01*0.1258    *       1380*
*(X       )*0.1189    *0.6956e-01*0.7356e-01*0.1258    *       1199*
*(Y       )*0.1209    *0.1067e-01*0.1111e-01*0.1258    *        181*
*            *          *          *          *          *          *
**************************************************************************
... END OF SIMULATION ...


              MEMORY USED:       29265 WORDS OF 4 BYTES
                ( 11.71  % OF TOTAL MEMORY)
nb of msg=      1757
* * * Delay Distribution:
       0.          0.          0.
  0.1000       0.8537e-02  0.8537e-02
```

0.2000	0.4269e-01	0.4269e-01
0.3000	0.7114e-01	0.7114e-01
0.4000	0.1059	0.1059
0.5000	0.1360	0.1360
0.6000	0.1394	0.1394
0.7000	0.1309	0.1309
0.8000	0.1167	0.1167
0.9000	0.9106e-01	0.9106e-01
1.000	0.5919e-01	0.5919e-01
1.100	0.3984e-01	0.3984e-01
1.200	0.2163e-01	0.2163e-01
1.300	0.1366e-01	0.1366e-01
1.400	0.7399e-02	0.7399e-02
1.500	0.6261e-02	0.6261e-02
1.600	0.5122e-02	0.5122e-02
1.700	0.2277e-02	0.2277e-02
1.800	0.1707e-02	0.1707e-02
1.900	0.	0.
2.000	0.5692e-03	0.5692e-03
2.100	0.	0.
2.200	0.	0.
2.300	0.	0.
2.400	0.	0.
2.500	0.	0.
2.600	0.	0.
2.700	0.	0.
2.800	0.	0.
2.900	0.	0.
3.000	0.	0.
3.100	0.	0.
3.200	0.	0.
3.300	0.	0.
3.400	0.	0.
3.500	0.	0.
3.600	0.	0.
3.700	0.	0.
3.800	0.	0.
3.900	0.	0.
4.000	0.	0.
4.100	0.	0.
4.200	0.	0.
4.300	0.	0.
4.400	0.	0.
4.500	0.	0.
4.600	0.	0.
4.700	0.	0.
4.800	0.	0.
4.900	0.	0.
5.000	0.	0.
5.100	0.	0.
5.200	0.	0.
5.300	0.	0.
5.400	0.	0.
5.500	0.	0.
5.600	0.	0.
5.700	0.	0.
5.800	0.	0.
5.900	0.	0.
6.000	0.	0.
6.100	0.	0.

```
    6.200           0.          0.
    6.300           0.          0.
    6.400           0.          0.
    6.500           0.          0.
    6.600           0.          0.
    6.700           0.          0.
    6.800           0.          0.
    6.900           0.          0.
    7.000           0.          0.
    7.100           0.          0.
    7.200           0.          0.
    7.300           0.          0.
    7.400           0.          0.
    7.500           0.          0.
    7.600           0.          0.
    7.700           0.          0.
    7.800           0.          0.
    7.900           0.          0.
    8.000           0.          0.
    8.100           0.          0.
    8.200           0.          0.
    8.300           0.          0.
    8.400           0.          0.
    8.500           0.          0.
    8.600           0.          0.
    8.700           0.          0.
    8.800           0.          0.
    8.900           0.          0.
    9.000           0.          0.
    9.100           0.          0.
    9.200           0.          0.
    9.300           0.          0.
    9.400           0.          0.
    9.500           0.          0.
    9.600           0.          0.
    9.700           0.          0.
    9.800           0.          0.
    9.900           0.          0.
* nb_max=          0
* * * Expected Delay :  0.6744
* * * Variance :  0.8476e-01
* * * Delay Bound:   1.500

   462
   463 /END/
```

REFERENCES

[1] L. Kleinrock, *Communication Nets - Stochastic Message Flow and Delay*, McGraw-Hill, NY, 1964.

[2] L. Kleinrock, *Queueing System, Vol.1: Theory*, Wiley- Interscience, NY, 1975.

[3] L. Kleinrock, *Queueing System, Vol.2: Computer Applications* , Wiley Interscience, NY, 1975.

[4] D. Bertsekas and R. G. Gallager, *Data Networks*, Prentice-Hall, New Jersey, 1987.

[5] M. Schwartze, *Telecommunication Networks*, Addison-Wesley, New York, 1987.

[6] R. J. Cypser, *Communications Architecture for Distributed Systems*, Addison Wesley, NY, 1978.

[7] G. D. Schultz, D. B. Rose, C. H. West and J. P. Gray, *"Executable Description and Validation of SNA"*, IEEE Transaction on Communication, vol.COM-28, no.4, Apr.,1980.

[8] P. E. Green, *Computer Network Architecture and Protocols*, Plenum:New York, 1982.

[9] J. W. Wong and S. S. Lam *"Queueing Network models of pachet-switching networks"*, Proc. Nat. Telecomm. Conf., Houston, TX, 1980.

[10] J. W. Wong, *"Distribution of end-to-end delay in message switched networks"*, Computer Network, vol.2, 1978.

[11] F. Baskett, K. M. Chandy, R. R. Muntz and F. Palacios, *"Open, Closed and Mixed Networks of Queues with Different Classes of Customers"*, JACM, 22, 1975.

[12] R. L. Disney, R. L. Farrell and P. R. Morais, *"A characterization of M/G/1 queues with renewal departures"*, Management Sci., 20, 1973.

[13] E. Reich, *"Waiting Times when Queues are in Tandem"*, Ann. Math. Statistics, 28, 1957.

[14] E. Reich, *"Notes on Queues in Tandem"*, Ann. Math. Statistics, 34, 1963.

[15] G. L. Fultz, *"Adaptive Routing Techniques for Message Switching Networks"*, Ph.D. dissertation, Dept. of Computer Sci., Univ. of California, Los Angelos, July, 1972.

[16] D. G. Cantor and M. Gerla, *"Optimal Routing in a packet Switching Computer Network"*, IEEE Tranc. Comput., vol.23, Oct.,1974.

[17] L. Kleinrock and S. S. Lam, *"Packet-Switching in a Slotted Satellite Channel"*, in AFIPS Nat. Comput. Conf. Proc., vol.42, June 1973.

[18] S. S. Lam, *"Packet-switching in a Multi-access Broadcast Channel with Application to Satellite Communication in a Computer Network"*, Ph.D. dissertation, Dept. of Compt. Sci., Univ. of California, Los Angeles, Apr., 1974.

[19] N. Abramson, *"The ALOHA System"*, Computer Communication Networks, N. Abramson and F. F. Kuo, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1973.

[20] F. F. Kuo and N. Abramson, *"Some Advances in Radio Communications for Computers"*, in Dig. Papers–COMPCON '73, Sancisco, CA, Feb. 1973.

[21] P. Kaiser, J. Midwinter and S. Shimada, *"Status and Future Trends in Terrestrial Optical Fiber Systems in North America, Europe and Japan"*, IEEE Commun. Mag., vol.25, Oct., 1987.

[22] L. Fratta, M. Gerla and L. Kleinrock, *"The Flow Deviation Methods - An Approach to Store and Forward Communication Network Design"*, Networks, 3, 1973.

[23] W. Chou and H. Frank, *"Routing Strategies for Computer Network Design"*, Symp. Computer Commun. Networks, Brooklyne, New York, Apr., 1972.

[24] J. R. Jackson, *"Job Shop-Like Queueing System"*, Management Sci., vol.10, Oct.,1963.

[25] H. Kobayashi and M Reiser, *"On Generalization of Job Routing behavior in Queueing Network Model"*, Res. Report RC5679, IBM Thomas J. Watson Res. Center, Yorktown Heights, NY, Oct., 1975.

[26] L. Kleinrock, *"A Conservation Law for a wide class of Queueing Disciplines"*, Naval Res. Log. Quart., vol.12, 1965.

[27] M. Gerla, W. Chou and H. Frank, *"Cost-Throughput treads in Computer Networks using satellite communications"*,Proc. Int. Conf. Commun., Minneapolis, MN, 1974.

[28] D. Huynh, H. Kobayashi and F. F. Kuo, *"Optimal Design of Mixed-Media Packet Switching Networks: Routing and Capacity Assignment"*, IEEE Tranc. Commun. com-25, no.1, Jan., 1977.

[29] K. Maruyama, *"Optimization of Mixed-Media Communication Networks"*, Computer Networks, no.2, 1978.

[30] D. Huynh, H. Kobayashi and F. F. Kuo, *"Design issues for Mixed-Media Packet Switching Networks"*,

[31] D. L. A. Barber and D. W. Davies, *"The NPL Data Network"*, Proc. Conf. Laboratory Automation, Novosibiribrsk, U.S.S.R., Oct.,1970.

[32] S. Butterfield, R. Rettberg and D. Walden, *"The Satellite IMP for the ARPA Network"*, in Proc. 7th Hawaii Int. Conf. System Sciences–Subcon. Comput. Nets, Jan, 1974.

[33] B. D. Wesler and R. B. Hovey, *"Public Packet-Switching Networks"*, Datamation, July 1974.

[34] CONTEL/ Federal System, *"System Specification for the Military Airlift Command(MAC) Command and Control(C2) Information Processing Systems(IPS)"*, IPS-S-102, March 1987.

[35] N. Abramson, *"The Aloha System – Another Alternative For Computer Communications"*, Proc. Fall Joint Computer Conf., AFIPS, Conf., 37, 1970.

[36] M. Gerla and L. Kleinrock, *"On the Topological Design of Distributed Computer Networks"*, IEEE Tranc. Commun., COM-25:48-60, 1977.

[37] A. Kershenbaum and R. R. Boorstyn, *"Centralized Teleprocessing Network Design"*, Networks, 13, 1983.

[38] C. L. Monma and D. D. Sheng, *"Backbone Network Design and Performance Analysis: A Methodology for Packet Switching Networks"*, IEEE J. Select. Areas: Commun., SAC-4. 1986.

[39] T. P. Yum, *"The Design and Analysis of a Semi-dynamic Deterministic Routing Rule"*, IEEE Tranc. Commum., vol. COM-29, no.4, Apr., 1981.

[40] R. G. Gallager, *"A Minimun Delay Routing Algorithm Using Distributed Computation"*, IEEE Tranc. Commun., vol. com-25, No.1, Jan., 1977.

[41] H. Rudin, *"On Routing and Delta-Routing: A Taxonomy and Performance Comparison of Techniques for Packet-Switching Networks."*, IEEE Tranc. Commun., vol. com-24, No.1, Jan., 1977.

[42] R. R. Boorstyn and A. Livne, *"A Technique for Adaptive Routing in Networks"*, IEEE Tranc. Commun., com-29, no.4, Apr., 1981.

[43] A. K. Agrawala, E. G. Coffman, J. R. Garey and S. K. Tripathi , *"A Stochastic Optimization Algorithm Minimizing Expected Flow Times on Uniform Processors"*, Preprint, 1983.

[44] R. L. Larsen, *"Control of Multiple Exponential Servers With Applications to Computer Systems"*, Ph.D. Thesis, Tech. Rep. no. TR-1041, University of Maryland, College.Park, Apr., 1981.

[45] I. Viniotis and A. Ephremides, *"Extensions of Optimality of the Threshold Policy in Heterogeneous Multiserver Queueing Systems"*, Tech. Rep., TR-87-15, System Res. Center, Univ. of Maryalnd, College Park, 1987.

[46] A. Ephremides, *Communication Networks: Lecture Notes*, Electrical Engineering Dept., Univ. of Maryland, College Park, Spring 1988.

[47] A. Ephremides, P. Varaiya and J. Walrand, *"A Simple Dynamic Routing Problem"*, IEEE Tranc. Automatic Control, vol.ac-25, no.4, Aug., 1980.

[48] T. P. Yum and M. Schwartz, *"The Join-Biased-Queue Rule and its Application to Routing in Computer Communication Networks"*, IEEE Tranc. Commun., vol.com-29, no.4, Apr., 1981.

[49] P. R. Kumar and J. Walrand, *"Individual Optimal Routing in Parallel Systems"*, SIAM J. Control and Optimization, May, 1984.

[50] T. P. Yum and H. C. Lin, *"Adaptive Load Balancing for Parallel Queues with Traffic Constrains"*,IEEE Tranc. Commun., vol.com- 32, no.32, Dec., 1984.

[51] W. Chou, A. W. Bragg and A. A. Nilsson, *"The Need for Adaptive Routing in Chaotic and Unbalanced Traffic Environment"*, IEEE Tranc. Commun., vol. com-29, No.4, Apr., 1981.

[52] J. L. Yuan, *"A Dynamic Routing Algorithms for Multimedia Networks"*,term paper, ENEE426/Univ. of Maryland, College Park, Spring, 1988.

[53] P. P. Varaiya, J. C. Walrand and C. Buyukkoc, " *Extensions of the Multiarmed Bandit Problem: The Discounted Case"*, IEEE Tranc. Auto. Control. vol.ac-30, no.5, May, 1985.

[54] J. C. Gittins, *"Bandit Process and Dynamic Allocation Indices"*,J. Roy. Statistics Soc., vol.41, 1979.

[55] J. C. Gittins and D. M. Jones, *"A Dynamic Allocation Index for Sequential Design of Experiments"*, Progress in Statistics, Euro. Meet. Statis., vol.1, 1972.

[56] P. Varaiya and J. Walrand, *"Interconnection of Marcov Chains and Quasi-Reversible Queueing Networks"*, Adv. Appl. Prob., 12, 1980.

[57] J. Walrand,*An Introduction to Queueing Networks*, Prentice-Hall, New Jersey, 1988.

[58] J. Walrand, *"Sojourn Times and the Overtaking Conditions in Jackson Networks"*, Adv. Appl. Prob., 12, 1980.

[59] H. Daduna, *"Passage Times for Overtake -Free Paths in Gordon-Newell Networks"*, Adv. Appl. Prob., 14, 1982.

[60] B. Melamed, *"Sojourn Times in Queueing Networks"*, Opera. Res., 7, 1982.

[61] J. D. C. Little, *"A Proof of the Queueing Formula $L = \lambda W$"*, Oper. Res., 9, 1961.

[62] S. D. Conte and C. Boor, *Elementary Numerical Analysis -An Algorithm Approach*,-3rd Ed., 1980.

[63] J. W. Cooley and J. W. Tukey, *"An Algorithm for the Machine Calculation of Complex Fourier Series"*, Math. of Comput., 19, 1965.

[64] W. Kaplan, *Advanced Mathematics for Engineers*, Addison-Wesley, 1981.

[65] E. Horowitze and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Series Press, Inc., 1982.

[66] W. Lin nd P. R. Kumar, *"Optimal Control of a Queueing System with Two Heterogeneous Servers"*, IEEE Tranc. Automatic Control, 1984.

[67] *QNAP2 Reference Manual*, Institut National de Recherchee en Informatique et en Automatique(INTRA), France, 1984.

[68] G. Bilardi and M Sarrafzadeh, *"Optimal Discrete Fourier Transform in VLSI"*, Proc. International Workshop on Parallel Computing and VLSI., Amalfi, Italy, May 1984.

[69] J. Ja Ja, *"High Speed VLSI Network for Computing the Discrete Fourier Transform"*, Conf. on Advanced Research in VLSI, Boston, MA., 1984.

[70] H. T. Kung, *"Why Systolic Architecture?"*, Computer Magazine, 37-46., 1982.

[71] K. J. Maclish, *"Mapping the Integration of the Artificial Intelligence into Telecommunications"*, IEEE J. Select. Areas in Commun., vol.6, no.5, June 1988.

[72] K. J. Macleish, S. Thiedke and D. Vennergrund, *"Expert System in Central Office Switch Maintenance"*, IEEE Commun. Mag., Sep., 1986.