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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Some Related Models and Background Review</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.1 The Heterogeneous-Server Queueing Model</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.2 The Multiple Output-Queues Model</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>2.3 The Multiple Sub-networks Model</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>Derivation of Indices for Messages Switching</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>3.1 The Analytical Model</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>3.2 Single-Path Delay Distribution</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>3.3 Delay Bounds for two Classes of Routing Algorithms</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>The Computation of Indices and the Switching Algorithms</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>4.1 Computation of Indices</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>4.2 The Switching Algorithms</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>Simulation Results</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>5.1 The Simulation Scenario</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>5.2 Simulations for the Expected-Delay Indices</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>5.3 Simulations for the Delay-Bound Indices</td>
<td>68</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>81</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
<td>83</td>
</tr>
<tr>
<td>A.1</td>
<td>Proofs of Lemma 3.3</td>
<td>83</td>
</tr>
<tr>
<td>A.2</td>
<td>Proofs of Theorem 4.1</td>
<td>84</td>
</tr>
<tr>
<td>A.3</td>
<td>QNAP2 Simulation Programs: the Expected Delay Case</td>
<td>86</td>
</tr>
<tr>
<td>A.4</td>
<td>QNAP2 Simulation Programs: the Delay Bound Case</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>111</td>
</tr>
</tbody>
</table>

iii
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
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..) feedback 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-
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, responding 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].
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-
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.
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 $\mu_2$) should be utilized only when the number of customers waiting in the buffer is greater than $T$ where $T$ is given by:

$$T = \frac{\mu_1}{\mu_2} - 1$$  \hspace{1cm} (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 $\tau$, such that the second media(link) should
be utilized only when the number of jobs in the buffer exceeds \( \tau \). This is proved by Lin and Kumar[66]. They also show that as \( \tau \) increases and the rate \( \lambda \) of Poisson arrivals decreases \( \tau \) will converge to (2.1) as \( \lambda \to 0 \). It is conjectured that this result can also be extended to the cases \( s \geq 2 \)(see [49]).

Recently, Viniotis and Epherides[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.
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 \( q_2 \).
The optimal flows are given by (see [4]):

\[
\begin{align*}
  x_1 &= r \\
  x_2 &= 0 \\
  x_1 &= \frac{\sqrt{C_1}}{\sqrt{C_1+C_2}}(r - (C_1 + C_2)) + C_1 \\
  x_2 &= \frac{\sqrt{C_2}}{\sqrt{C_1+C_2}}(r - (C_1 + C_2)) + C_2 
\end{align*}
\]  

for \( r < C_1 - \sqrt{C_1C_2} \) \( \quad \) (2.2) \( \geq \) \( \)

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

\[
\begin{align*}
  p_1 &= x_1/(x_1 + x_2) \\
  p_2 &= x_2/(x_1 + x_2)
\end{align*}
\]  

(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
the effective queue length for queue $j$ is given by:

$$L_j = \sum_k L_j^k / C_j$$  \hspace{1cm} (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
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 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].
In Chapter 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 networks 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(μ). 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 Cr, r = 1, 2, ..., 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
important quantities and notations we used for this model are listed below:

\[ N = \text{number of nodes in the network.} \]
\[ R = \text{number of classes(paths).} \]
\[ M = \text{number of links in the network.} \]
\[ C_i = \text{capacity of link } i, i = 1, 2, \ldots, M. \]
\[ (s_1, s_2, \ldots, s_M) = \text{the state of the network.} \]
\[ s_i = (n_{i1}, n_{i2}, \ldots, n_{iR}), i = 1, 2, \ldots, M. \]
\[ n_{ir} = \text{number of customers of class } r \text{ on link } i. \]
\[ a(r) = \{i_{r1}, i_{r2}, \ldots, i_{rl}\} : \text{the set of links on path } r \text{ with length } l. \]

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\cite{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:

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

for all time \( s \), where \( \perp \) denotes the independence relations between two random processes.
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\cite{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\cite{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:

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

The routing device $d_i$ of Fig.3.2 with routing defined above is a Marcov routing device.
Intuitively, Markov 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 $\lambda_i$ 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]:

$$
\lambda_i = \sum_c r^c_{0i} + \sum_c \sum_{j \neq i} \lambda_j r^c_{ji} \quad i = 1, 2, \ldots, M.
$$

(3.3)

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

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

(3.4)

and the total utilization of queue $i$ as:

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

(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 Markov routing devices defined in Def.3.2 has a product form invariant distribution:

$$
P(s_1, s_2, \ldots, s_M) = P_1(s_1)P_2(s_2) \cdots P_M(s_M)
$$

(3.6)

where $P_i(s_i), \quad i = 1, \ldots, M$, is the marginal distribution of the queue length process $\{S^i_t\}_{t \geq 0}$.
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 \( r, \quad r = 1, 2, \ldots, R. \) 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 \( x_i \in \mathcal{R}, i = 1, \ldots, M, \quad N \in \mathbb{Z}^+, \) the following equation always holds:

\[
\left( \sum_{i=1}^{M} x_i \right)^N = \sum_{n_1, n_2, \ldots, n_M \geq 0, n_1 + n_2 + \cdots + n_M = N} \frac{N!}{\prod_{i=1}^{M} n_i!} \prod_{i=1}^{M} x_i^{n_i} \tag{3.7}
\]

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

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

\[
n_r \triangleq \sum_{i=1}^{M} n_{i,r} = \sum_{i \in a(r)} n_{i,r} \tag{3.8}
\]
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:

\[
n_i \triangleq \sum_{r=1}^{R} n_{ir}
\]  

(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:

\[
P(s_i, n_i) = P\{n_{i1}, n_{i2}, \ldots, n_{iR}; \sum_{r=1}^{R} n_{ir} = n_i\} = A^{n_{i1} + n_{i2} + \cdots + n_{iR}} \frac{n_{i1}! \cdots n_{iR}!}{n_{i1}! \cdots n_{iR}!} \rho_{i1}^{n_{i1}} \rho_{i2}^{n_{i2}} \cdots \rho_{iR}^{n_{iR}}
\]  

(3.10)

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

\[
\sum_{n_i=0}^{\infty} \sum_{s_i} P(s_i, n_i) = \sum_{n_i=0}^{\infty} \sum_{(n_{i1}, \ldots, n_{iR})} A^{n_{i1}} \frac{n_{i1}! \cdots n_{iR}!}{n_{i1}! \cdots n_{iR}!} \prod_{r=1}^{R} \rho_{ir}^{n_{ir}}
\]

\[
= A \sum_{n_i=0}^{\infty} \left( \sum_{r} \rho_{ir} \right)^{n_i}
\]

\[
= A \sum_{n_i=0}^{\infty} \rho_i^{n_i} = 1
\]  

(3.11)

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

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

(3.12)

From (3.12), we obtain:

\[
P(n_i) = \sum_{s_i} P(s_i, n_i)
\]
\[ \begin{align*}
&= \sum_{s_i} (1 - \rho_i)(\sum_{r=1}^{R} n_{ir})! \prod_{r=1}^{R} \frac{\rho_{ir}^{n_{ir}}}{n_{ir}!} \\
&= (1 - \rho_i) \sum_{s_i} \frac{n_{i}!}{\prod_{r=1}^{R} n_{ir}!} \prod_{r=1}^{R} \rho_{ir}^{n_{ir}} \\
&= (1 - \rho_i) \rho_i^{n_i} \tag{3.13}
\end{align*} \]

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:

\[
T_r(s) = \prod_{i \in a(r)} E\{e^{-st_i}\} = \prod_{i \in a(r)} T_i(s) = \prod_{i \in a(r)} \frac{\mu C_i(1 - \rho_i)}{s + \mu C_i(1 - \rho_i)} \tag{3.21}
\]

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 \( t \), 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 Markov 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.
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:

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

where

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

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

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

Assume that $\mu C_s (1 - \rho_s) \neq \mu C_i (1 - \rho_i)$, $\forall s \neq t$.

**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 $P_r$. 

28
\[ P_r = \{s, i_1, \ldots, i_{m-2}, d\}, \text{ where } m \text{ is the path length of } r. \text{ We claim that:} \]
\[ \forall r' = (i,j), \quad i,j \in P_r \implies a(r') \subset P_r. \quad (3.24) \]
\[ \forall r'' = (k,l), \quad k,l \notin P_r \implies a(r'') \notin P_r. \quad (3.25) \]

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

$$t_v(x) = \left( \prod_{i \in \alpha(r)} \mu C_i (1 - \rho_i) \right) \left[ \sum_{i \in \alpha(r)} \left( \prod_{j \in \alpha(r), j \neq i} \frac{1}{\mu C_j (1 - \rho_j) - \mu C_i (1 - \rho_i)} \right) \times \exp(-\mu C_i (1 - \rho_i) x) \right]$$

(3.26)

Integrating (3.26) gives us the cumulative density function (cdf) of \( t_v \):

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

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

(3.27)

where \( a_i, b_i, c \) are defined as in (3.23). Setting \( F_{t_v}(B_0) = 0.99 \) 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
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

\[ a(r_j) = \text{denote the set of links for j.th path for class r.} \]

\[ q(r_j) = \text{the probability that the j.th path is selected for class r.} \]

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

**Theorem 3.3** Let \(r_j, \ j = 1, 2, \ldots, K(s, d)\) 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:

\[
T_r^*(s) = \sum_{j=1}^{K(s, d)} q(r_j) \prod_{i \in z(r_j)} \frac{\mu C_i(1 - \rho_i)}{s + \mu C_i(1 - \rho_i)}
\]

(3.28)
**Proof**

*From (3.19), we have:

\[ t_{r_j} = \sum_{i \in a(r_j)} t_i \quad (3.29) \]

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

\[ E\{e^{-st_r}|r = r_j\} = E\{e^{-st_{r_j}}\} \]

\[ = E\{e^{-\sum_{i \in a(r_j)} t_i}\} \]

\[ = \prod_{i \in a(r_j)} \frac{\mu C_i (1 - \rho_i)}{\mu C_i (1 - \rho_i) + s} \]

\[ T^{*}_r(s) = E\{E\{e^{st_r}|r\}\} \]

\[ = \sum_{j=1}^{K_r} q(r_j) E\{e^{-st_{r_j}}|r = r_j\} \]

\[ = \sum_{j=1}^{K_r} q(r_j) \prod_{i \in a(r_j)} \frac{\mu C_i (1 - \rho_i)}{\mu C_i (1 - \rho_i) + s} \quad (3.30) \]

*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
problem are as following:

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

constraints:

$$\begin{align*}
    f_{ik} &= \sum_j s_i(j) \phi_{ik}(j); \\
    s_i(j) &= t(r) + \sum_l s_l(j) \phi_{il}(j); \\
    \phi_{ik}(j) &\geq 0, \quad \forall i, j, k
\end{align*}$$  (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].
Now we denote the set \( \{r_1, r_2, \ldots, r_M\} \) as the paths enumerated from the path tree for \( r = (s, d) \). Each path \( r_i \) represents a sequence of nodes

\[
(s, k_{l_1}^{(i)}, k_{l_2}^{(i)}, \ldots, k_{l_{i-1}}^{(i)}, d) \quad i = 1, 2, \ldots, M
\]

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:

\[
t(r_i) = t(r) \prod_{j=1}^{l_i} \phi_{k_j^{(i)} k_{j+1}^{(i)}}(d)
\]  
(3.33)

where we define:

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

Let,

\[
q_i(r) \triangleq \prod_{j=1}^{l_i} \phi_{k_j^{(i)} k_{j+1}^{(i)}}(d)
\]

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

34
the average of delay for each path in $P_r$, for all $i$:

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

Where $q_i(r), P_r, \mu, C_j, \rho_j (j = 1, 2, \cdots, l_r)$ are as specified before.

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

$$t_r^*(x) = \sum_{i=1}^{m} q_i(r) \left( \prod_{l=1}^{M_i} \mu C_l^{(i)}(1 - \rho_l^{(i)}) \right) \times$$

$$\times \left( \sum_{j=1}^{M_i} \left( \prod_{k=1 \atop k \neq j}^{M_i} \frac{\text{exp}(-\mu C_j^{(i)}(1 - \rho_j^{(i)})x)}{\mu C_k^{(i)}(1 - \rho_k^{(i)}) - \mu C_j^{(i)}(1 - \rho_j^{(i)})} \right) \right)$$  \hspace{1cm} (3.35)$$

Rewriting (3.35), we obtain:

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

Where,

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

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

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:

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

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

By solving

\[ F_{c}^*(B_{*}) = 0.99 \]

which is:

\[ \sum_{i=1}^{m} \sum_{j=1}^{M_i} b_{ij} exp(-\mu C_{j}^{(i)}(1 - \rho_{j}^{(i)}) B_{*}^{c}) = c + 0.99 \quad (3.38) \]

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 Markov 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:

\[ T_r = \sum_{i \in P_r} \frac{1}{\lambda_i (1 - \rho_i)} \]  

(3.39)

\[ \sigma^2_{T_r} = \sum_{i \in P_r} \frac{1}{\lambda_i^2} \left( \frac{\rho_i}{1 - \rho_i} \right)^2 \]  

(3.40)

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

\[ T_r = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i (1 - \rho_i)} \]  

(3.41)

\[ \sigma^2_{T_r} = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i^2} \left( \frac{\rho_i}{1 - \rho_i} \right)^2 \]  

(3.42)

**Proof**

By (3.20), we have:

\[ T_r(s) = E\{ e^{-s T_r} \} \]  

(3.43)

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

\[ \frac{\partial}{\partial s} T_r(s) \bigg|_{s=0} = E\{ -t_r e^{-s T_r} \} \bigg|_{s=0} = E\{ -t_r \} \]  

(3.44)

\[ \frac{\partial}{\partial s^2} T_r^2(s) \bigg|_{s=0} = E\{ t_r^2 e^{-s T_r} \} \bigg|_{s=0} = E\{ t_r^2 \} \]  

(3.45)

From (3.22), one finds:

\[ \frac{\partial}{\partial s} T_r(s) = \sum_{i \in \Omega(r)} \frac{-\mu C_i (1 - \rho_i)}{(s + \mu C_i (1 - \rho_i))^2} \]  

(3.46)

\[ \frac{\partial^2}{\partial s^2} T_r(s) = \sum_{i \in \Omega(r)} \frac{2 \mu C_i (1 - \rho_i)}{(\mu C_i (1 - \rho_i))^2} \]  

(3.47)
From the equation:

\[ \sigma_{t_r}^2 = E\{t_r^2\} - E^2\{t_r\}. \]  \hspace{1cm} (3.48)

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

\[ E\{t_r^2\} = \sum_{j=1}^{K_r} q(r_j) \sum_{i \in P_{r_j}} \frac{1}{\lambda_i} \frac{\rho_i}{1 - \rho_i} \]  \hspace{1cm} (3.49)

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

The proof for the multiple paths case is similar. \hspace{1cm} 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

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

(4.1)

Proof Rewrite the definition of Laplace transform, one has:

\[
\mathcal{L}[f] = \int_{0}^{\infty} f(t)e^{-\sigma t}dt = \int_{0}^{\infty} f(t)e^{-\sigma t-j\omega t}dt
\]

\[
= \int_{0}^{\infty} (f(t)e^{\sigma t})e^{-j\omega t}dt
\]

\[
= \Phi[f(t)e^{-\sigma t}]
\]

(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:

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

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

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

where

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

and the inverse transform is defined by:

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

In our case, $X$ 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 $O(mn) \approx O(n)$ 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)** \(\beta \in \mathbb{C}\) is a primitive \(n\)th root of unity if \(\beta \neq 1\) satisfies:

\[
(1) \quad \beta^n = 1 \\
(2) \quad \sum_{k=0}^{n-1} \beta^{jk} = 0 \quad , j = 1, 2, \ldots, n - 1
\]

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 \(n = 2^n\) 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:

\[
x_k = \sum_{i=0}^{n-1} X_i w^{ki}, \quad k = 0, 1, \ldots, n - 1.
\]
Assume \( n = 2^p \), then (4.6) can be expanded into

\[
x_k = X_0 + X_1 w^k + X_2 w^{k^2} + \cdots + X_{n-1} w^{k(2^p-1)}
\]
\[
= (X_0 + X_2 w^{k^2} + X_4 w^{k^4} + \cdots + X_{n-2} w^{k(2^p-2)}) + \\
+ (X_1 w^k + X_3 w^{k^3} + \cdots + X_{n-1} w^{k(2^p-1)})
\]
\[
= (X_0 + X_2 (w^2)^k + \cdots + X_{n-2} (w^2)^{k(2^p-1)-1}) + \\
+ (X_1 + X_3 (w^2)^k + \cdots + X_{n-1} (w^2)^{k(2^p-1)-1}) w^k
\]  \( (4.7) \)

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

\[
y_k \triangleq \sum_{i=0}^{2^{p-1}} Y_i w^{ki} \quad (4.8)
\]
\[
z_k \triangleq \sum_{i=0}^{2^{p-1}} Z_i w^{ki} \quad (4.9)
\]

(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

\[
x_k = j_k + z_k w^k, \quad k = 0, 1, \ldots, \frac{n}{2} - 1.
\]  \( (4.11) \)

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

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

43
\[ \begin{align*}
&= \sum_{i=0}^{n-1} X_i (-w^{k-\frac{n}{2}})^i, \quad k = \frac{n}{2}, \frac{n}{2} + 1, \ldots, n. \\
&= \sum_{i=0}^{n-1} X_i (-w^{k'})^i, \quad k' = 0, 1, \ldots, \frac{n}{2} - 1. \\
&= X_0 - X_1 w^{k'} + X_2 w^{k'2} - \cdots - X_{n-1} w^{k'(2^{p-1})} \\
&= [X_0 + X_2 w^{k'2} + \cdots + X_{n-2} w^{k'(2^{p-2})}] \\
&\quad - [X_1 + X_3 w^{k'3} + \cdots + X_{n-1} w^{k'(2^{p-1})}] \\
&= [X_0 + X_2 (w^2)^{k'} + \cdots + X_{n-2} w^{2k'(2^{p-1})}] \\
&\quad - [X_1 + X_3 (w^2)^{k'} + \cdots + X_{n-1} w^{2k'(2^{p-1})}] w^k \\
&= y_k - z_k w^k, \quad k = \frac{n}{2}, \ldots, n - 1 \\
\end{align*} \]

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 Horowitz[65]:

**Algorithm** IDFT \((X, w, n)\);

begin

if \( n = 1 \) then \( x_0 \leftarrow X_0 \);
else

begin

\[ X \leftarrow [X_0, X_2, \ldots, X_{n-2}] ; \]

\[ \]
\[ Z \leftarrow \{X_1, X_3, \ldots, X_{n-1}\}; \]
\[ y \leftarrow \text{IDFT}(Y, w^2, n/2)); \]
\[ z \leftarrow \text{IDFT}(Z, w^2, n/2)); \]
\[ x_k \leftarrow y_k + z_kw^k, \quad k = 0, 1, \ldots, 2^{p-1}; \]
\[ x_k \leftarrow y_k - z_kw^k, \quad k = 2^{p-1}, \ldots, 2^p; \]
\]

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:

\[
T(n) = T(2^p) = 2T(2^{p-1}) + cn
\]
\[
= 2[2T(2^{p-2}) + c\frac{n}{2}] + cn
\]
\[
\vdots
\]
\[
= 2^pT(1) + \underbrace{cn + \cdots + cn}_{p}
\]
\[
= nT(1) + cnp = c'n + cn \log n \quad (4.13)
\]

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

\[
F_{t_r}(k) = \sum_{i=0}^{k} x_i \quad (4.14)
\]

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):

$$O(mn) + O(n \log n) + O(n)$$  \hspace{1cm} (4.15)

where

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

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:

$$B_n^i = B_0^i + \frac{bn}{a-n} \quad n = 0,1,\cdots$$  \hspace{1cm} (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 \( n \to \infty \).

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:

$$X_n^i = [L - B_n^i]^+$$  \hspace{1cm} (4.17)

where \( [a]^+ = \max(a, 0) \), \( \forall a \in \mathbb{R} \), \( L \) 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 \( \lambda \). 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
server takes one message for every time unit and allocates it to one of the sub-networks 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 sub-networks to assign those waiting customers such that the total discounted reward is maximized:

Formally, we want:

$$\max \left[ \sum_{t=1}^{\infty} \alpha^t r(t) \right]$$  \hspace{1cm} (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 $\sum_{t=0}^{\infty} \alpha^t X_{n_t} < +\infty$, $\forall i = 1, 2, ..., M$ where $n_t$ denotes the $n_t$ 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:

$$r_i^t = \sup_{r \geq 0} \left[ \frac{\sum_{m=t}^{r-1} \alpha^m X_i^m}{\sum_{m=t}^{r-1} \alpha^m} \right]$$  \hspace{1cm} (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:

\[ B_m^i \geq B_n^i, \quad \forall m \geq n \tag{4.20} \]

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 \( X^i_t \) denote one of the \( \{X_n^i, n = 0, 1, \ldots\} \)

\[ \delta(t) \triangleq \arg\max_{i=1\ldots M} \{X^i_{n_t}\} \tag{4.22} \]

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

**Proof**

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

\[ \sum_{t=1}^{\infty} a^i r(t) \]

where \( (i_1, i_2, \ldots, i_N) \) is the sequence for operating networks \( \{X^i_n, \forall n = 0, 1, \ldots N\} \) at
time $t = 1, 2, \ldots, N$ Assume there is some $s$ such that

$$
Pr\{X^{i*}_{n_s} \leq X^{i*+1}_{n_{s+1}}, n_{s+1} > n_s\} = 1
$$

thus,

$$
R(i_1, i_2, \ldots, i_s, i_{s+1}, \ldots, i_N) - R(i_1, i_2, \ldots, i_{s+1}, i_s, \ldots, i_N)
\quad = \quad \sum_{i=1}^{\infty} \alpha^n r_1(t) - \sum_{i=1}^{\infty} \alpha^n r_2(t)
$$

(4.24)

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

$$
r_1(t) = \begin{cases} 
r_2(t), & \forall t \neq s, s + 1; \\
\alpha^s X^{i*}_{n_s}, & t = s; \\
\alpha^{s+1} X^{i*+1}_{n_{s+1}}, & t = s + 1.
\end{cases}
$$

(4.25)

$$
r_2(t) = \begin{cases} 
r_1(t), & \forall t \neq s, s + 1; \\
\alpha^s X^{i*+1}_{n_{s+1}}, & t = s; \\
\alpha^{s+1} X^{i*}_{n_s}, & t = s + 1.
\end{cases}
$$

(4.26)

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

$$
\alpha^s X^{i*}_{n_s} + \alpha^{s+1} X^{i*+1}_{n_{s+1}} - \alpha^s X^{i*+1}_{n_{s+1}} - \alpha^{s+1} X^{i*}_{n_s}
\quad = \quad \alpha^s (X^{i*}_{n_s} - X^{i*+1}_{n_{s+1}}) + \alpha^{s+1} (X^{i*+1}_{n_{s+1}} - X^{i*}_{n_s})
\quad = \quad \alpha^s (1 - \alpha) (X^{i*}_{n_s} - X^{i*+1}_{n_{s+1}})
$$

(4.27)

Combining (4.27) and (4.23) we get:

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

(4.28)

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

$\text{QED}$
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
<table>
<thead>
<tr>
<th>interarrival time (mean)</th>
<th>service time mean</th>
<th>node proc. time</th>
<th>simulation time</th>
<th>data files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>medium A</td>
<td>medium B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5 sec</td>
<td>0.1 sec</td>
<td>0.2 sec</td>
<td>0.05 sec</td>
<td>2050 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4 sec</td>
<td>0.1 sec</td>
<td>0.2 sec</td>
<td>0.05 sec</td>
<td>1640 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.37 sec</td>
<td>0.1 sec</td>
<td>0.2 sec</td>
<td>0.05 sec</td>
<td>1537.5 sec</td>
</tr>
<tr>
<td>0.3 sec</td>
<td>0.1 sec</td>
<td>0.2 sec</td>
<td>0.05 sec</td>
<td>1230 sec</td>
</tr>
</tbody>
</table>

Table 5.1: data set I

<table>
<thead>
<tr>
<th>interarrival time (mean)</th>
<th>service time mean</th>
<th>node proc. time</th>
<th>simulation time</th>
<th>data files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>medium A</td>
<td>medium B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5 sec</td>
<td>0.12 sec</td>
<td>0.15 sec</td>
<td>0.01 sec</td>
<td>2050 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4 sec</td>
<td>0.12 sec</td>
<td>0.15 sec</td>
<td>0.01 sec</td>
<td>1640 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3 sec</td>
<td>0.12 sec</td>
<td>0.15 sec</td>
<td>0.01 sec</td>
<td>1230 sec</td>
</tr>
</tbody>
</table>

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

Table 5.3: data set III

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Shortest path algorithms for subnet 1</td>
</tr>
<tr>
<td>A2</td>
<td>Shortest path algorithms for subnet 2</td>
</tr>
<tr>
<td>A3</td>
<td>Index switching( using expected delay)</td>
</tr>
<tr>
<td>A4</td>
<td>Index switching( using delay bound)</td>
</tr>
</tbody>
</table>

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
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 does 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 to 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
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 reduce 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 computation time is tolerable. By this approach, we can use 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.
A.1 Proofs of Lemma 3.3

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

\[
\left(\sum_{i=1}^{k} x_i + x_{k+1}\right)^N = \sum_{j=0}^{N} \binom{N}{j} \left(\sum_{i=1}^{k} x_i \right)^j x_{k+1}^{N-j}
\]

\[
= \sum_{j=0}^{N} \binom{N}{j} \left( \sum_{n_1, n_2, \ldots, n_k} \frac{j!}{\prod_{i=1}^{k} n_i} \prod_{i=1}^{k} x_i^{n_i} x_{k+1}^{N-j} \right)
\]

\[
= \sum_{j=0}^{N} \frac{N!}{j!(N-j)!} \frac{j!}{\prod_{i=1}^{k} n_i} \prod_{i=1}^{k} x_i^{n_i} x_{k+1}^{N-j}
\]

\[
= \sum_{n_1, n_2, \ldots, n_k, n_{k+1} \geq N-j} \frac{N!}{\prod_{i=1}^{k+1} n_i} \prod_{i=1}^{k+1} x_i^{n_i}
\]

(1.1)

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

83
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:

$$1 + w + w^2 + \cdots + w^{2^p-1}$$

$$= (1 + w^2 + \cdots + w^{2^{p-2}}) + (w + w^3 + \cdots + w^{2^{p-1}})$$

$$= (1 + w^2 + \cdots + w^{2^{p-2}} + w(1 + w^2 + \cdots + w^{2^{p-2}}))$$

$$= (1 + w)(1 + w^2 + \cdots + w^{2^{p-2}}) = 0$$

(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)

\[(w^{k+\frac{n}{2}} - w^k)(w^{k+\frac{n}{2}} + w^k)\]

\[= (w^{k+\frac{n}{2}})^2 - (w^k)^2\]

\[= w^{2k+n} - w^{2k}\]

\[= w^{2k}w^n - w^{2k} = 0\]

(1.3)

Note that 1, w, w^2, ..., 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 Simulation Programs:
the Expected Delay Case

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

1 &
2 & Network model b3
3 & Routing: Use switching algorithms at node for class X
4 & Index for switching: The estimates for the expected delays
5 &
6 /DECLARE/
7 INTEGER N=7; & number of processing nodes
8 INTEGER S=4; & Background traffic sources
9 INTEGER lx=2000; & number of points for delay distr.
10 INTEGER lx_max=200; & printing limit for delay distr.
11 INTEGER nb_max; & nb of messages that exceed lx_max
12 INTEGER Nx; & number of X message sent to netw
13 INTEGER nb_msg; & number of X message received
14 INTEGER lx,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 tl,t2; & indexes for switching
20 REAL lambda; & temp var for throughput
21 REAL ct; & storage for message delay
22 REAL fx(lx); & 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 sbv_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 Jacksonian routage array
29 REAL temp;
30 REAL d_mean,d_var,d_bound; & delay statistics
31 &
32 QUEUE REAL d_time; & attributes declaration:
33 CUSTOMER REAL s_time; & nb/lambda for queuing delay
34 CUSTOMER INTEGER s_nb; & starting time for a message
35 &
36 QUEUE n(N); & sequential number for a message
37 QUEUE s(N); & processing nodes
38 QUEUE l(N,N); & background sources
39 &
40 REF CUSTOMER cx; & communication links
41 &
42 CLASS X,Y; & an X message
43 &
44 & two classes of traffic
45 &
46 &
47 & Followings are the variables 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};
REAL rout_pr(N,N) =
(1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0).
(0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0).
(0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0).
(0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0).
(0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0).
(0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0).

& routing probability (i,j) :
& node i destined for node j
& for class X messages
& path array for sp
& evolution array for min cost

INTEGER path(N);
REAL d(N,N); & the formal parameters
& k:source, 1:destination & &
& computation variables

PROCEDURE sp1(k1,k2):

BEGIN
FOR j1:=1 STEP 1 UNTIL N DO
BEGIN
d(1,j1):=graph(k1,j1);
path(j1):= IF graph(k1,j1)=0 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
BEGIN
m:=d(h-1,n1)+graph(n1,j1);
path(j1):=m1;
END;
END;
d(h,j1):=m;
END;
END;

WHILE (path(11)<>k1) AND (path(11)<>0) DO
BEGIN
FOR j1:=1 STEP 1 UNTIL N DO
BEGIN
rout_pr(path(11),j1):=0; & clear the row
rout_pr(path(11),j1):=1;
END;
END;

IF path(11)=k1 THEN BEGIN
END;
END;
END;
END;
END;
END;

PRINT("** The Shortest Path Matrix (1) * * ");
PRINT("1) The path array:");
PRINT(path(1 STEP 1 UNTIL N));
PRINT("2) The cost matrix");
FOR j1:=1 STEP 1 UNTIL N DO
PRINT(d(j1, 1 STEP 1 UNTIL N));
PRINT("3) The routing probability matrix:");
FOR j1:=1 STEP 1 UNTIL N DO
PRINT(rout_pr(j1, 1 STEP 1 UNTIL N));
END;

& The shortest path for the second network
PROCEDURE sp2(k1,k2):
INTEGER k1,k2;
INTEGER i1,j1;
BEGIN
IF k1=1 THEN
BEGIN
FOR j1:=1 STEP 1 UNTIL N DO
rout_pr(1,j1):=0;
rout_pr(1,4):=1.0;
END ELSE
IF k1=7 THEN
BEGIN
rout_pr(7,7):=1.0;
END ELSE
IF k1=4 THEN
BEGIN
FOR j1:=1 STEP 1 UNTIL N DO
rout_pr(4,j1):=0;
rout_pr(4,7):=1.0;
END
END;
ELSE
BEGIN
FOR i1:=(2,3,5,6) DO
BEGIN
FOR j1:=1 STEP 1 UNTIL N DO
rout_pr(i1,j1):=0;
rout_pr(i1,4):=1;
END;
END;
PRINT("* * * The Shortest Path Matrix (II) * * *");
PRINT("1) The path array:");
PRINT(path(1 STEP 1 UNTIL N));
PRINT("2) The cost matrix");
FOR i1:=1 STEP 1 UNTIL N DO
PRINT(d(i1, 1 STEP 1 UNTIL N));
PRINT("3) The routing probability matrix:");
FOR i1:=1 STEP 1 UNTIL N DO
PRINT(rout_pr(i1,1 STEP 1 UNTIL N));
END;

& Find the Jacksonian routing probability
&
PROCEDURE Jackson;
REAL r_sum: & sum of one row in jack_pr
INTEGER i1, j1;
BEGIN
FOR i1:=1 STEP 1 UNTIL N DO
BEGIN
r_sum:=0;
FOR j1:=1 STEP 1 UNTIL N DO
r_sum:=r_sum+(IF graph(i1, j1)=xxx THEN graph(i1, j1)
ELSE 0.0);
FOR j1:=0 STEP 1 UNTIL N DO
jack_pr(i1, j1):=IF graph(i1, j1)=xxx THEN graph(i1, j1)/r_sum
ELSE 0.0;
END;
PRINT("* * The Jacksonian Matrix * * *"): FOR i1:=1 STEP 1 UNTIL N DO
PRINT(jack_pr(i1, 0 STEP 1 UNTIL N));
END;
NAME=n(1);
SERVICE(X)=
BEGIN
CST(pc_time);
cx:=n(1).FIRST;
$MACRO l_delay(nodel, node2)
nb:=l(nodel, node2).NB;
lambda:=REALINT(l(nodel, node2).NBOUT)/TIME;
l(nodel, node2).d_time:=IF lambda<0 THEN REALINT(nb)/lambda
ELSE 0.0;
$END
$MACRO n_delay(node)
nb:=n(node).NB;
lambda:=REALINT(n(node).NBOUT)/TIME;
n(node).d_time:=IF lambda<>0 THEN REALINT(nb)/lambda
ELSE 0.0;
$END
$MACRO l_delay(1, 2) $n_delay(2) $l_delay(2, 3)
$n_delay(3) $l_delay(3, 6) $n_delay(6) $l_delay(6, 7) $n_delay(7)
$l_delay(1, 4) $n_delay(4) $l_delay(4, 7)
tl:=l(1, 2).d_time+n(2).d_time+l(2, 3).d_time+n(3).d_time+l(3, 6).d_t
== time
n(6).d_time+l(6, 7).d_time;
t2:=l(3, 4).d_time+n(4).d_time+l(4, 7).d_time;
IF tl<t2 THEN 
BEGIN
& PRINT(TIME, tl,"|", t2,"MESSAGE:", cx.s_nb,
& " sent to network1");
& TRANSIT(l(1, 2));
END
ELSE 
BEGIN
& PRINT(TIME, tl,"|", t2,"MESSAGE:", cx.s_nb,
RESULTS: s_nb, delay times, etc.

89
& "sent to network2");
TRANSIT(1.1.4);
END;
SERVICE(1) = CIST (pc_time);
TRANSIT(1) = (2.1 \text{ STEP 1 UNTIL N}), \text{ jack}_{\text{pr}} (1, 1 \text{ STEP 1 UNTIL N}), \text{OUT};
/STATION/
NAME=n(2);
SERVICE=CST (pc_time);
TRANSIT(X) = (2.1 \text{ STEP 1 UNTIL N}), \text{rou}_{\text{nt}_{\text{pr}}}(2, 1 \text{ STEP 1 UNTIL N});
TRANSIT(Y) = (2.1 \text{ STEP 1 UNTIL N}), \text{jack}_{\text{pr}} (2, 1 \text{ STEP 1 UNTIL N}), \text{OUT};
/STATION/
NAME=n(3);
SERVICE=CST (pc_time);
TRANSIT(X) = (3.1 \text{ STEP 1 UNTIL N}), \text{rou}_{\text{nt}_{\text{pr}}}(3, 1 \text{ STEP 1 UNTIL N});
TRANSIT(Y) = (3.1 \text{ STEP 1 UNTIL N}), \text{jack}_{\text{pr}} (3, 1 \text{ STEP 1 UNTIL N}), \text{OUT};
/STATION/
NAME=n(4);
SERVICE=CST (pc_time);
TRANSIT(X) = (4.1 \text{ STEP 1 UNTIL N}), \text{rou}_{\text{nt}_{\text{pr}}}(4, 1 \text{ STEP 1 UNTIL N});
TRANSIT(Y) = (4.1 \text{ STEP 1 UNTIL N}), \text{jack}_{\text{pr}} (4, 1 \text{ STEP 1 UNTIL N});
& no ou
===> t
/STATION/
NAME=n(5);
SERVICE=CST (pc_time);
TRANSIT(X) = (5.1 \text{ STEP 1 UNTIL N}), \text{rou}_{\text{nt}_{\text{pr}}}(5, 1 \text{ STEP 1 UNTIL N});
TRANSIT(Y) = (5.1 \text{ STEP 1 UNTIL N}), \text{jack}_{\text{pr}} (5, 1 \text{ STEP 1 UNTIL N}), \text{OUT};
/STATION/
NAME=n(6);
SERVICE=CST (pc_time);
TRANSIT(X) = (6.1 \text{ STEP 1 UNTIL N}), \text{rou}_{\text{nt}_{\text{pr}}}(6, 1 \text{ STEP 1 UNTIL N});
TRANSIT(Y) = (6.1 \text{ STEP 1 UNTIL N}), \text{jack}_{\text{pr}} (6, 1 \text{ STEP 1 UNTIL N}), \text{OUT};
/STATION/
NAME=n(7);
SERVICE=
BEGIN
CST (pc_time);
\text{cx} := \text{n}(7). \text{FIRST};
\text{IF cx.CCLASS} = \text{X} \text{THEN}
\text{BEGIN}
\text{ct} := \text{TIME} - \text{cx.s_time};
& \text{PRINT("MESSAGE":}, \text{cx}.s_nb, "delay":} \text{ct});
\text{IF 1x} > \text{1x} - 1 \text{THEN } \text{1x} := \text{1x} - 1; \& (\text{fx}(1999), \text{fx}(2000)) \text{is the last pa}
===> 1r
\text{IF 1x} < 1 \text{THEN } \text{1x} := 1;
\text{IF 1x} > \text{ix}_{\text{max}} \text{THEN nb}_{\text{max}} := \text{nb}_{\text{max}} + 1; \& \text{the delay that exceeds}
===> \text{ix}_{\text{max}}
\text{nb}_{\text{msg}} := \text{nb}_{\text{msg}} + 1;
\text{fx} ([1x] + 1) := \text{fx} ([1x] + 1) + 1;
\text{END;}
\text{TRANSIT=OUT;}
&
\text{background source nodes}
&
/STATION/
NAME=s(1);
TYPE=SOURCE;
SERVICE=
BEGIN
EXP(ia_time);
& X is created only when system is warm-up
IF (TIME>warm_up) AND DRAW(0.45) THEN
BEGIN
  cx:=s(l)._FIRST;
  cx.a_time:=TIME;
  cx.s_nb:=N
  Nx:=Nx+1;
  TRANSIT(m(l),X);
END
ELSE TRANSIT(m(l),Y);
END:
/STATION/
NAME=s(2);
TYPE=SOURCE;
SERVICE=EXP(ia_time);
TRANSIT=m(2),Y,1;
/STATION/
NAME=s(3);
TYPE=SOURCE;
SERVICE=EXP(ia_time);
TRANSIT=m(3),Y,1;
/STATION/
NAME=s(5);
TYPE=SOURCE;
SERVICE=EXP(ia_time);
TRANSIT=m(5),Y,1;
/STATION/
NAME=s(6);
TYPE=SOURCE;
SERVICE=EXP(ia_time);
TRANSIT=m(6),Y,1;
& & comm. links
$MACRO link(s_node,d_node,s_time) & definition for communication
/STATION/ NAME=l(s_node,d_node) & link stations
SERVICE=EXP(s_time);
TRANSIT=m(d_node),l;
$END
&
$link(1,2,sva_time) $link(1,3,sva_time)
$link(2,3,sva_time) $link(2,4,sva_time)
$link(3,2,sva_time) $link(3,6,sva_time)
$link(5,2,sva_time) $link(5,6,sva_time)
$link(5,7,sva_time)
$link(6,3,sva_time) $link(6,5,sva_time)
$link(6,7,sva_time)
&
$link(1,4,svb_time) $link(4,7,svb_time)
$link(2,4,svb_time) $link(3,4,svb_time)
$link(6,4,svb_time)
$link(4,2,svb_time) $link(4,3,svb_time)
$link(4,5,svb_time)
&
/CONTROL/
CLASS=ALL QUEUE:
ENTRY=BEGIN
PRINT("Inter-arrival time?"); ia_time:=GET_REAL;
PRINT("Service time for net 1?"); sva_time:=GET_REAL;
PRINT("Service time for net 2?"); svb_time:=GET_REAL;

91
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
355 END;
356 TMAT:=tmax;
357 EXIT:=BEGIN & print the delay distribution
358 PRINT("nb of msg");
359 PRINT(*)
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("" * * Expected Delay :");d_mean;
377 PRINT("" * * Variance :");d_var;
378 PRINT("" * * Delay Bound:");d_bound;
379 /
380 /EXEC/
381 BEGIN
382 sp1(1.7); & find the shortest path in network 1
383 sp2(4.7); & find the shortest path in network 2
384 & find the routing prob. for Y
385 PRINT("" * * Simulation result for single SHORTEST PATH * * ");
386 SIMUL;
387 END;

* * * The Shortest Path Matrix (I) * *
1) The path array:
   1   2   1   4   3   6
2) The cost matrix
   0.2000 1.000 1.000 1000.0 1000.0 1000.0
   0.2000 0.4000 1.000 3.000 1.200 6.000
   0.2000 0.4000 1.000 3.000 0.6000 1.400
   0.2000 0.4000 1.000 3.000 0.6000 0.8000
   0.2000 0.4000 1.000 3.000 0.6000 0.8000
   0.2000 0.4000 1.000 3.000 0.6000 0.8000
3) The routing probability matrix:
   0.1.000 0.0.0.0.0.0.0.0.0.
   0.1.000 0.0.0.0.0.0.
   0.0.0.0.1.000 0.0.
   0.0.0.0.1.000 0.0.
   0.0.0.0.1.000 0.0.
   0.0.0.0.1.000 0.0.

* * * The Shortest Path Matrix (II) * *
1) The path array:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.000</td>
<td>1.000</td>
<td>1000.0</td>
<td>1000.0</td>
</tr>
<tr>
<td>2</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>1.2000</td>
</tr>
<tr>
<td>3</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
</tr>
<tr>
<td>4</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
</tr>
<tr>
<td>5</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
</tr>
<tr>
<td>6</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
</tr>
</tbody>
</table>

2) The cost matrix:

<table>
<thead>
<tr>
<th></th>
<th>0.</th>
<th>0.2000</th>
<th>0.4000</th>
<th>1.000</th>
<th>3.000</th>
<th>1.2000</th>
<th>6.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>1.2000</td>
<td>6.000</td>
</tr>
<tr>
<td>2</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
<td>1.400</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
<td>0.8000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
<td>0.8000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
<td>0.8000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.2000</td>
<td>0.4000</td>
<td>1.000</td>
<td>3.000</td>
<td>0.6000</td>
<td>0.8000</td>
<td></td>
</tr>
</tbody>
</table>

3) The routing probability matrix:

<table>
<thead>
<tr>
<th></th>
<th>0.</th>
<th>0.</th>
<th>0.</th>
<th>0.</th>
<th>0.</th>
<th>0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>1.000</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>2</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>3</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>4</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>5</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
<tr>
<td>6</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
<td>0.</td>
</tr>
</tbody>
</table>

** 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 ***

** NAME * SERVICE * BUSY PCT * CUST NB * RESPONSE * SERV NB *

**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

93
<table>
<thead>
<tr>
<th>n</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.5002e-01</td>
<td>0.3469</td>
<td>0.4427</td>
<td>0.6384e-01</td>
<td>14216*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.5002e-01</td>
<td>0.2537</td>
<td>0.2949</td>
<td>0.5814e-01</td>
<td>10398*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5002e-01</td>
<td>0.2249</td>
<td>0.2572</td>
<td>0.5718e-01</td>
<td>9219*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.5002e-01</td>
<td>0.1813</td>
<td>0.2079</td>
<td>0.5737e-01</td>
<td>7429*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1</td>
<td>1</td>
<td>0.5094</td>
<td>1.000</td>
<td>1.000</td>
<td>0.5094</td>
<td>4023*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>2</td>
<td>0.4960</td>
<td>1.000</td>
<td>1.000</td>
<td>0.4960</td>
<td>4133*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>3</td>
<td>0.5044</td>
<td>1.000</td>
<td>1.000</td>
<td>0.5044</td>
<td>4064*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>4</td>
<td>0.4980</td>
<td>1.000</td>
<td>1.000</td>
<td>0.4980</td>
<td>4116*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5</td>
<td>5</td>
<td>0.5024</td>
<td>1.000</td>
<td>1.000</td>
<td>0.5024</td>
<td>4079*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l1</td>
<td>1</td>
<td>0.1191</td>
<td>0.6669e-01</td>
<td>0.6840e-01</td>
<td>0.1221</td>
<td>1148*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l2</td>
<td>2</td>
<td>0.1166</td>
<td>0.5465e-01</td>
<td>0.5522e-01</td>
<td>0.1178</td>
<td>961*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l3</td>
<td>3</td>
<td>0.1243</td>
<td>0.5973e-01</td>
<td>0.6324e-01</td>
<td>0.1316</td>
<td>985*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l4</td>
<td>4</td>
<td>0.1736</td>
<td>0.1601</td>
<td>0.1816</td>
<td>0.1970</td>
<td>1890*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l5</td>
<td>5</td>
<td>0.1790</td>
<td>0.7246e-01</td>
<td>0.7935e-01</td>
<td>0.1960</td>
<td>830*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l6</td>
<td>6</td>
<td>0.1694</td>
<td>0.8760e-01</td>
<td>0.1023</td>
<td>0.1978</td>
<td>1060*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l7</td>
<td>7</td>
<td>0.1253</td>
<td>0.7113e-01</td>
<td>0.7421e-01</td>
<td>0.1307</td>
<td>1164*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l8</td>
<td>8</td>
<td>0.1238</td>
<td>0.5804e-01</td>
<td>0.6005e-01</td>
<td>0.1281</td>
<td>961*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l9</td>
<td>9</td>
<td>0.1322</td>
<td>0.1309e-01</td>
<td>0.1416e-01</td>
<td>0.1430</td>
<td>203*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l10</td>
<td>10</td>
<td>0.1737</td>
<td>0.2655</td>
<td>0.3619</td>
<td>0.2367</td>
<td>3134*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l11</td>
<td>11</td>
<td>0.1737</td>
<td>0.2655</td>
<td>0.3619</td>
<td>0.2367</td>
<td>3134*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l12</td>
<td>12</td>
<td>0.1225</td>
<td>0.3018</td>
<td>0.4329</td>
<td>0.1758</td>
<td>5049*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l13</td>
<td>13</td>
<td>0.1225</td>
<td>0.3018</td>
<td>0.4329</td>
<td>0.1758</td>
<td>5049*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l14</td>
<td>14</td>
<td>0.1228</td>
<td>0.1240</td>
<td>0.1420</td>
<td>0.1407</td>
<td>2070*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l15</td>
<td>15</td>
<td>0.1228</td>
<td>0.1240</td>
<td>0.1420</td>
<td>0.1407</td>
<td>2070*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l16</td>
<td>16</td>
<td>0.1733</td>
<td>0.2567</td>
<td>0.3404</td>
<td>0.2297</td>
<td>3037*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l17</td>
<td>17</td>
<td>0.1733</td>
<td>0.2567</td>
<td>0.3404</td>
<td>0.2297</td>
<td>3037*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l18</td>
<td>18</td>
<td>0.1217</td>
<td>0.6816e-01</td>
<td>0.7016e-01</td>
<td>0.1253</td>
<td>1148*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l19</td>
<td>19</td>
<td>0.1217</td>
<td>0.5706e-01</td>
<td>0.5825e-01</td>
<td>0.1243</td>
<td>961*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l20</td>
<td>20</td>
<td>0.1216</td>
<td>0.1109e-01</td>
<td>0.1191e-01</td>
<td>0.1306</td>
<td>187*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l21</td>
<td>21</td>
<td>0.1698</td>
<td>0.1652</td>
<td>0.1940</td>
<td>0.1994</td>
<td>1994*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l22</td>
<td>22</td>
<td>0.1698</td>
<td>0.1652</td>
<td>0.1940</td>
<td>0.1994</td>
<td>1994*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l23</td>
<td>23</td>
<td>0.1771</td>
<td>0.1732</td>
<td>0.2117</td>
<td>0.2165</td>
<td>2004*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l24</td>
<td>24</td>
<td>0.1771</td>
<td>0.1732</td>
<td>0.2117</td>
<td>0.2165</td>
<td>2004*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l25</td>
<td>25</td>
<td>0.1747</td>
<td>0.1739</td>
<td>0.2066</td>
<td>0.2076</td>
<td>2039*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

94
| X  | 0.1747  | 0.2066  | 0.2076  | 2039*  |
| Y  | 0.1760  | 0.1784  | 0.2138  | 0.2109 | 2078*  |
|    | * 0.1760 | 0.1760  | 0.2138  | 0.2109 | 2078*  |
|    | * 0.1717 | 0.5068  | 1.012   | 0.3429 | 6052*  |
|    | * 0.1697 | 0.6886e-01 | 0.1206 | 0.2981 | 829*   |
|    | * 0.1720 | 0.4381  | 0.8917  | 0.3500 | 5233*  |
|    | * 0.1183 | 0.2933  | 0.4131  | 0.1666 | 5082*  |
|    | * 0.1183 | 0.2933  | 0.4131  | 0.1666 | 5082*  |
|    | 32 * 0.1706 | 0.2476  | 0.3352  | 0.2309 | 2976*  |
|    | 34 * 0.1230 | 0.1857  | 0.2293  | 0.1519 | 3094*  |
|    | * 0.1230 | 0.1857  | 0.2293  | 0.1519 | 3094*  |
|    | * 0.1206 | 0.1189  | 0.1378  | 0.1397 | 2021*  |
|    | * 0.1206 | 0.1189  | 0.1378  | 0.1397 | 2021*  |
|    | 38 * 0.1223 | 0.1249  | 0.1410  | 0.1381 | 2094*  |
|    | * 0.1223 | 0.1249  | 0.1410  | 0.1381 | 2094*  |
|    | * 0.1730 | 0.2645  | 0.3472  | 0.2271 | 3135*  |
|    | * 0.1730 | 0.2645  | 0.3472  | 0.2271 | 3135*  |
|    | 40 * 0.1210 | 0.1778  | 0.2112  | 0.1437 | 3012*  |
|    | * 0.1210 | 0.1778  | 0.2112  | 0.1437 | 3012*  |
|    | 42 * 0.1183 | 0.6613e-01 | 0.6781e-01 | 0.1213 | 1146*  |
|    | * 0.1193 | 0.5592e-01 | 0.5730e-01 | 0.1222 | 961*   |
|    | * 0.1193 | 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 msgf = 1790

** Delay Distribution:

<table>
<thead>
<tr>
<th>Delay</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.0</td>
</tr>
<tr>
<td>0.01</td>
<td>0.01229e-01</td>
</tr>
<tr>
<td>0.20</td>
<td>0.4581e-01</td>
</tr>
<tr>
<td>0.30</td>
<td>0.7313e-01</td>
</tr>
<tr>
<td>0.40</td>
<td>0.1112</td>
</tr>
<tr>
<td>0.50</td>
<td>0.1380</td>
</tr>
<tr>
<td>0.60</td>
<td>0.1425</td>
</tr>
<tr>
<td>0.70</td>
<td>0.1302</td>
</tr>
<tr>
<td>0.80</td>
<td>0.9832e-01</td>
</tr>
<tr>
<td>0.90</td>
<td>0.7933e-01</td>
</tr>
<tr>
<td>1.00</td>
<td>0.5922e-01</td>
</tr>
<tr>
<td>1.10</td>
<td>0.3575e-01</td>
</tr>
<tr>
<td>1.20</td>
<td>0.2793e-01</td>
</tr>
<tr>
<td>1.30</td>
<td>0.1620e-01</td>
</tr>
<tr>
<td>1.40</td>
<td>0.1006e-01</td>
</tr>
<tr>
<td>1.50</td>
<td>0.6145e-02</td>
</tr>
<tr>
<td>1.60</td>
<td>0.3352e-02</td>
</tr>
<tr>
<td>1.70</td>
<td>0.2235e-02</td>
</tr>
<tr>
<td>1.800</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>1.900</td>
<td>0.3352e-02</td>
</tr>
<tr>
<td>2.000</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>2.100</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>2.200</td>
<td>0.5587e-03</td>
</tr>
<tr>
<td>2.300</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>2.400</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>2.500</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>2.600</td>
<td>0.5587e-03</td>
</tr>
<tr>
<td>2.700</td>
<td>0.5587e-03</td>
</tr>
<tr>
<td>2.800</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>2.900</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.000</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.100</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.200</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.300</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.400</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.500</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.600</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.700</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.800</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>3.900</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.000</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.100</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.200</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.300</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.400</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.500</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.600</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.700</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.800</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>4.900</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.000</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.100</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.200</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.300</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.400</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.500</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.600</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.700</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.800</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>5.900</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.000</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.100</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.200</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.300</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.400</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.500</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.600</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.700</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.800</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>6.900</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.000</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.100</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.200</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.300</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.400</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.500</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.600</td>
<td>0.1117e-02</td>
</tr>
<tr>
<td>7.700</td>
<td>0.1117e-02</td>
</tr>
</tbody>
</table>
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.

* nh_max = 0
** Expected Delay : 0.6754
** Variance : 0.1022
** Delay Bound : 1.700

390
391 /END/
A.4 QNAP2 Simulation Programs:
the Delay-Bound Simulation 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 model for class X
4 & Index for switching: Delay bounds setup for each subnetwork
5 &
6 /DECLARE/
7 & number of processing nodes
8 & number of points for delay distr.
9 & printing limit for delay distr.
10 & number of messages that exceed IX_MAX
11 & nb of msgs that exceeds IX_MAX in
12 & nb of msgs that exceeds IX_MAX in
13 & number of X msg sent to netw
14 & number of X msg sent to netw
15 & number of X msg sent to netw
16 & number of X msg sent to netw
17 & number of X msg sent to netw
18 & a large cost for disconn. links
19 & system warm-up time
20 & indexes for switching
21 & temp var for thruput
22 & storage for message delay
23 & the delay distribution for X
24 & the delay distribution for net1
25 & the delay distribution for net2
26 & sources inter_arrival time
27 & links service time for net 1
28 & links service time for net 2
29 & nodes processing time
30 & simulation time
31 & the Jacksonian routage array
32 & the delay bound
33 & 99 percentile
34 & indices for delay bounds
35 & delay statistics
36 & attributes declaration:
37 & nb/lambda for queueing delay
38 & starting time for a message
39 & sequential number for a message
40 & subnet identifier
41 & processing nodes
42 & background sources
43 & communication links
44 & an X message
45 & two classes of traffic

CLASS X.Y:
& X: messages; Y: background

Followings are the variables for shortest path algorithms

REAL graph(N,0:N)=

5.0, xxx, 0.0, 0.2, 1.0, xxx, 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 cut flow

REAL rout_pr(N,N)=

(1.0, 0.0, 0.0, 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,
0.0, 0.0, 1.0, 0.0, 0.0, 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, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0)

& routing probability (i,j):
& node i destined for node j
& for class X messages
& path array for sp
& evolution array for min cost

INTEGER path(N);
REAL d(N,N);

The BELLMAN-FORD algorithm for computing
& the shortest path

PROCEDURE spl(k1,k2);
& the formal parameters
INTEGER k1,k2;
& k: source, i: destination
REAL m;
INTEGER i,j1,nl,h;
& computation variables

BEGIN
FOR j1:=1 STEP 1 UNTIL N DO
BEGIN
d[i,j1]:=graph[k1,j1];
path[j1]:=i
END;
FOR h:=2 STEP 1 UNTIL N DO
FOR j1:=1 STEP 1 UNTIL N DO
BEGIN
m:=d[h-1,j1];
FOR nl:=1 STEP 1 UNTIL N DO
IF (d[h-1,nl]+graph[nl,j1])<m THEN
BEGIN
m:=d[h-1,nl]+graph[nl,j1];
path[j1]:=nl;
END;
END;
END;
END;
END;

11:=k2;
WHILE (path[11]!=k1) AND (path[11]<>0) DO
BEGIN

99
FOR j1 := 1 STEP 1 UNTIL N DO
  rout_pr(path(i1), j1) := 0; & clear the row
  rout_pr(path(i1), i1) := 1;
  i1 := path(i1);
END;

IF path(i1) = k1 THEN BEGIN
  FOR j1 := 1 STEP 1 UNTIL N DO
    rout_pr(k1, j1) := 0; & clear the row
    rout_pr(k1, i1) := 1;
END;

PRINT("** * * The Shortest Path Matrix (1) * * **");
PRINT("1 The path array:");
PRINT(path(1 STEP 1 UNTIL N));
PRINT("2 The cost matrix");
FOR j1 := 1 STEP 1 UNTIL N DO
  PRINT(d(j1, 1 STEP 1 UNTIL N));
PRINT("3 The routing probability matrix:");
FOR j1 := 1 STEP 1 UNTIL N DO
  PRINT(rout_pr(j1, 1 STEP 1 UNTIL N));
END;

---

& The shortest path for the second network
&

PROCEDURE sp2(k1, k2);
  INTEGER k1, k2;
  INTEGER i1, j1;
BEGIN
  IF k1 = 1 THEN
    BEGIN
      FOR j1 := 1 STEP 1 UNTIL N DO
        rout_pr(1, j1) := 0;
        rout_pr(1, 4) := 1.0;
    END;
  END ELSE
  BEGIN
    IF k1 = 7 THEN
      BEGIN
        rout_pr(7, 7) := 1.0;
      END ELSE
    BEGIN
      FOR j1 := 1 STEP 1 UNTIL N DO
        rout_pr(4, j1) := 0;
        rout_pr(4, 7) := 1.0;
    END;
  END ELSE
  BEGIN
    FOR i1 := (2, 3, 5, 6) DO
      BEGIN
        FOR j1 := 1 STEP 1 UNTIL N DO
          rout_pr(i1, j1) := 0;
          rout_pr(i1, 4) := 1;
        END;
    END;

PRINT("** * * The Shortest Path Matrix (II) * * **");
PRINT("1 The path array:");
PRINT(path(1 STEP 1 UNTIL N));
PRINT("2 The cost matrix");
FOR i1:=1 STEP 1 UNTIL N DO
    PRINT(d(i1, 1 STEP 1 UNTIL N));
    PRINT("(i1, 1 STEP 1 UNTIL N): ");
    FOR i1:=1 STEP 1 UNTIL N DO
        PRINT(rout_pr(i1, 1 STEP 1 UNTIL N));
    END;

&-----------------------------------------------
& Find the Jacksonian routing probability
&-----------------------------------------------
PROCEDURE Jackson;
REAl r_sum;
INTEGER i1, j1;
BEGIN
    FOR i1:=1 STEP 1 UNTIL N DO
        BEGIN
            r_sum:=0;
            FOR j1:=0 STEP 1 UNTIL N DO
                r_sum:=r_sum*(IF graph(i1, j1)<x0 THEN graph(i1, j1)
                ELSE 0.0);
            FOR j1:=0 STEP 1 UNTIL N DO
                jack_pr(i1, j1):=IF graph(i1, j1)<x0 THEN graph(i1, j1)/r_sum
                ELSE 0.0;
        END;
        PRINT("** * The Jacksonian Matix * * **");
    FOR i1:=1 STEP 1 UNTIL N DO
        PRINT(jack_pr(i1, 1 STEP 1 UNTIL N));
    END;

& processing nodes
/STATION/
NAME=n(1);
SERVICE(X)=
BEGIN
    CStTpc_time;
    Ck:=n(1).FIRST;
    $MACRO 1_delay(nodel,node2,sva_time)
        nb:=1(nodel,node2).NB;
        l1(nodel,node2).d_time:=nb*sva_time;
    $END
    $MACRO n_delay(node)
        nb:=n(node).NB;
        n(node).d_time:=nb*pc_time;
    $END
    $l_delay(1,2,sva_time) $n_delay(2) $l_delay(2,3,sva_time)
    $n_delay(3) $l_delay(3,6,sva_time) $n_delay(6)
    $l_delay(6,7,sva_time) $n_delay(7)
    $l_delay(1,4,svb_time) $n_delay(4) $l_delay(4,7,svb_time)
    t1:=l(1,4).d_time+n(2).d_time+l(2,3).d_time+n(3).d_time+l(3,6).d_t
& New codes for A4 algorithm &

IF (b1<=b) AND (b2<=b) THEN
  IF t1<=t2 THEN
    cx.net_id:=1;
    TRANSIT(1(1,2));
  END
  ELSE
    BEGIN
      cx.net_id:=2;
      TRANSIT(1(2,4));
    END;
  IF (b1<=b) AND (b2>b) THEN
    BEGIN
      cx.net_id:=1;
      TRANSIT(1(1,2));
    END;
  IF (b1>b) AND (b2<=b) THEN
    BEGIN
      cx.net_id:=2;
      TRANSIT(1(1,4));
    END;
  IF (b1>b) AND (b2>b) THEN
    BEGIN
      IF b1 <= b2 THEN
        BEGIN
          cx.net_id:=1;
          TRANSIT(1(1,2));
        END;
      IF b1 > b2 THEN
        BEGIN
          cx.net_id:=2;
          TRANSIT(1(1,4));
        END;
    END;

SERVICE(1)=CST(pc_time);
TRANSIT(1)=1(1,1 STEP 1 UNTIL N), jack_pr(1, 1 STEP 1 UNTIL N), OUT;
/STATION/
NAME=n(3);
SERVICE=CST(pc_time);
TRANSIT(1)=1(2,1 STEP 1 UNTIL N), rout_pr(2,1 STEP 1 UNTIL N);
TRANSIT(1)=1(2,1 STEP 1 UNTIL N), jack_pr(2,1 STEP 1 UNTIL N), OUT;
/STATION/
NAME=n(4);
SERVICE=CST(pc_time);
TRANSIT(1)=1(3,1 STEP 1 UNTIL N), rout_pr(3,1 STEP 1 UNTIL N);
TRANSIT(1)=1(3,1 STEP 1 UNTIL N), jack_pr(3,1 STEP 1 UNTIL N), OUT;
/STATION/
NAME=n(5);
SERVICE=CST(pc_time);
TRANSIT(1)=1(4,1 STEP 1 UNTIL N), rout_pr(4,1 STEP 1 UNTIL N);
TRANSIT(1)=1(4,1 STEP 1 UNTIL N), jack_pr(4,1 STEP 1 UNTIL N); & no ou
TRANSIT(X)=1(5.1 STEP 1 UNTIL N), rout_pr(5.1 STEP 1 UNTIL N), OUT;
TRANSIT(Y)=1(5.1 STEP 1 UNTIL N), jack_pr(5.1 STEP 1 UNTIL N), OUT;
/SERVICE/
NAME=n(6);
SERVICE=GST(pc_time);
TRANSIT(X)=1(6.1 STEP 1 UNTIL N), rout_pr(6.1 STEP 1 UNTIL N);
TRANSIT(Y)=1(6.1 STEP 1 UNTIL N), jack_pr(6.1 STEP 1 UNTIL N), OUT;
/SERVICE/
NAME=n(7);
SERVICE=
BEGIN
GST(pc_time);
CX:=n(7).FIRST;
IF CX.CCLASS=X THEN
BEGIN
CT:=TIME:CX.S_TIME;
& PRINT("MESSAGE:" , CX.S_NB,".delay=", CT);
IX:=INTREAL(CX*10)*2+1:
IF IX>IX-1 THEN IX:=IX-1: & (FX(1999),FX(2000)) is the last pa
=> ix
IF IX<1 THEN IX:=1;
IF IX>IX_MAX-1 THEN NB_MAX:=NB_MAX+1: & the delay that exceeds
=> ix_MAX
NB_MSG:=NB_MSG+1;
FX(IX+1):=FX(IX+1)+1;
& New codes for A4 algorithm &
=> iX_MAX
N_ID:=CX.NET_ID;
IF N_ID=1 THEN
BEGIN
NB1_MSG:=NB1_MSG+1;
FX1(IX+1):=FX1(IX+1)+1;
IF IX>IX_MAX-1 THEN NB1_MAX:=NB1_MAX+1;
TAIL:=REALINT(NB1_MSG)*(1-B_PCT);
TEMP:=NB1_MAX;
JX:=IX_MAX;
WHILE (TEMP<TAIL) AND (JX>=1) DO
BEGIN
TEMP:=TEMP+FX1(JX);
JX:=JX-2;
END;
END;
B1:=REALINT(JX)/20.0;
END;
IF N_ID=2 THEN
BEGIN
NB2_MSG:=NB2_MSG+1;
FX2(IX+1):=FX2(IX+1)+1;
IF IX>IX_MAX-1 THEN NB2_MAX:=NB2_MAX+1;
TAIL:=REALINT(NB2_MSG)*(1-B_PCT);
TEMP:=NB2_MAX;
JX:=IX_MAX;
WHILE (TEMP<TAIL) AND (JX>=1) DO
BEGIN
TEMP:=TEMP+FX2(JX);
JX:=JX-2;
END;
END;
B2:=REALINT(JX)/20.0;
END;
& PRINT("received from net", n_id);
& PRINT("tail":tail,"temp":temp);

END;

END;

TRANSLATE=OUT;

& background source nodes

&

/STATION/

NAME=s(1);

TYPE=SOURCE;

SERVICE=

BEGIN

EXP(1a_time):

& X is created only when system is warm-up

IF (TIME>warm-up) AND DRAW(0.45) THEN

BEGIN

cx:=s(1).FIRST;

cx.s_time:=TIME;

cx.s_nb:=Nx;

Nx:=Nx+1;

TRANSLATE(n(1),X):

END

ELSE TRANSLATE(n(1),Y):

END;

/STATION/

NAME=s(2);

TYPE=SOURCE;

SERVICE=EXP(1a_time);

TRANSLATE=n(2),Y,1;

/STATION/

NAME=s(3);

TYPE=SOURCE;

SERVICE=EXP(1a_time);

TRANSLATE=n(3),Y,1;

/STATION/

NAME=s(5);

TYPE=SOURCE;

SERVICE=EXP(1a_time);

TRANSLATE=n(5),Y,1;

/STATION/

NAME=s(6);

TYPE=SOURCE;

SERVICE=EXP(1a_time);

TRANSLATE=n(6),Y,1;

&

& comm. links

&MACRO link(s_node,d_node,s_time) & definition for communication

/STATION/ NAME=l(s_node,d_node); & link stations

SERVICE=EXP(s_time);

TRANSLATE=m(d_node),1;

&

$link(1.2,sva_time) $link(1.3,sva_time) $link(1.4,sva_time) $link(2.5,sva_time) $link(3.6,sva_time) $link(5.2,sva_time) $link(5.3,sva_time) $link(5.7,sva_time) $link(6.3,sva_time) $link(6.5,sva_time) $link(6.7,sva_time)
&
#link(1,4,svb_time) #link(4,7,svb_time)
#link(2,4,svb_time) #link(3,4,svb_time) #link(5,4,svb_time)
#link(6,4,svb_time) #link(4,2,svb_time)
#link(4,5,svb_time) #link(4,3,svb_time) #link(4,5,svb_time)
&
/CONTROL/
CLASS=ALL QUEUE:
ENTRY=BEGIN
PRINT("Inter-arrival time?"); i_a_time:=GET(REAL);
PRINT("Service time for net 1?"); sva_time:=GET(REAL);
PRINT("Service time for net 2?"); svb_time:=GET(REAL);
PRINT("Processing time?"); pc_time:=GET(REAL);
PRINT("Delay bound?"); B:=GET(REAL);
PRINT("Simulation time?"); t_max:=GET(REAL);
FOR ix:=1 STEP 2 UNTIL IX-1 DO & initializing the delay distr. ar
ray
fx(ix)]:=REALINT(ix+1)/20.0;
END;
TMAX=t_max;
EXIT=BEGIN & print the delay distribution
PRINT("** * * Delay Distribution:");
FOR ix:=1 STEP 2 UNTIL IX_MAX-1 DO BEGIN
fx(ix+1):=fx(ix+1)/REALINT(nb_msg);
PRINT(fx(ix),fx(ix+1), .CURVE(fx,fx(ix+1)));
end;
fx(i):=fx(i)+fx(i+1);
d_mean:=d_mean+fx(i)*fx(i+1);
d_var:=d_var+fx(i)*fx(i)*fx(i+1);
END;
d_var:=d_var - d_mean*d_mean;
temp:=REALINT(nb_max)/REALINT(nb_msg);
jx:=ix_max;
WHILE (temp<0.01) AND (jx>=1) DO
BEGIN
END;
END;
/EXEC/
BEGIN
sp1[1,7]; & find the shortest path in network 1
sp2[4,7]; & find the shortest path in network 2
Jackson: & find the routing prob. for Y
PRINT("** * * Simulation result for single SHORTEST PATH **");
SIMUL.
END;

* * The Shortest Path Matrix (1) * *
1) The path array:
1
2
3
4
5
6
2) The cost matrix

0. 0.1200 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
3) The routing probability matrix:
   0. 0.100 0.0 0. 0. 0. 0.
   0. 0. 1.000 0. 0. 0. 0.
   0. 0. 0. 1.000 0. 0. 0.
   0. 0. 0. 1.000 0. 0. 0.
   0. 0. 0. 0. 0. 0. 0.
   0. 0. 0. 0. 0. 0. 0.

* * * The Shortest Path Matrix (II) * * *

1) The path array:
   1 2 1 4 3 6

2) The cost matrix
   0. 0.2000 1.000 1.000 1.000 0.0 1.000 0.
   0. 0.2000 1.400 1.000 3.000 3.000 1.000 0.0
   0. 0.2000 1.000 1.000 3.000 3.000 1.000 0.0
   0. 0.2000 1.000 1.000 3.000 3.000 1.000 0.0
   0. 0.2000 1.000 1.000 3.000 3.000 1.000 0.0

3) The routing probability matrix:
   0. 1.000 0. 0. 0. 0. 0.
   0. 0. 1.000 0. 0. 0. 0.
   0. 0. 0. 1.000 0. 0. 0.
   0. 0. 0. 0. 1.000 0. 0.
   0. 0. 0. 0. 0. 1.000 0.
   0. 0. 0. 0. 0. 0. 1.000

* * * The Jacksonian Matrix * * *
   0. 0. 0.9091e-01 0.4545 0.4545 0.
   0. 0. 0.1515e-01 0.2273 0.3788 0.2174e
   0. 0. 0.1538 0.3261 0.3261 0.3261 0.
   0. 0. 0.3571 0.2143 0.2143 0.
   0.1087 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 *
******************************************************************************

106
<table>
<thead>
<tr>
<th>n</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5002e-01</td>
<td>0.9690e-01</td>
</tr>
<tr>
<td>2</td>
<td>0.5002e-01</td>
<td>0.4288e-01</td>
</tr>
<tr>
<td>3</td>
<td>0.5002e-01</td>
<td>0.5402e-01</td>
</tr>
<tr>
<td>4</td>
<td>0.5002e-01</td>
<td>0.3567</td>
</tr>
<tr>
<td>5</td>
<td>0.5002e-01</td>
<td>0.2926e-01</td>
</tr>
<tr>
<td>6</td>
<td>0.5002e-01</td>
<td>0.3275</td>
</tr>
<tr>
<td>7</td>
<td>0.5002e-01</td>
<td>0.2630</td>
</tr>
<tr>
<td>8</td>
<td>0.5002e-01</td>
<td>0.2338</td>
</tr>
<tr>
<td>9</td>
<td>0.5002e-01</td>
<td>0.3406</td>
</tr>
<tr>
<td>10</td>
<td>0.5002e-01</td>
<td>0.1362e-01</td>
</tr>
<tr>
<td>11</td>
<td>0.5002e-01</td>
<td>0.3270</td>
</tr>
<tr>
<td>12</td>
<td>0.5002e-01</td>
<td>0.3435</td>
</tr>
<tr>
<td>13</td>
<td>0.5002e-01</td>
<td>0.3435</td>
</tr>
<tr>
<td>14</td>
<td>0.5002e-01</td>
<td>0.2577</td>
</tr>
<tr>
<td>15</td>
<td>0.5002e-01</td>
<td>0.2285</td>
</tr>
<tr>
<td>16</td>
<td>0.5002e-01</td>
<td>0.1795</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5161</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>0.5118</td>
<td>1.000</td>
</tr>
<tr>
<td>3</td>
<td>0.4875</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>0.5120</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>0.5004</td>
<td>1.000</td>
</tr>
<tr>
<td>6</td>
<td>0.1220</td>
<td>0.8267e-01</td>
</tr>
<tr>
<td>7</td>
<td>0.1220</td>
<td>0.7135e-01</td>
</tr>
<tr>
<td>8</td>
<td>0.1221</td>
<td>0.1131e-01</td>
</tr>
<tr>
<td>9</td>
<td>0.1255</td>
<td>0.6344e-01</td>
</tr>
<tr>
<td>10</td>
<td>0.1255</td>
<td>0.6344e-01</td>
</tr>
<tr>
<td>11</td>
<td>0.1661</td>
<td>0.1254</td>
</tr>
<tr>
<td>12</td>
<td>0.1611</td>
<td>0.4385e-01</td>
</tr>
<tr>
<td>13</td>
<td>0.1690</td>
<td>0.8154e-01</td>
</tr>
<tr>
<td>14</td>
<td>0.1348</td>
<td>0.1348e-01</td>
</tr>
<tr>
<td>15</td>
<td>0.1273</td>
<td>0.8716e-01</td>
</tr>
<tr>
<td>16</td>
<td>0.1348</td>
<td>0.1348e-01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>l</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1697</td>
<td>0.2640</td>
</tr>
<tr>
<td>2</td>
<td>0.1697</td>
<td>0.2640</td>
</tr>
<tr>
<td>3</td>
<td>0.1188</td>
<td>0.2907</td>
</tr>
<tr>
<td>4</td>
<td>0.1188</td>
<td>0.2907</td>
</tr>
<tr>
<td>5</td>
<td>0.1267</td>
<td>0.1276</td>
</tr>
<tr>
<td>X</td>
<td>0.1267</td>
<td>0.1276</td>
</tr>
<tr>
<td>----</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1</td>
<td>0.1731</td>
<td>0.2722</td>
</tr>
<tr>
<td>Y</td>
<td>0.1731</td>
<td>0.2722</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1132</td>
<td>0.784e-01</td>
</tr>
<tr>
<td>X</td>
<td>0.1138</td>
<td>0.6657e-01</td>
</tr>
<tr>
<td>Y</td>
<td>0.1101</td>
<td>0.1187e-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1705</td>
<td>0.1715</td>
</tr>
<tr>
<td>Y</td>
<td>0.1705</td>
<td>0.1715</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1731</td>
<td>0.1751</td>
</tr>
<tr>
<td>Y</td>
<td>0.1731</td>
<td>0.1751</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1741</td>
<td>0.1715</td>
</tr>
<tr>
<td>Y</td>
<td>0.1741</td>
<td>0.1715</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1747</td>
<td>0.1759</td>
</tr>
<tr>
<td>Y</td>
<td>0.1747</td>
<td>0.1759</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1671</td>
<td>0.4678</td>
</tr>
<tr>
<td>X</td>
<td>0.1602</td>
<td>0.4361e-01</td>
</tr>
<tr>
<td>Y</td>
<td>0.1678</td>
<td>0.4242</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1213</td>
<td>0.3016</td>
</tr>
<tr>
<td>Y</td>
<td>0.1213</td>
<td>0.3016</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1688</td>
<td>0.2435</td>
</tr>
<tr>
<td>Y</td>
<td>0.1688</td>
<td>0.2435</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1218</td>
<td>0.1773</td>
</tr>
<tr>
<td>Y</td>
<td>0.1218</td>
<td>0.1773</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1167</td>
<td>0.1136</td>
</tr>
<tr>
<td>Y</td>
<td>0.1167</td>
<td>0.1136</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1199</td>
<td>0.1206</td>
</tr>
<tr>
<td>Y</td>
<td>0.1199</td>
<td>0.1206</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1722</td>
<td>0.2556</td>
</tr>
<tr>
<td>Y</td>
<td>0.1722</td>
<td>0.2556</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1202</td>
<td>0.1783</td>
</tr>
<tr>
<td>Y</td>
<td>0.1202</td>
<td>0.1783</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.1192</td>
<td>0.8023e-01</td>
</tr>
<tr>
<td>X</td>
<td>0.1189</td>
<td>0.6956e-01</td>
</tr>
<tr>
<td>Y</td>
<td>0.1209</td>
<td>0.1067e-01</td>
</tr>
</tbody>
</table>

END OF SIMULATION

MEMORY USED: 29265 WORDS OF 4 BYTES

(* 11.71 % OF TOTAL MEMORY)

nb of msg= 1757

** Delay Distribution:

<table>
<thead>
<tr>
<th>Prob</th>
<th>0.1000</th>
<th>0.8537e-02</th>
<th>0.8537e-02</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

108
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2000</td>
<td>0.4269e-01</td>
<td>0.4269e-01</td>
<td></td>
</tr>
<tr>
<td>0.3000</td>
<td>0.7114e-01</td>
<td>0.7114e-01</td>
<td></td>
</tr>
<tr>
<td>0.4000</td>
<td>0.1059</td>
<td>0.1059</td>
<td></td>
</tr>
<tr>
<td>0.5000</td>
<td>0.1360</td>
<td>0.1360</td>
<td></td>
</tr>
<tr>
<td>0.6000</td>
<td>0.1394</td>
<td>0.1394</td>
<td></td>
</tr>
<tr>
<td>0.7000</td>
<td>0.1309</td>
<td>0.1309</td>
<td></td>
</tr>
<tr>
<td>0.8000</td>
<td>0.1167</td>
<td>0.1167</td>
<td></td>
</tr>
<tr>
<td>0.9000</td>
<td>0.9106e-01</td>
<td>0.9106e-01</td>
<td></td>
</tr>
<tr>
<td>1.000</td>
<td>0.5919e-01</td>
<td>0.5919e-01</td>
<td></td>
</tr>
<tr>
<td>1.100</td>
<td>0.3984e-01</td>
<td>0.3984e-01</td>
<td></td>
</tr>
<tr>
<td>1.200</td>
<td>0.2163e-01</td>
<td>0.2163e-01</td>
<td></td>
</tr>
<tr>
<td>1.300</td>
<td>0.1366e-01</td>
<td>0.1366e-01</td>
<td></td>
</tr>
<tr>
<td>1.400</td>
<td>0.7399e-02</td>
<td>0.7399e-02</td>
<td></td>
</tr>
<tr>
<td>1.500</td>
<td>0.6261e-02</td>
<td>0.6261e-02</td>
<td></td>
</tr>
<tr>
<td>1.600</td>
<td>0.5122e-02</td>
<td>0.5122e-02</td>
<td></td>
</tr>
<tr>
<td>1.700</td>
<td>0.2277e-02</td>
<td>0.2277e-02</td>
<td></td>
</tr>
<tr>
<td>1.800</td>
<td>0.1707e-02</td>
<td>0.1707e-02</td>
<td></td>
</tr>
<tr>
<td>1.900</td>
<td>0.5692e-03</td>
<td>0.5692e-03</td>
<td></td>
</tr>
<tr>
<td>2.000</td>
<td>0.5692e-03</td>
<td>0.5692e-03</td>
<td></td>
</tr>
<tr>
<td>2.100</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.200</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.300</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.400</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.500</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.600</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.700</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.800</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>2.900</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.000</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.100</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.200</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.300</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.400</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.500</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.600</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.700</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.800</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>3.900</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.000</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.100</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.200</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.300</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.400</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.500</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.600</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.700</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.800</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>4.900</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.000</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.100</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.200</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.300</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.400</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.500</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.600</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.700</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.800</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>5.900</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>6.000</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
<tr>
<td>6.100</td>
<td>0.</td>
<td>0.</td>
<td></td>
</tr>
</tbody>
</table>
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


112


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

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

[35] N. Abramson, "The Aloha System – Another Alternative For Computer Com-

[36] M. Gerla and L. Kleinrock, "On the Topological Design of Distributed Com-

[37] A. Kershenbaum and R. R. Boorstyn, "Centralized Teleprocessing Network

[38] C. L. Monma and D. D. Sheng, "Backbone Network Design and Performance
 Analysis: A Methodology for Packet Switching Networks", IEEE J. Select.


 Comparison of Techniques for Packet-Switching Networks.", IEEE Tranc.


tic Optimization Algorithm Minimizing Expected Flow Times on Uniform

114


