Telecommunication Networks and Computer Systems

Thomas G. Robertazzi

Computer Networks and Systems: Queueing Theory and Performance Evaluation

Springer-Verlag

Thomas G. Robertazzi

Computer Networks and Systems: Queueing Theory and Performance Evaluation

With 92 Illustrations

Springer-Verlag
New York Berlin Heidelberg London
Paris Tokyo Hong Kong Barcelona

Thomas G. Robertazzi
Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794-2350
USA

Series Editors

Mario Gerla
Department of Computer Science
University of California
Los Angeles, CA 90024, USA

Aurel Lazar
Department of Electrical
Engineering
and Center for
Telecommunications Research
Columbia University
New York, NY 10027, USA

Paul Kuehn
Institute of Communications
Switching and Data Technics
University of Stuttgart
D-7000 Stuttgart
Federal Republic of Germany

Printed on acid-free paper.

Camera-ready text prepared by the author using troff.
Printed and bound by: R.R. Donnelley and Sons, Harrisonburg, Virginia.
Softcover reprint of the hardcover 1st edition 1990

9 8 7 6 5 4 3 2 1

ISBN-13:978-1-4684-0387-9 e-ISBN-13:978-1-4684-0385-5
DOI: 10.1007/978-1-4684-0385-5

To Marsha and Rachel

Preface

Statistical performance evaluation has assumed an increasing amount of importance as we seek to design more and more sophisticated communication and information processing systems. The ability to predict a proposed system's performance without actually having to construct it is an extremely cost effective design tool.

This book is meant to be a first year graduate level introduction to the field of statistical performance evaluation. As such, it covers queueing theory (chapters 1-4) and stochastic Petri networks (chapter 5). There is a short appendix at the end of the book which reviews basic probability theory. At Stony Brook, this material would be covered in the second half of a two course sequence (the first half is a computer networks course using a text such as Schwartz's *Telecommunications Networks*). Students seem to be encouraged to pursue the analytical material of this book if they first have some idea of the potential applications.

I am grateful to B.L. Bodnar, J. Blake, J.S. Emer, M. Garrett, W. Hagen, Y.C. Jenq, M. Karol, J.F. Kurose, S.-Q. Li, A.C. Liu, J. McKenna, H.T. Mouftah and W.G. Nichols, I.Y. Wang, the IEEE and Digital Equipment Corporation for allowing previously published material to appear in this book.

My appreciation of this material has been enhanced by interaction with my students in Stony Brook's Congestion and Delay in Communications Systems Course. I am grateful to S. Rappaport for encouraging me to teach this course. Thanks are due for editorial assistance to Z. Ruder, K. Dreyhaupt, J. Day and J. Abrahams of Springer-Verlag and to A. Lazar of Columbia University. Thanks are due to M. Gerla for reviewing the manuscript. This book benefited from the drawing ability of L. Koh. The graph of the Poisson distribution was made by J. Shor. This book would not have been possible without the use of computer facilities supervised by J. Murray, M. Lee and G. Liu. I would like to gratefully acknowledge the support, in the course of this work, of the National Science Foundation (grant no. NCR-8703689) and of the SDIO/IST and managed by the U.S. Office of Naval Research (grant no. N00014-85-K0610).

Finally, I would like to thank my wife and daughter for their encouragement and support during the course of this project.

Parts of this book have benefited from the influence of a number of earlier, classic works which no personal library of performance evaluation should be without. These recommended readings are:

Bruell, S.C. and Balbo, G., *Computational Algorithms for Closed Queueing Networks,* North-Holland, N.Y., 1980.

Cooper, R.B., *Introduction to Queueing Theory,* North-Holland, N.Y., 1981.

Gross, D. and Harris, C.M., *Fundamentals of Queueing Theory,* Wiley, New York, 1974, 1985.

Kelly, F.P., *Reversibility and Stochastic Networks,* John Wiley & Sons, New York, 1979.

Kleinrock, L., *Queueing Systems, Vol. I: Theory,* Wiley, New York, 1975.

Kobayashi, H., *Modeling and Analysis: An Introduction to System Performance Evaluation,* Addison-Wesley, Reading, Mass., 1978.

Schwartz, M., *Telecommunications Networks: Protocols, Modeling and Analysis,* Addison-Wesley, Reading, Mass., 1987.

A solution manual is available from the author for instructors who adopt this book for a course. Please send requests on university stationery.

T.G.R.
Dept. of Electrical Engineering,
SUNY at Stony Brook,
Stony Brook, N.Y. 11794,
May 1990

Table of Contents

complex model appears in [BASK 75].

3.3 Algebraic Topological Interpretation of P.F. Solution

3.3.1 Introduction

Why does the product form solution exist and how can one explain its simple form? During the mid 1980's an interesting and aesthetically pleasing interpretation of the existence of the product form solution became possible. The basic idea is the realization that the state transition diagrams of product form networks can be decomposed into an aggregation of elementary "building blocks". Because these building blocks do not interact with each other in a substantial way, they can be solved in isolation for the relative state probabilities. The product form solution, then, for a specific equilibrium state probability in terms of a reference probability is really a recursion that patches together these building block solutions along a "path" from the state of interest back to the reference state.

This algebraic topological interpretation of the product form solution is important for several reasons. It gives a constructive means for arriving at the product form solution. It helps to explain why certain models have a product form solution and why others do not. It becomes possible to modify or select models so that they have a product form solution [HAMI 86a,86b,89] or to construct product form models which provide performance bounds for the more difficult to analyze non-product form networks [VAND].

In what follows these ideas will be expanded upon. The sources for this material are from the work of A. Lazar of Columbia University [LAZA 84a, 84b, 84c], I. Wang of the New Jersey Institute of Technology [WANG 86] and the author.

3.3.2 A First Look at Building Blocks

Consider the cyclic three queue network in Figure 3.3. The service times are exponential random variables. The state transition diagram is illustrated in Figure 3.4. The state description for this model is two dimensional since this is a closed network. That is, if n_1 and n_2 are the numbers of customers in the upper queues, then $N-n_1-n_2$ is automatically the number in the lower queue where N is the total number of customers in the network.

Now let us look at the local balance equations of this product form network. Recalling the rule that allows us to pair transitions for local balance, in Figure 3.5 the paired transitions which exhibit local balance are shown. Observing the lower left corner of the state transition diagram one can see that the local balance pairings mean that the flow from (0,0) to (1,0) equals that from (1,0) to (0,1). But the flow from (1,0) to (0,1) must equal that from (0,1) to (0,0). Finally the flow from (0,1) to (0,0) must equal that from (0,0) to (1,0).

Thus an alternative view of local balance is that there is a circular flow of probability flux along the edges of the triangular sub-element: (0,0) to (1,0) to (0,1) and back to (0,0). With this circular flow, each edge has the same magnitude of probability flux.

Continuing with this logic, each triangular sub-element (or building block) in the state transition diagram has a circular flow about its edge. This is illustrated in Figure 3.6.

Because of the decomposition of the flow of probability flux that is thus possible, one can effectively remove one building block at a time from the remaining building blocks without disturbing its flow pattern, except for a renormalization. The renormalization is necessary so that the probabilities of the remaining states sum to one. A completely decomposed state transition diagram is shown in Figure 3.7.

If one wants to solve for p_{21} for instance, one solves building block C in this last figure to obtain the local balance equation

$$\mu_1 p_{11} = \mu_2 p_{21} \tag{3.37}$$

and the solution:

$$p_{21} = \frac{\mu_1}{\mu_2} p_{11} \tag{3.38}$$

Solving building block E one obtains the local balance equation

$$\mu_3 p_{11} = \mu_1 p_{10} \tag{3.39}$$

and the solution:

$$p_{11} = \frac{\mu_1}{\mu_3} p_{10} \tag{3.40}$$

Finally, the building block D yields the local balance equation

$$\mu_1 p_{00} = \mu_2 p_{10} \tag{3.41}$$

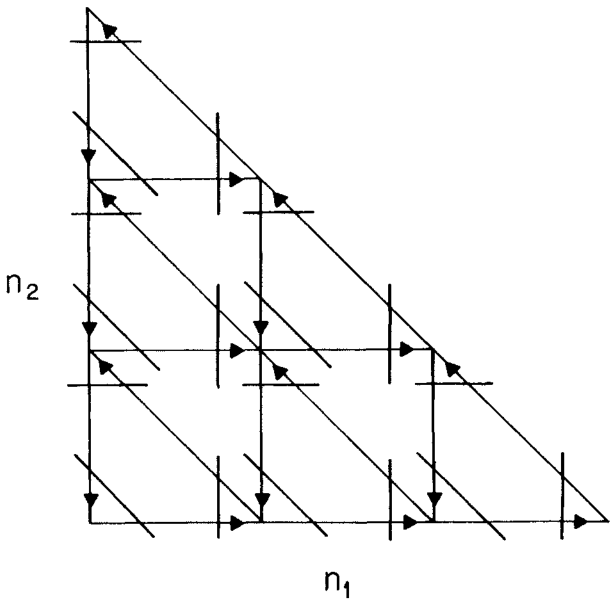


Fig. 3.5: Local Balance Pairings

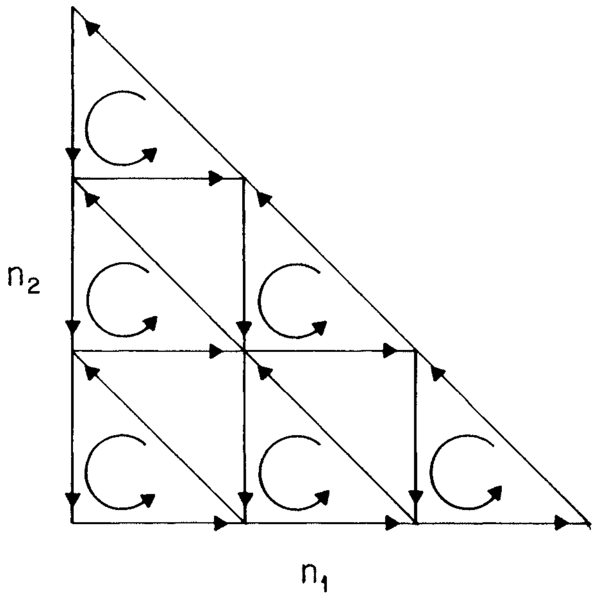


Fig. 3.6: Circular Flows

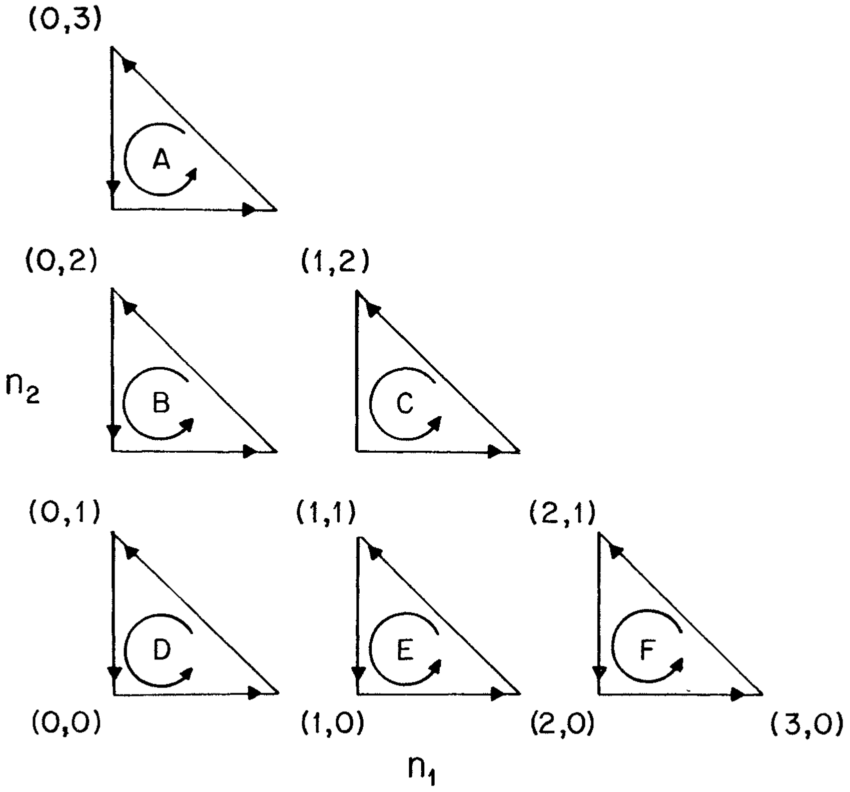


Fig. 3.7: Decomposed State Transition Diagram

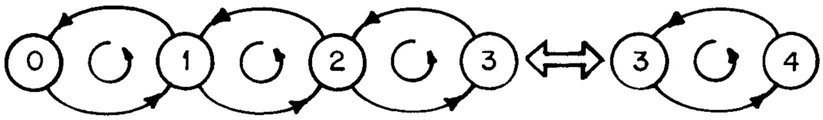


Fig. 3.8: M/M/1 System Building Blocks

and the solution:

$$p_{10} = \frac{\mu_1}{\mu_2} p_{00} \tag{3.42}$$

Substituting these solutions back into one another yields

$$p_{21} = \frac{\mu_1}{\mu_2} \frac{\mu_1}{\mu_3} \frac{\mu_1}{\mu_2} p_{00} = \left(\frac{\mu_1}{\mu_2}\right)^2 \left(\frac{\mu_1}{\mu_3}\right) p_{00} \tag{3.43}$$

or the product form solution. What we have done is solved the building blocks along a path from the desired state to the reference state. Actually this is a conservative system. Any path between the two states would have produced the same results.

If we solve for all the state probabilities we can deduce that:

$$p_{n_1 n_2} = \left(\frac{\mu_1}{\mu_2}\right)^{n_1} \left(\frac{\mu_1}{\mu_3}\right)^{n_2} p_{00} \tag{3.44}$$

One more point needs to be made. The state transition diagram of Figure 3.4 is the same as that for an open network of two queues in series, with an upper bound on the *total* number of customers in both queues. In the diagram the upper bound, N, takes the form of the boundary $n_1+n_2 \leq N$. As $N \rightarrow \infty$ and we go to an open network of two infinite buffer queues in series the state transition diagram becomes infinite in extent but everything that has been said about the building block structure still holds.

Example: Consider the (open) M/M/1 queueing system. The state transition diagram is constituted of building blocks consisting of two transitions each. For the *infinite* buffer M/M/1 queueing system there are an infinite number of such building blocks. In the case of the M/M/1/N *finite* buffer system, Figure 3.8, a finite number of such building blocks are pasted together. Note that the same state transition diagram models a (closed) cyclic queueing system of two queues.

Naturally there is only one path from any state to the usual reference state, p_0.

∇

Example: Consider a closed cyclic network of four queues with exponential servers and with only two customers in the network. The state transition diagram is three dimensional and appears in Figure 3.9. The basic building block consists of four edges along a tetrahedron shaped volume of space. One could show that there are circular and equal valued flows of probability flux along the edges of these tetrahedrons and that any state probability can be found by solving these tetrahedrons along a path back to the reference probability.

Note that the state transition diagram of Figure 3.9 is the same as that for an open network of three queues in series, with an upper bound on the *total* number of customers in the queues. This bound takes the form of the plane boundary $n_1+n_2+n_3 \leq N$ in the state transition diagram. As $N \rightarrow \infty$ one has a system of infinite buffer queues and the state transition diagram consists of an infinite number of tetrahedral building blocks.

∇

We have seen that the state transition diagrams of a number of product form networks can be decomposed into elementary building blocks. In fact, as we shall see, this is true of any product form network. One can also look at the reverse case, the construction of state transition diagrams out of elementary building blocks. We will call the process of aggregating building blocks to form a state transition diagram, *geometric replication.* The reason is that the same geometric building block is usually replicated to form the overall diagram. Note that in the language of mathematical topology one would call a building block a "cell" and a state transition diagram would be called a "complex". Thus [LAZA 84a]:

Definition: The constructive process of aggregating multi-dimensional cells into a complex is referred to as *geometric replication.*

The geometric replication of cells or building blocks is useful in explaining the nature of the product form solution. It will also be useful later in devising models with this solution.

3.3.3 Building Block Circulatory Structure

In this section the ideas of the last section will be made more precise [WANG 86].

Definition: The *circulatory structure* of a state transition diagram is the pattern of probability flux on the diagram transitions.

Perhaps the most important property of building blocks for product form networks is that they can be embedded into the overall state transition diagram without affecting the flow pattern of the rest of the diagram, except for a renormalization. This property is what allows one to solve

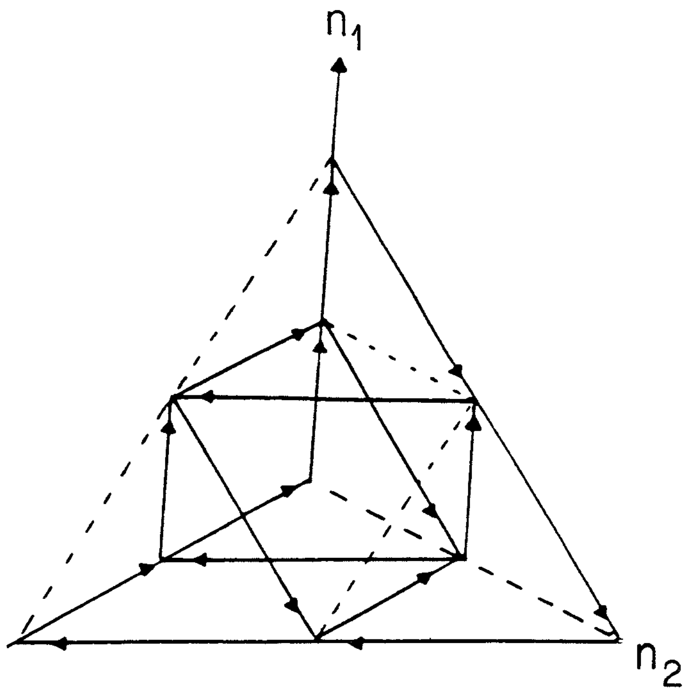


Fig. 3.9: State Transition Diagram for Four Queue Cyclic Network

building blocks in isolation in order to generate the recursive product form solution.

Definition: An *isolated circulation* consists of the probability flux along a subset of state transition diagram edges for which there is conservation of this flow at each adjacent state when the edges are embedded in the overall state transition diagram. Thus the relative equilibrium probabilities for these states can be solved for in isolation without considering the rest of the state transition diagram.

Definition: A *cyclic flow* is an isolated circulation in the form of a single cycle.

The "circular flows" of the building blocks of the previous section are cyclic flows. We can summarize by saying:

Duality Principle: There is a duality between the existence of local balance and the existence of isolated circulations. That is, the existence of local balance leads to isolated circulations and the presence of isolated circulations leads to local balance.

Proof: The former follows from the usual queueing interpretation of local balance. The latter follows from the definition of an isolated circulation.

∇

In other words, in observing local balance for a pair of transitions at a state, one is observing an isolated circulation passing through the state.

We will now characterize the building blocks:

Definition: A cyclic flow about L edges is said to be of length L.

Now let's recall the examples of the previous section for both open and closed networks. We can construct a table showing the lengths of the building blocks for the various cases:

Table 3.1: # of Queues vs. L

Closed	Open	L
2	1	2
3	2	3
4	3	4

This suggests the following:

Theorem: For a Markovian product form queueing network where the state of a queue is described by the number of customers in the queue, a closed path of L queues corresponds to an L length cyclic flow and an open path of L queues (into, through, and out of the network) corresponds to a L+1 length cyclic flow.

Proof: To prove this for a closed network is straight forward. Each queue in a closed path of L queues contributes one transition so the building block is of length L. For an open network of L queues each queue contributes one transition. There is also one additional transition that can be thought of as being due to a "virtual queue" that connects the output to the input. The transition rate of this additional transition is just the arrival rate to the network. This gives building blocks of length L+1.

∇

Next, let's characterize the sequence of events in a queueing network that are associated with, or serve to generate, a building block:

Definition a: For a closed network a *relay sequence* is a series of successive events consisting of a customer leaving queue i to enter queue j, followed by a departure from queue j which enters queue k, ... finally followed by an arrival back into queue i. Each of the queues is distinct and they form a closed cyclic path through the queueing network.

The sequence of events associated with a building block is called a relay sequence because a customer arriving at a queue is followed by a customer leaving the same queue. For a FIFO discipline this is not the same customer unless the queue is empty. It is as if customers are carrying a baton around a closed path, as in a relay race. There is an analogous definition for an open network:

Definition b: For an open network a "relay sequence" is a series of successive events corresponding to an arrival into queue i from outside the network, followed by a departure from queue i to queue j, followed by a departure from queue j to queue k, ... followed by a departure from the network. Each of the queues is distinct.

These ideas will now be expanded upon through a series of examples:

Example: Consider two queues, with exponential servers, in series forming an open network with Poisson arrivals, as in Figure 3.10. As has been mentioned, if the system has an infinite number of buffers the state transition diagram has building blocks as in Figure 3.7 but is infinite in

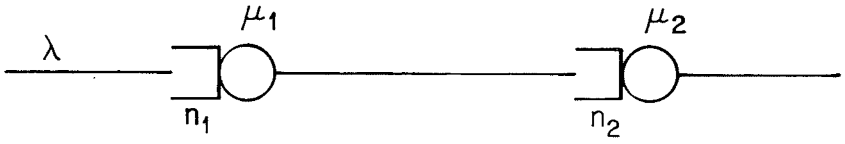


Fig. 3.10: Series (Tandem) Open Network

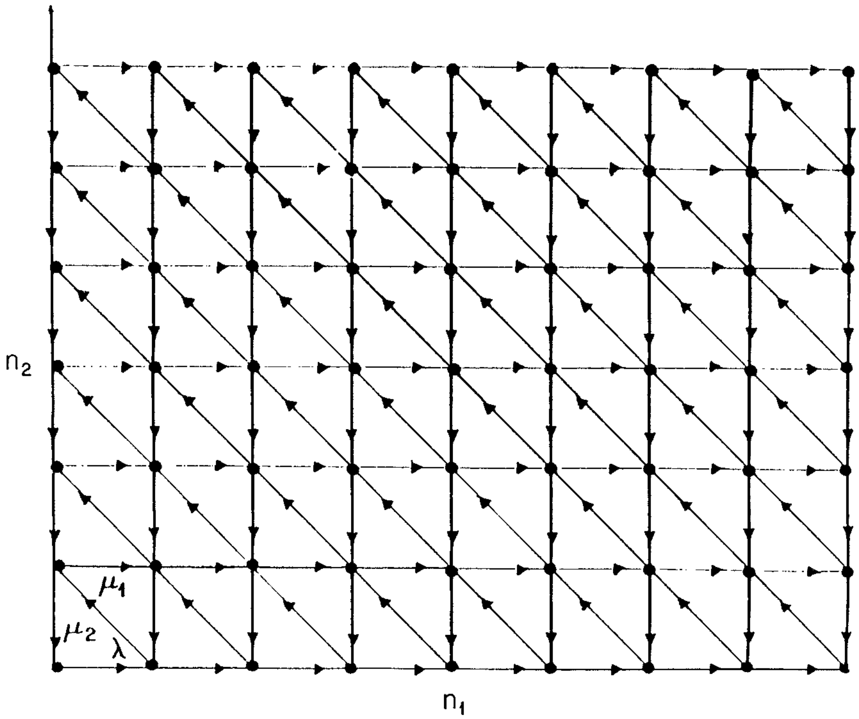


Fig. 3.11: Blocking State Transition Diagram

extent. The product form solution for this network is:

$$p_{n_1n_2} = \left(\frac{\lambda}{\mu_1}\right)^{n_1}\left(\frac{\lambda}{\mu_2}\right)^{n_2} p_{00}$$

Suppose now that there is an upper bound on the number of customers in *each* queue: $n_1 \leq n_{1,max}, n_2 \leq n_{2,max}$ This gives rise to the state transition diagram of Figure 3.11 with vertical and horizontal boundaries. This blocking network is now a non-product form network.

Why is this so? What has happened is that there are no longer integral building blocks along the upper and right side rectilinear boundary. Put another way, the pairing of transitions for local balance has been disrupted along these boundaries. That is, there are transitions along the boundary that are missing their natural partner for local balance pairing. This deficiency along the boundary disrupts the flow of probability flux throughout the state transition diagram so that there are no longer cyclic flows about the remaining integral building blocks and the product form solution no longer exists. Nico van Dijk of the Free University, The Netherlands, has written [VAND] that there is a "failure" of local balance in such a case. At this point the only way to calculate the state probabilities directly is by numerically solving the set of linear global balance equations.

Not every boundary destroys the product form solution. For instance, suppose that we have an upper bound on the *total* number of customers in both queues. As has been mentioned this leads to the state transition diagram like that of Figure 3.4. This queueing network has the same product form solution, except for the normalizing value of p_{00}, as that for a similar infinite buffer network. What is different, compared to the blocking network, is that the diagonal boundary $n_1 + n_2 \leq N$ allows integral building blocks and local balance pairing for all transitions, and thus the product form solution exists.

Finally, consider another modification where if either of the queues is full, customers will "skip" over it. The state transition diagram is shown in Figure 3.12. There are now integral building blocks of length two along the upper and right boundaries and integral building blocks of length three throughout the interior. Again there is local balance, cyclic flows and the same product form solution holds.

∇

Example: In this example we will look at the effect of random routing on the building block structure. The example involves a multiprocessor model.

A multiprocessor system [HWAN] consists of a group of processors interconnected with a group of memory modules. A closed Markovian queueing model of a multiprocessor system is shown in Figure 3.13. That

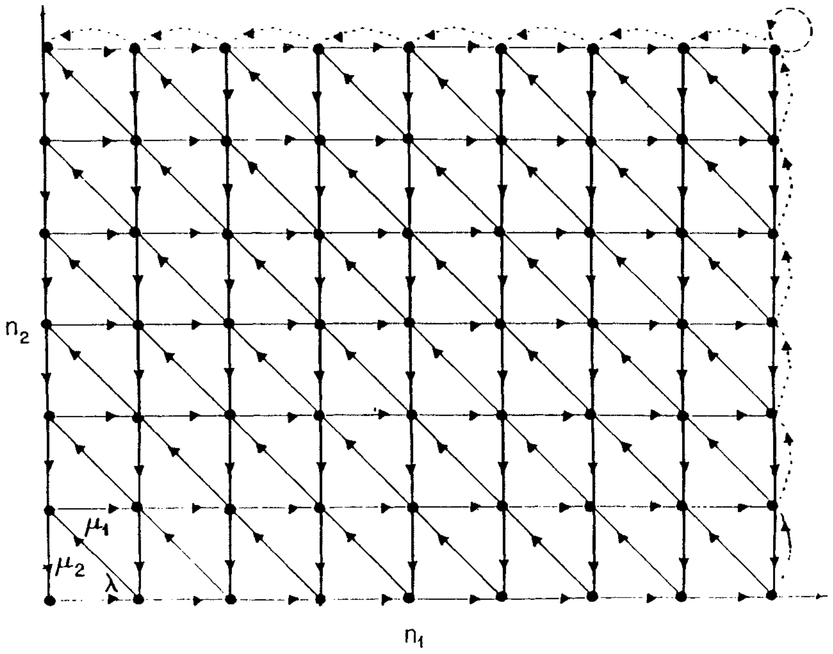


Fig. 3.12: Skipping State Transition Diagram

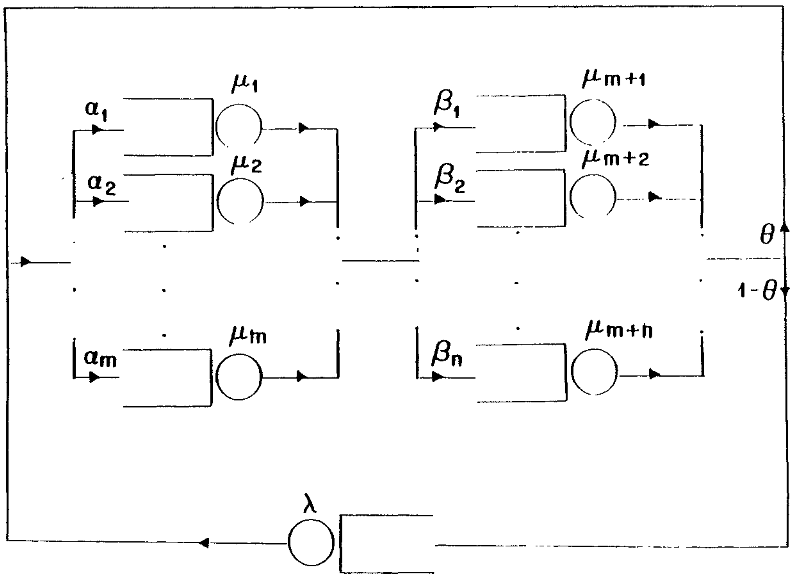


Fig. 3.13: Multiprocessor Queueing Model

is, the service times are exponential. From before, we know that this is a product form network. This model consists of three queueing systems arranged in a cyclic fashion. Two of the systems are comprised of a number of parallel queues and represent, respectively, the CPU's and memory modules. Routing into each of the two banks of parallel queues is random. That is, customers (really jobs) enter the m CPU's with probability $\alpha_1, \alpha_2, \cdots \alpha_m$ and customers enter the n memory modules with probabilities $\beta_1, \beta_2, \cdots \beta_n$. The third system is a single queue, departures from which represent submissions of jobs (or interactive commands) to the computer system.

In this multiprocessor system a customer leaving the memory module bank can proceed directly back to the CPU's with probability θ or it may go to the job submission queue to wait for its next execution with probability $1-\theta$. To simplify the following discussion we will concentrate on the special case when $\theta=0$.

For this reduced system, a "relay sequence" is a sequence of successive events corresponding to a customer entering a CPU queue followed directly by a customer departure from that CPU into a memory module followed directly by a customer departure from that module back to the job submission queue. Again, the departing customers are not necessarily the same as the arriving customers.

We will look at two types of building blocks, the second of which is a decomposed part of the first. The structure of the first building block for this m CPU and n memory module system is shown in Figure 3.14. Assume that "a" is a state $(k_1, k_2, \cdots k_l \cdots k_m, k_{m+1} \cdots k_{m+n})$ where k_l is the # of customers in the lth queue and the job submission queue is nonempty. A customer leaving the job submission queue corresponds to a transition into one of the 1,2 ... m CPU states, i.e., $(k_1, k_2, \cdots k_{l+1}, \cdots k_m, k_{m+1}, \cdots k_{m+n})$. A customer leaving a CPU corresponds to a transition into one of the m+1, m+2 ... m+n memory module states. Finally a memory module departure corresponds to a transition back to state a.

Thus, in queueing terms, this first building block is generated by *all possible* relay sequences starting from state a. The building block's net flow into and out of the block's states balance when the block is embedded in the state transition diagram. The building block is a subgraph whose relative state probabilities can be solved for in isolation.

Now let us consider the second type of building block. Let p_l be the state probability that the customer is in the CPU queue l if $1 \leq l \leq m$ and in memory module l if $m+1 \leq l \leq m+n$. Using flux conservation at a CPU state

$$\alpha_l \lambda p_a = \sum_{j=1}^{n} \beta_j \mu_l p_l = \mu_l p_l \qquad l=1,2,3 \ldots m \tag{3.45}$$

and at a memory module state:

$$\mu_l p_l = \beta_{l-m} \sum_{i=1}^{m} \mu_i p_i \qquad l=m+1,m+2,...m+n \tag{3.46}$$

Solving these for p_l yields:

$$p_l = \frac{\alpha_l \lambda}{\mu_l} p_a \qquad l=1,2,3...m \tag{3.47}$$

$$p_l = \frac{\beta_{l-m} \lambda}{\mu_l} p_a \qquad l=m+1,m+2,...m+n \tag{3.48}$$

The circulatory structure of the building block of Figure 3.14 can be decomposed into an aggregation of smaller cyclic flows which are associated with smaller building blocks. For instance, $\alpha_i \lambda$, the flux from state a to state i, where $1 \leq i \leq m$, can be decomposed as $\sum_{j=1}^{n} \alpha_i \beta_j \lambda$ in which $\alpha_i \beta_j \lambda$ is the portion of flow which corresponds to departures from the ith CPU destined for the jth memory module. The flow corresponding to departures from memory module j can also be decomposed into sub-flows. These are proportional to $\alpha_1, \alpha_2, \cdots \alpha_m$.

The circulatory structure of the first building block of Figure 3.14 with its splits and joins of flux can thus be decomposed into an aggregation of simple cyclic flows embedded along smaller building blocks. Each such flow corresponds to a *specific* relay sequence originating from state a. One such cyclic flow/building block is shown in Figure 3.15. It illustrates a relay sequence with m=n=3 involving the 2nd CPU and the 1st memory module.

These cyclic flow building blocks can also be solved in isolation for the equilibrium probabilities. They are pasted together along edges to form the larger building block of Figure 3.14. They form the most basic decomposition possible of the overall circulatory pattern of the multiprocessor state transition diagram with this state description.

∇

Example: Consider now a different queueing system example. This is a FIFO queue processing two different classes of customers. The server is exponential with rate μ for both classes. The state consists of the class of the job in each queue position. The state transition diagram of this queueing system is shown in Figure 3.16. The numbers associated with each state represent the class of each job in each queueing system position. In

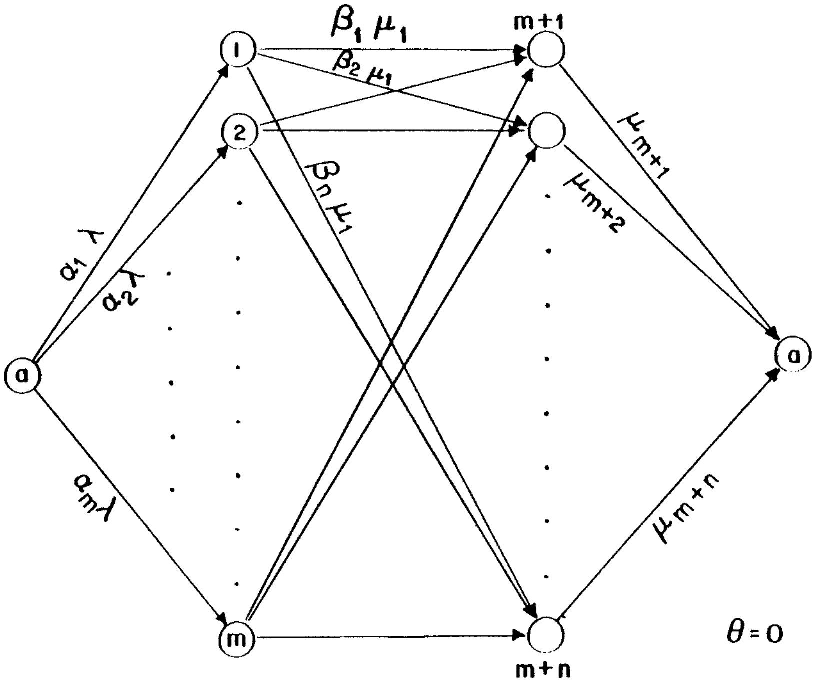


Fig. 3.14: First Multiprocessor Building Block

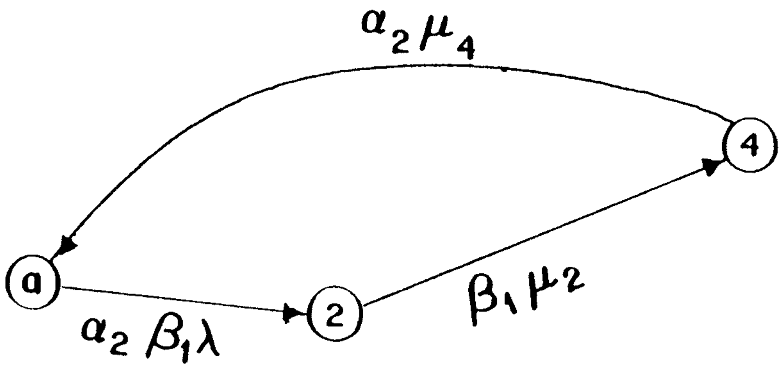


Fig. 3.15: Second Multiprocessor Building Block

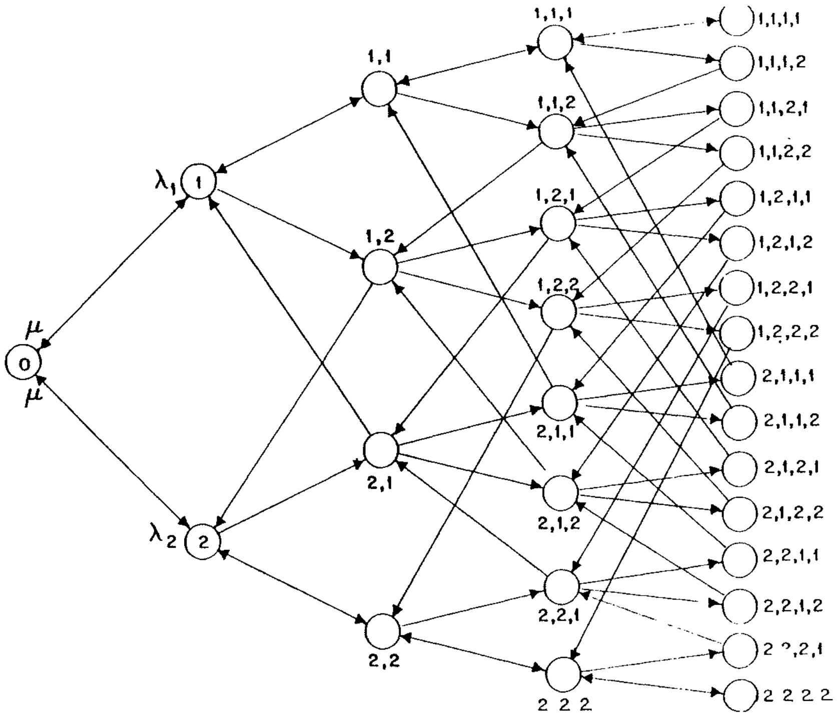


Fig. 3.16: FIFO Single Queue State Transition Diagram

the diagram's notation they are served from left to right.

This is a product form network. There are cyclic flows in the state transition diagram. In terms of queueing system events these correspond to a shift register like behavior:

Definition: A "shift register sequence" for a FIFO multiclass product form queue where the state involves the customer class at each queue position is a sequence of successive queueing events corresponding to a class i departure followed by a class i arrival followed by a class j departure (j may =i) followed by a class j arrival, and so on, terminating when the queue returns to the original state. This sequence of events generates the state transition diagram building blocks.

For instance the state transition sequence:

$$121 \rightarrow 21 \rightarrow 211 \rightarrow 11 \rightarrow 112 \rightarrow 12 \rightarrow 121$$

corresponds to a cyclic flow of length six. The state transition diagram of this queueing system consists of cyclic flows of different lengths. There is a group of local balance equations for each cyclic flow which can be solved without regard to the rest of the state transition diagram.

Let stage i=1,2... in the diagram consist of the transitions between states with i and i-1 total customers in those states. The cyclic flows exist in a single stage. Moreover, there are cyclic flows of different lengths within the same stage (Table 3.2). This is because the shift register like behavior may return the system to a starting state with a number of shifts less than the queue length (consider 1212).

Table 3.2: Number of Cyclic Flows

	Stage					
Circulation Length	1	2	3	4	5	6
2	2	2	2	2	2	2
4	-	1	-	1	-	1
6	-	-	2	-	-	2
8	-	-	-	3	-	-
10	-	-	-	-	6	-
12	-	-	-	-	-	9
Total	2	3	4	6	8	14

If n_1 is the number of class one customers in the queueing system and n_2 is the number of class two customers in the queueing system then:

$$p(n_1,n_2) = \frac{\lambda_1^{n_1}\lambda_2^{n_2}}{\mu^{n_1+n_2}} p(0,0)$$

Note that in having the state of the system indicate the queueing order of the two classes, one has a more detailed decomposition of the state transition diagram then that when the state indicates only the total number of each class at a queue. The cyclic flows in the state transition diagram of Figure 3.16 are a decomposition of isolated circulations in the diagram where state does not indicate queueing order.

For networks of such FIFO queues the basic geometric building block is also generated by shift register like sequences. As an example, consider the cyclic two queue system of Figure 3.17. The state transition diagram is shown in Figure 3.18 for four customers. Here μ_1^{II}, for instance, indicates the service rate of the 1st queue for class II customers. To obtain a product form solution, service rates must be class independent. Then

$$p(n_1,n_2) = \left(\frac{\mu_2}{\mu_1}\right)^{n_1+n_2} p(0,0)$$

where n_1 and n_2 are the number of class I and class II customers in the upper queue, respectively.

Cyclic flows exist between adjacent columns of states in the diagram of Fig. 3.18. One such cyclic flow corresponds to the state transition sequence:

$$1{,}122 \rightarrow 11{,}22 \rightarrow 1{,}221 \rightarrow 12{,}21 \rightarrow 2{,}211 \rightarrow 22{,}11 \rightarrow 2{,}112 \rightarrow 21{,}12 \rightarrow 1{,}122$$

where the state of the upper queue appears at left.

Note that the sequence starts with a class 1 customer leaving the lower queue followed by a class 1 departure from the upper queue. While this is reminiscent of a relay sequence, one has not returned to the original state since the customer order in both queues is changed. The sequence thus continues cycling until it returns to the original state. It is as if a shift register sequence, in the context of a queueing network, is comprised of successive relay sequences. Let's make precise the sequence of events that generate the state transition diagram building blocks.

Definition: For a closed FIFO product form network in which the state indicates the class of customer at each queue position a "shift register sequence" is a series of successive queueing events beginning with a customer leaving queue i to enter queue j, followed by a departure from queue j which enters queue k... followed by an arrival back into queue i. If the

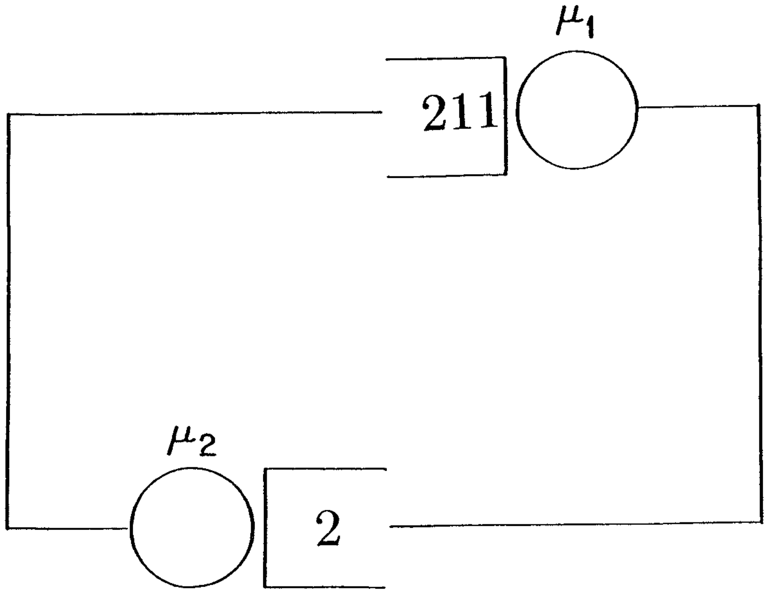


Fig. 3.17: Closed FIFO Network

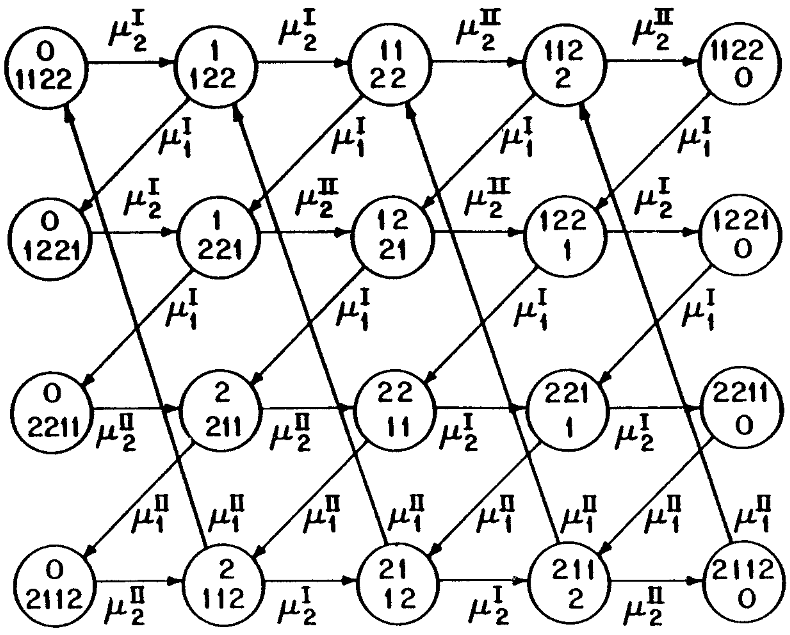


Fig. 3.18: Closed FIFO Network State Transition Diagram

network is not now in the original state, the previous series of events repeats until the network returns to the original state. Again, each of the queues is distinct and they form a closed cyclic path through the queueing network.

Definition: For an open FIFO network in which the state indicates the class of customer at each queue position a "shift register sequence" is a series of successive events beginning with an arrival into queue i from outside of the network followed by a departure from queue i to enter queue j, followed by a departure from queue j which enters queue k... followed by a departure from the network. If the network is not now in the original state, the previous series of events repeats until the network, returns to the original state. In the sequence, network departures of one class are followed by network arrivals of the same class. Each of the queues is distinct.

Theorem: For a FIFO product form queueing network where the state is described by the class of each customer in each queue position, and for which there is a consistent (see below) set of local balance equations, a closed path of N queues corresponds to a cyclic flow of length M ($M \leq N$) multiplied by the total number of customers in the path. An open path of N queues corresponds to a cyclic flow of length M ($M \leq N+1$) multiplied by the total number of customers in the path.

Proof: The cyclic flow of greatest length occurs when each customer must be shifted completely around the cyclic path of queues (N for a closed path and N+1 for an open path with returns through a virtual queue). For certain states customers may not have to be shifted around the entire path to bring the system back to the original state. This accounts for the inequalities in the Theorem.

∇

For networks of LCFS queues in which the state indicates the class of customer at each queue position the basic geometric building block is generated by a relay type sequence. The difference is that, because of the LCFS discipline, the job departing the first queue in a closed path is the *same* job that eventually returns. For an open path a job of the same class returns.

3.3.4 The Consistency Condition

Recall that the approach adopted in the previous sections calls for decomposing the state transition diagram into its building blocks and for solving the associated local balance equations. In the two dimensional case, we have noted the existence of distinct paths connecting some arbitrary states with the reference state. This property is not observed in the one dimensional case since the state transition diagram is a tree and hence a unique path links any node to the reference solution.

Thus the problem of consistency of the set of local balance equations arises as a result of the existence of distinct paths from an arbitrary state to the reference state. If, by taking alternate paths, the equilibrium probabilities for the same state are different, the set of local balance equations is not consistent.

As in the lower dimensional case the set of global balance equations must be decomposed into a set of local balance equations. Such a decomposition is, however, not unique. In addition it is not clear whether the resulting set of local balance equations is consistent. If the local balance equations are consistent, however, they are equivalent with the global balance equations. This is because the latter has a unique solution.

To derive the necessary and sufficient conditions for the consistency of a set of local balance equations, we define the following *consistency graph* [LAZA 84C]:

Definition: A consistency graph is an oriented graph, topologically equivalent with the original state transition diagram. In addition, it has the property that the probability at each node is equal to the probability of any adjacent node multiplied with the value of the associated edge connecting the two nodes.

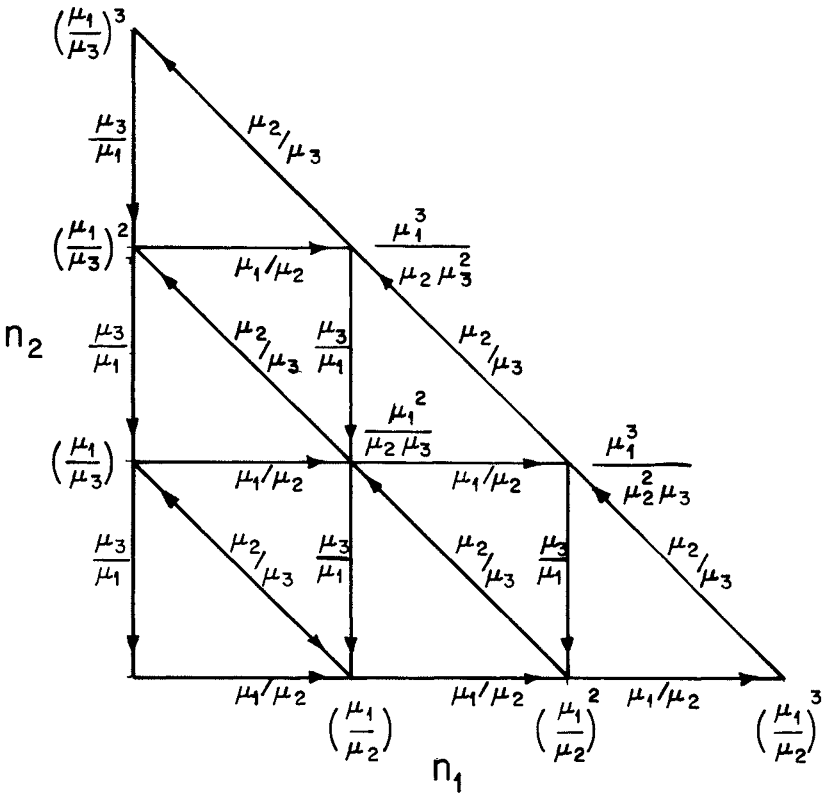
The class of consistency graphs of interest to us derives the algebraic values associated with the arcs directly from the set of local balance equations. Thus, the consistency graph can be seen as an "easy to read" graphical representation of the *proposed solution* for the local balance equations.

Example: Consider the state transition diagram of Figure 3.4 for the three queue cyclic Markovian queueing network of Figure 3.3. The consistency graph for this system appears in Figure 3.19. It is the consistency graph for what is, at this point, a *proposed* solution and associated local balance equations. For instance, the relationship between p_{21} and p_{11} is:

$$p_{21} = \frac{\mu_1}{\mu_2} p_{11}$$

Now choose any path (open or closed) on the consistency graph. Associated with it is an algebraic value called the *product* of the edges or simply the product. For example, with the path from state (0,0) to (0,1) to (1,1) to (1,2) is associated the product:

$$\frac{\mu_1}{\mu_3} \frac{\mu_1}{\mu_2} \frac{\mu_1}{\mu_3}$$



$\left(\frac{\mu_1}{\mu_3}\right)^3$
μ_3/μ_1
μ_2/μ_3
$\left(\frac{\mu_1}{\mu_3}\right)^2$
μ_1/μ_2
$\frac{\mu_1^3}{\mu_2\mu_3^2}$
n_2
$\frac{\mu_1^2}{\mu_2\mu_3}$
$\left(\frac{\mu_1}{\mu_3}\right)$
$\frac{\mu_1^3}{\mu_2^2\mu_3}$
$\left(\frac{\mu_1}{\mu_2}\right)$
$\left(\frac{\mu_1}{\mu_2}\right)^2$
$\left(\frac{\mu_1}{\mu_2}\right)^3$
n_1

Fig. 3.19: Consistency Graph for Fig. 3.4.

∇

Note that reversing the orientation of an arc of the consistency graph with value a results in an arc with value a^{-1}. We can now state the following consistency theorem [LAZA 84c]:

Theorem: A system of local balance equations is consistent if and only if any closed path of the consistency graph has the product equal to one.

Proof: Assume that there is a closed path in the consistency graph such that its product is not equal to one. Therefore, for at least one node belonging to this path, there are two distinct paths leading to the reference node which do not have the same product. Thus two different values for the probability associated with such a node can be obtained. Hence, the local balance equations are not consistent.

The sufficiency part of the theorem can be shown by direct computation. The product of an arbitrary path connecting node $(n_1,n_2,n_3, \cdots n_m)$ with the reference node $(0,0,0, \cdots 0)$ is an algebraic invariant. This is because, by assumption, the product of any closed path is equal to one and reversing the orientation of any edge with value a results in an edge with value a^{-1}. The probability at node $(n_1,n_2,n_3, \cdots n_m)$ is, therefore, equal to the product of the path connecting this node with the reference node times the probability of the reference node.

∇

Hence to determine the consistency of the set of local balance equations, the product of any closed path of the consistency graph has to be computed and compared with the value one. This will be referred to as the ***consistency condition.*** Naturally, the consistency condition is redundant for some closed paths. The minimum set of products needed to verify the consistency condition is investigated from a topological point of view in [LAZA 84c].

It is easy to verify for Figure 3.19 that the product of any closed path is one and thus there is a consistent set of local balance equations, a building block decomposition and a product form solution.

3.4 Recursive Solution of Non-Product Form Networks

3.4.1 Introduction

In terms of applications, there are many important and useful non-product form models. In general, these do not have closed form solutions for their equilibrium probabilities. However quite a few models are amenable to setting up a series of recursive equations to determine their equilibrium probabilities.

The idea of using recursions was first discussed in a systematic way by Herzog, Woo and Chandy [HERZ][SAUE 81]. To date such recursions have usually been developed for relatively simple systems with one or two queues. However there has been some work on algorithmically determining such recursions [PARE] [ROBE 89].

In what follows two classes of non-product form models for which recursive, or at least decomposed, solutions are possible will be presented.

In [LAZA 87] a class of non-product form networks is described whose state transition diagrams can be shown to be equivalent to a lattice tree of simplexes. In this "flow redirection" method the state transition diagram geometry is manipulated by equivalence transformations. Sequential decomposition refers to the related process of solving one subset of states at a time for the equilibrium probabilities:

Definition: A solvable subset of queueing network states is a subset of states for which the equilibrium probabilities can be determined without regard to the equilibrium probabilities of the remaining unknown states. Here probabilities are determined with respect to a reference probability.

Note that the direct solution of linear equations takes time proportional to the cube of the number of equations. If P states can be solved as S subsets then the computational effect is proportional to $(P/S)^3 *S$ rather than P^3.

Two types of structure which allow the state transition diagram to be decomposed into solvable subsets will now be presented. The first type of structure [ROBE 89], is illustrated in Fig. 3.20. Here each circular subset represents a state or a group of states. For the ith subset the rule is that there must be only one state external to the subset, with *unknown* probability, from which a transition(s) entering the subset originates. The subsets are solved sequentially, starting from the first subset to the second and so on. There is no restriction on the number of transitions which may leave the ith subset for destinations in the j=i+1,i+2,... subsets. This type of structure will be referred to as type A structure.

The second type of structure is illustrated in Fig. 3.21. Here the first subset consists of a single state. The remaining subsets each consist of a state or a group of states. These are arranged in a tree type of configuration with the flow between subsets from the top of the diagram to the bottom and a return flow from the bottom level back to the top level. The subsets may be solved from the top to the bottom. Transitions may traverse several levels (e.g. dotted line in the figure) as long as the direction of flow is downward. This type of structure will be referred to as Type B structure.

It is possible to write recursive equations for the equilibrium probabilities when each subset in the Type A or Type B structure consists of a

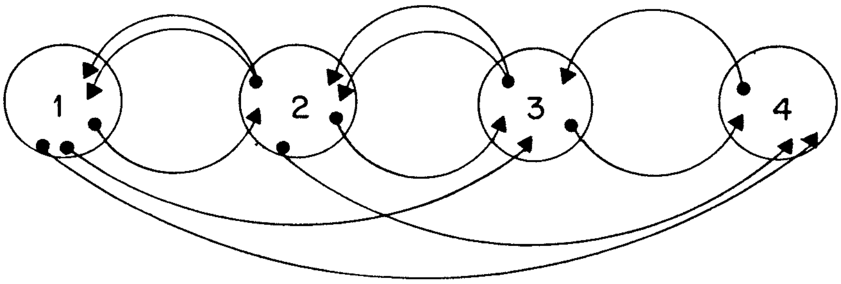


Fig. 3.20: Type A Structure
Copyright 1990 IEEE

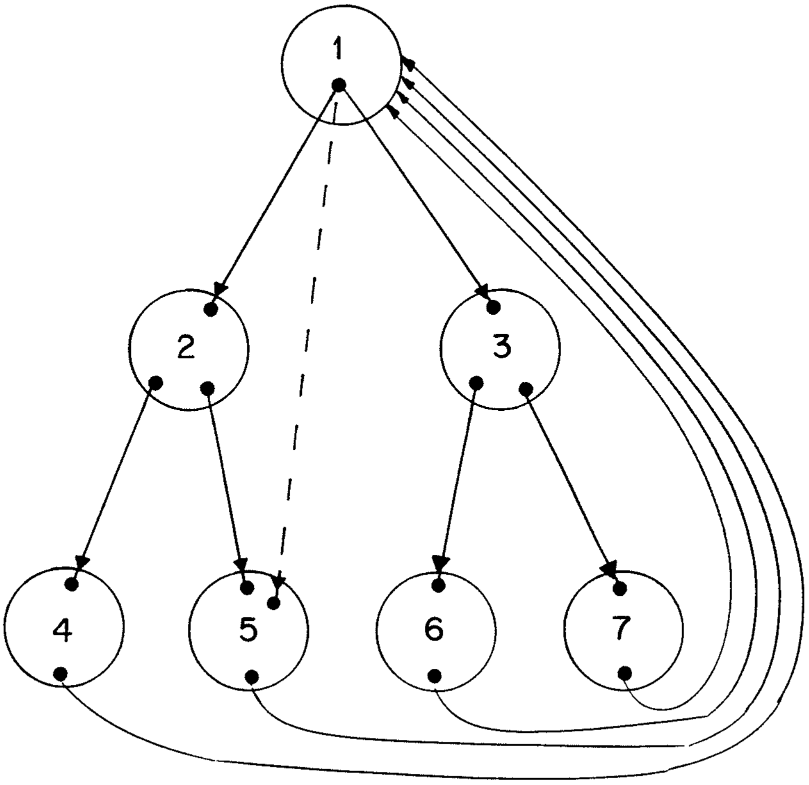


Fig. 3.21: Type B Structure
Copyright 1990 IEEE

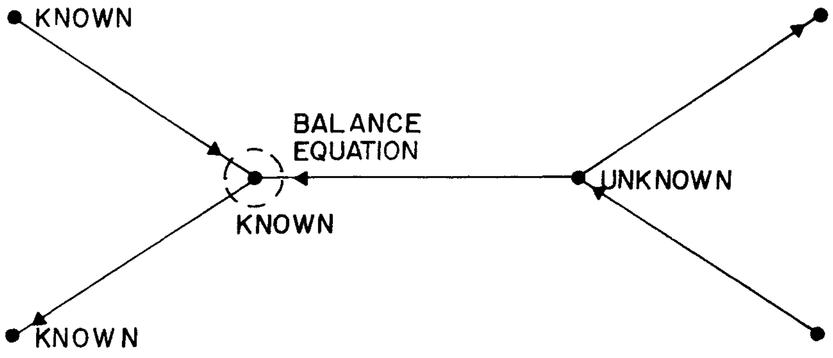


Fig. 3.22: Type A Structure: One State Per Subset
Copyright 1989 IEEE

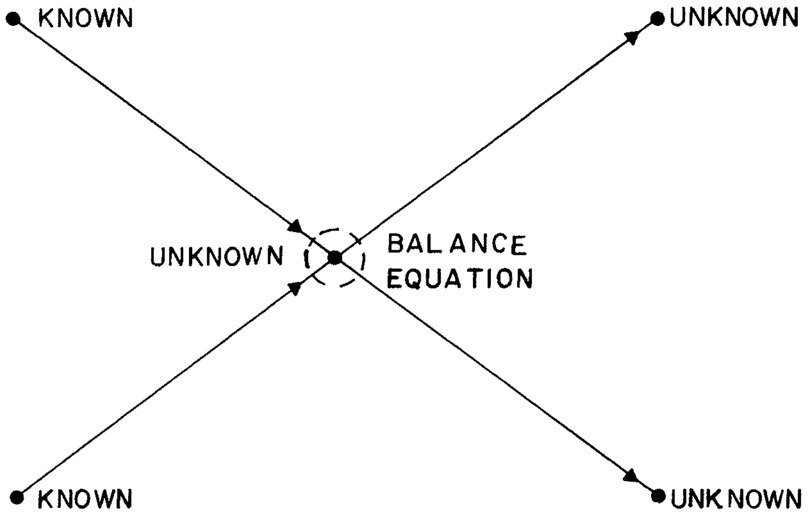


Fig. 3.23: Type B Structure: One State Per Subset
Copyright 1989 IEEE

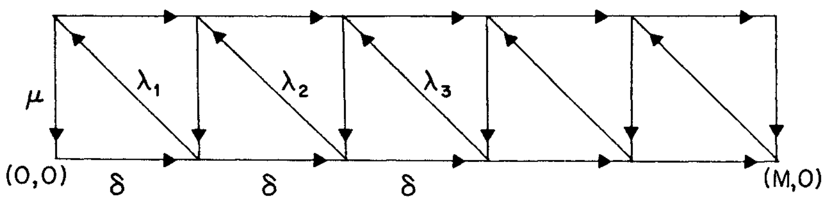


Fig. 3.24: Tandem Queues: Buffer Size 1 for 2nd Queue
Copyright 1990 IEEE

single state. In order to understand how recursive solutions are produced it is helpful to take a closer look at the Type A and B structures when one state at a time is decomposed. Figure 3.22 illustrates how a solution is arrived at for the Type A structure. A balance equation is solved for a state whose probability is already known. The state must have only one incoming transition from another state with unknown probability. When the balance equation is solved this other state probability becomes known. Often, this other state provides the next balance equation for solution.

Figure 3.23 illustrates how a solution is arrived at for the Type B structure. A balance equation is solved for a state with unknown probability where all incoming transitions originate from states with known probabilities.

A number of examples of the recursive solution of practical queueing networks follows. They are from [WANG].

3.4.2 Recursive Examples

A. Two Blocking Tandem Queues

This first example concerns two queues in tandem with finite buffers. The first queue has a buffer size of M and the second queue has a buffer size of one. Here δ is the system arrival rate and λ_i and μ are the service rates of the first and second queues, respectively. The arrival rate of the first queue is state dependent. As with all examples in this correspondence, the arrival process is Poisson and service rates follow an exponential distribution. Recursive equations for this case appear below. The state transition diagram is illustrated in Fig. 3.24.

Initialize:

$$p(0,0)=1.0 \tag{3.49}$$

$$p(0,1)=(\delta/\mu)p(0,0)$$

Iterate: i=1,...,M-1

$$p(i,0)=(\delta/\lambda_i)[p(i-1,0)+p(i-1,1)]$$

$$p(i,1)=(\delta/\mu)[p(i-1,1)+p(i,0)]$$

Terminate (Right Boundary):

$$p(M,0)=(\delta/\lambda_M)[p(M-1,0)+p(M-1,1)]$$

$$p(M,1)=(\delta/\mu)p(M-1,1)$$

These equations have a simple form. The structure is Type A. It should be noted that the equations are not unique. It is also possible to take state dependent arrival rates into account for this and the remaining examples. Recursive equations for the case when the second queue's buffer is of size two appears below. Here μ_i is the state dependent service rate of the second queue. The state transition diagram is illustrated in Fig. 3.25.

Initialize:

$$\begin{gathered} p(-1,2)=0.0 \\ p(0,0)=1.0 \\ p(0,1)=(\delta/\mu_1)p(0,0) \end{gathered} \tag{3.50}$$

Iterate: i=1,2,...,M-1

$$p(i,1)=\delta f(i)/g(i)$$

where

$$\begin{gathered} f(i)=(\delta+\lambda_i)(\delta+\mu_2)[p(i-1,0)+p(i-1,1)] \\ +\delta(\delta+\lambda_i)p(i-2,2)-\lambda_i(\delta+\mu_2)p(i-1,0) \\ g(i)=\lambda_i[\mu_1(\delta+\mu_2)+(\delta+\lambda_i)(\delta+\mu_2)-\delta(\delta+\lambda_i)] \end{gathered}$$

$$\begin{gathered} p(i,0)=[1/(\delta+\lambda_i)][\delta p(i-1,0)+\mu_1 p(i,1)] \\ p(i-1,2)=[1/(\delta+\mu_2)][\lambda_i p(i,1)+\delta p(i-2,2)] \end{gathered}$$

Terminate (Right Boundary):

$$p(M,1)=\delta f'(M)/g'(M)$$

where

$$\begin{gathered} f'(M)=\lambda_M(\delta+\mu_2)p(M-1,1)+\delta\lambda_M p(M-2,2) \\ g'(M)=\lambda^2_M\mu_2+\lambda_M\mu_1(\delta+\mu_2) \end{gathered}$$

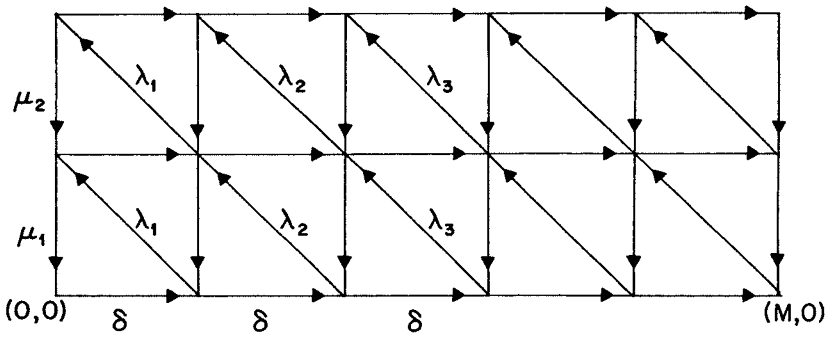


Fig. 3.25: Tandem Queues: Buffer Size 2 for 2nd Queue
Copyright 1990 IEEE

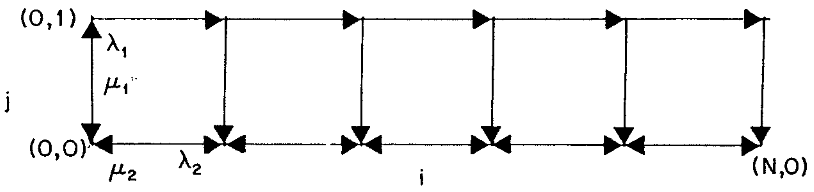


Fig. 3.26: ISDN State Transition Diagram
Copyright 1990 IEEE

$$p(M,0)=[1/(\lambda_M)][\delta p(M-1,0)+\mu_1 p(M,1)]$$
$$p(M-1,2)=[1/(\delta+\mu_2)][\lambda_M p(M,1)+\delta p(M-2,2)]$$
$$p(M,2)=(\delta/\mu_2)p(M-1,2)$$

Again, the structure is Type A. The first subset consists of (0,0). Then subsets consist of the states (i-1,1), (i-1,2) and (i,0) for i=1,2...M. At the right boundary the last subset consists of (M,1) and (M,2). It appears that these equations can not be extended to a larger buffer size for the second queue because of the resulting geometry of the state transition diagram. The case where the first queue's buffer is of size one or two and the second queue's buffer is of size M can be handled through a straightforward change of variables.

B. An ISDN Protocol

Consider a channel that can service either voice or data transmissions [SCHW 87]. The transmitter contains a buffer of size N for data packets. A voice call can seize the channel with rate λ_1 if no data packets are in the buffer. It is serviced at rate μ_1. If a voice transmission has seized the channel, then data packets can not be serviced but are buffered. Data packets arrive at rate λ_2 and are serviced at rate μ_2. The state transition diagram appears in Fig. 3.26. Interestingly the upper row of the diagram has the Type B structure and the lower row has the Type A structure. The recursive equations appear below. We note that the recursive expression for the row 1 appears in [SCHW 87].

Initialize:

$$p(0,0)=1.0 \tag{3.51}$$
$$p(0,1)=[\lambda_1/(\lambda_2+\mu_1)]p(0,0)$$

Iterate: i=1,...,N-1

$$p(i,0)=(\lambda_2/\mu_2)[p(i-1,0)+p(i-1,1)]$$
$$p(i,1)=[\lambda_2/(\lambda_2+\mu_1)]p(i-1,1)$$

Terminate (Right Boundary):

$$p(N,0)=(\lambda_2/\mu_2)[p(N-1,0)+p(N-1,1)]$$
$$p(N,1)=(\lambda_2/\mu_1)p(N-1,1)$$

C. A Window Flow Control Protocol

Consider the queueing system of Fig. 3.27 which models a window flow protocol [SCHW 87]. Packets are sent from the lower queue (source) through a channel (upper queue) to the destination (W box). The W box only releases packets when all W packets in the closed queueing network are in the W box. This models a reception acknowledgement of the entire group of packets. The state transition diagram appears in Fig. 3.28. The structure is type B. The recursive equations are:

Initialize:

$$p(0,0)=1.0 \tag{3.52}$$

Iterate: i=1,...,W-1

$$p(i,0)=[\lambda/(\lambda+\mu_i)]p(i-1,0)$$
$$p(W,0)=(\lambda/\mu_W)p(W-1,0)$$

Iterate: j=1,...,W-1

$$p(0,j)=(\mu_1/\lambda)p(1,j-1)$$

Iterate: i=1,...,W-j-1

$$p(i,j)=[(1/(\mu_i+\lambda)][\lambda p(i-1,j)+\mu_{i+1}p(i+1,j-1)]$$
$$p(W-j,j)=$$
$$(1/\mu_{W-j})[\lambda p(W-j-1,j)+\mu_{W-j+1}p(W-j+1,j-1)]$$

3.4.3 Numerical Implementation of The Equations

Scaling problems are possible with the use of some of these equations. As an example, consider equation (3.49). The first queue may be viewed approximately as an M/M/1 queue. The equilibrium probability at the ith state is equal to λ/μ times the equilibrium probability at the (i-1)th state. Here, λ is the arrival rate, and μ is service rate. Thus, for instance, the probability that there are one hundred packets in the buffer is $(\lambda/\mu)^{100}$ times the probability that the buffer is empty. This can naturally lead to numerical scaling and underflow problems.

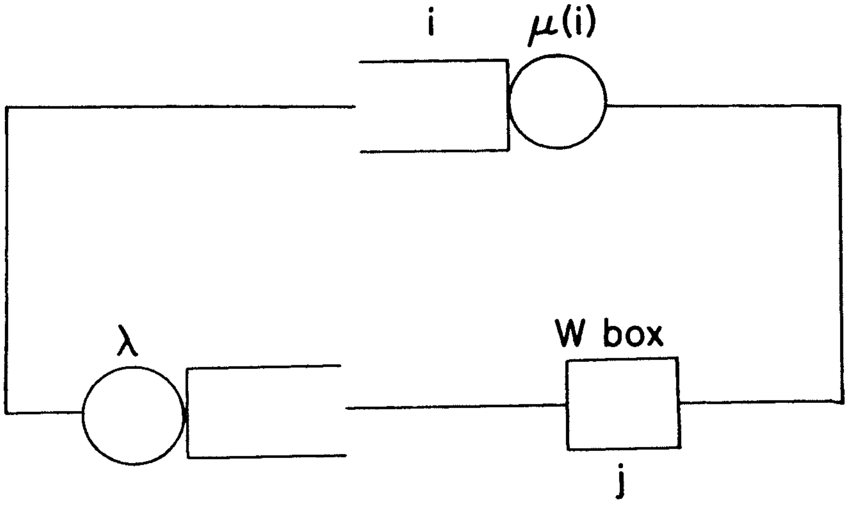


Fig. 3.27: Window Flow Control Model
Copyright 1990 IEEE

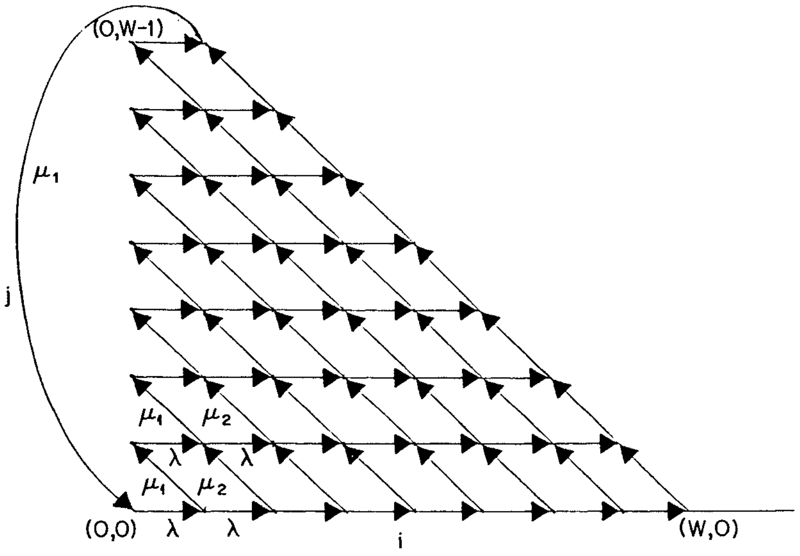


Fig. 3.28: State Transition Diagram of Fig. 3.27
Copyright 1990 IEEE

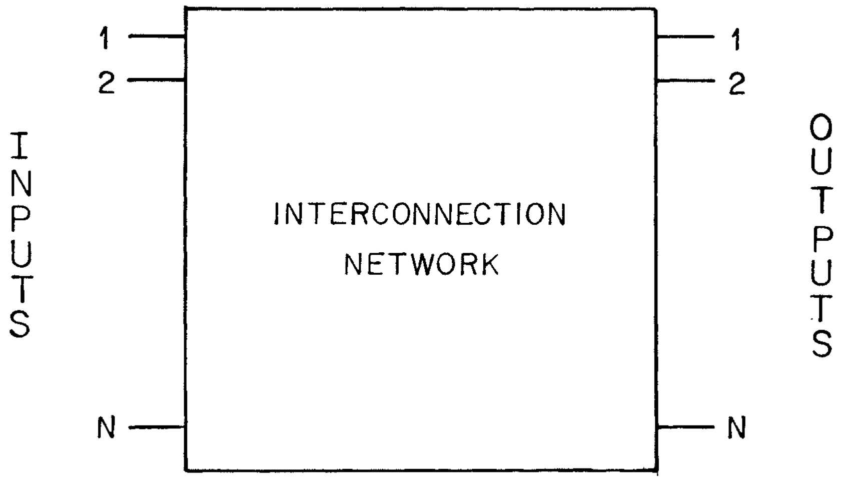


Fig. 3.29: Interconnection Network

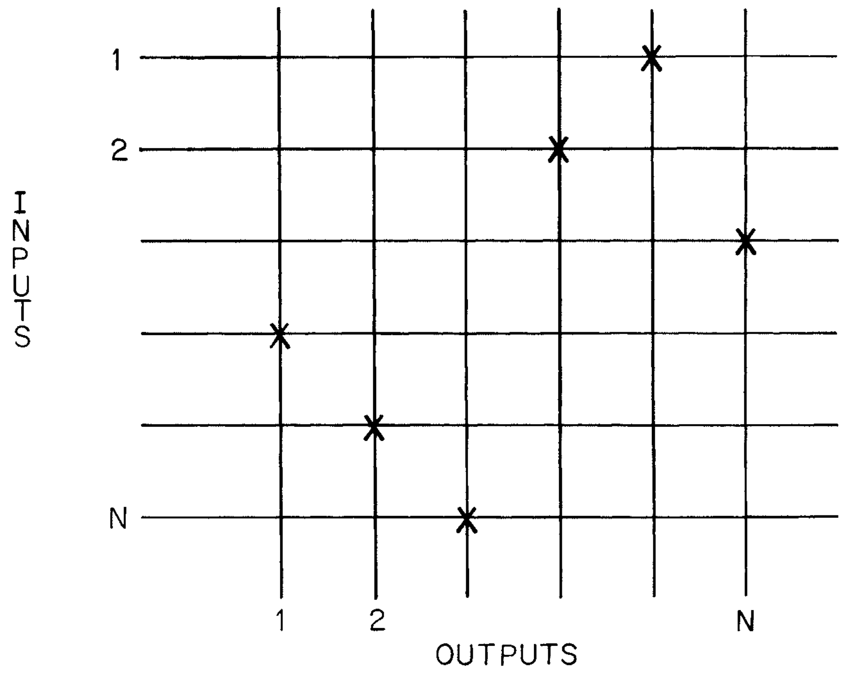


Fig. 3.30: NxN Crossbar Switch

One form of automatic scaling is possible in this if it is clear which section of the state transition diagram corresponds to light traffic and which corresponds to heavy traffic. This is the case for the linear state transition diagram associated with equation (3.49). The equilibrium probabilities are then calculated in this roughly monotonic order. When the size of equilibrium probability begins to approach the computer register size, it is scaled downward by a factor which brings the equilibrium probability down to the lower limit of machine register size.

3.5 Case Study I: Queueing on a Space Division Packet Switch

3.5.1 Introduction

As long as there have been telephones, and more recently with multiprocessor systems, a basic problem has been the interconnection of N inputs and N outputs. That is, how does one design a box, as in Figure 3.29, that simultaneously provides connection paths from the multiple inputs to multiple outputs. In traditional telephony the paths are actual circuits through the interconnection network (or simply the "switch") that are in use for the duration of a call. In the more modern packet switching technology, packets of bits pass through the interconnection network. A packet is a finite series of digital bits, some of which represent control information such as the destination address and some of which represent the actual information being transmitted.

A simple way to design an interconnection network is with a cross-bar architecture. In an NxN crossbar switch, as in Figure 3.30, inputs are connected to say, horizontal wires (buses) and outputs to vertical buses. There is a switch, which can be independently closed or opened, everywhere a horizontal bus crosses a vertical bus. These are the interconnection crosspoints. An "X" in the figure indicates a closed switch, creating a path from input to output. Such a cross-bar switch is said to be a "space-division" switch in that the distinct paths through it are spatially separated.

There are a wide variety of interconnection architectures available [WU]. Some are blocking networks. These reduce the number of crosspoints at the expense of blocking. That is, not every input can be connected to a different output at the same time. An input that can't find a path to an output is said to be blocked. Blocking networks can be advantageous as they reduce the number of crosspoints, a traditional measure of switch complexity, and because it may be that traffic patterns are such that only a small number of inputs will be active at one time.

In what follows though, we will consider the case of a non-blocking packet switch, such as the crossbar, with queues at either the inputs