Imperial College of Science and Technology

(University of London)

Department of Computing and Control

THE EVALUATION OF COMPUTER PERFORMANCE BY MEANS OF STATE-DEPENDENT QUEUEING NETWORK MODELS

by

A.W. Mecklenburg

A Thesis submitted for the Degree of Doctor of Philosophy

July, 1978

TITLE:

'THE EVALUATION OF COMPUTER PERFORMANCE BY MEANS OF STATE-DEPENDENT QUEUING NETWORK MODELS'

AUTHOR: A. W. MECKLENBURG

ABSTRACT

Computing system performance is influenced not only by the service capacity of processing resources, but also by capacity limitations of storage and data resources. These effects, although more subtle, are no less profound in their implications to system management and system architecture.

Herein, computing systems are abstractly interpreted as a collection of active (processor) and passive (storage media, data objects) resources. Such resources, and the processes they serve, can be represented analytically by queuing network models.

Recent development in queuing theory, particularly separable queuing networks, have greatly increased the usefulness of queuing models for the evaluation of system performance. Unfortunately, these methods do not generally allow solutions to constrained networks. Such constraints are imposed by the presence of infeasible states which arise naturally due to finite processing and occupancy limitations of the resources they represent.

This work investigates separable network solutions to constrained networks involving two phenomena: skipping and blocking. A customer, on its journey through the network, either (1) skips the next node or (2) blocks the current node, if the next transition would lead to an infeasibility. Models of these phenomena, considering local and joint state dependent service functions and state dependent routings, lead to the conclusion that the separable results extend simply to networks with skipping; but do not, in general, admit solutions to the blocking problem. However a reasonably general class of networks (e.g. the central server model) do have product form solutions and are simply analysed in the context of separable networks.

ACKNOWLEDGEMENTS

I gratefully acknowledge the support and cooperation of my friends and associates in the Systems Methodology Group.

I would like to thank my supervisor, Professor M.M. Lehman, for his continuous encouragement and, in particular, for his helpful comments which greatly improved the presentation of this thesis.

Special thanks are due Peter Harrison for the numerous productive discussions during the course of this research and especially for his meticulous review of this manuscript.

I am also indebted to Sarah Billows for her expert and sympathetic advise and to Lesley Myers and Doris Abeysekura for their aid in typing.

Finally, I gratefully acknowledge the IBM Corporation for not only providing the opportunity, but also for their encouragement and generous financial support.

To

Paul and Heidi

...for their gentle and innocent encouragement

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

I N T R O D U C T I O N

Fifteen years have passed since analysts have actively employed queuing network models for the evaluation of computing systems. This activity arose naturally out of the obvious congestion analogy; requests for the time-space capacity of finite resources result in congestion, i.e.,performance degration. Computing jobs and processors correspond, conveniently, to queuing customers and servers, respectively. Insofar as processing resources are assumed to be *independent* of other resources, these models are not only convenient but remarkably accurate.

Unfortunately, in contemporary systems, the complex interaction of resources often weakens the independence assumpt- ion. Storage and data objects, considered as system resources, tend to limit the performance of other resources and, therefore, the performance of the system. Such limitations impose logical constraints on the queuing network and its relevant state space.

Herein, it is asserted that resources are *not* necessarily independent. In particular, physical limitations of storage and data objects, called passive resources, impose operational limitations on processors. These limitations are manifest in queuing networks as *infeasible* sub-states in the network state space. Hence the service rate, or routings in the network, necessarily prohibit transitions to infeasible states. Networks with these constraints are called *state- dependent*, since services and/or routings are functionally dependent on the state of the network.

In this thesis, a special type of state dependent network, a *blocking* network , is studied and modelled. Briefly, blocking is a condition whereby one node blocks the service of another. Such conditions are often encountered in system evaluation due to capacity limitations of system resources. Simple blocking models are presented which though not fully general, yield theoretically interesting and potentially useful solutions.

It is our contention that the competition for passive resources such as data and storage objects have, and will continue to have, a significant effect on the performance of the system. These passive resources, being of finite capacity, will block or inhibit processor service and consequently limit system performance. The abstraction of these phenomena, construction of corresponding queuing models and development of compact solutions are the objectives of this work.

1.1 Thesis Organisation

This dissertation is conventionally organised. The first two chapters are mostly bibliographical- or definitional- in order to establish the context for the analysis and results presented in subsequent chapters.

Chapter 2 reviews and reaffirms the object Performance Evaluation. System resource and workload definitions are extended for the consideration of data and storage objects we call this collection of definitions the performance model.

Chapter 3 reviews Queuing Network theory, emphasising recent developments in solution methods. Particularly, network models of system performance are discussed. Network definitions are extended to be compatible with the performance model; this yields a queuing network model (Markovian) which is notably characterised by constraints on its state space.

The interpretation of state-space constraints and the disposition of such constraints by the use of state dependent parameters follow in Chapter 4. An interesting queuing network construct, the multiple server, is derived which has both theoretical interest and practical application in system models.

The most notable results are mainly, but not exclusively, contained in Chapter 5 wherein the models of blocking and skipping are presented.

The final chapter is a reiteration of the thesis; it summarises the assertions and results, comments on unsuccessful investigations, and points to promising areas of future research on constrained networks.

CHAPTER 2

THE PERFORMANCE OF COMPUTING SYSTEMS

2.0 Introduction

In the last few years, the performance evaluation of computing systems has become an increasingly fertile, yet complex area of investigation. Conference papers journal articles, and research projects are proliferating at an increasing rate; and not without good reason. For many of the management objectives, design criteria and theoretical investigations in computing science are fundamentally linked to performance issues. Multiprogramming, multiprocessing, scheduling, paging, spooling, access methods, sorting, etc. are essentially constructs for improving performance. Indeed the distinction between function and performance seems very vague— as computing systems become more complex, so do their evaluations.

And yet even with its undeniable importance, computer performance has defied formal definition and resisted rigorous scientific treatment; it remains a lively art without theory. Because the measures of performance commonly used today lack precision, clarity, uniformity and generality, they are as diversified as are systems, applications and investigators.

This chapter contains three sections: the first is primarily bibliographical and traces computer performance evaluation (CPE) in relation to the historical development of computing systems. Then follows a brief commentary on CPE, interpreted in its most general form.

In the final section, as a prelude to quantitative analysis, a *performance model* is presented which abstracts computing system components and the use of those components. This model serves as the foundation for the construction of subsequent *Queuing network* models; its significance is that it, in principle, extends the domain of applicability of queuing models to storage and data resources - treating these as finite and performance limiting objects of the system.

2.1 Computer Performance Evaluation: Perspectives

Without presenting a thorough history of Computing Performance Evaluation (CPE), we note its general development with respect to the succession of computer systems generations. This classification, though informal (see for example ROSE 76a) provides a useful chronology of the development of CPE.

2.1.1 The First Generation (1951-1958)

Although many kinds of calculating machines and analytic engines existed prior to 1946, the computer as we know it today had its genesis in the concept of a stored program in the early digital machines of the mid-forties. The first commercial computer, the UNIVAC I, was delivered in 1951.

One could argue that the whole motivation of the continuing development of computing machines was one of performance; i.e., to *perform* tedious calculations with more ease, speed, reliability and accuracy. However early designers of such systems were more concerned with function (that the systems actually worked) then with their performance; the latter was merely the speed rating of the available component technology and the emphasis was on *automatic* computation.

One of the first papers on CPE produced the legendary Grosch's Law* [GROS 53] which perceptively (although facetiously)

---

*that the cost of a system was proportional to the square root of its 'performance'.

attempted to axiomise the relationship between performance of a system and its cost. But most reports were content with rating of components or analysis of instruction distributions HERM55 .

2.1.2 Second Generation (1959-1963)

Second generation systems were born of two phenomena: a remarkable advance in component technology (e.g., solid state circuits and magnetic core memories) and the consolidation of software systems (compilers, programming languages, and primitive operating systems).

Yet most performance interest was still on raw speed of devices. Since most systems were dedicated to sequential jobs, the whole was usually the sum of the parts. Merely by measuring the parts and summing, a reasonable comparison could be made between various computing systems. This procedure , known as *Benchmarking* [DOPP62 , GOSD62] , was a kind of electronic digital olympics whereby key data processing jobs raced against the clock on different machines. Fastest was still best.

2.1.3 Third Generation (1964-1968)

Although there remained the persistent technological advances (e.g. monolithic integrated circuits), the third generation of computing systems is mostly characterised by the broad implementation of multiprogramming, concurrent processing and time sharing operating systems.

This period may well be regarded as the birth of CPE. With the introduction of multiprogramming, concurrent processing and other shared resource architectures, performance evaluation became a non-trivial task. Consequently there was a rapid development in computer modelling and measuring techniques (for surveys see [BUCH69], [CALI67], [DRUM69]); queuing theory was rediscovered [SCHE65] and computers were put to self-analysis in the form of discrete simulation models [KATZ66,HOLL68].

2.1.4 Generation "3.5" (1969-present)

Although there was no change in computing systems during this period important enough to warrant a new generation designation, subtle architectural changes such as virtual storage, virtual machines, multi-layered storage hierarchies, and shared data bases demanded more sophisticated CPE techniques. Paging behaviour and storage hierarchy models [BELA68 , DENN69 , MATT70] were introduced to cope with these new complexities.

By this time CPE has its own journals (EDP Performance Review, SIGMET); a recent survey [EDPR77] referenced hundreds of articles in 60 CPE categories for the year 1976. Nevertheless CPE has mostly been an application vehicle for statistical, analytic and simulative modelling; there still exists no standard measure [JOHN70 , CONN76], or definitions of performance, or even what constitutes a computing system.

2.1.5 The Next Generation

If one accepts the trend towards distributed systems, special purpose mini-computers linked into networks, and notes the greater tendency towards the development of user oriented, turn-key systems (e.g., very high level languages and transaction oriented systems), then another level of complexity is demanded of CPE- the anticipation and satisfaction of computer users. The characterisation and prediction of this growth (often referred to as user *Workload*) constitutes a major research area in CPE (SCHW76 , CONN76).

In the absence of a systematic and predictable design theory and methodology for these complex systems, one must for sometime to come, rely on CPE to predict the expected performance (behaviour) of projected or installed systems so that users at least know what to expect and can plan accordingly [LEHM78] .

Despite the vigorous research and accumulated literature in CPE, there is still no formal theory. The need for such a theory has long been recognised [JOHN70, SEKI72] .
While there have been a few attempts [KOLE72, CONN76] , none have yet demonstrate their usefulness or generality.

2.2 A Macroscopic View of Performance Evaluation

One of the difficulties with CPE is that it is so pervasive; the performance of a system can be discussed at many levels with respect to three different viewpoints. First, there is the computing system which must be organised and managed effectively to provide an efficient use of its collective resources, satisfy user demand, and remain cost effective. Secondly there is the computing user on whose behalf the system exists. The user has an entirely different view of performance, being not so interested in resource efficiency, as he is in its functional capability, quality and cost of service. Finally there are the designers whose main interests are enhancing or extending capabilities, and providing greater efficiency at lower cost.

The physical level of service and its cost are the underlying issues. Thus performance has two apparent aspects: Its *physics* and its *economics*. By physics we mean the efficiency in which computing system resources do work in time and space; by economics we mean the supply and demand, production and consumption, of computing services viewed as a (service) commodity.

This study is only concerned with the physical interpretation, i.e., the rate at which computing systems do work, the delays they impose, and the consumption of the time/space facilities of the system resources.

Hence the interesting, but immensely complicated, questions concerning the relationship of quality, price, and value of computing services are set aside. Furthermore, we ignore the dynamic (market) behaviour between computing users and suppliers. These are, of course, very difficult concepts to quantify and remain a subject for future research.

2.3 A Descriptive Performance Model

The modelling procedure in this study abstracts the CPE problem in two stages. The first is a *qualitative* reduction which describes and defines the system. This we call the *Performance model*. The second abstraction uses this semantic model as a basis for some suitably chosen *quantitative* models- in our case, queuing network models. Evaluation of a real system is then a consequence of sequential interpretations of model results.

The Performance model consists of two types of objects: computing system resources and system processes.

2.3.1 Computing System Resources

A computing system is a collection of physical or logical resources which collaborate to satisfy the aggregate user demand. We define three types of computing system resources:

1. Processors- The operational resources that transform the values in storage and realise two mappings:

   (1) a physical mapping on the storage space and
   (2) a physical delay, i.e., a mapping in physical time.

   Examples are accessing mechanisms, controllers, channels, ALU's, clocks, card readers, tape drives, etc.

2. Storage Media- The physical repository on which data objects are stored. This includes magnetic, electronic, chemical and paper storage.

   Some examples are magnetic cores, bubbles, drums, disks tapes; electronic registers, buffers, latches, light sensitive films, visual displays, punched cards, etc.

3. Data Objects- Collections of data which form the set of values of the computing system. This includes system and user *programs*, system objects (e.g., directories, maps, control blocks, tables, etc.), shared data bases and unique user data.

2.3.2 Processes and Requests

Our computing system is a service system; through its resources, it provides service to demands made on behalf of its users. The elements which result in the consumption (utilisation) of resources are the *requests* which are collected into procedures called *processes*. A process is now defined as a set of requests:

$$P = \{R_1 , R_2 , \ldots\ldots R_n\}$$

Each request has two aspects: (1) a logical mapping (work required) which specifies the function to be executed* and (2) an *implementation* specifying a resource trajectory, a specific binding of resources which result in either an execution (hence request satisfaction) or a sub-creation of another process.

The logical part of a request describes <u>what</u> function (transformation on the data) is to be performed+ while the implementation specifies <u>how</u> the request(s) are given service: the servicing of the request is achieved by issuing a set of (sub)requests at a lower level. In turn, the servicing of these requests will give rise to other sets of requests, the process terminating when the physical level is attained, i.e., when the requests are physically executable primitives of the system.

*the logical role of a request defines a mapping from the set of values. Clearly the set of values depends on the nature of the objects to be processed. Usually they will be a set of algebraic values in the modulo arithmetic of the system. But this may be extended to include complex data structures or programs.

+it is worth noting that the logical part of a request can be extended to include the sequence constraints governing the order in which requests of a given process are serviced. This can have important performance consequences insofar as these constraints may hinder the servicing of a request. However, such constraints unnecessarily complicate analysis and are ignored in this study.

To summarise, the logical role of the requests is to define the mathematical mappings to be applied to the arguments of the process. The implementation role represents a hierarchy of (sub)requests at a lower level which are eventually serviced at the leaves of the tree by the executable primitives of the system (processor resources). Thus we interpret a request as specifying both the mathematical transformation and the sub-tree of interactions representing the actual servicing of the request. A simple example is provided in figure 2.1.

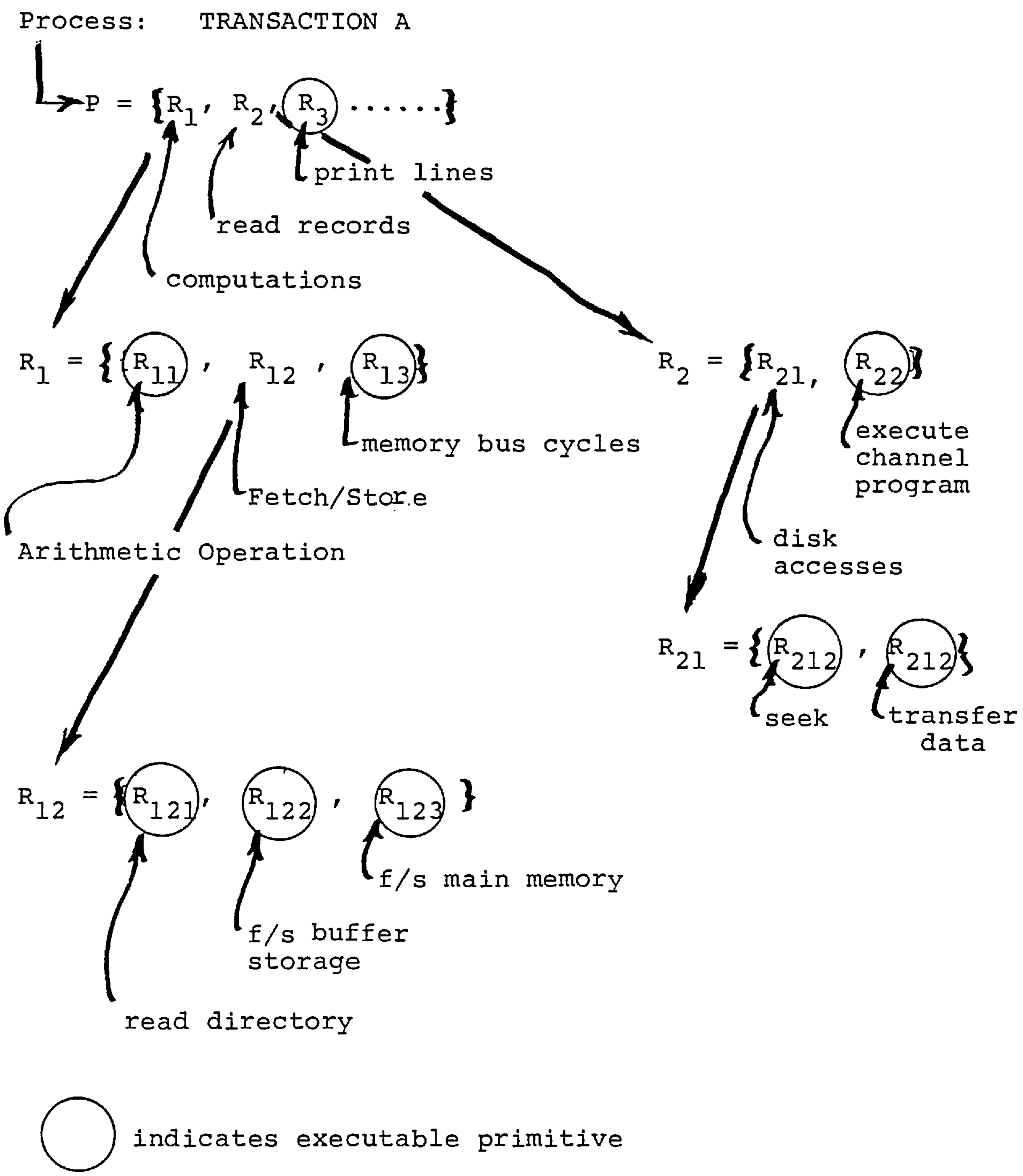


Figure 2.1 Example of Hierarchical structure of Processes and Requests

2.3.3 The process hierarchy as a workload specification

The specification of processes, i.e., the process/request hierarchy is a description of the demand functions (in the mathematical sense) imposed on the resources of the system. Since this specification is hierarchical, the parent process (the apex of the tree) can be regarded as a work specification to the collection of target resources; that is, the parent process is the unit of work at the computing system interface. Such processes are conventionally called *transactions* or *jobs* (is not a job a batch of unrequited transactions?). And the collection of transactions, executable on a particular computing system, is called the *workload* with respect to that computing system.

Thus the process specification can be viewed as the *workload specification* for a set of transactions at the system interface. Said another way the process specification (workload) is a function that maps user demand(applications) into resource requests.

2.4 Concluding remarks

The collection of resources and process/requests is merely a descriptive model of the real objects and events of a computing system. As such, it is intentionally limited in scope. We are interested in evaluating not what a system does but *how well* it performs. It is presumed that the system computes correctly, that the resources are available, and that the transformations are proper. Hence the simple performance model is adequate if we restrict our attention to the less spectacular issues of "how well" and particularly, "how much".

The notion of "how much" deserves some elaboration. The enumeration and measurement of system performance has mostly been an imprecise science [JOHN71]. The metrics of CPE are ambiguous and there are few universally recognised and accepted measures.

In this study we consider two performance metrics: *thruput* and *delay*. Thruput is [defined to be] the *rate* at which the system renders service to the consumer. Hence thruput can be interpreted at the component, device, sub-system and system level for any request (or set of requests/processes). This is also true of delay.

Delay is the distribution of satisfaction time of a request (or group of requests, e.g., a process or transaction). This metric nominally includes service, waiting, reservicing and blocking times[1].

Thruput and delay are necessary performance metrics; but they are rarely sufficient. Utilisation, contention, and effective capacity are familiar terms in the CPE vocabulary[2]. We call these metrics performance *indicators*. Rather than measure performance directly, they indicate potential problems (or opportunities).

In the queuing network models which follow, we consider thruput and delay at the dominant metrics. Accompanying their derivation, other indicators also appear: utilisations, queue lengths, capacities and state probabilities, to mention a few.

In this work queuing network models are the quantitative forms of the performance model. That is queuing (network) theory, deposited on the descriptive model substrate, is the medium for quantitative evaluation of computer system performance. We now attend to these models.

1. may also include 'down-time'; but this kind of performance model is usually called a "reliability" model.

2. see SVOB76 for an extensive list

CHAPTER 3

QUEUES, NETWORKS AND MODEL CORRESPONDENCE

3.0 Introduction

In the preceeding chapter, a model was postulated which viewed computer systems usage (production) in the context of two sets; one being a connected set of resources and the other being a hierarchical set of processes which make requests on the time-space facilities of the resources. These processes (hence requests) represent the system *internal workload* which descends from the *external workload* (i.e. user demand) of the system.

The objective of this chapter is to connect this computing system (CS) model to a *Queuing Network* model . The purpose of this union is to establish a methodology for analytic and quantitative experimentation. Queueing models have been often contrived, resulting in (sometimes) simplistic, yet effective representation of system perfomance. But nearly all implementations consider only a single independent resource type, i.e., processor resources. The intention here is to extend this *modelling* procedure to treat subordinate storage and data resources; and, in subsequent chapters, to analyse and explore solution techniques for this extended model.

In the following sections of this chapter, we intend to:

(1) Briefly define and specify queueing networks (QN)

(2) Summarise recent developments in the analysis and solutions of queuing networks and review their application to computer performance evaluation.

(3) Introduce an important subclass:state dependent QN's.

(4) Specify the constrained queuing network corresponding to the performance model of 2.3.

3.1 Queuing Networks and Theory

Queuing theory is fundamentally a model of *congestion*; this congestion results from the (random) demands that entities place on a system of finite resources. Since its first conception by Erlang in 1910, there has accumulated a vast amount of literature (for survey, see BHAT69) in the development and application of this theory. Most of this earlier work deals with solution methods of specific conditions on a *single resource* and is of no concern here. Our point of departure is *networks of queues* which have only been actively investigated since the mid 1960's. The definitions which follow are somewhat generalised; this is necessary to include resource constraints which naturally arise in the analysis of computing systems.

3.1.1 Queuing Networks (QN)

A *queuing network* QN, is:

(a) A set of *service nodes*,

(b) A set of *customers* admissible to the set of nodes,

(c) A set of *constraints* which limit the populations, capacities, and joint capacities of the network.

3.1.2 Nodes

A node (queue, service center, service facility, server) is an individual resource of a queuing network consisting of at most three elements: a waiting area, a service mechanism(s) or channel(s) and a queuing discipline.

The *waiting area* may be divided into classes according to customer type or workload required of the server. A customer in this area is said to be waiting, delayed or enqueued.

The *service* mechanism describes the service (hence the consumption in time and space) provided by the node; servers process at a speed called the *service rate* which, in the most general case, may be a joint function of the state of the network.

The *queue discipline* is a dispatching rule which decides which customer from which class next receives service in the node.

A node is said to be *passive* if it has no service mechanism; its only purpose is to support another active node. Or, said another way; if passive nodes fail to support an active node, they may constrain the service rate of that active node. In this thesis, nodes not explicitly designated as passive, will be assumed to be active.

3.1.3 Customers

The entities that migrate among and place service demands on the nodes in the network are called *customers*. Customers *consume* the resources provided by the nodes in time and space and are characterised in the network in three basic ways:

(1) Customers may exist as a single indistinguishable *class* (homogeneous) or in several classes whereby each customer class makes specific (to its class) demands on the nodes and has a class dependent routing among the nodes (non-homogeneous).

(2) A customer places a service demand on each node which is assumed to be a random variable generated by a stochastic process sampled from a distribution called the *service request distribution*, SRD*.

(3) For each customer class, a (stochastic or deterministic) *routing* is specified to describe the visitation of each customer to each node. If *source* and *sink* nodes are included in the network the arrival and departure process (of customers) are specified by the routing process (in conjunction with the service rate distributions).

*this SRD terminology is similar to the *service capacity* terminology introduced by Kobayashi and Reiser (KOBA75b and REIS76) and differs from the ordinary service time distribution in queuing theory. The effect is to separate the demand variation of customers from the service rate variation of the nodes. This, of course, suits computing system models very well and the terminology will be adopted in this work.

3.1.4 Constraints

At this point, the concept of constraints is introduced in the specification of QN's; it is especially important for the class of models formulated to make explicit the specific (joint) limitations of the nodes. Conventional queueing theory rarely specifies finiteness in a parametric way, rather it is implicit in the definition of the models.*

In many cases these constraints naturally are incorporated into the service rate functions; this is usually possible when service rates of one node are independent of those of another node. But when service rates may be a joint function of multiple nodes (particularly passive nodes), an explicit specification is required.

In subsequent chapters, constrained networks will be considered wich not only limit the number in service, but also limit the number allowed to wait.

*for example, a single server queue has a *service capacity* constraint of exactly one customer; a finite population model has a network capacity constraint of, say n, customers. A finite capacity queue (blockable queue) has a parameter constraining the number in service and waiting.

3.2 Methods of Solution

Once a QN has been specified, i.e., its nodes, customers, routing, and constraints have been established, there remains the problem of finding its solution. By solution we mean the (possibly time dependent)probability of the network being in each of its possible discrete states or aggregates of these states. From these one may deduce the performance of the QN by computing the departure rates of the network, the time delays through a set of nodes, and the usage indicators of each node (such as utilisation, busy time, blocked time, etc.).

There are two basic methods of analyzing queuing networks: by the construction and solution of either mathematical or simulation models. Each of these methods may be divided into two dominant sub-methods: mathematical into exact and approximate analytic solutions; simulation into pure and hybrid methods.

3.2.1 Exact Mathematical Models

All of the major works in deriving exact results have been within a class of QN models called Markovian networks. These models will be discussed in section 3.3.1.

3.2.2 Approximate Mathematical Models

There appear to be three predominant approximation techniques for the solution of QN's: (1) diffusion, (2) iterative and (3) decomposition/recomposition.

(1) The application of continuous state, continuous time Markov chain theory, or "diffusion theory" (GAVE63, GAVE68, KOBA74a,b, GELE75, REIS75) to study discrete state, continuous time Markov chains is known as diffusion approximation. The advantage of continuous state Markov chains is that analytical methods which can often be applied e.g. differential equations and integration, are often better developed than those for discrete analysis. The accuracy of the approximation improves as the values for the time variable increase compared to the interval between consecutive transitions; hence its use in heavy traffic systems.

(2) Iterative procedures generally make simplifying assumptions about a sub-network of the QN to be solved. The basic procedure is to iteratively alter the true service rates of the nodes of a simplified network (consisting of two nodes) and solve by ordinary Markovian methods. The thruput and queue length statistics of this reduced system are then tested for compatability with the original network within specified tolerances; if they fail this test, the virtual service rates are altered and the iteration continues. Unfortunately this method only has empirical evidence supporting its accuracy (CHAN74); further research is required to determine bounds and error functions on the approximation.

(3) Decomposition as first described by Courtois (COUR71) is really more a modelling technique than a method of QN solution. The method essentially describes

conditions whereby large QN's may be sub-divided (decomposed) into sub-models which may be solved either by ordinary analytic or simulative methods. The aggregate solution of these sub-models then becomes the parameters of a higher level model; this process continues until the highest level (the original QN) of modelling is realised. This procedure is sometimes known as hierarchical modelling (BROW5b).

In most QN's, sub-modelling into independent models is rarely feasible, so that in practice the network is divided into sub-networks said to be "nearly decomposable". The basic requirement for "near-complete decomposition" is that the subsystem has transient time constants which are far shorter than the mean time between interactions of the subsystem and supersystem. (COUR75).

Of course, the results obtained are only approximations. But the degree of approximation is known and predictable. It can be proved that the error made at each level of aggregation remains of the same order of magnitude as the ratio of the *inter* subsystem to *intra* subsystem interactions and is dependent on the degree of irreducibility of the network. A method to determine the degree of approximation has been developed (COUR75).

3.2.3 Simulation

Probably the most general technique for solving QN's is the *modelling* technique commonly known as discrete system simulation (WHIT75). It can be thought of as performing experiments on a queuing network. Since these "experiments" can be developed in a very detailed and pragmatic way (but usually at great expense), the models may be very general indeed.* Hence simulation is the most used (but, perhaps, the least understood) method for solving QN's. Clearly the emphasis is on efficiency in the design and execution of simulations and many computer programmes have been implemented for this purpose (IRAN71, FOST74, SAUE75). However, the main disadvantage of this technique, aside from development expense, is that it is still an approximation technique (for stochastic networks) and, more importantly, the solution is numerical, not parametric, so that changes in parameters of a QN often necessitate a re-simulation of the network.

3.2.4 Hybrid Simulations

Hybrid simulation is the common term used to describe solution techniques employing both discrete simulation and mathematical techniques within the same model (GOMM75, SAUE77). There appear to be two trends in the development of hybrid simulations; one is to insert pre-derived analytic functions within the event structure of the simulations which reduces the state space, making the simulation more efficient.

*there are some QN problems which may not be solved by ordinary simulation methods, but may be amenable to analytic techniques. Consider the problem of simulating a storage hierarchy where the relative access rates between the top and bottom levels may be ten orders of magnitude. Then to get a statistically significant sample at the bottom, billions of events must be simulated at the top; an unrealistic proposition.

The other method uses the principle of decomposition mentioned above. The idea is to decompose the QN model into submodels which are mathematically tractable and those which are not. The latter are simulated and reduced to analytic functions which are provided as parametric input, along with the other analytic submodels, for recomposition into the original network.

3.3 Queuing Network Definitions

While a complete and formal definition of QN is feasible we shall not attempt one here (see DISN75 for an attempted QN taxonomy). Instead we define a reduced class of QN, known as Markovian queuing networks, which are both analytically tractable and yet general enough to serve as an analytic model for the CS model presented in section 2.3.

3.3.1 Markovian Queuing Networks (MQN)

Consider the general QN described in 3.1, and define a set of elements which represent the discrete condition of the entire network. The collection of these elements (finite or denumerable) in time constitute the time dependent state space of the network. The state space variable may be, in the case of a single node with homogeneous population, a scalar or more generally, a complex data structure representing the demographics of the entire network. Furthermore transitions among these states are restricted to be due to a very special lind of random process called a Markov process and form a Continuous Time Markov Chain (CTMC). The following properties and relationships of CTMC are well known (see KLEI75) and form the basis for the definition of MQN:

(1) CTMC posses the Markov property which states that the way in which the past trajectory of the process (transitions among states) influences the future transitions is *completely* described by the current state of the process;

(2) this implies in particular that the time a process spends in any state is "memoryless" (of past states)*;

(3) this further implies that the CTMC must have exponentially distributed state times (KLEI75). As will be seen, this often is not a severe restriction since other distributions may be emulated by a method of collecting exponential times known as the method of stages (COX55).

(4) for the models presented here, we shall be interested only in the time-homogeneous solution and the time independent solution. It is argued that this solution is unique and efficiently computed; other solutions which are time specific and transient are exceedingly difficult to compute and depend on specific boundary conditions (of which there may be infinitly many) and contribute with diminishing returns to the knowledge of the system. It is important to note that the dynamics of the system are still included, being inherent in the derivation.

(5) for an irreducible homogeneous Markov chain it can shown that the limiting distribution ,p, (often called the steady-state distribution), *always exists* and is independent of the initial state of the system. Furthermore it is *uniquely* determined by the system of *linear* equations

*this is not so much a theoretical limit as a practical one, since we may redefine a state space to include a previous state or states but only at a cost of geometrically increasing the size of the state space.

$pR = 0$ and the normalising condition $\Sigma p = 1$ (3.1)

The system R is known as the transition rate matrix and represents the balance equations of the network.

Queueing networks which satisfy these conditions are called Markovian Queueing Networks, MQN, (REIS75). It is apparent that this is a very general class of QN and affords a significant amount of modelling flexibility. From equation (3.1) it is also apparent that the solution is obtained 'simply' by solving a set of linear equations. The problem is that the state space exhibits a combinatorial growth in the number of equations. Even for very small queueing networks analytic results, even computational results, are not easily achieved; and in the case of infinite state space, perhaps impossible.

3.3.2 Separable Queuing Networks

Fortunately there is a large sub-class of MQNs that have a compact and computationally tractable solution; these forms have been called Separable Queuing Networks, SQN, so called because they can be derived by the method of separation of variables (GORD65, KRZE77). When conditions, called local balance (CHAN72) exist in the balance equations, product-form solutions are obtainable.

3.3.3 Closed, Open, Mixed Networks

If a QN has a finite customer population, with no permissible arrivals or departures, then it is said to be a *closed* network; if arrivals are allowed it is *open*. If there exist classes of customers such that some classes are finite and some have external arrivals then the network is *mixed*. In this work analysis is restricted, but not limited to, closed networks.

3.3.4 Routings

The routings for a MQN may be deterministic or stochastic. A stochastic routing refers to a state independent rule whereby after service completion at a node, a successor node and customer class is chosen at random. Networks which have a single fixed routing (only one successor) are called *cyclic* and *serial* networks for closed and open networks, respectively.

3.3.5 Local and Joint State Dependencies

The QN speficication in section 3.1 allowed for the service rate of any node to be a function of the state of the system. This function is usually restricted to the state of the node itself and is referred to as *local state dependent*, (LSD). Networks which have nodes whose service rates are a function of more one are called *joint state dependent*, JSD. This definition expands the class of models used to represent QN's. While local state dependent rates are allowed in SQN, joint state dependent ones are not. The modelling and solution of certain JSD networks will be explored in Chapters 4 and 5.

3.3.6 Queue Disciplines

It is usual to describe queue disciplines as a rule which specifies which customer from the waiting area next receives service in the node (scheduling); some common disciplines are:

FCFS: first come, first serve

LCFS:PR: last come, first serve, preemptive resume

PRIORITIES:

PS: Processor Shared, customers all share the node but at a diminishing rate.

IS: Infinite Server-all customers share the node without diminished rate

Since we are extending the QN model to possibly include finite waiting areas, other rules may be required to admit customers to waiting areas of (blocked) nodes. These rules will be discussed in Chapter 4.

3.4 Exact Analysis of Separable Queuing Networks (SQN)

In this section, SQN solutions are reviewed; assumptions, restrictions and the scope of the models are noted.

3.4.1 Jacksoń's Theorem

Although simple tandem (2 node, serial network) queues have been studied since the 1950's (REIC57), it was not until the 1960's when Jackson presented his remarkable results that a major solution of queuing networks became available (JACK57, JACK63). With the exception of a few notable extensions, it remains the major result.

A Jackson QN is shown in figure 3.1: a set of N service nodes are interconnected arbitrarily Customers enter the network from an infinite source and are routed to a service node. Let $k = (k_1, k_2, \ldots k_i, \ldots k_N)$ be a vector of integers, k_i being the number of customers at each service node and denote $K = \Sigma k_i$ the total number of customers in the network. The customer service request distribution, SRD, is assumed to be the exponential distribution with mean w_i, and each service node, i, has a local state dependent service rate of $c_i(k_i)$; therefore the *departure rate* is a function of k_i denoted $\mu(k_i)$ and $\mu(k_i) = c_i(k_i)/w_i$

Further specify a routing matrix $Q = \{q_{ij}\}$ such that

$q_{ij} = \text{PROB}\left\{\text{a customer departing node i, goes to node j}\right\}$

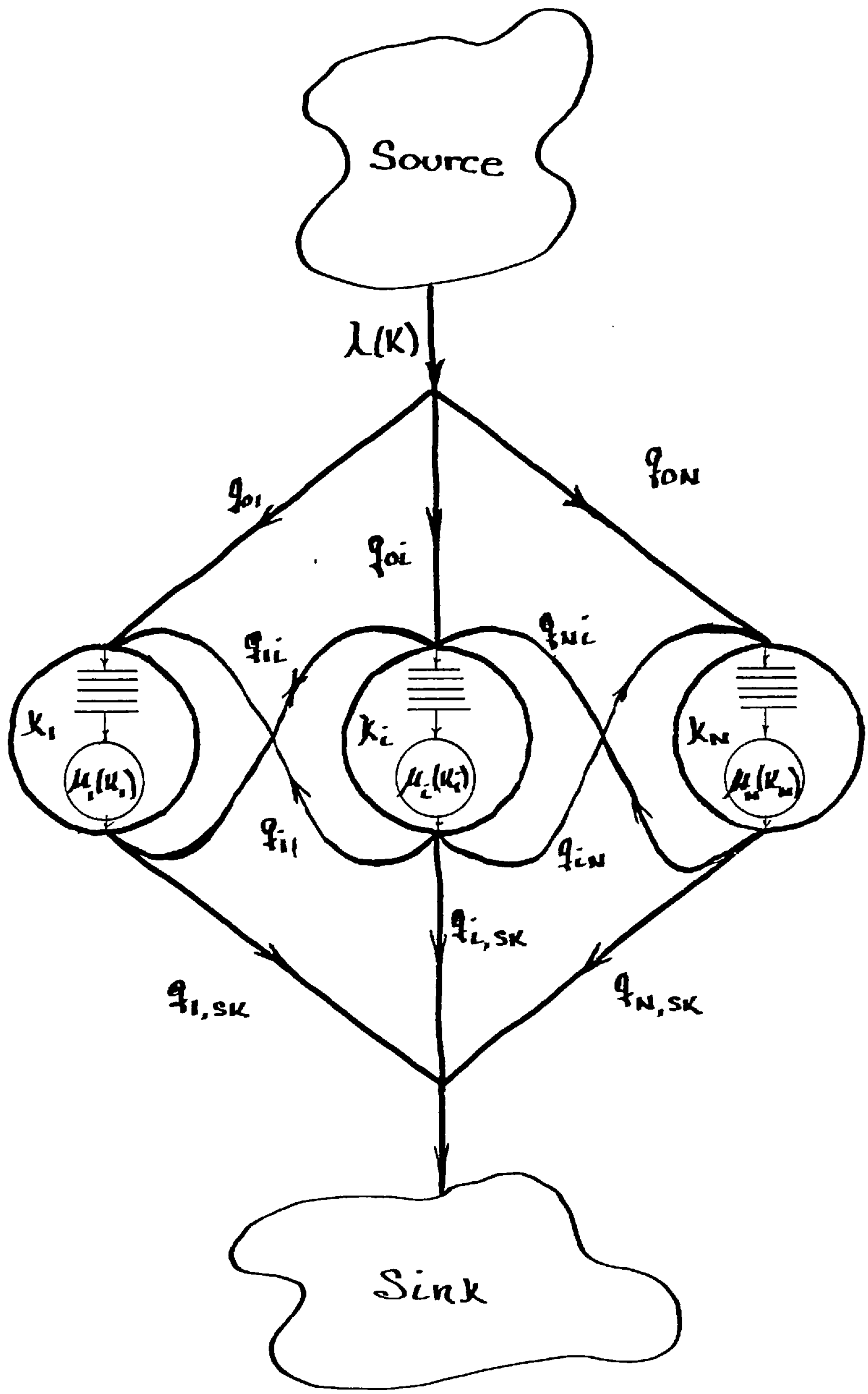


Figure 3.1 Jackson Queuing Network

where node O is assumed to be the source and node n + 1 the sink. The arrival process is assumed to be Poisson with rate $\lambda(K)$ customers are assumed homogeneous. With these assumptions the number of customers at each node constitute a CTMC.

Jackson's solution to this system of equations, which he ingeniously deduced*, and can be proved by direct substitution into the balance equations, is

$$p(\underline{k}) = G^{-1}(K) \prod_{m=0}^{k-1} \lambda(m)\; H(\underline{k}) \quad \text{where} \tag{3.2}$$

$$H(\underline{k}) = \prod_{i=1}^{N} \beta_i(k_i)\,(W_i)^{k_i} \quad \text{and} \quad \beta_i(k_i) = \prod_{j=1}^{k_i} \frac{1}{c_i(j)} \tag{3.3}$$

$$W_i = e_i w_i$$

$G(K)$ is the normalising constant such that

$$G(K) = \sum_{K'=0}^{\infty} \prod_{m=0}^{K'} \lambda(m) \sum_{k \varepsilon \underline{S}} H(\underline{k}); \quad \underline{S} \text{ is the set of all} \tag{3.4}$$

$$k \text{ such that } \Sigma k_i = K$$

where e_i is the solution to the following system of linear equations:

$$e_i = q_{oi} + \sum_{j=1}^{N} e_j \cdot q_{ji} \quad i = 1,2,\ldots N \tag{3.5}$$

and is interpreted as the number of visits a customer makes to node i during its lifetime in the network; hence W_i is the *expected workload* demand a customer places

*to date, no constructive proof or derivation exists.

on the node during its life.

If the arrival rate function is a constant, $\lambda(K) = \lambda$, and all service rates are constant, $c_i(k_i) = c_i$, then 3.3 simplifies to

$$p(\underline{k}) = \prod_{i=1}^{N} (1-p_i)p_i^{k_i} \quad \text{and } p_i = \frac{\lambda W_i}{c_i} \qquad (3.6)$$

that is, the joint distribution of $\underline{k}$ is decomposable into the product of marginal distributions of the individual nodes; this is often referred to as Jacksons Decomposition Theorem.

3.4.2 Closed Networks

A few years after the publication of Jacksons Theorem, Gordon and Newell (GORD67) presented a solution to a similar network; the only significant difference being that the network had a finite population of K (no arrivals or departures). Their solution turns out to be a special case of Jackson's result. Although it does not extend the applicability of the Jackson Model, the importance of the GN*model is that a different derivation method was employed (separation of variables), which, although still not constructive, provides more insight than direct substitution.

*Gordon and Newell closed network (GORD67)

3.4.3 Computational Algorithms

Although Jacksons and the GN results were available since the mid 1960's, it was not until the early 70's that they were put to use. The problem was that the normalising constant, G, had to be summed over the set S defined in 3.4. It can be seen that this set contains $\binom{K+N-1}{K}$ terms; hence for even very modest networks, direct enumeration is difficult. In 1971 Buzen (BUZE71) and shortly thereafter others (REIS73) produced a simple recursive algorithm to calculate this constant. With these solutions and computational forms in hand, Jackson and GN models became popular modelling vehicles for performance evaluation of computing systems.

Note that the above solutions both have product form solutions and therefore are Separable Queuing Networks (SQN). Yet this model has several shortcomings in the applicability to computing system modelling; they are:

(1) inability to distinguish among classes of customers with distinct stochastic behaviour.

(2) restriction of SRD and interarrival time distributions which must be exponential.

(3) restriction on probabilistic routing behaviour given by first order Markov chains.

(4) inability to accommodate queue disciplines other

than FCFS.

(5) exclusion of nodes in which service time parapeters depend on the number, and properties, of customers in a subnetwork, i.e., joint state dependencies.

Currently, there is no SQN model which alleviates all of these problems; however, the following extension significantly extends QN model applicability.

3.4.4 The 'BCMP' Theorem

In 1975, the Jackson model was significantly extended to allow for:

(1) Non-homogeneous customer classes each may have its own routing among nodes *and* classes.

(2) Service disciplines other than FCFS.

(3) Relaxation of the exponential SRD for some node types, as defined in (2)

This extension was developed by Baskett et al (BASK75) and is referred to as the 'BCMP' Theorem.

The network topology is the same as in figure 3.1 except that multiple customer classes, $\ell = 1,2...L$, and routings are admitted and service nodes, i, $1 \leq i \leq N$, are of four types:

(1) Node i has a FCFS discipline and an exponential SRD with parameter $w_{i\ell}$

(2) Node i has a PS discipline (cf. 3.3.6) and the SRD may be modelled by the method of stages (cf. 3.3.1).

(3) Node i is an infinite server, i.e. $c_i(k_i) = k_i c_i$ for all i, and the SRD is nearly general.

(4) Node i has a LCFS:PR discipline (cf 3.3.6) and a nearly general SRD.

For the BCMP network, the solution is of the form*:

$$P(\underline{k}) = G^{-1} d(k) \prod_{i=1}^{N} g_i(\underline{k}_i) \tag{3.7}$$

where

$$d(k) = \begin{cases} \prod_{m=0}^{k-1} \lambda(m) & \text{if the QN is open} \\ = 1 & \text{if it is closed} \end{cases} \tag{3.8}$$

G is the normalising constant and $g_i(\underline{k}_i)$ are product forms dependent on the node type and degree of state aggregation

Further generalisations by Gelenbe and Muntz (GELE76) and Kobayashi and Reiser (KOBA75a) resulted in the more recent extensions:

(1) deterministic or n-th order routings may be specified.

*Only a very condensed form of BCMP results are displayed here, the complete results are available in the original paper (or KRZE77)

(2) Routing transitions need not be instantaneous, but may have a nearly general delay distribution.

These analytical developments were matched by the design of efficient numerical evaluation techniques (MUNT74, KRZE77) so that relatively large and general networks can be solved. Note that the solution of the BCMP network is still a product form and belong to the class of separable networks. Although these results greatly expand the Jackson solution *there are still no general results* for:

(1) FCFS queues with general SRD or non-homogeneous workload

(2) Priorities

(3) Blocking or limited access to subsystems

(4) Simultaneous occupancy of Resources

(5) Waiting time distribution

(6) Transient Solutions

3.5 Queuing Network Models of Computer Systems

The development and generalisation of SQN models has inspired many queuing network representations of computer systems. These have been extensively surveyed (MCKI69, ADIR72, WYSZ75, and MUNT75) and we shall briefly summarize the key models and their contributions.

3.5.1 Machine Repair Analog

Scherr used the classical machine repair model (FELL57) to evaluate a multiprogramming computing system (SCHE65) where the 'machines' were jobs in I/O processing and the 'repairman' was analogous to CPU processing. It is interesting to note that even though the service times and routings did not conform to the model assumptions, Scherr reported good results with respect to direct system measurement.

3.5.2 Cyclic Queues

Two node, cyclic queues were used extensively to study paging behaviour, supervisor overhead, and I/O delays (LEWI71, GAVE73). Work was divided into two types, CPU cycles and data transfer, and was represented by two nodes. These models could often be solved without imposing the exponential assumption (yet still FCFS) which violates the BCMP restriction.

3.5.3 The Central Server Model

The first extensive use of Jackson's networks for the evaluation of computing systems were reported by Moore (MOOR71) and particularly Buzen (BUZE71). Besides the computational algorithms previously mentioned, Buzen also introduced the 'central server model' to describe the behaviour of a computing system where K jobs are permitted to circulate endlessly among the N resources, and all routings are through the central server (CPU). Under its simple assumptions, many interesting and useful results have been derived.

3.5.4 BCMP Implementations

Shortly after the appearance of the BCMP theorem and its companion computational algorithms, computer programming packages became available. QNET4 (REIS75a), programmed in APL, provided an application oriented conversational language, and SNAP (KRZE76) furnished a batch FORTRAN version. These routines have often been used for the evaluation of computing systems (REIS76, KRZE77, HARR78) and the models have been successfully validated against measurements of existing systems (GIAM76, ROSE76, KRIT77).

3.6 MQN Specification

Implicit in the above discussion is the idea that a computer system can be represented by a QN whose parameters are the quantifiable demands, transformations, and constraints of the actual system: and that the solution variables could be reinterpreted as a representation of the performance of the actual system. These models are making fundamental assumptions about (1) the objects of the system, i.e., the processes and system resources, and (2) the numerical assessment of the customer demands and node service rates; that is, the time-space requirements and constraints of the processes and resources, respectively.

The most conspicuous limitation of the SQN deployed in the evaluation of systems is that the service rates of the nodes must be independent - that joint state dependencies are not allowed. This deficiency severly limits the kinds of resources which may be abstracted by a SQN.

In particular, finite storage and data objects cannot be accommodated; processor resources which inhibit or *block* service of other resources are disallowed.

It is our thesis that future development of computing systems are moving in the direction of greater concurrent processing with more emphasis on shared data objects (e.g. data bases, directories, etc.) and shared storage (multi-level hierarchies). Hence the performance of these systems will be strongly influenced by the contention for these finite resources rather than speed or configuration changes of hardware processors. Therefore performance models must be capable of predicting the effects of limited resources.

3.6.1 Performance Queuing Network Model

The following is a Markov Queuing Network (MQN) specification of the qualitative model introduced in section 2.3. Its purpose is to provide the correspondence between the performance and queuing models, to specify the system and workload, and to define the notation.

3.6.2 Specifications

3.6.2.1 Resource *Specification*

For the resource in the considered computing system, let there be

(1) a set of *service nodes*, $\underline{N} = \{1,..n,..N\}$, assuming one for each *active* (processor) resource

(2) a set of *passive* nodes, $\underline{N}' = \{N+1,..n'...N'\}$, assuming one for each *passive* (storage or data) resource

3.6.2.2 Resource *Parameters*

(1) For each active resource, $n \in \underline{N}$, let $c_n(\underline{k})$ be a positive real function, the *service rate* (work units/sec) of node (resource) n when the system

is in state $\underline{k}$. (c.f. 3.6.2.5)

(2) Let d_n be the *limiting capacity* of resource n, $n \in \underline{N} \cup \underline{N}'$ *. In the case of active resources, d_n, this will be a positive integer representing the maximum finite queue population of the processor node. In the case of passive resources it will represent the total finite units of resource available (e.g. 10KB of storage).

3.6.2.3. Process Specification (workload)

For each process type (requests, transactions, jobs), let $\underline{L}$ be the set $\{1,2,...\ell...L\}$ of customer classes.

3.6.2.4 Workload *Parameters*

(1) Let $w_{n\ell}$ be the mean of the service request distribution (SRD), representing the *mean service request* of process ℓ on *active* resource n (work units/request). In this work we shall always assume exponential service so that stage indicies in the state space specification are unnecessary. Furthermore the nodes will all be considered simple nodes(complex node types being emulated by various $c_n(\underline{k})$ functions , c.f. 3.6.2.2.(1)).

(2) Let $g_n(\underline{k})$, $n \in \underline{N} \cup \underline{N}'$, over the state space $\underline{k}$, be a set of vector valued positive functions with integer range which represent the requirements of active and passive resources as a function of the the active nodes. Such functions will have associated variables or constants representing the number of passive units per active unit (e.g., 5KB storage/CPU process).

*this capacity limitation could easily extended to handle customer classes.

(3) Let $\lambda_{\ell}(K)$ be the mean (Poisson) *arrival rate* of process type ℓ to the network (system) in processes/sec.

(4) Let *routing*, $q_{i\ell,jm}(\underline{k})$ represent the probability that a process of type $\ell \in \underline{L}$ upon leaving node i proceeds instantly to node j, ($i,j \in \underline{N}$) and becomes a process of type m, $m \in \underline{L}$; the routing may be a function of $\underline{k}$. We further assume that the routings are finite and irreducible.

3.6.2.5 The State Space

Let $\underline{k} = (\underline{k}_1, \underline{k}_2,,, \underline{k}_n .. \underline{k}_N)$ be the vector of process types representing the population (occupancy at each active resource) where $\underline{k}_n = (k_{n,1}, k_{n,2} .. k_{n,\ell} .. k_{n,L})$ and $k_{n,\ell}$ is the population of process type ℓ at node n. S, such that $\underline{k} \in \underline{S}$, is called the *state space*.

(Note for notational convenience, the stage of service index required if the SRD is non-exponential has been ignored; this is handled as in CHAN75. Furthermore special node types requiring station balance(CHAN77) are not considered.

3.6.2.6 The Feasible State Space

Let $\underline{F} \subseteq \underline{S}$ be the feasibel state space such that for all $g_n(\underline{k}) \leq d_n$, $n \in \underline{N} \cup \underline{N}'$, the states $\underline{k}$ are feasible, i.e., $\underline{k} \in \underline{F}$.

3.6.2.7 The Solution

The solution (if it exists) to the MQN model is determined by finding $p(\underline{k}) \varepsilon \underline{F}$, where $p(\underline{k})$ is the time indepenedent probability that the network is in state $\underline{k}$. Given the previous conditions above for the unconstrained MQN network (especially with respect to 3.6.2.4(1) and (4)) it is well known that the solution exists and is unique (GELE76). However, once the constraints (i.e., infeasible states) are considered, solutions may not exist if such constraints lead to inconsistencies in the balance equations.

3.7 Summary

In this chapter, a class of queuing network models, called MQN, have been described. These models have sufficient structure to represent quantitative models of the production process of computing systems. They may also have compact and relatively simple solutions. Unfortunately the subclass of these networks, which have known analytic solutions, do not allow for the modelling of joint state dependent nodes.

Consideration of these stated dependencies will be the subject of the remainder of this thesis.

CHAPTER 4

S T A T E   D E P E N D E N T   Q U E U E I N G N E T W O R K   M O D E L S

4.0 INTRODUCTION

The previous chapter reviewed QN models and their application to system performance. Within the class of Markovian Queueing Networks, a performance model was specified such that passive resources are considered to be performance limiting objects of the system These limitations constrain the feasible network states.

In this chapter, we

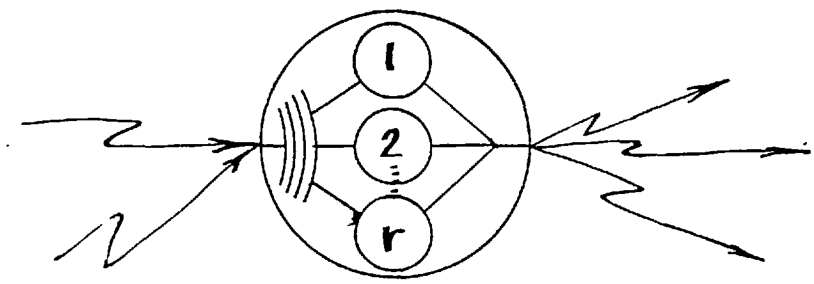
(1) define local and joint state dependent service rates,
(2) derive a local state dependent function corresponding to a system resource pool; this pool is referred to as a *multiple server*,
(3) interpret the constrained state space and consider its disposition,
(4) present a simple, but revealing, example of passive resource blocking.

4.1 Local State Dependent Service Rates

When Jackson presented his results on queueing networks [JACK63], he introduced the notion of service rate function, i.e., the service rate of a node may be any positive function of the number of processes at the node. In this thesis the service rate function has been separated into two components: one being the mean service *requirement* of the processes, sampled from a service request distribution (SRD),and the second being a service function possibly a *Joint Function* of the population of the nodes of the network. (c.f. 3.1.3) We call networks which have joint dependent service rates *joint State Dependent* (JSD) networks. Furthermore networks whose nodes *only* allow rate variation as a function of the state of its own node (such as Jackson Networks) are called *Local State Dependent* (LSD); finally networks whose nodes all have *constant* service rate are called *State Independent.*

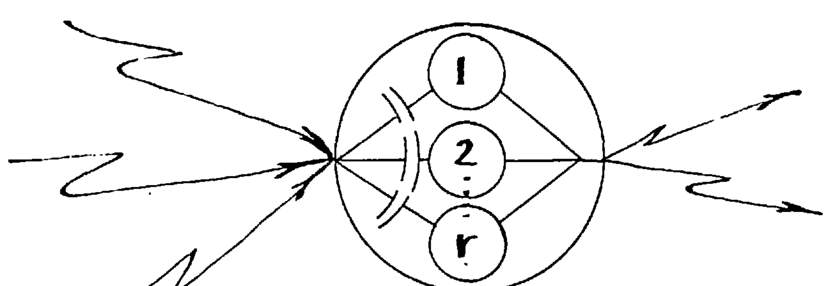
4.1.1 Local State Dependent Processor Node Models

Much of the usefulness of LSD functions is due to their facility for compactly modelling processor nodes via simple analytic expressions. In the GN networks [GORD67] a simple linear function was used to model multi-server nodes. (Nodes which allow parallel processing, figure 4.1a). This node, together with its limiting form, the infinite server, has proven to be very useful in modelling computing systems.



$$c(k) = \begin{cases} 1 & 0 \le k \le 1 \\ k & 1 \le k \le r \\ r & k \geq r \end{cases}$$

a) Multiserver



$$c(k) = \begin{cases} 1 & 0 \le k \le 1 \\ \dfrac{kr}{k+r-1} & k \geq 1 \end{cases}$$

b) Multiple Server

figure 4.1 Multi-server and Multiple server nodes

4.2 The Multiple Server

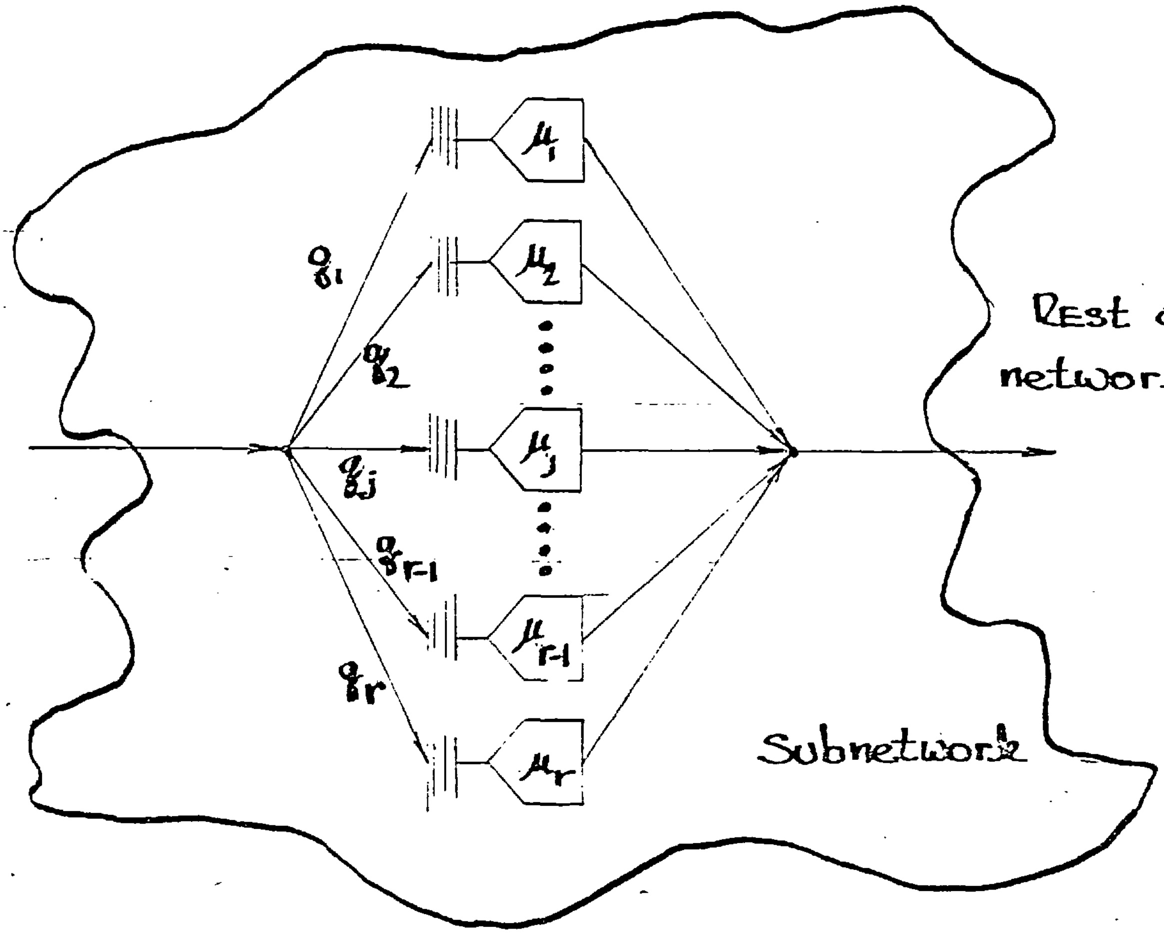
In the modelling of computing systems, the occasion often arises where there are multiple, functionally equivalent processors which may *not* service requests from the same queue; each must have its own local queue. For example storage modules (i.e. disk drives, memory modules) which are randomly accessed, but each request may *only* be serviced by a specific device; another example is the distribution of processes (messages) to communication processors. To model this phenomena a new queueing construct is introduced, the *Multiple Server* node (figure 4.1b); its specification is as follows:

Consider the sub-network shown in figure 4.2a consisting of r state-independent, FCFS processor nodes with mean departure rate μ_j, j = 1,2...r. Furthermore processes arriving at the sub-network are routed to node j with probability:

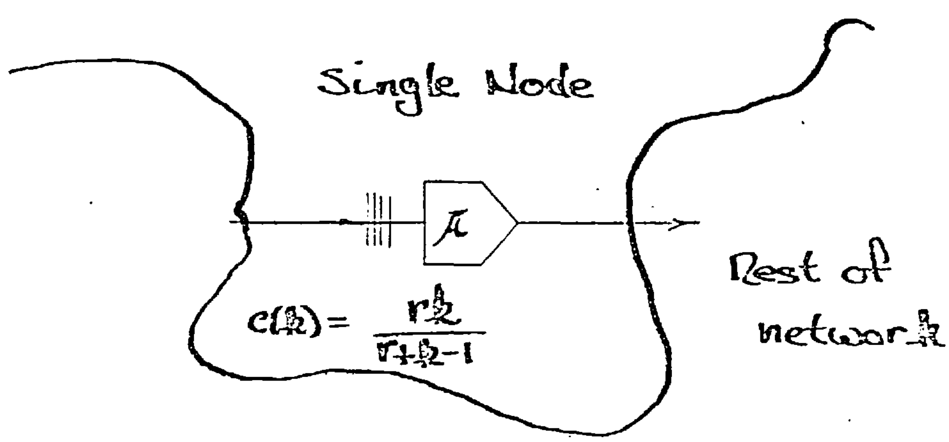
$$q_j = \frac{\mu_j}{r\bar{\mu}} \qquad \text{where } \bar{\mu} = r^{-1}\sum_j \mu_j \qquad j \in \underline{N}$$

This *sub-network* can be replaced by an equivalent node with an LSD function and is referred to as a *multiple server node.* The result is stated and proved in two parts, beginning with:

THEOREM 4.1: Given the sub-network above, containing k processes in a closed cyclic network, the thruput of the



a) Subnetwork containing multiple identical processors



b) Equivalent replacement node – the multiple server

figure 4.2 Equivalent networks

network is

$$T(k) = \frac{kr}{k+r-1}\,\bar{\mu}$$

Proof:

The closed network state independent thruput results are known to be (see BUZE75).

$$T_i(k) = \frac{e_i G(k-1)}{G(k)} \qquad i = 1,2\ldots r \tag{4.1}$$

The visitation rates e_i are the solution to linear system (c.f. 3.4.3)

$$e_i = \sum_j e_j\, q_{ji} = \frac{\mu_i}{\bar{\mu} r} \sum_j e_j \qquad i = 1,2\ldots r \tag{4.2}$$

A solution is $e_i = \mu_i \rho$ where (4.3)

ρ is a constant.

and
$$G(k) = \sum_{\substack{\Sigma k_i = k \\ k_i \geq 0}} \prod_{i=1}^{k_i} \rho^{k_i} \tag{4.4}$$

$$= \sum \rho^{\Sigma k_i} = \rho^k \sum_{\substack{\Sigma k_i = k \\ k_i \geq 0}} \tag{4.5}$$

The sum in 4.5 is well known, so that

$$G(k) = \binom{r+k-1}{k} \rho^k \tag{4.6}$$

Substituting 4.6 and 4.3 into 4.1 produces

$$T_i(k) = \frac{e_i \rho^{k-1}}{\rho^k} \frac{\binom{r+k-2}{k}}{\binom{r+k-1}{k}} = \mu_i \frac{k}{r+k-1} \quad . \tag{4.7}$$

Total thruput must be the sum of the individual thruputs

$$T(k) \triangleq \sum_{i=1}^{r} T_i(k) = \frac{kr}{k+r-1} \bar{\mu} \tag{4.8}$$

which proves the theorem.

COROLLARY 4.1.1

The utilisation of the network with r nodes and k customers is

$$U(k) = \frac{k}{k+r-1}$$

Proof: $$U_j(k) \triangleq \frac{T_j(k)}{\mu_j} = \frac{k}{r+k-1} \qquad \text{(by 4.7)} \tag{4.9}$$

and

$$U(k) \triangleq \frac{T(k)}{\sum\limits_{j=1,r} \mu_j} \quad \frac{k}{r+k-1} \tag{4.10}$$

where, as before $\bar{\mu} = \Sigma \mu_j / r$ (4.11) .

COROLLARY 4.12

The expected queue lengths and expected response times for each node in the network are respectively,

$$L(k) = \frac{k}{r} \quad ; \quad t(k) = \frac{r+k-1}{r\mu} \tag{4.12}$$

Proof: For a state independent network it is easily shown that (see for example BUZE75) the expected queue length is

$$L(k) = \sum_{\ell=1}^{k} \rho^{\ell} \frac{G(k-\ell)}{G(k)} = G(k)^{-1} \sum_{\ell=1}^{k} \rho^{\ell} G(k-\ell) \tag{4.13}$$

substituting 4.6 into 4.13 yields

$$L(k) = G(k)^{-1} \sum_{\ell=1}^{k} \rho^{\ell} \binom{r+k-\ell-1}{r-1} \rho^{k-\ell} = G(k)^{-1} \rho^{k} \sum_{\ell=0}^{k-1} \binom{r-1+\ell}{\ell}$$

$$= G(k)^{-1} \rho^{k} \binom{r+k-1}{k-1} \tag{4.14}$$

substitution once again of 4.6 provides

$$L(k) = \frac{\rho^{k} \binom{r+k-1}{k-1}}{\rho^{k} \binom{r+k-1}{k}} = \frac{k}{r} \tag{4.15}$$

The mean response time is derived by a straight-forward application of Little's Theorem (L=Tt),

$$t(k) = \frac{L(k)}{T(k)} = \frac{k/r}{k\mu/r+k-1} = \frac{r+k-1}{r\mu} \tag{4.16}$$

REMARKS

If a resource pool of processors exists and processes are routed to the individual nodes in this network in proportion to their service rates (faster nodes receive proportionally more processes), then theorem 4.1 and its corollaries provide remarkably simple formulae for the evaluation of the pool. Notice that queue lengths and utilisations of the resource pool are independent of the mean service rates.

The LSD function for the multiple server node follows directly according to

THEOREM 4.2

Let a simple state dependent resource (*multiple* server) node replace a resource pool (sub-network) which has r nodes each with FCFS, state independent service rates μ_j and routing to each node $q_j = \mu_j/r\bar{\mu}$; then this replacement is stochastically equivalent to the original network if the LSD function of the replacement node is:

$$c(k) = \frac{kr}{r+k-1} \qquad (4.17)$$

Proof: This may be proved in several ways, but the most compact form relies on Nortons Theorem for Queueing Networks (CHAN75a)*. Other proofs are often special cases of this theorem . Nortons Theorem states

---

*an alternative proof has been recently published in HARR78 that for a separable network, take any isolated sub-network which can be 'shorted', i.e., closed, and then solve for its thruput (numerically or analytically) as a function of the population of the sub-network. Then this sub-network may be replaced by a composite node with this thruput function as its LSD function (figure 4.2a,b). The queue length distribution of this new network is identical to that of the original network.

Since the conditions of this theorem have been satisfied by theorem 4.1., the proof is immediate and

$$T(k) = \bar{\mu}\; c(k) = \frac{kr}{r+k-1}\; \bar{\mu} \qquad (4.18)$$

COROLLARY 4.2.1

The maximum utilisation and thruput of a multiple server node (with parameter r) in any closed network with population K are given by

$$U_{max} = \frac{K}{K+r-1}\,; \qquad T_{max} = \frac{Kr}{K+r-1}\,\bar{\mu} \qquad (4,19)$$

Proof: The results for a closed network consisting of only a multiple-server are given in Theorem 4.1 and Corollary 4.1.1. This assumes, essentially, that the surrounding network contributes no delay (i.e., is infintaly fast) to the closed subnet. Therefore any active delay nodes in other parts of the network may add additional delays or reduced arrival rates to the multiple server node; therefore the upper bounds must be those provided in Theorem 4.1 and Corollary 4.1.1.

This remarkable result indicates that a multiple-server node may be a bottleneck (the limiting node) in a network even though its servers have a very low utilisation. For example, consider a computing system with a central processor (CPU) and 32 disks and a level of multiprogramming of say, 8. Then the *maximum* utilisation by Corollary 4.2.1 is

$$U_{max}(8) = \frac{8}{8+32-1} \simeq 21\%$$

Even though analysis of the network indicates that the CPU is much more highly utilised, no further improvements in performance are possible by (erroneously) speeding up or adding CPU processors. This theorem tends to expose the flaws in performance evaluation based on utilisations alone (which is probably the most often 'used' metric in CPE)

The characteristics of the multiple server may be compared with those of the multi-server. For the same service demand parameter, μ, a plot of LSD functions, for both node types, is given in figure 4.3. Note that these functions are identical when the node population is either 1 or becomes arbitrarily large.

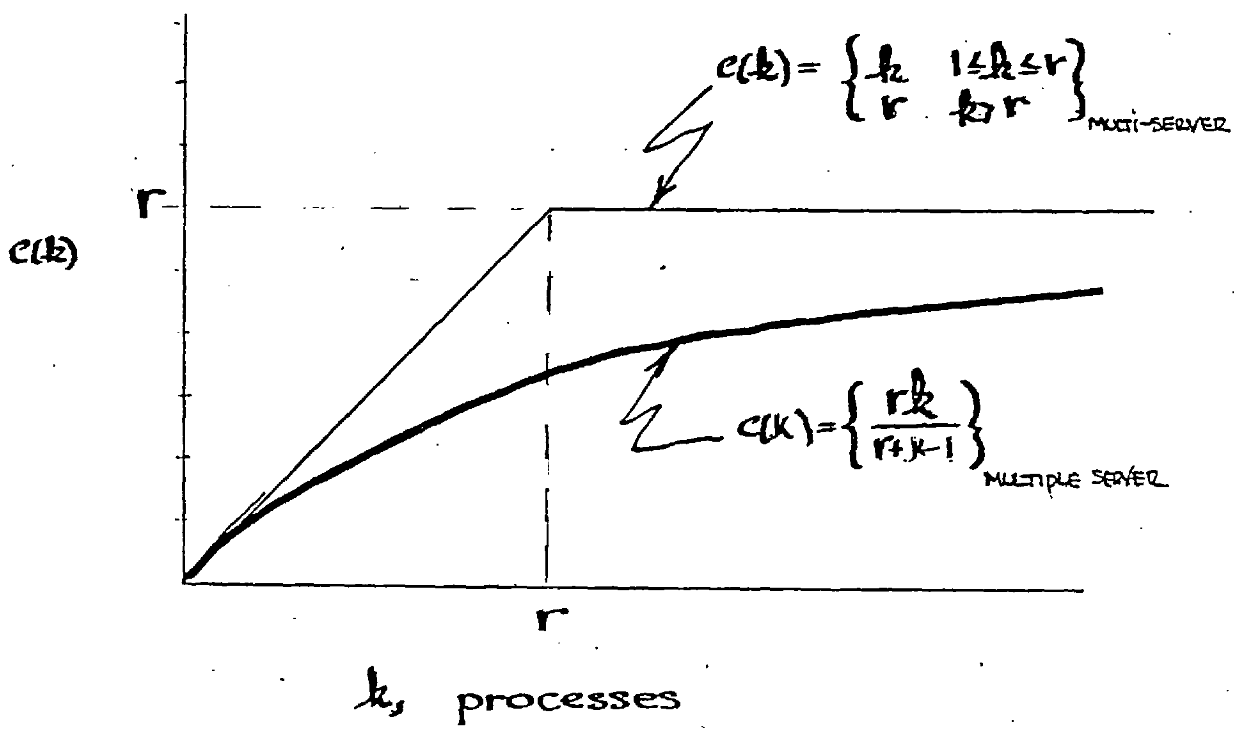


figure 4.3 Comparison of Multi- and Multiple servers

LEMMA 4.1

Let $T_{M-s}(k)$ be the thruput of a *multi-server* node with r, $(r>1)$, parallel servers and service parameter μ and $T_{Ms}(k)$ be the thruput of a *multiple* server with identical parameters then the *multi-server* always provides better performance where the maximum thruput ratio occurs at $k=r$ and is: $2r-1/r$.

Proof: Let $R(k) \triangleq \dfrac{T_{M-s}(k)}{T_{Ms}(k)}$ (4.20)

then by direct substitution of their respective LSD functions,

$$R(k) = \begin{cases} \dfrac{k+r-1}{r} & k \le r \\ \dfrac{2r-1}{r} & k=r \\ \dfrac{k+r-1}{k} & k \ge r \end{cases} \tag{4.21}$$

Note that $R(k) \ge 1$ for all positive values of k.

for arbitrary positive $\delta>0$: we need to show that

(1) $R(r-\delta) < R(r)$ and (2) $R(r+\delta) < R(r)$ (4.22)

for (1) of 4.22 by direct substitution :

$$\frac{(r-\delta) + \delta-1}{r} < \frac{2r-1}{r} \tag{4.23}$$

only if $\delta>0$ which is true; and for (2) in 4.23

$$\frac{(r+\delta) + r-1}{r+\delta} < \frac{2r-1}{r} \tag{4.24}$$

$r < 2r-1$ which is true for $r > 1$.

This point is clearly a global maximum.

Comment: The node designs implied by this result are obvious - the *performance* of a multi-server node is always better than that of a multiple server; but only up to twice as good ($k=r$; r large). For example, there would be a *performace* advantage in designing a simple disk storage unit with two accessing mechanisms rather than two identical units of half the capacity and a single mechanism, (a maximum performance improvement of 50%).

With respect to computational forms, the multiple server node offers a convenient generating function which is invertible and useful in the convolution algorithms (see REIS75).

LEMMA 4.2 For a multiple server node with parameters r, e, and μ and LSD function $c(k) = \frac{rk}{k+r-1}$ its generating (or capacity) function is given by

$$a(z) = \left(1 - \frac{e\mu z}{r}\right)^{-r} \tag{4.25}$$

Proof: Following REIS75, define the generating function,

$$a(z) \triangleq \sum_{k=0}^{\infty} \alpha(k) z^k \qquad \alpha(k) \triangleq \prod_{\ell=0}^{k} \frac{e\mu}{c(\ell)} \tag{4.26}$$

$$\alpha(k) = \left(\frac{e\mu}{r}\right)^k \prod_{\ell=0}^{k} \frac{\ell+r-1}{\ell} = \left(\frac{e\mu}{r}\right)^k \binom{k+r-1}{k} \tag{4.27}$$

$$a(z) = \sum_{k=0}^{\infty} \binom{k+r-1}{k} \left(\frac{e\mu z}{r}\right)^k \tag{4.28}$$

using the identity $\binom{k+r-1}{k} = \binom{-r}{k} (-1)^k$

$$a(z) = \sum_{k=0}^{\infty} \binom{-r}{k} \left(-\frac{e\mu z}{r}\right)^k = \left(1 - \frac{e\mu z}{r}\right)^{-r} \qquad (4.29)$$

where (4.29) follows immediately from the binomial theorem.

This result may be incorporated in the usual convolution algorithms [REIS75] for the efficient treatment of multiple server nodes. Furthermore the use of this node will, for most computing system models, greatly reduce the size of the network. For example if we have an 'ordinary' network which models a CPU (with storage) 4 drum, 64 disk, and 32 tape storage devices, then the numbers of nodes in the network is 101, but the storage devices all satisfy the conditions of the multiple server so that this network is replacable by one of only 4 nodes.

4.3 The State Space

The performance model described in chapter 2 is represented as an MQN; which, is analytically specified by transitions among states. The purpose of these states is to compactly specify the requisite knowledge of the system. Theoretically, this poses no problem, since one may arbitrarly assign names (symbols) to each state and solve the system of linear equations(c.f. 3.31). In practice, however, this is rarely possible; there being two problems: one of complexity and one of multiplicity.

The first problem is due to the variety of complex conditions to be studied (e.g.number of processes at each processing node, the class of each process, their position in the queue, their resource requirements, their priority rules, etc.); the second problem is due to the sheer number of states (hence linear equations) which grow geometrically in the number of processes and nodes.

4.3.1 The State Transition Diagram

As an aid in visualising the transition rates amongst the states and the effects of constraints, a directed graph (figure 4.4) is defined such that the nodes of the *graph* (not to be confused with the nodes of the *network*) are the states, $\underline{S}$, and the directed arcs represent the probability *flow* between them; where in general $\mu_i(\underline{k})$is the departure rate of node i as a function of the current state $\underline{k}$ and $q_{ij}(\underline{k}')$ i the routing probability from node i to node j as a function

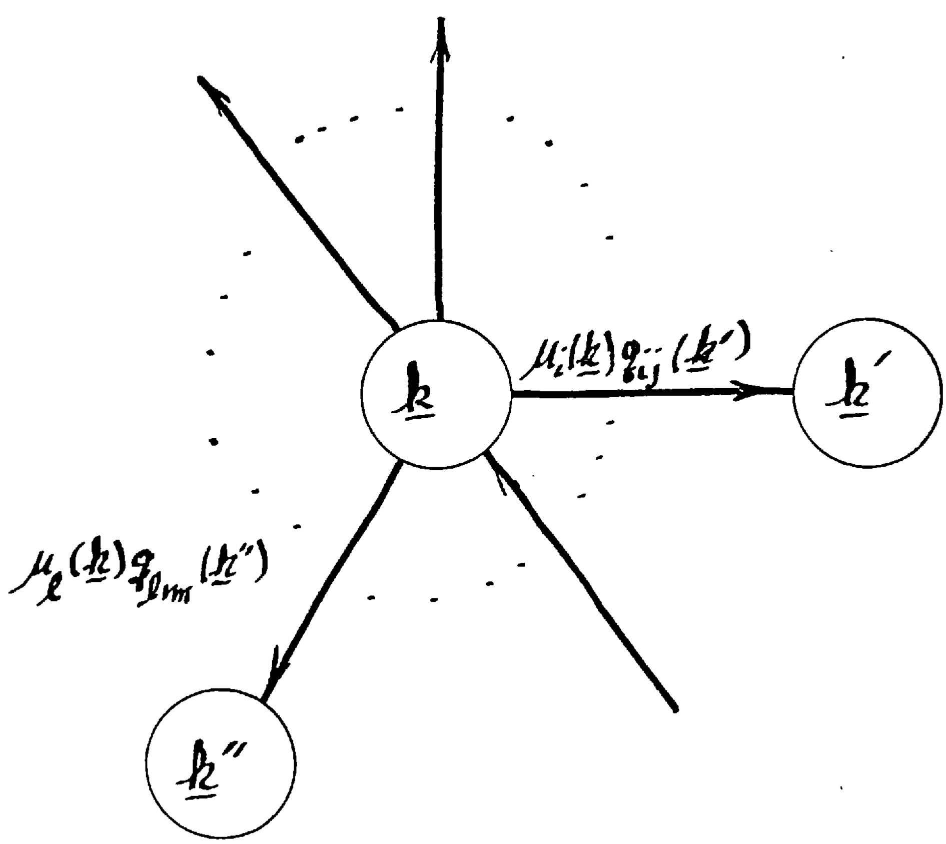


Figure 4.4 State Transition Diagram

of the state $\underline{k}'$ (target state

Figure 4.5 and 4.6 display various state transition diagrams for open and closed networks, respectively. These diagrams will be used not only to illustrate the transitions among states, but to display *constraints* on the space.

4.3.2 Constraints

State space constraints may be represented on the transition diagrams as a region of infeasible states (figures 4.5 and 4.6).

In figure 4.5a, two constraints, C_1, C_2 have been introduced and effectively 'cut' the state space, these constraints have the following effect on the state space :

all states $\underline{k} \in \underline{S}$

Constraints		Conditions
none :		s.t. $k_i \geq 0$
C_1 :		s.t. $k_1 \leq k_1^*$
C_2 :		s.t. $k_1 \geq 2$
C_1&C_2 :	s.t.	$2 \leq k_1 \leq k_1^*$

In figure 4.5b, constraints C_3, C_4, and C_5 have been added yielding the following effects

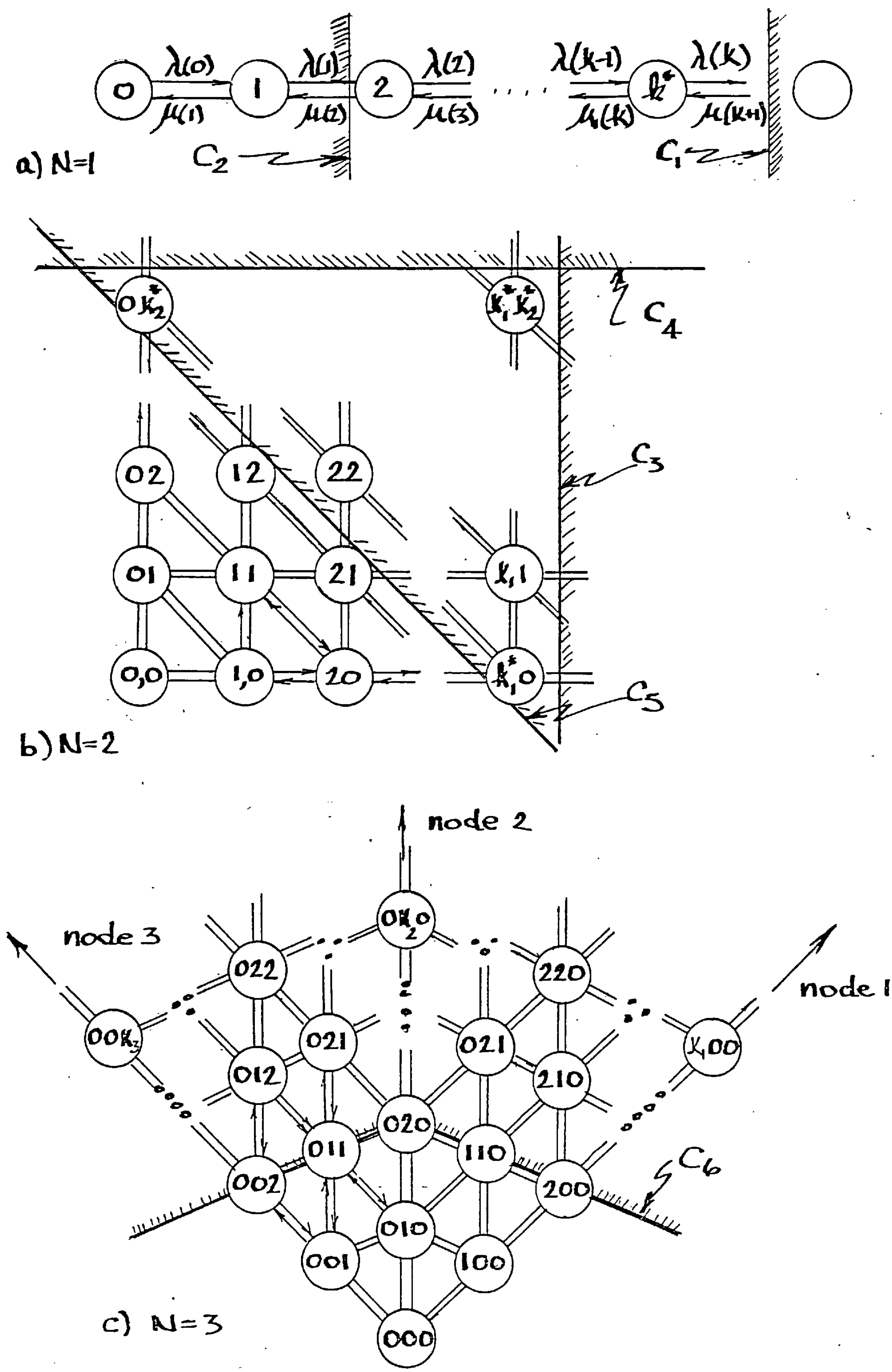


figure 4.5 State transition diagrams (Open Networks)

all $s(\underline{k})$ such that $k_1, k_2 \geq 0$ and

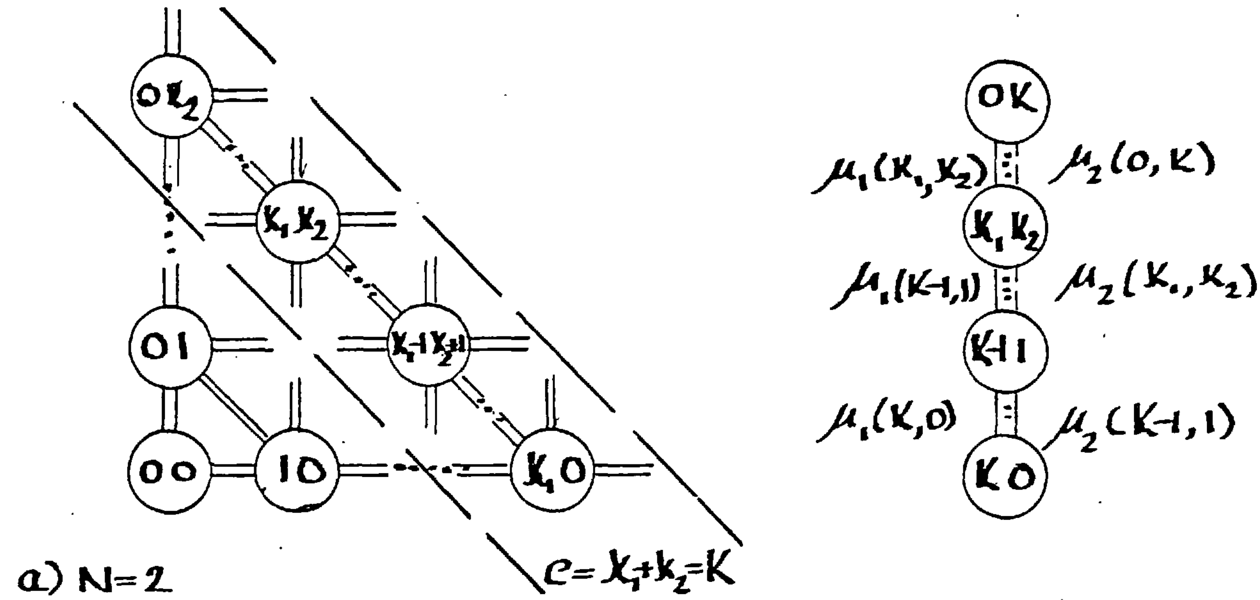
$$\left\{ \begin{array}{lll} C_3 : & k_2 \leq k_2^* & \\ C_4 : & k_1 \leq k_1^* & \\ C_3 : C_4 : & (k_1 \leq k_1^*) \wedge (k_2 \leq k_2^*) & \\ C_5 : & & k_1 + k_2 \leq k_1^* - 1 \end{array} \right\}$$

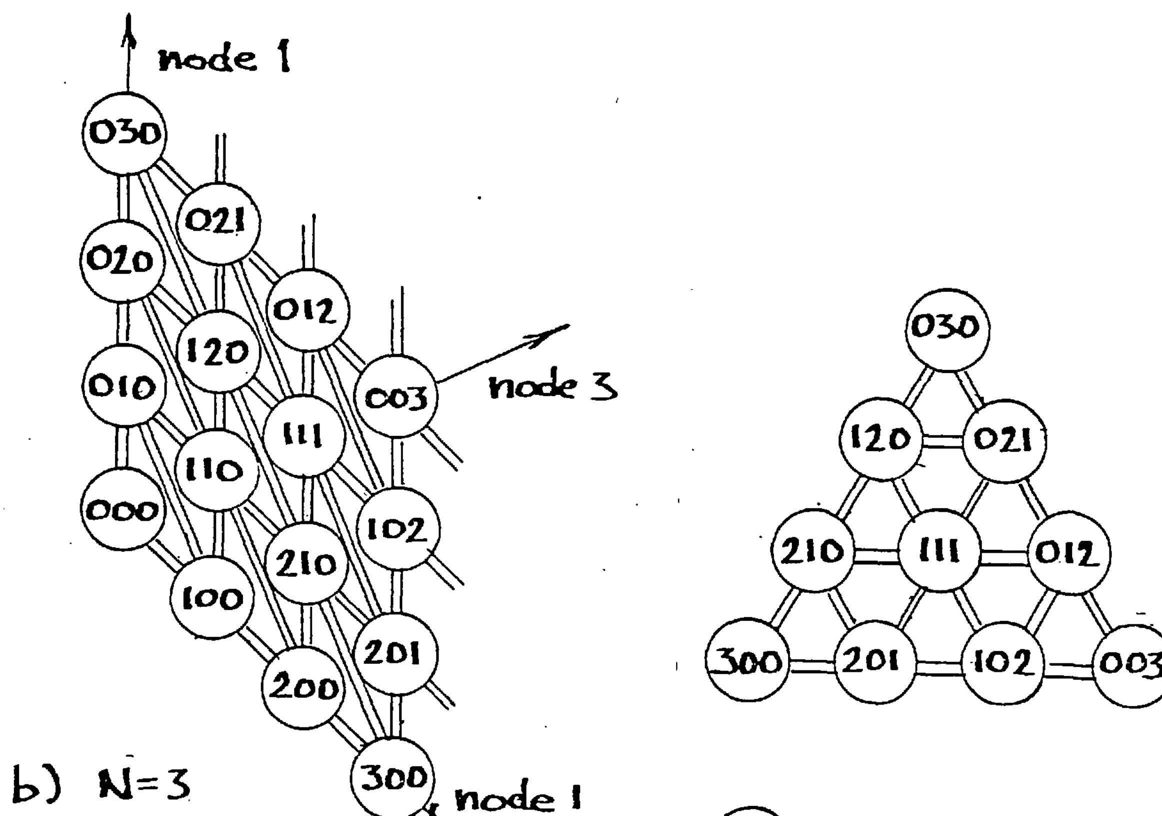
In figure 4.5 c a constraint plane, C_6 cuts the graph so that the population of the system is limited to 2 processes ($K \leq 2$).

In figure 4.6 , the graphs of these networks are displayed with the open graph appearing on the left and the closed network on the right. By introducing two constraints C_1 and C_2 such that for C_1: $\Sigma k_i \leq K$ and C_2: $\Sigma k_i \geq K-1$, the network must have a constant finite population of K processes, which is projected on the next lower dimension in the right hand side of each figure. Note that in general an N-node network is reduced to an N-1 dimensional simplex with K-1 nodes along each base of the simplex, further note the topological equivalence of an N+1 node closed network and an N-node open one.

4.3.3 Disposition of constraints

At this point the crucial question is, how are the state space constraints resolved in the solution of queueing network? There seem to be three basic alternatives: (1) do nothing, i.e. ignore the constraints, (2) allow infeasibilities, appraise the effects or degree of





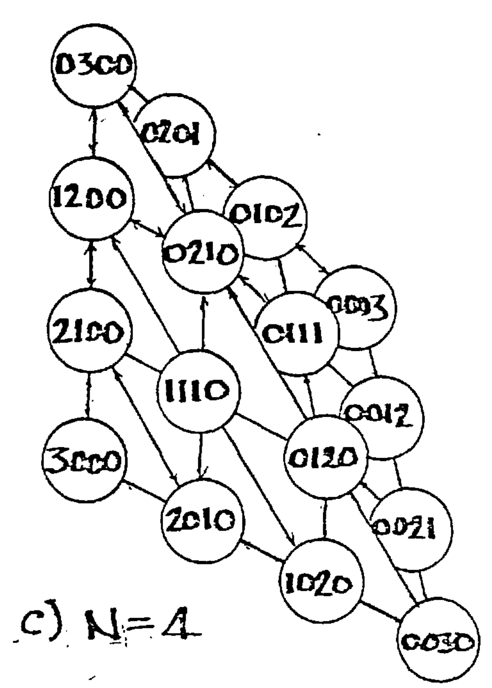


figure 4.6 Closed networks
with constraints

infeasibility and (3) modify the structure of the model such that infeasibilities are prohibited.

The first alternative, to disregard the constraints, amounts to solving the network under ideal conditions. In such cases the model presented is stochastically equivalent to the BCMP network and hence has known solutions. Such behaviour while expedient, is inconsistent with our hypothesis that competition for scarce resources may be crucial to system performance.

The next alternative is to assess the impact of the constraints; this is quite simply done in principle, by solving the unconstrained network by the usual methods and using $p(\underline{k})$ to derive:

(1) the distribution of (passive) resources occupied

(2) the expected resources held

(3) the probability of constraint violation

Ultimately, models are desired which not only provide analytic solutions to weakly constrained networks, but also enforce the (state) constraints implied by resource limitations.

4.3.4 Estimating Constraint Effects

Each resource, active and passive, may have a limit to its queueing (not just service) capacity, these limits being specified by parameters, d_i ,$i \in \underline{N} \cup \underline{N}'$
Also recall that the model allowed the specification of resource demand functions $g_i(\underline{k})$. If $\underline{F}$ is the set of feasible state and $\underline{I}$ is a set of infeasible states then

$$\left\{\begin{array}{ll} \underline{k} \in \underline{F} & g_i(\underline{k}) \leq d_i \\ \underline{k} \in \underline{I} & \text{otherwise} \end{array}\right\} \quad i \in \underline{N} \cup \underline{N}' \tag{4.30}$$

$$\text{where } \underline{F} \cup \underline{I} = \underline{S} \tag{4.31}$$

If $p(\underline{k})$ is the solution to the unconstrained network $(\underline{k} \in \underline{S})$ then the probability that the network is in an infeasible state is

$$\sum_{\underline{k} \in \underline{I}} p(\underline{k}) \tag{4.32}$$

and the expected demand on resource i is

$$\sum_{\underline{k} \in \underline{S}} g_i(\underline{k}) p(\underline{k}) \tag{4.33}$$

the probability that the demand for the ith resource exceeds capacity is

$$\sum_{\substack{g_i(\underline{k}) > d_i \\ \underline{k} \in \underline{S}}} p(\underline{k}) \tag{4.34}$$

With the above estimates it may be possible to test the adequacy of the ideal model. If the constraints are never or rarely exceeded then there is no need to attempt a solution of the much more complex joint state-dependent model. However if the violations are judged significant, then it is necessary to pursue a solution method which enforces the system constraints.

4.4 A Limited Storage Example

To illustrate the concept of finite passive resource limitations in a queueing network, a simple example is presented. Its purpose is to demonstrate that, not only do such models yield quantitative results, they also provide insight into the behaviour of constrained systems..

4.4.1 The System

Consider a small system consisting of three resources:

(1) A 2 channel CPU processor } active
(2) A single I/O processor }
(3) A storage module } passive

The workload of the system consists of two types of processes. Furthermore the processes require a minimum amount of storage in order to obtain service from the CPU processor; if storage is unavailable then the process must wait for storage to be released and blocks service of a processor channel.

4.4.2 The Model

In figure 4.7 a cyclic network consisting of two nodes with $K_1 + K_2 = K$, processes is shown. Assume

(1) node 1 is a 2 way *multiserver* with parameter μ_1 for each process type

(2) node 2 is a single server with parameter μ_2

(3) node 3 is passive; processes of type ℓ demand s_ℓ units of storage when entering service at node 1; it has a capacity of d_3 so that

$$g_3(\underline{k}) = \sum_{\ell=1}^{2} k_{1,\ell}\, s_\ell \le d_3$$

4.4.3 Evaulation

Let $L = 2$;	$K_1 = 2$;	$K_2 = 1$	processes
$s_1 = 1$;	$s_2 = 2$;	$d_3 = 2$	storage units
$1/16 \le \mu_1 \le 8$;		$\mu_2 = 1$	processor units

This model satisfies the MQN conditions and is solved algebraically (Appendix A) for both the constrained and unconstrained models. Comparative system thruputs appear in figure 4.8. The expected number of blocked processes and the mean number waiting at node 1 are shown in figures 4.9 and 4.10, respectively.

Observe that for very fast node 1 processing (relative to node 2), the expected number of blocked processes is very small - an expected result. Also note that as node 1 slows down, blocking becomes more prominent. The surprising result is that the blocking reaches a peak and then begins to diminish as the service rate of node 1 slows (mystery 1).

This remarkable outcome may be reconciled by comparing the waiting and blocking for processes at node 1. *Waiting* (ordinary queueing) results from the finite processing capacity at node 1 (capacity constraint) while *blocking* is a manifestation of the passive-resource constraint. As node 1 slows, one would expect the number of processes waiting to increase. In figure 4.10, above observe that waiting is indeed consistent with our hypothesis for type 2, but again note the surprising result for type 1 processes- an eventual *decrease* in waiting (mystery 2).

These mysteries are resolved by recalling that there are two type 1 processes which may share the passive resource, but type 2 must have the entire resource to proceed. When node 1 becomes much slower than node 2, processes of type 1 depart and re-arrive at node 1 before its companion process finishes hence pre-empting process 2. Thus processes of type 1 will experience less waiting or blocking because its companion process acts as a '*place-holder*' (mystery 2 resolved). While a process of type 1 is place-holding, type 2 is blocked; but as soon as type 1 re-arrives, type 2 is no-longer blocked but waiting. Hence as the node gets ever faster the duration of place-holding

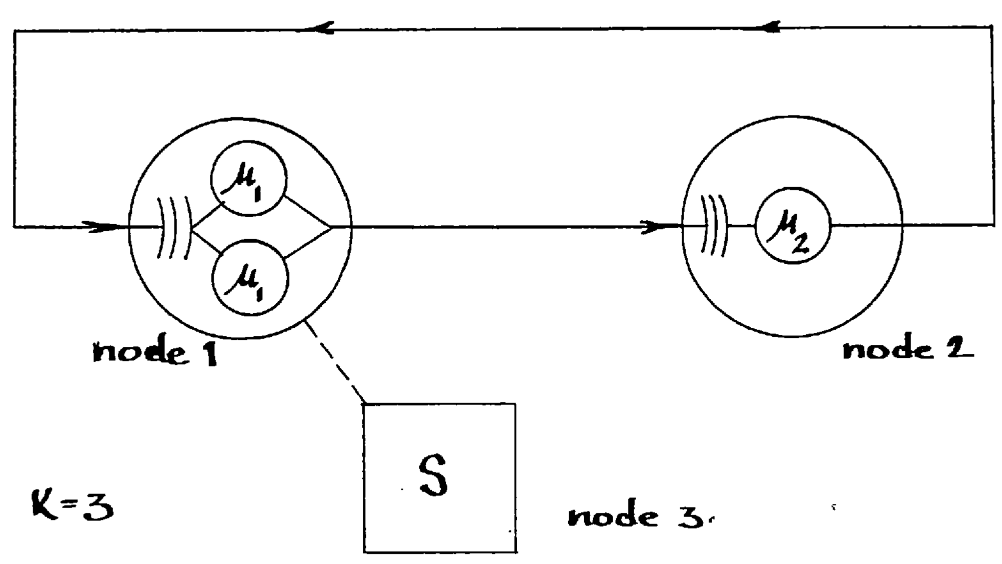


figure 4.7 2-node cyclic network (2 classes)

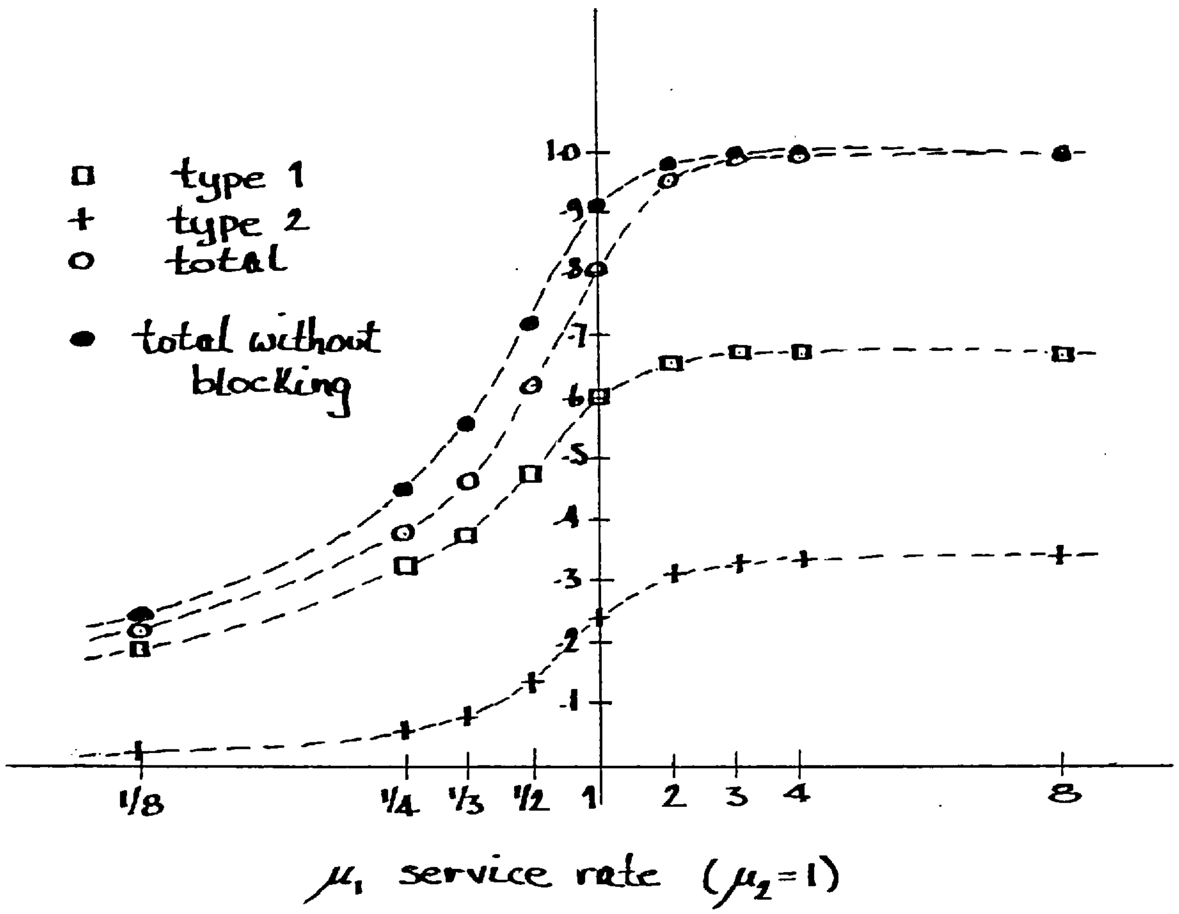


figure 4.8 Comparative Thruputs

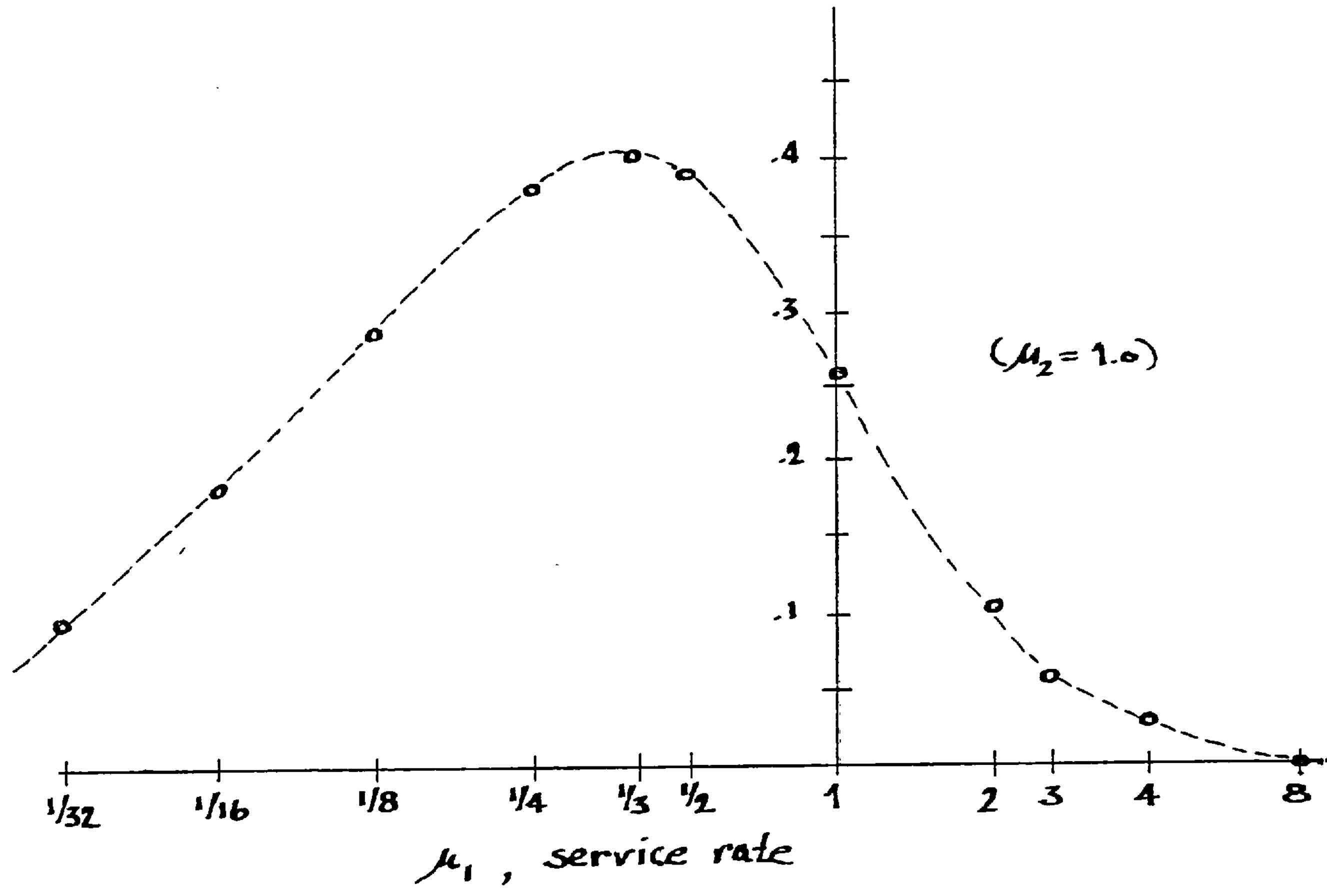


figure 4.9 Total expected blocked processes

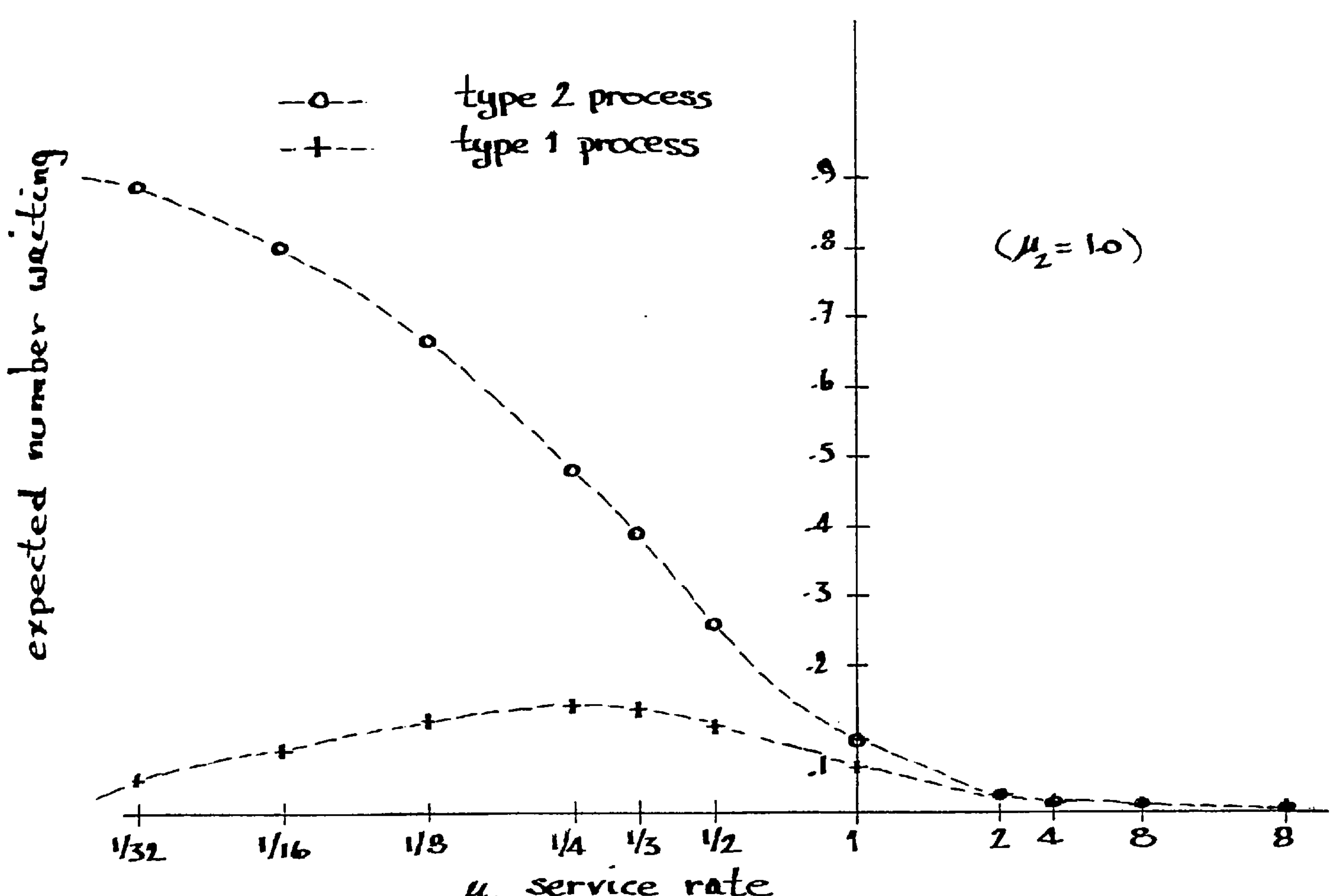


figure 4.10 Expected number of waiting processes at node 1

becomes smaller and hence blocking vanishes (mystery 1 resolved). In the limit ($\mu_1 \to \infty$), one would expect no blocking whatever, no waiting of type 1, and type 2 would never be serviced (i.e. waiting time $\to \infty$).

4.4.4 Conclusion

The above example, and its rather protracted explanation, suggests some of the value of state-dependent queueing models - even for a very simple case, counter-intuitive behaviour was predicted and subsequently explained. Such predictions would obviously be of profound importance in the design, installation and maintenance of computing systems wherein complex sharing of finite capacity storage and data objects may occur.

These results are important insofar as we are now able to make *quantitative* statements about the performance of the system commensurate with (passive) storage resource effects. Yet the solution method used in this example was mostly ad hoc and not easily generalised.

In the next chapter, models of constrained networks with more general *solutions* are presented; however these models are necessarily limited in scope. Nevertheless MQN assumptions remain valid so that it is usually possible to model the constrained network, if not solve it.

CHAPTER 5

C O N S T R A I N E D   N E T W O R K   M O D E L S : B L O C K I N G   A N D   S K I P P I N G

While the method of 4.3.4 may usefully estimate the degree of infeasibility, it yields no information whatsoever about the effects of constraint violation on the performance of the network. To more accurately predict performance variation induced by changes in the (quality and quantity of) system resources, models which explicitly maintain state space feasibility are required.

In terms of the Markovian queuing network, one requires (steady-state) solutions such that the probability of dwelling in an infeasible state is nil. We postulate that this is accomplished in two different ways: either (1) the infeasible states may be bypassed (instantly passed through) whenever they are encountered, or (2) transitions to infeasible states are simply prohibited. In this thesis, these two phenomena are called <u>skipping</u> and <u>blocking</u>, respectively.

5.1 Skipping and Blocking

5.1.1 Skipping

Skipping is effected by forcing an instantaneous transition *through* an infeasible state. By greatly increasing the service rate of the appropriate node, service, hence transition, becomes instantaneous. That is if a transition to a node, say i, would lead to an infeasible state, $\underline{K}\varepsilon\underline{I}$, then the occurence of that infeasibility can be effectivly eliminated i.e., $p(\underline{k}')\to\emptyset$ by letting $c_i(\underline{k}')\to\infty$.

Skipping phenomena occur in many queuing models under different names. "Customer lost" (KLEI75) wherein a customer departs without service if the service centre is full is one example.

Subsequently, it will be shown that skipping problems have very simple, if not useful, solutions.

5.1.2 Blocking

Blocking*, while quite common as a real phenomenon has received little attention in queuing network models -

*the term 'blocking' as it appears in the literature, refers to networks which have nodes of finite customer capacity - our use of the word is a generalisation of this phenomenon since *physical* capacity is considered a passive resource.

primarily due to the lack of general (even particular) solutions. Previous investigators have taken two paths: (1)providing methods for compactly or automatically building the *balance equations* (GRAU75, GORD67a, HILL67) or (2) providing analytic results for tandem queues (2 node networks) (KOBA77, NEUT68). Even tandem queue results are exceedingly complex and very difficult to apply.

One significant complication in the specification and analysis of blocking problems is the need to define the order in which blocked processes 'unblock' if more than one blocked node has processes seeking entry to the same blocking node (e.g. first blocked, first served). Add to this further complexities in describing how service is suspended at the blocked node (e.g. Halt immediate, finish current processes, etc) and the possible alternative routing strategies(such as having a secondary routing if the primary one is blocked),then the problem may become too complex for compact solutions. These alternative queuing disciplines and routings we regard as *scheduling* problems and are beyond the scope of this work.

Blocking conditions arise in many ways in computer system models. One common example is the blocking of storage devices when its data channel is busy, in fact any processing where more than one service node is simultaneously required.An example appears in terminal oriented systems containing two nodes - one dispatching processes and another servicing them;

in this case the processing node may have a finite capacity (e.g. limited degree of multiprogramming). Devices with finite buffers are further examples of blocking; when the buffers fill they often inhibit the sending node (there are numerous applications dealing with communications processors). But probably the most common occurence of blocking is where a processor (node) is inhibited or blocked due to limitations of some passive resource such as storage media or shared data objects (such as the example given in 4.4.3).

5.1.3 Markovian Blocking

At this point a simplification is introduced in order that the complex scheduling and service resumption algorithms associated with general blocking phenomena may be disregarded. In the rest of this thesis, only a special type of blocking called *Markovian blocking*, is considered: Markovian blocking is defined accordingly:

> Any node whose service completion would result in a transition to an infeasible state has its service *immediately* suspended. Such a node is said to be blocked; service is instantaneously resumed when the potential infeasibility is removed.

This definition allows for a very simple representation of blocking: namely that the departure rates of nodes which induce transitions to infeasible states, approach zero.

5.1.4 Joint State Dependent Representations of Blocking and Skipping

Given the definition of skipping and blocking, their representation in MQN follows immediately If k' is an infeasible state(s) and $\underline{k}''$ is a subset of states with defined transitions to infeasible states, then

(1) for *skipping* $c_i(\underline{k}') \to \infty$ $\quad i \varepsilon \underline{N}$
where i is any node for which an arrival induces a transiton to $\underline{k}' \varepsilon \underline{I}$.

(2) for *Markovian blocking* $c_i(\underline{k}'') \to 0$ $\quad i \varepsilon \underline{N}$ (5.1)
where i is a set of nodes whose service completion would result in a transtion to $\underline{k}'' \varepsilon \underline{I}$

Note that these are joint state dependent service rates* and as such deny the independence assumption of SQN networks; however they still enjoy the MQN assumptions.

5.1.5 Balance Equations

For the following state dependent networks, assume

(1) the network contains N active nodes,
$\underline{N} = \{1,2...N\}$.

(2) the network is *closed* and may have L classes,
$\underline{L} = \{1,2..L\}$, of k_ℓ customers each, $\ell \varepsilon \underline{L}$

*blocking and skipping phenomena need not be discrete binary events. It is easy to conceive of situations whereby the joint state of nodes will cause service rates to diminish or increase without approaching their limiting values. Hence blocking nodes may only be *hindering* while skipping nodes may be cooperating.

(3) each active node may have a state dependent departure rate

$$\mu_{i\ell}(\underline{k}) = c_i(\underline{k})/w_{i\ell} \qquad i \in \underline{N},\ \ell \in \underline{L}$$

where $w_{i\ell}$ are mean demands with *exponential SRD's* and $c_i(\underline{k})$ is the JSD service rate of node i.

(4) with routings denoted $q_{i\ell;jm}$ $\quad i,j\in\underline{N} \quad \ell,m\in\underline{L}$ where the routing matrix Q defines a two dimentional Markov chain. It is assumed that the set $\underline{S}$ of all possible states can be partioned into J closed subsets such that each subset is aperiodic and that each state in $\underline{S}_J$ can communicate with each other. That is, the ergodic conditions.

(5) the network state space $\underline{S}$ consists of all $\underline{k}$ where $\underline{k} \overset{\Delta}{=} \{\underline{k}_1\ \underline{k}_2\ ..\ \underline{k}_i\ ..\ \underline{k}_N\}$ and

$$\underline{k}_i \overset{\Delta}{=} \{k_{i1} k_{i2}\ ..\ k_{i\ell}\ ..\ k_{iL}\}$$

$$k_{i\ell} \quad \emptyset \quad i\in\underline{N},\ell\in\underline{L}$$

With these assumptions, the network is a MQN and has global balance equations:

$$p(\underline{k}) \sum_{j\in\underline{N}} \sum_{m\in\underline{L}} \mu_{jm}(\underline{k}) = \underbrace{\sum_{j\in\underline{N}} \sum_{m\in\underline{L}} \sum_{i\in\underline{N}} \sum_{\ell\in\underline{L}}}_{\underline{k}_{jm;i\ell}\ \in\underline{S}} \mu_{i\ell}(k_{i\ell} + 1) q_{i\ell;jm} p(\underline{k}_{jm;i\ell}) \qquad \underline{k}\in\underline{S} \tag{5.2}$$

where $\quad k_{jm;i\ell} = \{k_{11}..k_{i\ell}+1\ ..\ k_{jm}-1\ ..k_{NL}\}$

This system of linear equations with the normalising condition $\sum_{\underline{k} \varepsilon \underline{S}} p(\underline{k}) = 1$, has a unique solution for the unconstrained case (KLEI75, p.52).

This system of equations is valid for blocking and skipping conditions as defined in 5.1; so that, in principle, any *Markovian* blocking problam can be treated by direct subsitution into (5.2) and solving the linear system.

With the exception of very small problems (such as the example in 4.4.3), direct solutions of (5.2) are unmanageable. In the subsequent sections, four models of blocking or skipping are presented which, depending on assumptions and conditions, provide compact product form solutions to the balance equations (5.2).

5.2 A Skipping Model

Recall that skipping is the instantaneous expulsion by a node of any arrival which induces an infeasible state. Furthermore, the parametric interpretation is $c_i(\underline{k}') \to \infty$ for $\underline{k}'$ infeasible. If the i^{th} node is blocking the arrival, it suffices that $c_i(\underline{k}_i') \to \infty$ to relieve the condition.

For notational convenience, assume the network of 5.1.5 but with *homogeneous population* so that balance equations (5.2) become

$$p(\underline{k}) \sum_{j \varepsilon \underline{N}} \frac{c_j(k_j)}{w_j} = \underbrace{\sum_{j \varepsilon \underline{N}} \sum_{i \varepsilon \underline{N}}}_{\underline{k}_{ji} \varepsilon \underline{S}} \frac{c_i(k_i+1)}{w_i} q_{ij}\, p(\underline{k}_{ji}) \quad \underline{k},\ \underline{k}_{ij} \,\varepsilon\, \underline{S} \qquad (5.3)$$

where $\underline{k}_{ji} = \{\ldots k_i+1 \ldots k_j-1 \ldots\}$

THEOREM 5.1 (Skipping)

If the MQN is a skipping problem, i.e. $\underline{k}'$ infeasible => $c_i(k_i') \to \infty$, $i \,\varepsilon\, \underline{N}$, then the solution has product form and is the ordinary Separable Network solution renormalised about the feasible states. Or

$$p(\underline{k}) = \left\{ \begin{array}{ll} G^{-1} \prod_{i \varepsilon \underline{N}} \beta_i(k_i)(e_i w_i)^{k_i} & \underline{k} \,\varepsilon\, \underline{F} \\ & \\ \emptyset & \underline{k} \not\varepsilon\, \underline{F} \end{array} \right\} \qquad (5.4)$$

where $$G = \sum_{\underline{k} \varepsilon \underline{F}} \prod_{i \varepsilon \underline{N}} \beta(k_i)\ (e_i w_i)^{k_i} \qquad (5.5)$$

and e_i is the assummed solution to the system

$$e_i = \sum_j e_j q_{ji} \qquad i \in \underline{N} \tag{5.6}$$

and
$$\beta_i(k_i) = \prod_{j=1}^{k_i} 1/c_i(j) \qquad i \in \underline{N} \tag{5.7}$$

PROOF:

Assume local balance, i.e., for each node $j\varepsilon\underline{N}$

$$p(\underline{k})\frac{c_j(k_j)}{w_j} = \sum_{\substack{i\varepsilon N \\ \underline{k}_{ji}\varepsilon\underline{S}}} \frac{c_i(k_i+1)}{w_i} q_{ij}\, p(\underline{k}_{ji}) \qquad \underline{k},\underline{k}_{ji}\varepsilon\ \underline{S} \tag{5.7.1}$$

then
$$\sum_{i\,\varepsilon\underline{N}} \frac{p(\underline{k}_{ji})}{p(\underline{k})}\ \frac{w_j}{w_i}\ \frac{c_i(k_{i+1})}{c_j(k_j)} = 1 \qquad j\ \varepsilon\ \underline{N} \tag{5.8}$$

substituting (5.4) into (5.8) yields (5.6) if

$$\frac{\beta_j(k_{j-1})}{\beta_j(k_j)} = c\ (k_j) \text{ and } \frac{\beta_i(k_{i+1})}{\beta_i(k_i)} = 1/c_i(k_i) \tag{5.9}$$

$$i,j\ \varepsilon\ \underline{N}$$

there are four cases to verify for each $j\ \varepsilon\ \underline{N}$

$$\begin{array}{llll} (1) & \underline{k}\ \varepsilon\ \underline{F} & \forall_i \underline{k}_{ji}\ \varepsilon\ \underline{F} & \\ (2) & \underline{k}\ \varepsilon\ \underline{F} & \exists_i \underline{k}_{ji}\ \not\varepsilon\ \underline{F} & \qquad (5.10) \\ (3) & \underline{k}\ \not\varepsilon\ \underline{F} & \forall_i \underline{k}_{ji}\ \varepsilon\ \underline{F} & \\ (4) & \underline{k}\ \not\varepsilon\ \underline{F} & \exists_i k_{ji}\ \not\varepsilon\ \underline{F} & \end{array}$$

in each of these cases $c_\ell(k_\ell) \underset{\text{infeas}}{\to} \infty \Rightarrow \beta_\ell(k_\ell) \to 0$ and consequently by 5.4 $p(\underline{k}) \to 0$ $\underline{k}\ \varepsilon\ \underline{F}$

There is no reason that this result cannot be extended to the BCMP result, so that renormalisation over feasible states seems to be the correct interpretation for skipped service.

REMARKS

This result implies that constrained networks which may be rationalised by skipping are easily solved by generating known SQN solutions, forcing the illegitimate state probabilities to zero and renormalising. In fact one can now interpret a closed network as one which is normally open except that the system is infeasible when the networks population is not equal to its closed population value. In such instances, a node or subnet of nodes are skipped such that the network remains feasible (its population equal to the closed population parameter). The normalising constant merely reflects the adjustment necessary to make the probability function proper.

Although skipping corrects the state space, it does so artificially with respect to resource service. Networks with skipping appear to have improved throughput and reduced delays. Hence for most networks* skipping is an incorrect interpretation of the resource service.

We now consider the more interesting case of Markovian blocking (c.f. 5.1.3). Three models are introduced: the first subscribes joint service rates to each node in the network, the second considers blocking *gates* and the last considers state dependent *routings*.

*but not always-the skipping solution is equivalent to the blocking solution for a cyclic two-node network (GORD67a).

5.3 Blocking Model I: Joint Service Rates

It should be apparent that the form of the global equations allows for complete specification of parameters over *all* states. It is therefore possible in principle to solve the blocking problem for any combination of blocked states. Even if one could find symbolic (even numeric) solutions for a non-trivial network, the specification task would be enormous.

A less ambitious goal is to allow service rate functions for each node which depend only on their own state and each of the other nodes pairwise, i.e.,

$$c_i(k_i, k_j) \quad i,j \in \underline{N}; \; i \neq j \tag{5.11}$$

For this model assume only homogeneous networks; then the global balance equations, (5.2), become

$$p(\underline{k}) \sum_{j \in \underline{N}} \frac{c_j(k_j, k_i)}{w_j} = \sum_{\substack{j \in \underline{N} \\ \underline{k}_{ji} \in \underline{S}}} \sum_{i \in \underline{N}} \frac{c_i(k_i+1, k_j-1)}{w_i} q_{ij} \, p(\underline{k}_{ji}) \tag{5.12}$$

so for this model we have:

THEOREM 5.2 (Blocking, Joint Service Rates)

A homogeneous network with joint service rates, 5.11, has a solution given by

$$p(\underline{k}) = G^{-1} \prod_{i \in \underline{N}} \beta_i(k_i)(e_i w_i)^{k_i} \tag{5.13}$$

where G, and e are the same as in (5.5) and (5.6) and $\beta_i(k_i)$ are determined by:

$$\beta_i(k_i+1) = \beta_i(k_i)\ f_{iN}\ (k_i,1) \qquad i \in \underline{N}, i \neq N \quad (5.14)$$

$$\beta_N(k_N+1) = \beta_N(k_N) f_{jN}(0,1) f_{Nj}(k_N 1) \text{ any } j \neq N \quad (5.15)$$

$$\beta_i(0) = 1 \qquad i \in \underline{N}, \quad (5.16)$$

$$\beta_N(1) = 1 \quad (5.17)$$

where $$f_{ij}(k_i,k_j) \triangleq \frac{c_j(k_i,k_j)}{c_i(k_i+1,k_j-1)} \quad (5.18)$$

and $c_j(k_j\ k_i)$ are subject to the constraints,

$$f_{ij}(k_i,k_j+1)\ f_{jN}(k_j,k_N+1)\ f_{Ni}(k_N,k_i+1) = 1 \quad (5.19)$$

$$i,j = 1,2..N-1 \quad i \neq j$$

PROOF: Using a similar substitution used to derive (5.9).

$$\frac{\beta_i(k_i+1)\ \ \beta_j(k_j-1)}{\beta_i(k_i)\ \ \beta j(k_j)} = \frac{c_i(k_i k_j)}{c_i(k_i+1,k_j-1)} \triangleq f_{ij}(k_i,k_j) \quad i,j \in \underline{N} \quad (5.20)$$

note the identity

$$f_{ij}^{-1}\ (k_i,k_j) = f_{ji}(k_i+1,k_j-1) \quad (5.21)$$

to prove the result, it must be shown that (5.14) with constraints (5.19) satisfy (5.21). This is done by substitution as follows:

<u>For $i \neq N$</u>

substituting 5.14 the left hand side of 5.20 becomes

$$f_{iN}(k_i,k_N+1)\ f_{jN}^{-1}(k_j-1,k_N+1) = f_{iN}(k_i,k_N+1)\ f_{Nj}(k_j,k_N) \quad (5.22)$$

where the identity 5.21 has been used. Shifting (5.19) and applying the identity (5.21) evaluates the right hand side of expression (5.20)

$$f_{jN}^{-1}\ (k_j-1,k_N+1)\ f_{Ni}^{-1}\ (k_N,\ k_i+1)\ =\ f_{iN}(k_N+1,k_i)f_{Nj}(k_jk_N) \quad (5.23)$$

so that from (5.22) ;(5.23), (5.20) is proved.

Similar substitution of (5.14) and (5.15) into (5.20) and using conditions (5.16):(5.17) prove the result for i = N.

Note that N can be an arbitrary node in the network and j any *other* node in the network.

COMMENT

The above result is, in some sense, a generalisation of the Gordon-Newell result (GORD67). If the joint state dependent service rates are restricted to local state dependency, i.e.

$$c_i(\underline{k}) = c_i(k_i) \quad i \in \underline{N}$$

then $f_{ij}(k_ik_j) = \dfrac{c_j(k_j)}{c_i(k_i+1)}$ and it is easily verified that the constraints (5.19) are always satisfied so that

$$\beta_i(k_i+1) = \beta_i(k_i)/c_i(k_i+1)$$

$$\beta_i(0) = 1$$

which is identical with the GN result.

If it were not for the embarassing set of constraints (5.19), this result appears to be very useful. Unfortunately the constraints effectively eliminate all but very trivial problems. These constraints arise for two reasons. First they guard against inconsistent specification of joint service functions, and secondly they suggest that product forms 5.13 with general routing do not even exist.

The conclusion is that only in very unusual circumstances will blocking problems have representations satisfying the conditions of Theorem 5.2. It is this conclusion which prompts us to look for particular routings which have product form solutions.

5.4 Blocking Model II: Blocking Gates

Consider the MQN described by global balance equations (5.2) except that we limit the network to a homogeneous population of K customers with service rates $c_i(k_i)$. In addition define *admittance* rate functions, $b_j(k_j)$ which is the rate at which the j^{th} node will admit customers. So for blocking

$$b_j(k_j) \triangleq \begin{Bmatrix} 1 & k_j < k_j^* \\ 0 & k_j \geq k_j^* \end{Bmatrix} \tag{5.24}$$

These act as gates prohibiting entry to node j when it has a maximum capacity k_j^*.

For this model the state transition rate from node i to node j is

$c_i(k_i)b_j(k_j)q_{ij}$ and the balance equations are:

$$p(\underline{k}) \sum_{i\varepsilon\underline{N}} \sum_{j\varepsilon\underline{N}} q_{ij}\frac{c_i(k_i)}{w_i} b_j(k_j)$$

$$= \sum_{\substack{i\varepsilon\underline{N} \\ \underline{k}_{ji}\varepsilon\, \underline{S}}} \sum_{j\varepsilon\underline{N}} q_{ij} \frac{c_i(k_i+1)}{w} b_j(k_j-1)p(\underline{k}_{ji}) \qquad \underline{k}\varepsilon\underline{S} \tag{5.25}$$

For this model, we have

THEOREM 5.3 (Blocking Gates)

For a homogeneous network of exponential servers with blocking gates $b_j(k_j)$ and balance equation (5.25), the solution is

$$p(\underline{k}) = G^{-1} \prod_{i\varepsilon\underline{N}} \beta_i(k_i)(e_i w_i)^{k_i} \qquad \underline{k} \in \underline{S} \quad (5.26);$$

where G is the usual normalising constant

$$\beta_i(k_i) = \prod_{r=1}^{k_i} b_i(r-1)/c_i(r) \qquad (5.27)$$

we seek e's such that

$$\sum_{i\varepsilon\underline{N}} \sum_{j\varepsilon\underline{N}} (q_{ij}-q_{ji}e_j/e_i)\ c_i(k_i)b_j(k_j) = 0 \qquad \underline{k} \in \underline{S} \quad (5.28)$$

which are the necessary conditions for solution (5.26) to the balance equations (5.25).

PROOF: Again, by substitution (5.26) into the balance equations (5.25)

$$\sum_i \sum_j q_{ij} \frac{c_i(k_i)}{w_i} b_j(k_j) = \sum_j \sum_i q_{ji} \frac{e_j}{e_i w_i} c_i(k_i)b_j(k_j) \qquad (5.29)$$

$$\underline{k} \in \underline{F}$$

rearranging and noting that w_i is a non-zero scaling constant, we may redefine c_i , so that

$$\sum_i \sum_j h_{ij}\ c_i(k_i)\ b_j(k_j) = 0 \qquad (5.30)$$

where $h_{ij} = q_{ij} - q_{ji}e_j e_i^{-1}$ Q.E.D. (5.31)

COROLLARY 5.3.1

If there is no blocking, then the solution is the ordinary SQN (or GN) solution.

PROOF: $b_i(k_i) = 1$, for all k_i, so that (5.30) becomes

$$\Sigma\, c_i(k_i) \sum_j h_{ij} = 0 \tag{5.32}$$

Since all customers may be at one centre, $i\varepsilon\underline{N}$ say, so that $c_j(k_j)=0$ for $k_j=0$ $i\neq j$, it is necessary that

$$\sum_j h_{ij} = 0 \quad i \varepsilon \underline{N} \tag{5.33}$$

or

$$e_i = \sum_{j\varepsilon\underline{N}} e_j\, q_{ji} \tag{5.34}$$

which are the GN visitations Q.E.D.

DEFINITION: If the routing matrix is specified such that

$$e_i q_{ij} = e_j q_{ji} \qquad i,j\ \varepsilon\ \underline{N} \tag{5.35}$$

then the network is said to be *reversible**

COROLLARY 5.3.2

A reversible network, with or without blocking, satisfies condition (5.30) and therefore has product form solution (5.26).

PROOF: From expression (5.35) a reversible network has

$$h_{ij} = 0 \qquad i,j\ \varepsilon\ \underline{N} \tag{5.36}$$

so that (5.30) is always true. This means that any blocking problem having a reversible network has the simple product form solution 5.26. Reversible networks will be further discussed in the next section.

*by analogy with Kendall (KEND59)

COROLLARY 5.3.3

If a network has a finite capacity at all nodes and it is *possible* for each centre to be idle while all other centres are at capacity, call this a completely constrained network; such a network satisfies (5.30) and has visitations given by:

$$\sum_{i \in \underline{N}} c_i h_{ij} = 0 \qquad j \in \underline{N} \qquad (5.37)$$

which, from (5.31), is the linear system,

$$\sum_{i \in \underline{N}} c_i q_{ij} = e_j \sum_{i \in \underline{N}} c_i e_i^{-1} q_{ji} \qquad (5.38)$$

PROOF: by assumption, if node i is idle $c_i(0) = 0$ then all other nodes may be at capacity, i.e.,

$$b_j(k_j) = \begin{Bmatrix} 1 & j = i \\ 0 & j \neq i \end{Bmatrix} \qquad (5.39)$$

Note that these conditions satisfy (5.30). This can be seen by observing that $c_i(k_i) = 0$ removes the i^{th} row of the $\{c_i h_{ij}\}$ matrix and the columns will only sum to zero if all the columns $j = i$ are removed. Q.E.D.

An example of this type of network appears in figure 5.1.

This last result is surprising in that it apparently has product form solution without being separable - there are no local balance equations. However note that it is restricted to state independent service rates.

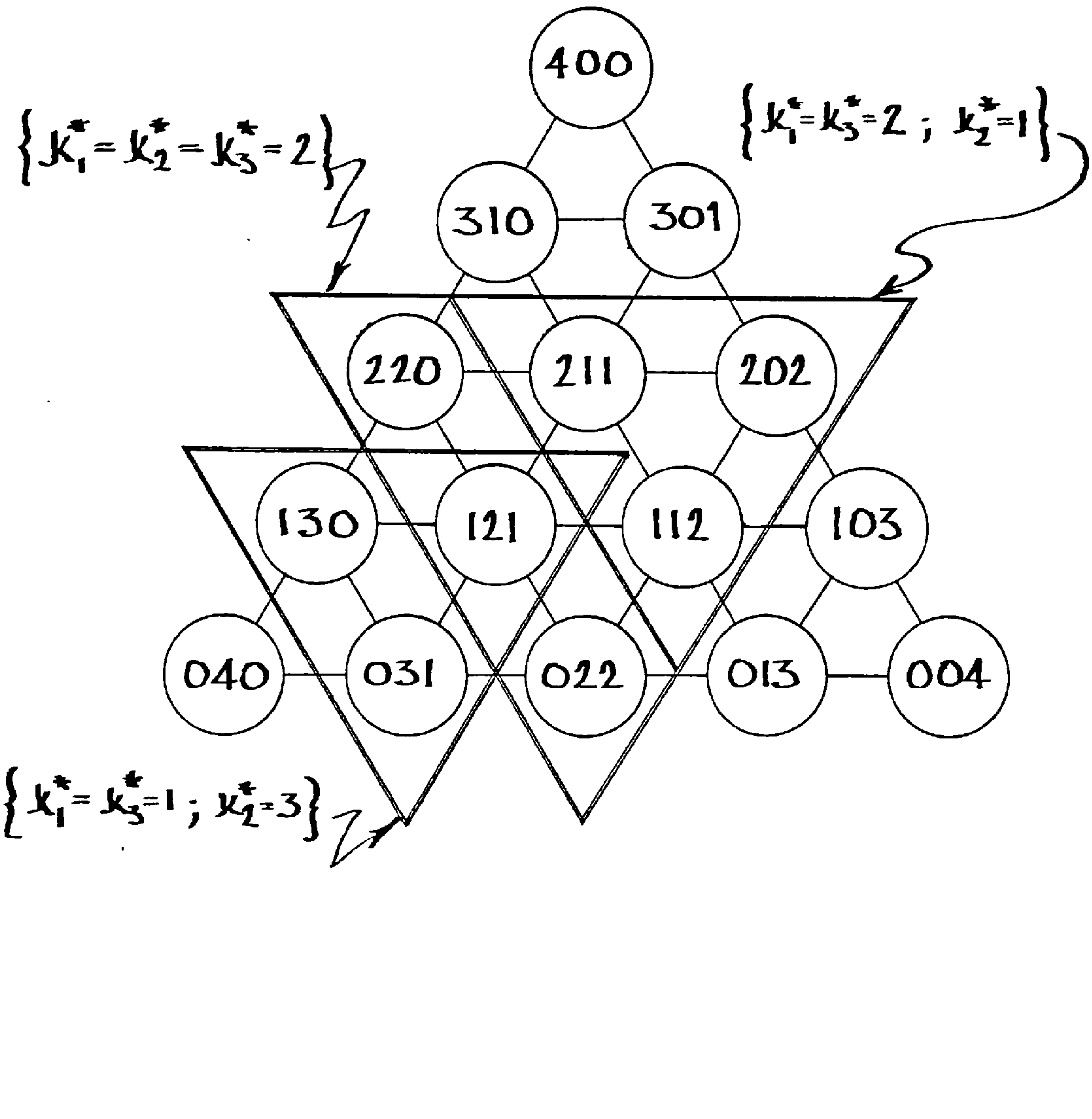


figure 5.1 Example of a *completely* constrained network. (K=4 ; N=3)

Now that there is analytic (and empirical) evidence that routings appear to be significant in the search for product form solutions, a model which has state dependent routing is investigated.

5.5 Blocking Model III: State Dependent Routings

From the previous models, it is increasingly apparent that compact blocking solutions are partially dependent on the routing. In this model, state dependent routings are considered such that, upon service completion, routings resulting in the transition to infeasible states are disallowed. In order to simulate blocking, we assume that the offending customer is re-routed back to the just-departed node.

For this model, the BCMP multi-class network with *exponential* SRD is assumed. For this slightly less general exponential case the departure-rate functions are

$$\mu_{i\ell}(k_{i\ell}) = \left\{\begin{array}{ll} c_i(k_i)/w_i & \text{(FCFS)} \\ k_{i\ell}/k_i w_i & \text{(PS)} \\ k_{i\ell}/w_{i\ell} & \text{(INF)} \\ 1/w_{i\ell} & \text{(LCFS-PR)} \end{array}\right\} \quad \begin{array}{l} i \in \underline{N} \\ \ell \in \underline{L} \end{array} \qquad (5.40)$$

DEFINE $\underline{A}_{i\ell}(\underline{k})$ to be the set of admissable transitions given a service completion of class ℓ at node i in state $\underline{k}$, or

$$\underline{A}_{i\ell}(\underline{k}) \stackrel{\Delta}{=} \{((j,m) \in \underline{N}\times\underline{L}) \text{ and } \underline{k}_{i\ell;jm} \in \underline{F}\} \qquad (5.41)$$

Recall that $\underline{F}$ is the set of feasible states and

$$k_{i\ell;jm} = \{\ldots k_{i\ell}-1, \ldots k_{jm}+1 \ldots\}$$

LEMMA 5.1

If $\underline{k}$ and $\underline{k}_{jm;i\ell} \varepsilon \underline{F}$, then the admissable set given a completion of class m at node j for state $\underline{k}$, then $A_{jm}(\underline{k})$ is identical to $A_{i\ell}(\underline{k}_{jm;i\ell})$, the admissable set for class ℓ at node i at state $\underline{k}_{jm,i\ell}$.

PROOF: from definitions (5.41)

$$A_{i\ell}(k_{jm;i\ell}) \triangleq A_{i\ell}(\{\ldots k_{jm}-1,\ldots k_{i\ell}+1 \;\ldots\})$$
$$\triangleq \{(r,s) \;\varepsilon\; \underline{N}\times\underline{L} \mid \{\ldots k_{jm}-1 \ldots k_{i\ell}+1-1 \ldots k_{rs}+1\ldots\} \;\varepsilon\; \underline{F}\}$$
$$\triangleq A_{jm}(\underline{k}) \qquad (5.42)$$

And trivially true for $(jm)=(i\ell)$ Q.E.D.

Represent the *state dependent* routings:

$$q_{i\ell;jm}(\underline{k}) = \left\{\begin{array}{lll} \emptyset & (j,m) \notin A_{i\ell}(\underline{k}) & \\ q_{i\ell;jm} & & (i,\ell) \neq (j,m) \\ q_{i\ell;i\ell} + \sum\limits_{(r,s)\notin A_{i\ell}(\underline{k})} q_{i\ell;rs} & & (i,\ell) = (j,m) \end{array}\right\} \quad (5.43)$$

where the first expression in (5.43) prohibits a transition if (j,m) is inadmissable, the second term grants the normal transition and the third term adds the sum of the inadmissable routing probabilities to the re-enqueuing routing probability.

For this model the balance equations can be expressed

$$p(\underline{k}) \sum_{j\varepsilon\underline{N}} \sum_{m\varepsilon\underline{L}} \mu_{jm}(k_{jm})$$

$$= \sum_{j\varepsilon\underline{N}} \sum_{m\varepsilon\underline{L}} \sum_{(i\ell)\varepsilon A_{jm}(\underline{k})} \mu_{i\ell}(k_{i\ell}+1)\, q_{i\ell;jm}(\underline{k}_{jm;i\ell})\, p(\underline{k}_{jm;i\ell})$$

$$\underline{k}\ \varepsilon\ \underline{F} \qquad (5.44)$$

with assumed local balance equations

$$p(\underline{k})\ \mu_{jm}(k_{jm}) = \sum_{(i\ell)\varepsilon A_{jm}(\underline{k})} \mu_{i\ell}(k_{i\ell}+1)\, q_{i\ell;jm}(\underline{k}_{jm;i\ell})\, p(\underline{k}_{jm;i\ell})$$

$$\underline{k}\ \varepsilon \underline{F}\ ; j\ \varepsilon\ \underline{N}\ ; \ell\ \varepsilon \underline{L} \qquad (5.45)$$

For this model we offer

THEOREM 5.4

Given the multi-class model of exponential servers of (5.40) with state dependent routings (5.43) and represented by balance equations (5.44), then the network has product form solution

$$p(\underline{k}) = G^{-1} \prod_{i\varepsilon\underline{N}} \beta_i(k_i) \prod_{\ell\varepsilon\underline{L}} (e_{i\ell} w_{i\ell})^{k_{i\ell}} \qquad \underline{k}\varepsilon\underline{F} \qquad (5.46)$$

where

$$\beta_i(k_i) = \left\{ \begin{array}{lr} \prod_{r=1}^{k_i} 1/c_i(r)\ \left[w_{i\ell}=w_i\right] & \text{(FCFS)} \\ k_i!\ \prod_{\ell\varepsilon\underline{L}} 1/k_{i\ell} & \text{(PS)} \\ \prod_{\ell\varepsilon\underline{L}} 1/k_{i\ell} & \text{(INF)} \\ 1 & \text{(LCFS-PR)} \end{array} \right\} \qquad (5.47)$$

and e'_{jm} is given by

$$e'_{jm} = e'_{jm}(q_{jm;jm} + \sum_{rs \notin A_{jm}(\underline{k})} q_{jm;rs})$$

$$+ \sum_{\substack{i\ell \in A_{jm}(\underline{k}) \\ (i\ell) \neq (jm)}} e'_{i\ell}\, q_{i\ell;jm} \qquad j\in\underline{N},\ m\in\underline{L} \quad \underline{k}\in\underline{F} \qquad (5.48)$$

PROOF:

The first part of the proof follows along the same lines as the derivation of the BCMP model with substitutions of (5.46) and (5.47) into (5.45) (these substitutions being similar to that leading to (5.29)). These substitutions yield

$$e'_{jm} = \sum_{i\ell \in A_{jm}(\underline{k})} e'_{i\ell} q_{i\ell;jm}(\underline{k}_{jm;i\ell}) \qquad (5.49)$$

then using the lemma (5.42), reduce the routings (5.43) to

$$q_{i\ell;jm}(\underline{k}_{jm;i\ell}) = \left\{ \begin{array}{ll} q_{i\ell;jm} & (i\ell) \neq (j,m) \\ \\ q_{jm;jm} + \sum_{(rs) \notin A_{jm}(\underline{k})} q_{jm;rs} & (i\ell) = (j,m) \end{array} \right\} \qquad (5.50)$$

(note that $(j,m) \notin A_{jm}(\underline{k})$ is null) substituting (5.50) into (5.49) produces (5.48). Therefore the product form (5.46) with (5.48) satisfies the *local* balance equations, hence the global balance equations, and proves the theorem.

COROLLARY 5.4.1

The visitations, e_{jm}, of the *unconstrained* BCMP network,

$$e_{jm} = \sum_{i\varepsilon \underline{N}} \sum_{\ell \varepsilon \underline{L}} e_{i\ell}\, q_{i\ell;jm} \qquad j \;\varepsilon\; \underline{N},\; m \;\varepsilon\; \underline{L} \tag{5.51}$$

satisfy (5.43) iff,

$$e_{jm} \sum_{rs \not\in A_{jm}(\underline{k})} q_{jm;rs} = \sum_{i\ell \not\in A_{jm}(\underline{k})} e_{i\ell}\, q_{i\ell;jm} \qquad \begin{matrix} j\varepsilon\underline{N} & m\varepsilon\underline{L} \\ \underline{k}\varepsilon\underline{F} & \end{matrix} \tag{5.52}$$

Proof: By definiton (5.51) becomes,

$$e_{jm} = \sum_{(i,\ell)\varepsilon A_{jm}(\underline{k})} e_{i\ell}\, q_{i\ell;jm} + \sum_{(i,\ell)\not\in A_{jm}(\underline{k})} e_{i\ell}\, q_{i\ell;jm} \qquad \underline{k}\varepsilon\underline{F} \tag{5.52.1}$$

$$= e_{jm} \sum_{(r,s)\not\in A_{jm}(\underline{k})} q_{jm;rs} + \sum_{(i,\ell)\varepsilon A_{jm}(k)} e_{i\ell}\, q_{i\ell;jm} \qquad \underline{k}\varepsilon\underline{F} \tag{5.53}$$

on substitution of (5.52). Hence from (5.48), $e_{jm} \propto e'_{jm}$ for all $\underline{k}$. Finally to prove that 5.52 holds, given $e_{jm}=e'_{jm}$, substitute for each of equations (5.48) given (5.53). Then using (5.52.1) in (5.53) we arrive at (5.52). Q.E.D.

COROLLARY 5.4.2

For networks without class *changes*, Corollary 5.4.1 becomes:

$$e_{jm} \sum_{r\not\in A_{jm}(\underline{k})} q_{jr;m} = \sum_{i\not\in A_{jm}(\underline{k})} e_{im}\, q_{ij;m} \qquad j\varepsilon\underline{N},\; m\;\varepsilon\underline{L} \tag{5.54}$$

where $q_{ij;m} \equiv$ the pr $\Big[$ a customer of the type m, having finished service at node i proceeds to node j$\Big]$

PROOF: follows directly from Corollary 5.4.1.

COROLLARY 5.4.3

For networks with *homogeneous* populations, Corollary 5.4.1 becomes:

$$e_j \sum_{r \notin A_j(\underline{k})} q_{jr} = \sum_{i \notin A_j(\underline{k})} e_i q_{ij} \qquad j \in \underline{N} \tag{5.55}$$

PROOF: Follows immediately from Corollary 5.4.1, 5.4.2.

Theorem 5.4 and its companion corollaries reveal conditions on the routings whereby product form solutions of the blocking model are realisable.

In particular these results are valid if the networks are *reversible* (cf. 5.3) (5.35-5.55) are satisfied for reversible networks:

(i) non-homogeneous with class changes

$$e_{jm} q_{jm;i\ell} = e_{i\ell} q_{i\ell;jm} \quad (j,m),(i,\ell) \in \underline{N} \underline{X} \underline{L}$$

(ii) non-homogeneous networks without class changes

$$e_{j\ell} q_{ji;\ell} = e_{i\ell} q_{ij;\ell} \quad i,j \in \underline{N} \quad \ell \in \underline{L}$$

(iii) homogeneous classes (5.56)

$$e_j q_{ji} = e_i q_{ij} \quad i,j \in \underline{N}$$

<u>Comments</u> In this thesis, theorem 5.4 provides the most general results; they are valid for multi-class networks and require only that conditions (5.48) be satisfied for the solution to have the simple SQN solution, (5.46) Regrettably the conditions (5.48) still limit the generality and therefore the applicability of this model.

<u>Reversibility</u> is a severe constraint. Notice that reversibility implies

$$\left\{\begin{array}{lll} q_{jm;i\ell} > 0 & \Rightarrow & q_{i\ell;jm} > 0 \\ q_{ji;\ell} > 0 & \Rightarrow & q_{ij;\ell} > 0 \\ q_{ji} > 0 & \Rightarrow & q_{ij} > 0 \end{array}\right\} \qquad (5.57)$$

therefore, for example, cyclic networks are generally not reversible and the state dependent routing does not apply*. However, it is easily verified that reversibility always holds if the routings are symmetric, i.e.,

$$\left\{\begin{array}{l} q_{jm;i\ell} = q_{i\ell;jm} \\ q_{ji;\ell} = q_{ij;\ell} \\ q_{ji} = q_{ij} \end{array}\right\} \quad i,j \in \underline{N}\ ;\ell,m \in \underline{L} \qquad (5.58)$$

Furthermore, the popular computing system queuing model, the Central Server Model (BUZE71) is also seen to be reversible.

---

*the exception is the 2-node cyclic network which is *always* reversible. This is the reason why this network has yielded simple analytic solutions (GORD67a) while other cyclic networks have not. Inspection of this result reveals that it is identical to the solution presented in (GORD67a).

5.6 Metrics of Blocked Networks

In all of the models of this chapter, the solution forms have product forms which are identical in form to unblocked networks. Yet it is expected that the normal network metrics, i.e., thruput, response time, utilisation will be different from blocked networks (otherwise this work would be pointless).

The difference lies in the interpretation and disposition of the state space variables. This is most evident in networks where blocking and skipping have identical results but the performance of the networks are completely different. This difference is reconciled in the thruput calculations where *intrinsically* a blocked node may have no thruput but still not be empty.

The thruput of blocked networks is defined to be

$$T_i(K) = \sum_{k_i=1}^{K} \mu_i(k_i)\,(p_i(k_i)(1 - B_i(k_i))) \tag{5.59}$$

where $\mu_i(k_i)$ is the usual departure rate (cf. 5.1.5)
$p_i'(k_i)$ is the marginal distribution of centre i

$$p_i(\ell) \triangleq \sum_{\substack{k\varepsilon F \\ \bar{k}_i = \ell}} p(\underline{k}) \tag{5.60}$$

and $B'_i(k_i)$ is the probability that node i is blocked when there are k_i customers at the centre,

$$B'_i(\ell) = \sum_{j\varepsilon\underline{N}} \sum_{\substack{\underline{k}\varepsilon\underline{F} \\ \overline{k}_i=\ell \\ k_j=k_j^*}} p(\underline{k})q_{ij} \qquad (5.61)$$

It is also worth mentioning that the traditional definition of utilisation (1-prob(k_i=0)) is inadequate since a centre may be occupied but not servicing (i.e. blocked). Utilisation definitions can be corrected simply

$$\text{Utilisation}_i \triangleq 1-p_i(0)- B_i \qquad (5.62)$$

where

$$B_i \triangleq \sum_{\ell=1}^{K} B_i(\ell) \text{ is the blocking probability.} \qquad (5.63)$$

5.7 Summary

In this chapter four models have been introduced as simple representations of constrained networks. One of these represents skipping, a by-passing of service to avoid infeasibility. The model proved that this phenomena is simply modelled as an ordinary SQN renormalised about the feasible states.

Three models of blocking were presented. Model I attempted to extend the Gordon-Newell model by considering joint service rates. This model produced an interesting but nearly useless solution due to the emergence of an embarassing set of constraints. Model II also considered a GN network and yielded results whenever the network was reversible or was fully constrained; this latter result is interesting insofar as it produces a product form solution *without* the local balance assumption. Finally a third model which considered state dependent routings yielded results for a multi-class network. These results also have practical use if the networks are reversible.

Mostly the efforts of finding compact blocking solutions have been frustrated by the constant appearance of unwanted condition. Of course there is no reason that such solution forms should exist and perhaps the discovery of even these flawed solutions should be considered good fortune.

CHAPTER 6

C O N C L U S I O N S, O B S E R V A T I O N S, R E F L E C T I O N S

At this point, it is conventional to circumspectly summarise the assumptions, assertions and important results of this work. To this end, we briefly reiterate ...

6.1 ... the thesis restated ...

(1) the performance of future systems is dependent on the *finite capacity* of storage and data objects (the passive resources),

(2) their *performance* effects are to limit the service and/or queuing capacity of the system processors (the active resources),

(3) this leads logically to finite capacity *constraints* on the system,

(4) which may be modelled by constrained *queuing* networks,

(5) such networks may be represented by Markovian queuing networks with blocking and skipping,

(6) these phenomena are modelled by joint state dependent functions.

6.2 ... the results restated ...

(1) finite passive and active resource effects are important theoretical performance constructs, occuring in real systems

(2) they are provocative modelling constructs as shown by example

(3) they are readily adapted to Markovian Queuing networks with appropriate assumptions and state dependent service considerations

(4) a new SQN modelling construct, the multiple server, is introduced, this being useful not only in network reduction but also significant on its own as a performance 'law'.

(5) skipping is shown to have the usual SQN solution renormalised about the feasible states

(6) three models of blocking, all with simple product forms, are introduced; they have the following significance

(a) Model I -(Blocking with joint service rate functions) - they exist but are so constrained that they are of no practical use except for trivial problems.

(b) Model II-(Blocking gates) - solutions have conditions on the visitation rates. These models have immediate practical value if the network is *reversible* or is *fully* constrained.

(c) Model III-(State dependent routing) - provides a solution to multi-class networks. Again conditions on the visitations appear which are satisfied if the network is reversible.

6.3 Other Results

In studying constrained networks, one is immediately confronted with the notion of blocking; a concept which is easily deposited on the MQN substrate. But in trying (perhaps over eagerly) to apply Separable Queuing Network methods in the pursuit of compact product form solutions, we are continually frustrated by the unwanted appearance of special network conditions.

These conditions give rise to doubts about the existence of simple product forms. In fact a simple numerical experiment suffices to show that they do not always exist.

Consider, for example, a simple non-reversible network with 3 nodes, 4 customers and all service parameters unity; simple enough so that the balance equations can be solved numerically. Then hypothesise a product form solution,

$$p(k_1k_2k_3) = G^{-1} e_1^{k_1} e_2^{k_2} e_3^{k_3} \qquad (6.1)$$

or even the more elaborate

$$p(k_1k_2k_3) = G^{-1} e_1(k_1)^{k_1} e_2(k_2)^{k_2} e_3(k_3)^{k_3} \qquad (6.2)$$

Conduct a least squares fit, considering the e's as regression coefficients and the MQN solutions as dependent variables. The results of this experiment are reported in

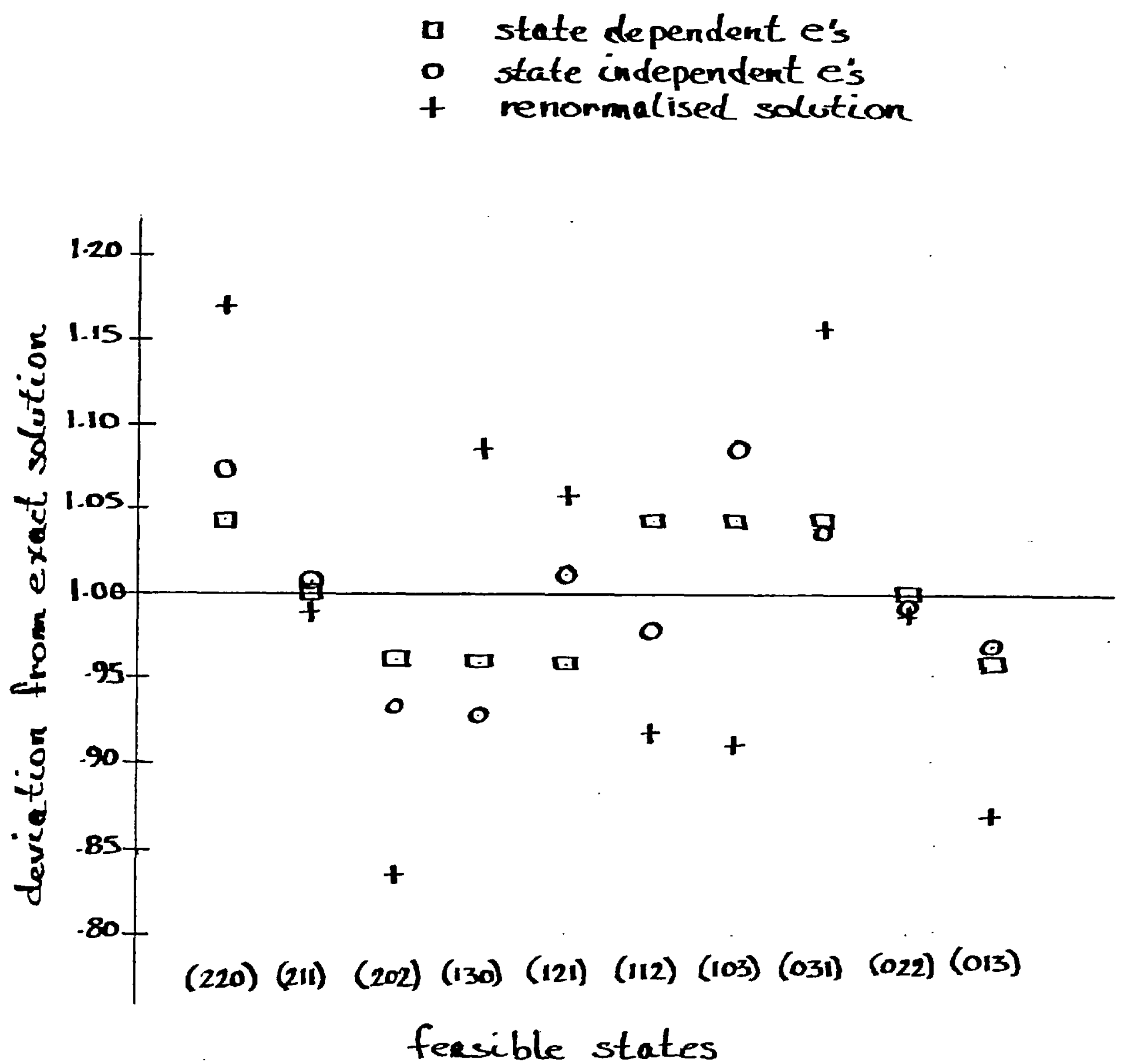


figure 6.1 Comparisons: Product form and exact solutions
($N=3$; $K=4$; $K_1^*=2$, $K_2^*=K_3^*=3$; $\mu_1=\mu_2=\mu_3=1$)

figure 6.1. It is evident that there is no *exact* product form (for *reversible* networks, both product form models coincide with SQN results). This disappointing, but not unexpected result, prompted us to consider alternative solution forms.

6.3.1 Blocked Cyclic Networks: A Special Case

For simple cyclic networks of greater than 2 nodes (recall these are *never* reversible) linear difference equation models representing (probability) flows in the network are analysed.

Consider the 3 server cyclic queue shown in Figure 6.2. This network has a limited queue capacity of two processes ($K_2^* = 2$) at node 2 (the remaining nodes may also be constrained, but this simply cuts the state space and adds no more complexity to the problem).

Two properties of MQN are invoked in the analysis of this network: the first is the conservation of flow (or probability) for an MQN in equilibrium- and the second is a direct consequence of the first, that the flow out of any state must be related by a simple ratio of the service parameters to any other outflow from the same state. The state space and implementation of the second property are produced in figure 6.3.

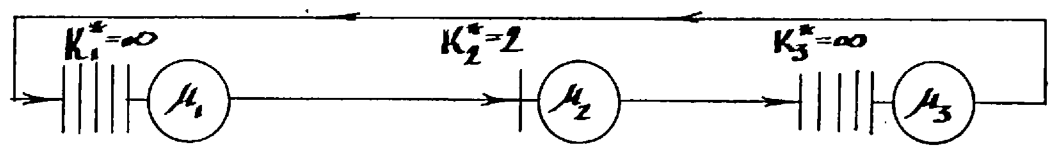


figure 6.2 Cyclic Constrained Network (N=3: $K_2^*=2$)

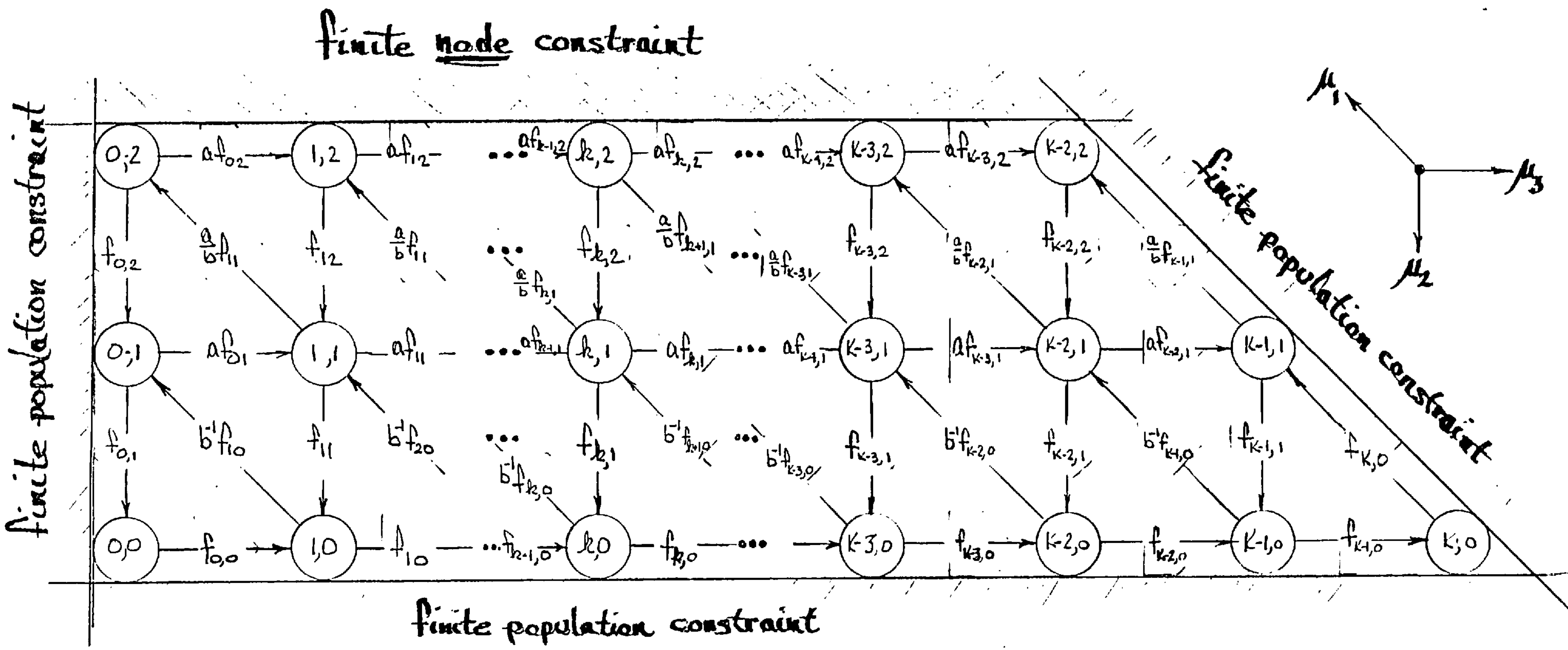


figure 6.3 State flow diagram

In the same transition diagram we have, for notational convenience, defined the flows $f_{k_1 k_2}$ as,

$$
\begin{aligned}
f_{k,0} &= \mu_3 p(k\ ,0,K-k\ ) && k=0,1,\ldots K-1 \\
f_{k,1} &= \mu_2 p(k\ ,1,K-k\ -1) && k=0,1,\ldots K-1 \\
f_{k,2} &= \mu_2 p(k\ ,2,K-k\ -2) && k=0,1,\ldots K-2
\end{aligned}
\tag{6.3}
$$

Note that the last element description in p is superfluous and dropped for notational convenience.

Implementation of the conservation of flow property leads directly to the set of difference equations

$$
\begin{aligned}
(1+c)f_{i,0} &= f_{i-1,0}+f_{i,1} && i=1,2..K-1 \\
(a+ac+1)f_{i,1} &= cf_{i+1,0}+af_{i-1,1}+f_{i,2} && i=1,2..K-2 \\
(a+1)f_{i,2} &= acf_{i+1,1}+af_{i-1,2} && i=1,2..K-3
\end{aligned}
\tag{6.4}
$$

where a and b are dummy parameters

$$
a = \mu_3/\mu_2;\ b = \mu_3/\mu_1\ ;\quad c = \mu_1/\mu_3 \tag{6.5}
$$

The results of this analysis and its extention appear in Appendix B; but only for very simply cyclic networks did this method produce usable analytic results.

6.3.2 Symbolic Solutions

It is worth reasserting that the balance equations (5.2), with appropriate state dependent service rates have unique solutions which solve the Markovian blocking problem.

$$p(\underline{k})R = \emptyset \; ; \; \Sigma p(\underline{k}) = 1 \tag{6.6}$$

The solution to this system will be, in general, the ratio of two polynomials; The denominator is the normalising value, i.e., the sum of the numerator polynomials. It is not unreasonable to solve (6.6) for the general state-dependent case in terms of parameters of the problem, e.g., $c_i(\underline{k})$; $\underline{k} \in \underline{S}$, $i \in \underline{N}$.

Since the state space, $\underline{S}$, is usually large, the rate matrix, R, is enormous. This makes symbolic evaluation nearly unthinkable unless algebraic simplifications can be implemented to continuously purge hidden identities in the polynomials (product forms are merely the 'left-overs' after the common factors have been absorbed into the normalising constant).

In order to solve $pR = \emptyset$ uniquely, an arbitrary row is replaced by $\Sigma p = 1$, call this the *normalising* row and denote the resulting matrix R'. Further define vector $\underline{b}$ to be a column vector containing zero's except for a one in the row corresponding to the normalising row.

From the fundamental assumptions of MQN, it is well known that $pR' = b'$ has a unique solution which implies that the determinant of R', Δ, is non-zero.

Let $\Delta(\underline{k})$ denote the determinant of R' obtained by replacing the column corresponding to the state $\underline{k}$ in R' by the column b'. Expanding by cofactors, it is not difficult to see that

$\Delta(\underline{k}) = \pm \Delta'(\underline{k})$, where $\Delta'(\underline{k})$ is the determinant of the matrix resulting from deletion of the normalising row and the column corresponding to $\underline{k}$ in R'.

By Cramer's rule

$$p(\underline{k}) = G^{-1}|\Delta(k)| \quad \text{where } G = \Delta$$

$\Delta(k)$ is merely the determinant of the rate matrix R with an arbitrary row deletion and a column deletion corresponding to $\underline{k}$.

The above result is a consequence of elementary algebra on the balance equations. Nevertheless the size of the matrix R prohibits symbolic (even numeric) evaluation. However it is possible to greatly reduce the amount of (symbolic) evaluation by taking advantage of symmetry in the transition matrix. Even though the balance equations are very general, the state space is very structured. The consequence of this orderliness is a *structured* symmetry of the rate matrix. This is best described by example.

Consider the topology of the state transition diagram for the cyclic network (N = 3; K=2) with *global* state dependent transition rates in figure 6.4. Note that there are three vertex nodes and three edge nodes; furthermore that the balance equations about the vertex nodes are identical, in form, differing only by the labels applied to the parameters. Similarly for the edges.

Thus the state space can be partitioned into cyclic permutation groups, there being one solution for each group (the others obtained simply by a permutation of the *parameters* of each group).

To experiment with symbolic solutions of the general state dependent model, a symbolic analyser (APL) program was written which when executed, produces the *symbolic* evaluation of determinants (6.7) for each cylclic permutation group.

For the example (figure 6.4) there are two cyclic permutation groups (3 members each) with solutions:

$$P(2\ 0\ 0) = G^{-1}\mu_2(0\ 2\ 0)\mu_2(0\ 0\ 2)\mu_3(1\ 0\ 1)B_1$$
$$P(1\ 0\ 1) = G^{-1}\mu_1(2\ 0\ 0)\mu_2(0\ 2\ 0)\mu_3(0\ 0\ 2)B_1 \tag{6.8}$$

$$B_1 = \mu_2(0\ 1\ 1)\mu_1(1\ 1\ 0)+\mu_2(1\ 1\ 0)\mu_2(0\ 1\ 1)+\mu_2(1\ 1\ 0)\mu_3(0\ 1\ 1)$$

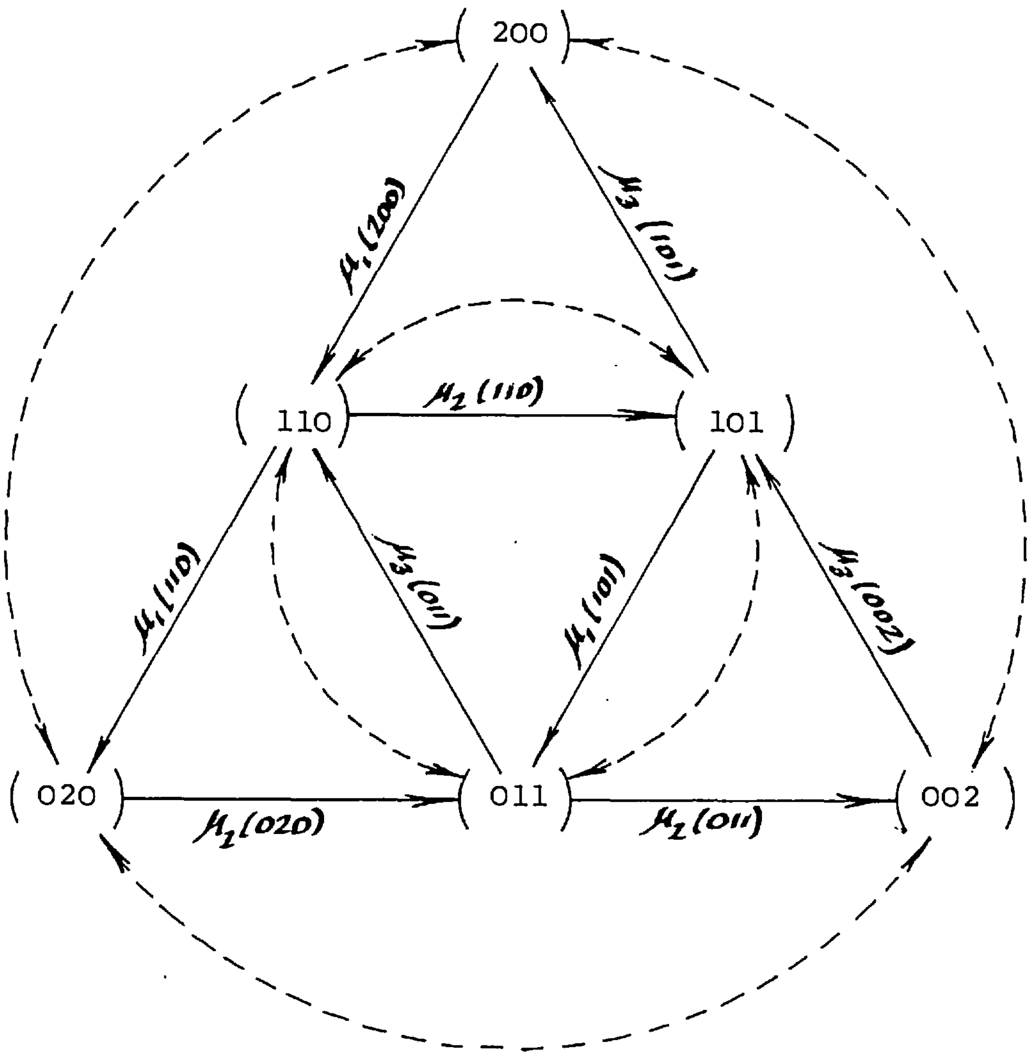


figure 6.4 Cyclic permutations of the general state dependent network ($K=2;N=3$)

Results of larger models appear in Appendix C.

Observations

(1) The solution is not in general a simple product form.

(2) The solution is valid for both blocking and skipping assumptions e.g., if $\underline{k}$ = (2 0 0) is a *blocked* state let $\mu_3(1\ 0\ 1) \to 0$ if $\underline{k}$ = (2 0 0) is a *skipped* state, let $\mu_1(2\ 0\ 0) \to \infty$.

(3) Only cyclic permutations of state (2 0 0) can have non-degenerate blocking and skipping solutions.

(4) With the local state dependent assumption, $\mu_i(\underline{k}) = \mu_i(k_i)$, then, after factoring, the terms, B_1^i are identical and may be absorbed into the normalising constant. The solution is then a product form and is equivalent to the SQN solutions (as it must be).

(5) These problems do not *appear* to have compact solutions (or even ones that can be written down by inspection).

These solutions are perhaps *too* general (the specification alone is overwhelming) and certainly they are not compact. The research problem seems to lie in finding analytic solutions, possibly not as neat as simple product forms, but hopefully less complicated than those in (6.8). These results must be produced if poorly conditioned blocking networks are to yield useful analytic forms.

6.3.3 Remarks on Numerical Evaluation

In many instances analytic solutions of the blocking equations are not only unknown, but unnecessary. On these occasions it may be expedient or necessary to produce numerical solutions. The numerical analysis of large linear systems has been extensively researched elsewhere and such techniques that are available will have obvious application to the evaluation of equations (5.2). The numerical analysis of blocking networks is beyond the scope of this work, but we remark:

(1) The size of matrix R is $\binom{N+K-1}{N-1}^2$ elements.

(2) Each row has between 2 and N^2+1 non-zero elements and for large K the number of non-zero elements is of $O(N^2)$. Thus such matrices are not in general very sparse although specific problems may have sparse matrices .

(3) For the general case, the matrix has a symmetrical *structure* (but not necessarily symmetrical in value) Special cases may unbalance this structural symmetry.

(4) All matrices have columns which sum to zero.

An effective numerical approximation method for such systems of linear equations is the power iteration method (WALL66).

If P^i is the i^{th} iterate, then

$$P^{i+1} \leftarrow P^i(kR+I)$$

where k is a scalar and I is the identity matrix. P^o is an initial guess which might be chosen as the solution to the best approximation SQN solution.

Other numerical approximation techniques such as decomposition (c.f. 3.2.2) and perturbation may provide the key to practical applications of large network models. Decomposition techniques have already been successfully applied (HINE77, COUR77); and while we know of no use of perturbation it is intuitively appealing to search for solutions of non-separable networks in the vicinity of SQN solutions.

The numerical and approximate analysis (including simulation) of blocked networks remains a subject for future research.

6.4 Post Mortem

In the attainment of its most ambitious goal - the explicit compact solution of the blocking problem - this work has been less than successful. This defeat, although not unexpected is nevertheless disappointing. While convenient analytic solutions do not exist in general, this study has shown that some simple, yet important, models have useful and theoretically interesting solutions.

Once again it is reasserted that the blocking class of queuing models is important to the analysis and understanding of finite resource computer systems. Eventually these models will be resolved - if not analytically, then approximately or through exhaustive simulation; these failing then, as usual, by pragmatic trial and error.

APPENDIX A FINITE CAPACITY EXAMPLE (SOLUTION)

This appendix contains the solution for the constrained queuing network example introduced in section 4.4. This network represents a passive resource limited computing system. Even though it is a reasonably naïve model, it is a non-trivial queuing problem - complexities are introduced into the balance equations due to potential blocking conditions at node 1. The model is solved explicitly for specific numerical values.

A.1 Solution

The model is solved for the case $\{L=2;\ K_1=2\ K_2=1\}$ with $s_1=1$, $s_2=2$ and $d_3=2$ storage units. This model satisfies the MQN conditions; Define the following states.

Let $\underline{k}$ be an array $= \left\{\begin{matrix} w_1, r_1 \\ w_2, r_2 \\ n_2, t \end{matrix}\right\}$ where (A.1)

w_ℓ, r_ℓ are the number of processes of type ℓ respectively waiting and executing at node 1 $(\ell=1,2)$

n_2 is the number of processes at node 2

t is the process type in service at node 2

then the state space is $\underline{k}\varepsilon\underline{F}$ such that

$\{(r_1 s_1+r_2 s_2)\leq 2\}, \{w_\ell+r_\ell \leq K_\ell ; \ell=1,2\}$ and $\{w_\ell, r_\ell, k_2 \geq 0;\ \ell=1,2\}$ (A.2)

The state space and its transition rate diagram are shown in figure A.1 (note the shading on the inhibited states).

By assumption this system is Markovian with solution

$$p R = \underline{0} \qquad \sum_{\underline{k} \varepsilon p} p(\underline{k}) = 1 \tag{A.3}$$

where

$$p = \left\{ p\begin{pmatrix}00\\00\\32\end{pmatrix} p\begin{pmatrix}00\\00\\31\end{pmatrix} p\begin{pmatrix}01\\00\\21\end{pmatrix} p\begin{pmatrix}01\\00\\22\end{pmatrix} p\begin{pmatrix}00\\01\\21\end{pmatrix} p\begin{pmatrix}02\\00\\12\end{pmatrix} p\begin{pmatrix}01\\10\\11\end{pmatrix} p\begin{pmatrix}10\\01\\11\end{pmatrix} p\begin{pmatrix}20\\01\\00\end{pmatrix} p\begin{pmatrix}02\\10\\00\end{pmatrix} \right\} \tag{A.4}$$

and

$$\underline{0} = (0,0,0,0,0,0,0,0,0,0) \qquad \text{and}$$

the transition rate matrix,

$$R = \begin{pmatrix}
-\mu_2 & & & \mu_1 & & & & & & \\
 & -\mu_2 & \mu_1 & & \mu_1 & & & & & \\
 & \tfrac{1}{2}\mu_2 & -(\mu_1+\mu_2) & & & & & \mu_1 & & \\
 & \tfrac{1}{2}\mu_2 & & -(\mu_1+\mu_2) & & 2\mu_1 & & & & \\
\mu_2 & & & & -(\mu_1+\mu_2) & & \mu_1 & & & \\
 & & \mu_2 & & & -(2\mu_1+\mu_2) & & & \mu_1 & \\
 & & & \mu_2 & & & -(\mu_1+\mu_2) & & & 2\mu_1 \\
 & & & & \mu_2 & & & -(\mu_1+\mu_2) & & \\
 & & & & & & & \mu_2 & -\mu_1 & \\
 & & & & & \mu_2 & \mu_2 & & & -2\mu_1
\end{pmatrix} \tag{A.5}$$

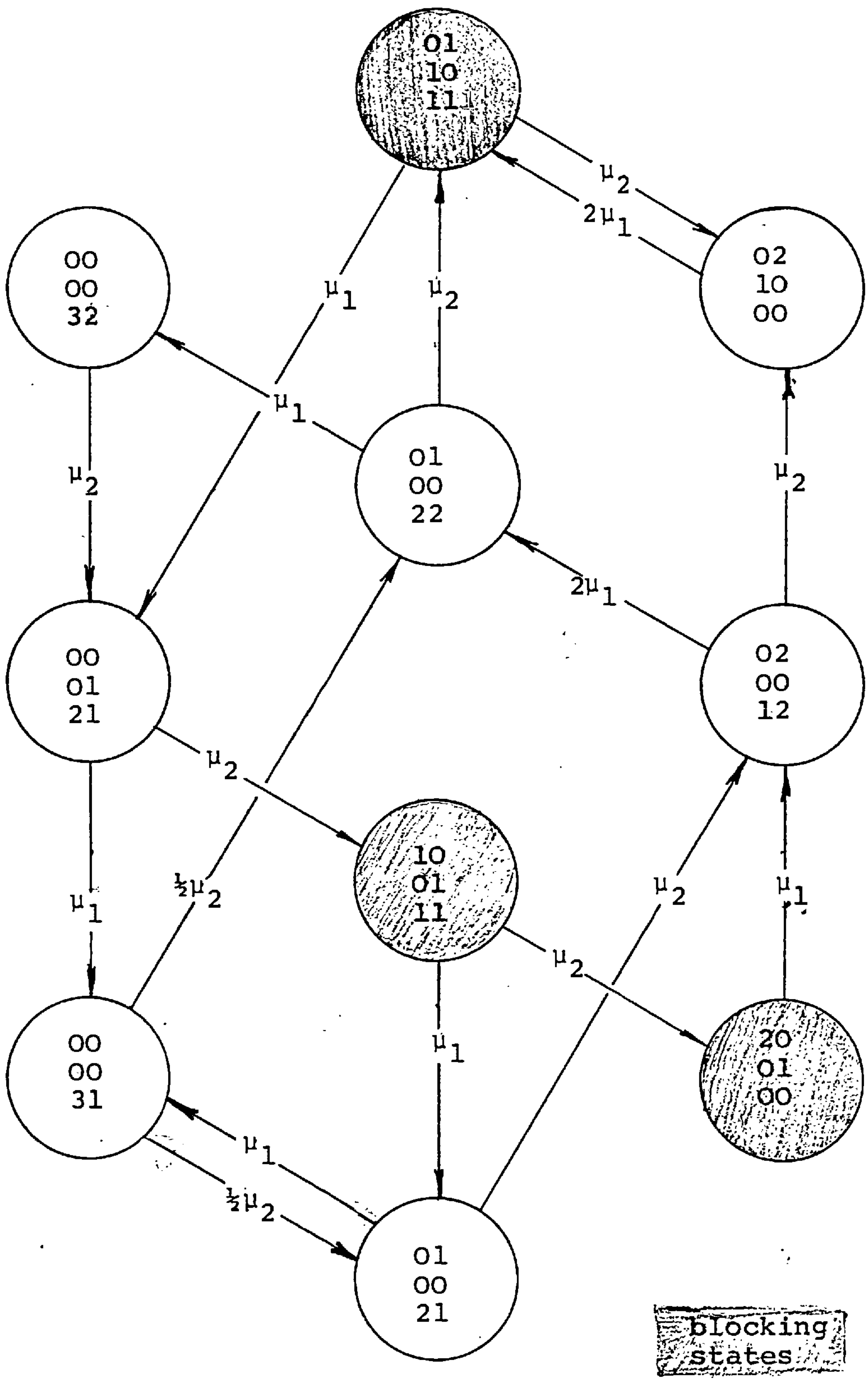


figure A.1   State Transition Diagram

A.2 Evaluation

The linear system (A.3) is solved numerically by ordinary linear techniques (for various parameters μ_1 normalised about μ_2). From p, the performance metrics for the network may be calculated and are summarised in table A.1, including

Prob[customer type ℓ is blocked] = b_ℓ

$$b_1 = p_{\begin{pmatrix}10\\01\\11\end{pmatrix}} + p_{\begin{pmatrix}20\\01\\00\end{pmatrix}} \;;\quad b_2 = p_{\begin{pmatrix}01\\10\\11\end{pmatrix}} \tag{A.6}$$

which in this example corresponds with the expected number of processes of type ℓ blocked.

A.3 Comparison with Unconstrained Results

If we relax the passive resource constraint in the example we have a simple cyclic 2-node network, easily solved by SQN methods (c.f. 3.3.2). Thus it is easily seen that

$$e_1 = 1 \;;\quad c_1(k_1) = \begin{cases} 1 & k_1 < 2 \\ 2 & k_1 \geq 2 \end{cases} ;\; \mu_1 = \mu_1$$

$$e_2 = 1 \;;\quad \underset{k_2 \geq 0}{c_2(k_2) = 1} \;;\quad \mu_2 = 1 \tag{A.7}$$

therefore,

$$\beta_1(1) = 1/\mu_1; \beta_1(2)=1/2\mu_1^2, \ \beta_1(3)=1/4\mu_1^3 \ ; \ \beta_2(k_2)=1 \qquad (A.8)$$

so $p(k_1, k_2) = G^{-1}(K)\beta_1(k_1)$ where

$$G(k) = \sum_{k_1+k_2=3} \beta_1(k_1)\,\beta_2(k_2) = \sum_{k_1=0}^{3} \beta_1(k_1) = \frac{4\mu_1^3+4\mu_1^2+2\mu_1+1}{4\mu_1^3} \qquad (A.9)$$

so that

$$p(k_1, 3-k_1) = \frac{4\mu_1^3}{4\mu_1^3+4\mu_1^2+2\mu_1+1} \ \beta_1(k_1) \qquad (A.10)$$

From (A.10) all performance metrics are easily derived and are computed for the same range of μ_1 as in section A.2 and are tabulated in Table A.2. Thruputs of both models are graphed in figure 4.8; and we see, as expected, a performance degradation for the blocking condition which vanishes for infinitely fast or slow processing (at node 1) and appears to be maximised at $\mu_1 \simeq .33\mu_2$ (when the blocking expectation is maximal).

Table A.1 Constrained Network Performance

Service Rate μ_1		class	8	4	3	2	1	½	1/3	¼	⅛	1/16	1/32
Prob [Idle]		1	.87800	.76300	.69200	.56300	.29700	.08980	.03900	.01940	.00307	.00043	.00006
		2	.00097	.00657	.01400	.03780	.15500	.38800	.53500	.62600	.79000	.88700	.94100
Expected number at node 1	In Service	1	.0833	.1660	.2210	.3280	.6040	.9520	1.1400	1.2700	1.5200	1.7100	1.8000
		2	.0416	.0820	.1080	.1530	.2400	.2720	.2520	.2270	.1560	.0959	.0542
	Waiting	1	.0005	.0033	.0067	.0170	.0601	.1210	.1420	.1450	.1240	.0849	.0509
		2	.0005	.0033	.0072	.0207	.0952	.2670	.3930	.4810	.6670	.8020	.8900
	Blocked	1	.0046	.0164	.0269	.0511	.1200	.1810	.1890	.1810	.1390	.0903	.0525
		2	.0049	.0181	.0302	.0591	.1440	.2160	.2200	.2080	.1520	.0951	.0541
	Total	1	.0884	.1857	.2546	.3961	.7841	1.2540	1.4710	1.5960	1.7830	1.8852	1.9034
		2	.0469	.1034	.1454	.2328	.4792	.7550	.8650	.9160	.9750	.9930	.9983
Expected number at node 2	In Service	1	.6670	.6650	.6630	.6550	.6040	.4760	.3810	.3170	.1900	.1070	.0574
		2	.3320	.3280	.3230	.3070	.2400	.1360	.0839	.0567	.0196	.0060	.0017
	Waiting	1	1.2400	1.1500	1.0800	.9490	.6110	.2700	.1440	.0885	.0254	.0069	.0018
		2	.6210	.5690	.5320	.4600	.2800	.1090	.0510	.0277	.0054	.0009	.0001
	Total	1	1.9070	1.8150	1.7430	1.6040	1.2150	.7460	.5250	.4055	.2154	.1139	.0592
		2	.9530	.8970	.8550	.7670	.5200	.2450	.1349	.0844	.0250	.0069	.0018
Thruput	Node 1	1	.6670	.6650	.6630	.6550	.6040	.4760	.3810	.3170	.1900	.1070	.0574
		2	.3320	.3280	.3230	.3070	.2400	.1360	.0839	.0567	.0196	.0060	.0017
	Node 2	1	.6670	.6650	.6630	.6550	.6040	.4760	.3810	.3170	.1900	.1070	.0574
		2	.3320	.3280	.3230	.3070	.2400	.1360	.0839	.0567	.0196	.0060	.0017
Response Time	Node 1	1	.13	.28	.38	.60	1.30	2.64	3.87	5.03	9.38	17.60	33.80
		2	.14	.32	.45	.76	1.99	5.55	10.30	16.20	49.90	166.00	590.00
	Node 2	1	2.87	2.73	2.63	2.45	2.01	1.57	1.38	1.28	1.13	1.06	1.03
		2	2.87	2.73	2.65	2.50	2.17	1.80	1.61	1.49	1.27	1.15	1.08

TABLE A.2 Unconstrained Network Performance

Service Rate μ_1	node	8	4	3	2	1	½	1/3	¼	⅛	1/16	1/32
Prob [Idle]	1	.8820	.7780	.7150	.6040	.3640	.1430	.0656	.0345	.0059	.0009	.0001
	2	.0004	.0030	.0066	.0189	.0909	.2860	.4430	.5520	.7570	.8760	.9380
Expected Queue Length	1	.125	.252	.338	.509	1.000	1.710	2.110	2.340	2.700	2.660	2.960
	2	2.870	2.750	2.660	2.490	2.000	1.290	.885	.655	.302	.139	.066
Thruput	1	1.000	.997	.993	.981	.909	.714	.557	.448	.243	.124	.062
	2	1.000	.997	.993	.981	.909	.714	.557	.448	.243	.124	.062
Response Time	1	.13	.25	.34	.52	1.10	2.40	3.79	5.23	11.10	23.10	47.00
	2	2.88	2.76	2.68	2.54	2.20	1.80	1.59	1.46	1.24	1.12	1.06

APPENDIX B CYCLIC NETWORKS WITH BLOCKING (N=3)

The special case of the 3-node cyclic network led to the system of linear difference equations, (6.4), with boundary conditions:

$$\left\{\begin{aligned} f_{00} &= f_{01} \\ (a+1)f_{01} &= cf_{10}+f_{02} \\ (a+1)f_{02} &= acf_{11} \end{aligned}\qquad i = 0\right\}$$

$$\left\{\begin{aligned} f_{K,0} &= f_{K-1,0} && i = K \\ (1+ac)f_{K-1,1} &= f_{K-1,0}+af_{K-2,1} && i = K-1 \\ f_{K-2,2} &= acf_{K-1,1}+af_{K-3,2} && i = K-2 \end{aligned}\right\} \tag{B.1}$$

where a,b,c are defined in equation (6.5). Elementary linear operations on these balance equations provide recursion equations:

$$\left\{\begin{aligned} f_{00} &= G \\ f_{10} &= b((a+1)f_{01}-f_{02}) \\ f_{i,0} &= (ab+a+b)f_{i-1,0}-abf_{i-2,1}-bf_{i-1,2} && i=2,..K-1 \\ f_{K,0} &= f_{K-1,0} \end{aligned}\right\} \tag{B.2}$$

$$\left\{\begin{aligned} f_{01} &= G \\ f_{i,1} &= (1+c)f_{i,0}-f_{i-1,0} && i = 1,2...K-1 \end{aligned}\right\} \tag{B.3}$$

$$\left\{\begin{array}{l} f_{02} = \dfrac{a(a+b+ab)}{a(1+b)+b(1+a)}G \\ f_{i,2} = \dfrac{(b+1)\left[(1+a+ac)f_{i,1}-af_{i-1,1}\right]-f_{i,0}+af_{i-1,2}}{a+b+2} \quad \ldots\ i=1,..K-3 \\ f_{K-2,2} = af_{K-3,2}+acf_{K-1,1} \end{array}\right\} \qquad (B.4)$$

These results, although suitable computational forms are not very compact and fail to offer a clue towards generalisation. To generalise this result consider a variable blocking parameter K*.

The state transition diagram is shown in figure B.1. Note that there are nine different types of linear difference equations required to specify the balance equations. There are four corners, four sets of edge equations and an internal equation (set). These equation 'atoms' are shown in Figure B.2 and by linking them together into larger 'molecules' it is possible to construct the state transition balance equations, figure B.3.

The flow balance equations are:

a  $\mu_3 p_{00} = \mu_2 p_{01}$

b  $\mu_1 p_{K0} = \mu_3 p_{K-1,0}$

c  $\mu_2 p_{K-K^*,K} = \mu_3 p_{K-K^*-1,K^*} + \mu_1 p_{K-K^*+1,K^*-1}$

d  $(\mu_2+\mu_3)p_{0,K^*} = \mu_1 p_{1,K^*-1}$

corners (B.5)

$$\left.\begin{array}{lll}
\text{e} & (\mu_1+\mu_3)p_{i,0}=\mu_3 p_{i-1,0}+\mu_2 p_{i,1} & (1\le i\le K-1) \\
\text{f} & (\mu_1+\mu_2)p_{K-i,i}=\mu_3 p_{K-i-1,i}+\mu_1 p_{K-i+1,i-1} & (1,K^*-2) \\
\text{g} & (\mu_2+\mu_3)p_{i,K^*}=\mu_3 p_{i-1,}+\mu_1 p_{i+1,K^*-1} & (1,K-K^*-1) \\
\text{h} & (\mu_2+\mu_3)p_{0,i}=\mu_2 p_{0,i+1}+\mu_1 p_{1,i-1} & (1,K^*-1)
\end{array}\right\}\ \text{edges} \qquad (B.5)$$




$$\text{i} \quad (\mu_1+\mu_2+\mu_3)p_{i,j}=\mu_3 p_{i-1,j}+\mu_1 p_{i+1,j-1}+\mu_2 p_{i,j+1} \ \Big\}\ \text{interior}$$

where the p's are the state probability form of the flow equations (c.f. 6.3).

This system can be solved by multiple linear substitutions for the cases $K^*=1$ and $K^*=2$, yielding, for example for $K^*=1$,

$$\begin{pmatrix} p_{i,0} \\ p_{i,1} \\ p_{i+1,0} \end{pmatrix} = G^{-1} \begin{pmatrix} 0 & 0 & 1 \\ -a & 0 & a(1+c) \\ -b(1+a) & -b & (1+b)(1+a) \end{pmatrix}^{i} \begin{pmatrix} 1 \\ a \\ b(1+a) \end{pmatrix} \quad i=0,1..K-3 \qquad (B.6)$$

Note that for the balanced network $\mu_1=\mu_2=\mu_3=\text{CONST}\Rightarrow a,b,c=1$

$$\left\{(p_{i,0},p_{i,1}\right\}= G^{-1}\left\{(1,1)(2,3),(5,8),\ldots\right\}\Big|\ i=0,1,2\ldots$$

the Fibonacci numbers, so that a closed form expression is available for this subcase (with appropriate care in handling the termination conditions).

Similar results are simply, but tediously derived for the case $K^*=2$:

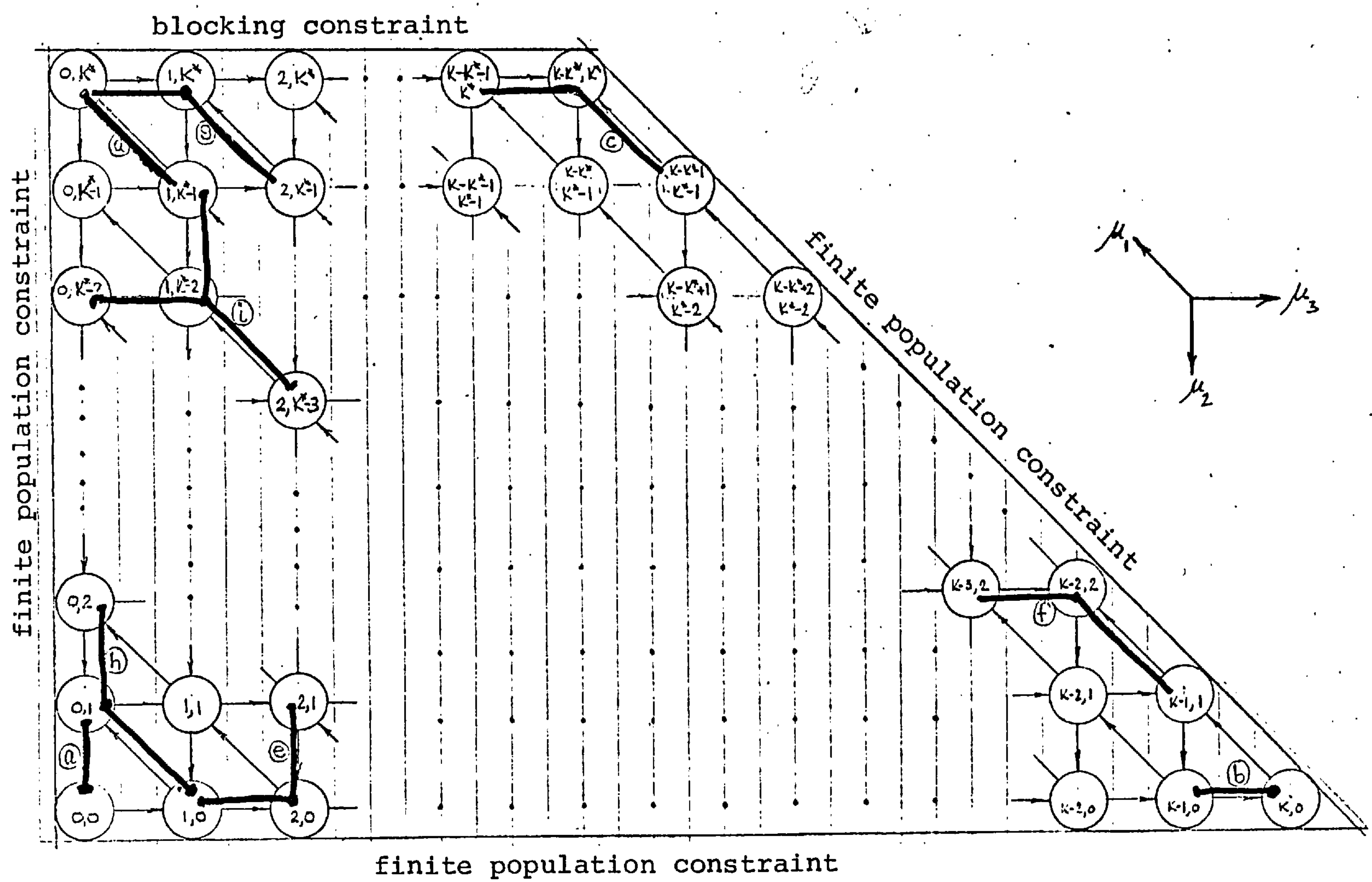


figure B.1 Cyclic Queuing Network (N=3)

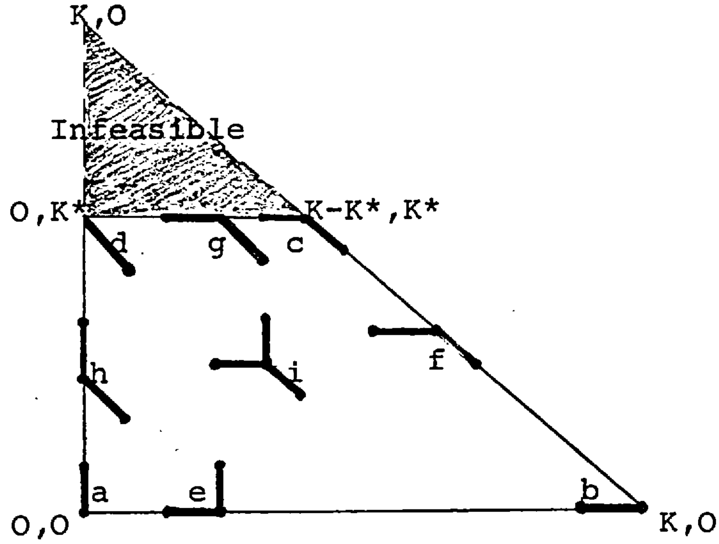


figure B.2 Balance Equation Elements

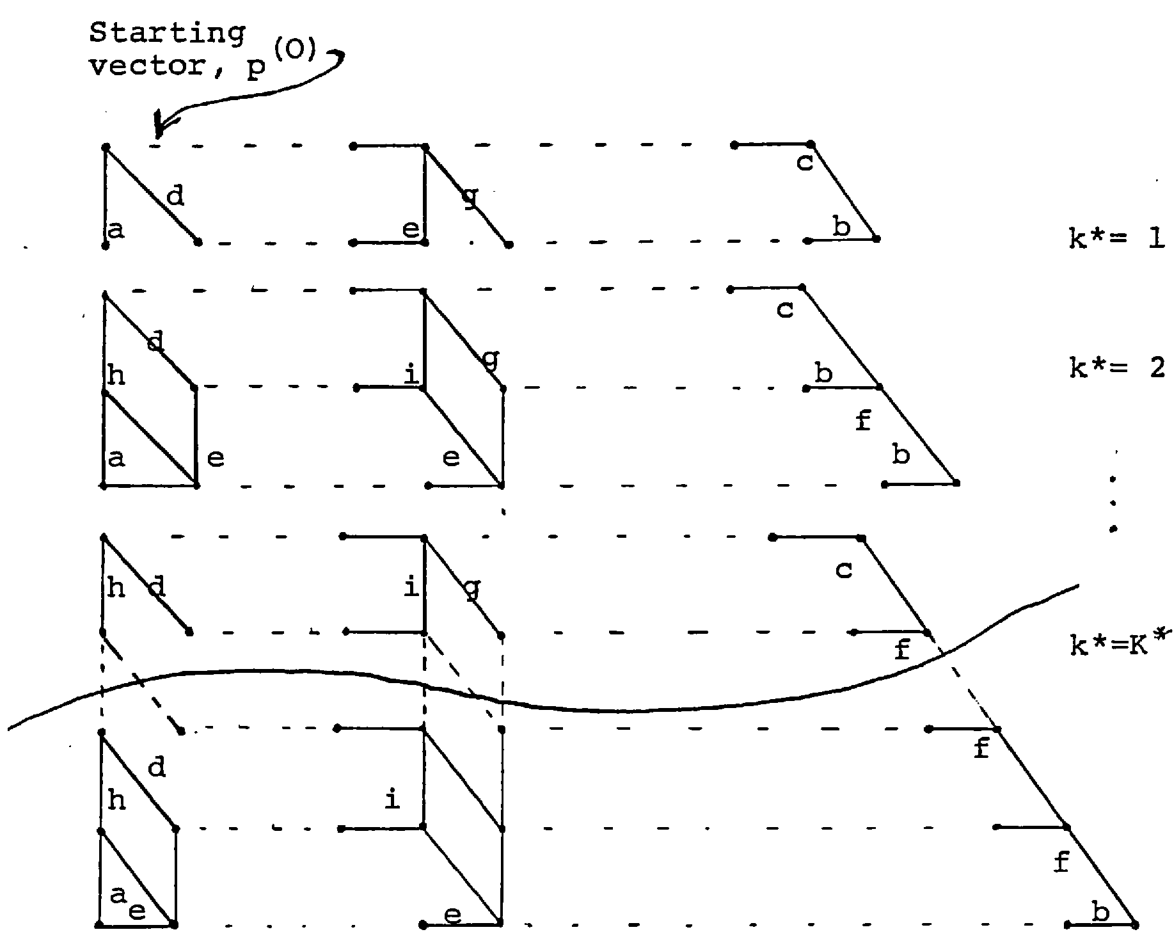


figure B.3 Assembly of Difference Equations

$$\begin{pmatrix} p_{i,1} \\ p_{i,2} \\ p_{i+1,0} \\ p_{i+1,1} \end{pmatrix} = G^{-1} r^{-1} \begin{pmatrix} 0 & 0 & 0 & r \\ -a(b+1) & ab & -a & \frac{(b+1)d}{b} \\ -b^2(a+1) & -b^2 & b & \frac{b(a+1)d}{a} \\ -b(b+1)(a+1) & b(b+1) & b(a+1) & \frac{(b+1)(a+1)d}{a} \end{pmatrix}^{i} \begin{pmatrix} 1 \\ a d e^{-1} \\ (b^2(a+1)^2 + ba)e^{-1} \\ b(a+1) d e^{-1} \end{pmatrix}$$

where $d = ab+a+b$

$r = ab+2b+1 \qquad ..i=0,1,2...N-2 \qquad$ (B.7)

$e = a(b+1)+b(a+1)$

However hopes are dimmed when attempting to produce results for blocking constraints $K^*\geq 2$. This unfortunate situation arises from an inability to solve for the starting vector $p^{(0)}$ (or the base of the recursion).

For the cases $K^*=1,2$ observe that the base $p^{(0)}$ is routinely determined (figure B.4); but for $K^*>2$ the starting vector is always underdetermined. Since it is known that the entire system is over determined, the conclusion is that starting conditions are reconciled at the terminating boundary. This means that this solution method is only useful for $K^*\leq 2$. Either we have failed to discover the solution; or it is possible, even likely, that no compact solution for this non-reversible network exists.

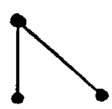
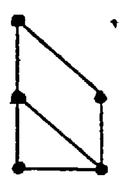
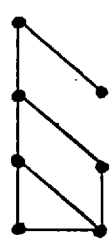
	network	unknowns	equations
k*=1		2	2
k*=2		4	4
k*=3		6	5
⋮			
k*=K*	⋮	2K*	K*+2

figure B.4 Initialisation Equations, $p^{(0)}$.

APPENDIX C GENERAL STATE DEPENDENT SOLUTIONS FOR TWO-CYCLE NETWORK (K=3;N=3) and (K=2,N=4)

The following are sample solutions for an element in the cyclic permutation group for two models.

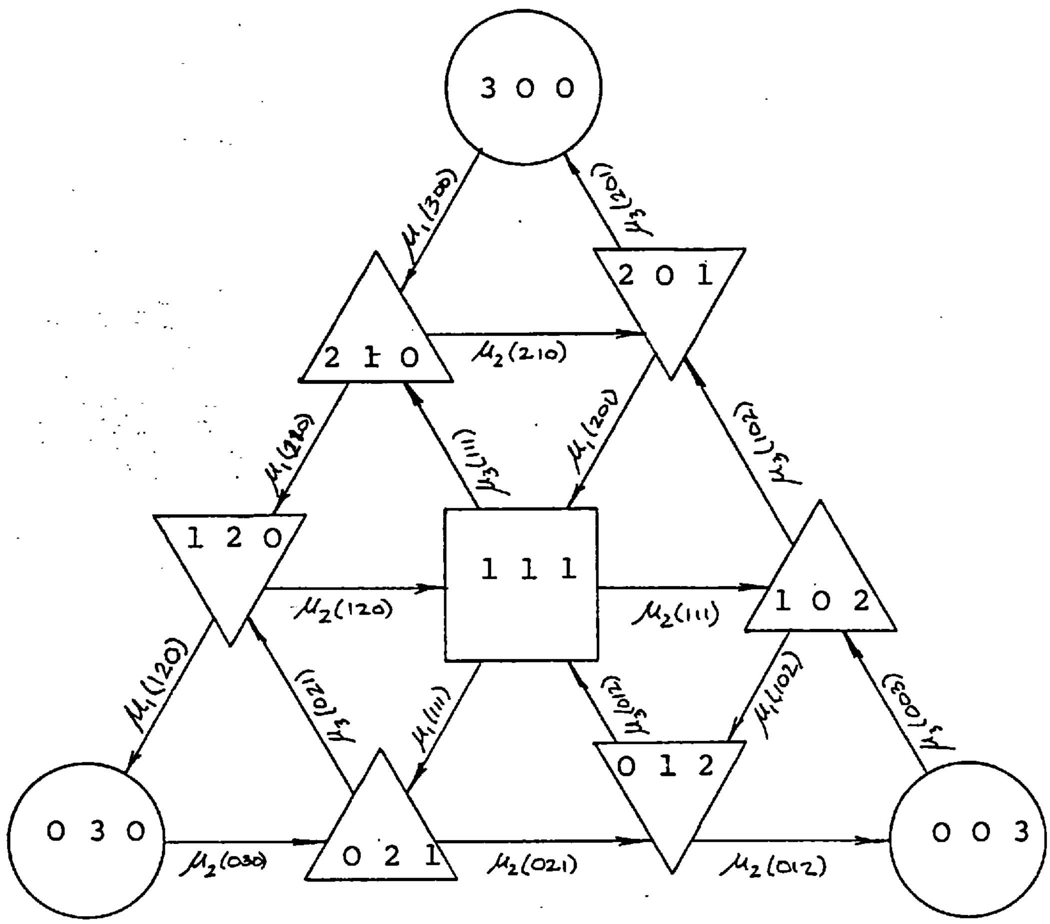
The first is a cyclic network of three nodes containing three customers (K=3;N=3), figure C.1, and the second has parameters (K=2;N=4) figure C.2.

The (K=3;N=3) model has 15 states which partition into four cyclic permutation groups

$$\begin{array}{l} \{(300)\ (030)\ (003)\} \\ \{(210)\ (021)\ (102)\} \\ \{(120)\ (012)\ (201)\} \\ \{(111)\} \end{array} \qquad (C.1\text{-}C.4)$$

Symbolic Evaluation produces the solution for cyclic permutation group (C.1).

```
P300=G^-1{ U2[030]
         x
              U3[003]
            x
                U2[111]×U3[102]×U3[021]×U1[120]×U3[201](U3[012] + U2[012])(U2[210] + U1[210])
               +
                U3[021]×U3[111]×U2[210]×U1[120]×U3[201](U3[102] + U1[102])(U3[012] + U2[012])
           -
                U3[003]
              x
                   U3[102]×U2[021]×U3[111]×U1[210]×U2[012]×U1[120]×U3[201]
                  +
                   U1[102]×U3[021]×U3[111]×U2[210]×U2[012]×U1[120]×U3[201]
   +
     U2[030]
    x
            U3[003]
          x
                U2[111]×U3[102]×U3[201](U3[012] + U2[012])(U2[021] + U3[021])(U1[120] + U2[120])
              x
                (U2[210] + U1[210])
             +
                U3[111]×U2[210]×U3[201](U3[012] + U2[012])(U2[021] + U3[021])(U1[120] + U2[120])
              x
                (U3[102] + U1[102])
       +
            U3[003]
          x
              U3[102]×U1[111]×U2[021]×U2[012]×U3[201](U1[120] + U2[120])(U2[210] + U1[210])
             -
              U1[102]×U3[111]×U2[210]×U2[012]×U3[201](U1[120] + U2[120])(U2[021] + U3[021])}
```



○ { (300) (030) (003) }
△ { (210) (021) (102) }
▽ { (120) (012) (201) }
□ { (111) }

figure C.1 General State Dependent Network (K=3;N=3)

For the model (K=2,N=4), there are three cyclic permutation groups:

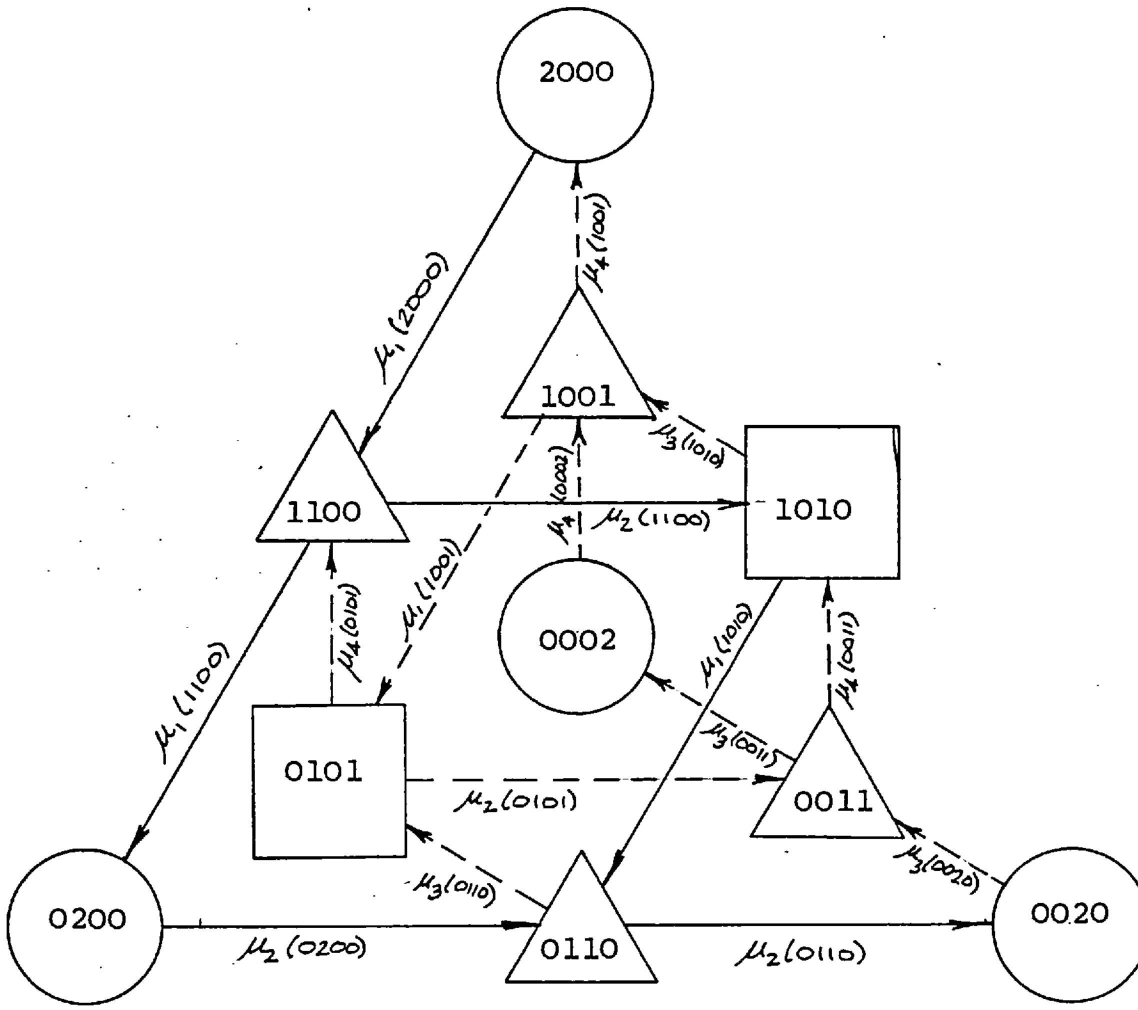
$$\{(2000)\ (0200)\ (0020)\ (0002)\}$$
$$\{(1100)\ (0110)\ (0011)\ (1001)\} \qquad (C.5\text{-}C.7)$$
$$\{(1010)\ (0101)\}$$

with Symbolic Solution for (C.5).

$P(2000) = G^{-1}\Big\{$

```
U3[0020]×U2[0200]
 x
    U4[1001]×U3[1010]×U4[0011]×U4[0101]×U4[0002]×U2[0110]×U1[1100]
   +
    U4[1001]×U4[0101]×U3[0011]×U4[0002]×U2[0110]×U1[1100](U3[1010] + U1[1010])
+
U2[0200]
 x
    U4[1001]×U1[1010]×U4[0101]×U2[1100]×U3[0011]×U4[0002]×U2[0110]×U3[0020]
   +
      U3[0020]
     x
         U4[1001]×U2[0101]×U3[0011]×U4[0002](U3[1010] + U1[1010])(U3[0110] + U2[0110])(U2[1100] + U1[1100]
        +
            U4[0002]
           x
                U4[1001]×U3[1010]×U2[0101]×U4[0011](U3[0110] + U2[0110])(U2[1100] + U1[1100])
               +
                U4[1001]×U3[1010]×U4[0101]×U2[1100](U3[0110] + U2[0110])(U3[0011] + U4[0011])
```

$\Big\}$



○ { (2000) (0200) (0020) (0002) }
△ { (1100) (0110) (0011) (1001) }
□ { (1010) (0101) }

figure C.2 General State Dependent Network (K=2;N=4)

BIBLIOGRAPHY

ADIR72 Adiri, I, "Queuing Models for Multiprogrammed Computers", Proc. Int. Symposium of Computer Communication Networks and Teletraffic, Polytech Press, Brooklyn, N.Y. pp 441-448, 1972

BASS75 Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F.G., "Open Closed and Mixed Networks of Queues with Different Classes of Customers" J. of ACM, Vol.22, No.2, pp 248-260, April 1975.

BELA69 Belady, L.A., and Kuehner, C.J., "Dynamic Space Sharing in Computer Systems", Comm. ACM, May 1969

BHAT63 Bhat, U.N., "Sixty Years of Queuing Theory", Management Science, Vol.15, pp B280-B294, 1963

BRAN77 Brandwain, A., "A Queuing Model of Multiprogrammed Computer Systems Under Full Load Conditions" J. of ACM, Vol24, No.2, pp 222-240, April 1977

BROW75 Browne, J.C. et al., "Hierarchical Techniques for the Development of Realistic Models of Complex Computer Systems", Proc. IEEE, 63.6, June 1975, pp 966-975

BROW75b Browne, J.C. et al., "Hierarchical Techniques for the Development of Realistic Models of Complex Computer Systems, Proc. IEEE, Vol.63, No.6, pp 966-975, June 1975

BUCH69 Buchholz, W., " A Selected Bibliography of Computer System Performance Evaluation", IEEE Computer Group News, pp 21-22, March 1969

BUZE71 Buzen, J. "Queuing Network Models of Multiprogramming", Ph.D Thesis, Division of Engineering and Applied Science, Harvard University, Cambridge Massachusetts, 1971

BUZE73 Buzen, J., "Computational Algorithms for Closed Queuing Networks with Exponential Servers", J. ACM , Vol. =6, No. 9, 1973.

CALI67 Calingaert, P., "System Performance Evaluation - A Survey and Appraisal", C.ACM, Vol.10, No.1 pp 12-18, January 1967

CHAN72 Chandy, K.M., "The Analysis and Solution for General Queuing Networks", Proc. 6th Annual Princeton Conf. on Information Sciences and Systems, Princeton Univ. March 1972

CHAN75 Chandy, K.M., Herzog, U., and Woo, L. "Approximate Analysis of General Queuing Networks" IBM J. of Res. and Dev., Vol.19, No.1, pp 43, January, 1975

CHAN75a Chandy, K.M., Herzog, U., and Woo, L., "Parametric Analysis of Queuing Networks", IBM J. of Res. and Dev., January 1975

CHAN77 Chandy, K.M., Howard, J. and Towsley, D., "Product Form and Local Balance in Queuing Networks", J. of ACM, Vol.24, No.2, April 1977

CHNG72 Chang, A and Lavenberg, S., "Work rates in Closed Queuing Networks and General Independent Servers", IBM Research Report RS989, March 1972

CONN76 Conner, W.M., Salako, A and Taulbee, O.E. A Framework for Computer System Performance Measurement and Evaluation", Tech. Report 76-1, University of Pittsburgh, 1976

COUR72 Courtois, P.J., "On the Near-Complete Decomposibility of Networks of Queue and of Stochastic Models of Computer Systems", Scientific Report CMU-CS-72-11, Carnegie-Melon Institute November 1972

COUR75 Courtois, P.J., "Error Analysis in Nearly Decomposable Stochastic Systems", MBLE Report R214, March 1975

COUR75b COURTOIS, P.J., "Decomposability, Instabilities and Saturation in Multiprogramming Systems", Comm. ACM, Vol.18, pp 371-377, 1975

COUR77 Courtois, P.J., "Decomposability - Queuing and Computer System Application", Academic Press, New York, 1977

COX55 Cox, D.R., 'A Use of Complex Probabilities in the Theory of Stochastic Processes', Proceedings Cambridge Philosophical Society 51, 313-319 (1955)

CRAN74 Crane, M.A., and Englehart, D.I. "Simulating Stable Stochastic Systems I: General Multiserver Queues", J.ACM, Vol.21, No.1, pp 103-113, 1974

DENN67 Denning, P.J., "The Working Set Model for Program Behaviour", C.ACM, Vol.11, No.3, pp 323-333, 1967

DISN75 Disney, R.L., "Random Flow in Queuing Networks A Review and Critique", AIIE Transactions, Vol.7, No.3, pp 268-286, September 1975

DOPP62 Dopping, A., "Test Problems Used for Evaluation of Computers, Bit Vol 2, No.4, pp 197-202, 1962

DRUM69 Drummond, M.E., "A Perspective on System Performance Evaluation", IBM Systems Journal, Vol.8, No.4, pp 252-263

EDPR77 EDP Performance Review, Vol.5, No.6, Applied Computer Research, Phoenix Arizona, 1977

FELL57 Feller, W., "An Introduction to Probability Theory and its Applications", 2nd Edition New York, John Wiley & Sons, 1957

FERD71 Ferdinand, A.E., "An Analysis of the Machine Interference Model", IBM Systems Journal, Vol.10, No.2, pp 129-142

GAVE73 Gaver, D.P., Schedler, G.S., "Processor Utilisation in Multiprogramming Systems via Diffusion Approximation", Operations Research, Vol.2, pp 569-576, 1973

GAVE68 Gaver, D., "Diffusion Approximation and Models for Certain Congestion Problems", J. Applied Prob., Vol.5, pp 607-623, 1968

GELE75 Gelenbe, E., "On Approximate Computer System Models", J. of ACM, 21, pp 261-269, 1975

GELE76 Gelenbe, E. and Muntz, R.R., "Probabilistic Models of Computer Systems- Part I (Exact Results)", ACTA Informatica, Vol.7, No.35, pp 35-60, 1976

GIAM76 Giampo, T., "Validation of a Computer Performance Model of the Exponential Queuing Network Family", Proceedings of the Internation Symposium on Computer Performance, Modelling, Measurement and Evaluation, pp 44-58, Harvard University, Cambridge Mass., 1976

GORD67 Gordon, W.J. and Newell, G.F., "Close Queuing Systems with Exponential Servers", Operations Research, Vol.15, pp 254-265, 1967

GORD67a Gordon, W.J. and Newell, G.F., "Cyclic Queuing Systems with Restricted Length Queues", Operations Research, Vol.15, pp 266-277, 1967

GOSD62 Godsen, J.A., Sisson, R.L., "Standardised Comparisons of Computer Performance", Proc. 1962 IFIP Cong. pp 57-61

GRAU75 Grau, K. and Byers, J., "An Analysis of a Network of Finite-Queue, Multiple-Server Facilities", University of Missouri Computer Science Department Report, November 1975.

GROC72 Grochow, J.M., "Utility Functions for Time-Sharing System Performance Evaluation", Computers, pp 16-19, September/October, 1972

GROS53 Grosch, H.R., "High Speed Arithmetic - the Digital Computer as a Research Tool", J. of Optical Society of America, Vol.43, No.4, April, 1953

HARR78 Harrison, P.G., "Structured Modelling of a Multiprogramming Computer System Configuration using Queuing Network Analysis", Imperial College Technical Report, June 1978

HERB55 Herbst, E.H., Metropolis, N., and Wells, M.B., "Analysis of Problem Codes on MANIAC", Math Tables and Other Aids to Computation", Vol.9, No.9, pp 4-21, 1955

HILL67 Hillier, F.S. and Boling, R.W., "Finite Queues in Series with Exponential or Erland Service Times - A Numerical Approach", Operations Research, Vol.15, pp 286-303, 1967

HINE77 Hine, J.H., Mitrani, I and Tsur, S., "The Control of Response Times in Multi-Class Systems by Memory Allocation", University of Newcastle upon Tyne, Report No. 98, 1977

HOLL68 Holland, F.C. and Merikallio, R.A., "Simulation of a Multiprocessing System using GPSS", IEEE Transactions, Systems and Cybernetics, Vol.SSC-4, No.4, pp 395-400, November 1968

JACK57 Jackson, J.R., "Networks of Waiting Lines" Operations Research, Vol.5, pp 518-521, 1957

JACK63 Jackson, J.R., "Jobshop-like Queuing Systems", Management Science, Vol.10, pp 131-142, 1963

JOHN71 Johnson, R.R., "Needed - A Measure for a Measure", Datamation, Vol.16, No.17, pp 22-30, December 1971.

KATZ66 Katz, J.H., "Simulation of a Multiprocessor Computer System", AFIPS Joint Computer Conf., Vol.28, pp 126-139, 1966

KEND59 Kendall, D.G., "Unitary Dilations of One-Parameter Semigroups of Markov Transition Operators, and the Corresponding Integral Representations for Markov Processes with a Countable Infinity of States", Proc. London Math Soc., (3), 9, pp 417-431.

KIMB72 Kimbleton, S.R., "Performance Evaluation - A Structured Approach", AFIPS 1972 SJCC, pp 411-416, 1972

KING69 Kingman, J.F.C., "Markov Population Processes" Journal of Appl. Prob., Vol.6, pp 1-18, 1969

KLEI75 Kleinrock, L., "Queuing Systems: Volume I: Theory" John Wiley & Sons, Inc., New York, 1975

KLEI76 Kleinrock, L., "Queuing Systems: Volume II: Computer Applications", John Wiley & Sons, Inc., New York,1976

KOBA74a Kobayshi, H., "Application of the Diffusion Approximation to Queuing Networks I: Equilibrium Distributions", J. of ACM, Vol.21, No.2, 1974.

KOBA74b Kobayashi, H., "Application of Diffusion Approximation to Queuing Networks II: Non-equilibrium Distributions and Applications to Computer Modelling", J. of ACM, Vol.21, No.3, pp 459-469, 1974

KOBA 75a Kobayashi, H. and Reiser, M., "On Generalization of Job Routing in a Queuing Network Model", IBM Rese. Report RC5252, Feb. 1975

KOBA75a Kobayashi, H., "System Design and Performance Analysis using Analytic Models", IBM Res. Report RA-75, December, 1975

KOBA77 Kobayashi, H and Konheim, A., "Queuing Models for Computer Communication System Analysis" IEEE Trans. and Communication, Vol. Com-25, No.1, pp 2-29, Jan. 1977

KOLE72 Kolence, K.W.,"Management Control of Installation Efficiency", Computer Systems Measurement, pp 447-461, 1972

KONH76 Konheim, A.G. and Reiser, M., "Finite Capacity Queuing Systems with Application in Computer Modelling", IBM Research Report RC 5827, 1975

KRZE76 Krzesinski, A, and Teunissen, P., "Stochastic Network Analysis Program (SNAP) - Users Manual" Report No. RW76-02, Dept. of Computer Science University of Stellenbosch, South Africa, 1976

KRZE77 Krzesinski, A. and Teunissen, P., "Efficient Computational for the Normalising Constant and the Statistical Measures of Mixed, Multiclass Queuing Networks", Univ. of Stellenbosch, Report No. RW77-04, South Africa, 1977

KRZE77b Krzesinski, A., and Teunissen, P., "A Multi-class Network Model of a Demand Paging Computer SystemP, ACTA Informatica, (to be published).

KRIT77 Kritzinger, P.S., Krzesinski,A.E., and Teunissem,P., "Performance Prediction of a Large Operational Computer System Using a Multiclass Queueing Network Model", Univ. of Stellenbosch Technical Report No. RW77-03, June, 1977

LEHM78 Lehman, M.M., "Performance Evaluation, Phenomenology, Computer Science", Proc. International Conference on the Performance of Computer Installations, North-Holland Publishing Company, 1978.

LAM77 Lam, S.S., "Queuing Networks with Population Size Constants", IBM J. of Res. and Devl., July 1977

LEWI71 Lewis, P.A. and Shedler, G.S., "A Cyclic-Queue Model of System Overhead in Multiprogrammed Computer Systems", J. of ACM, Vol.18, pp 199-220, 1971.

MATT70 Mattson, Gecsei, J., Slutz,D.R., and Trager, I.L., "Evaluation Techniques for Storage Hierarchies", IBM Systems J., Vol.9, No.2, pp 78-117, 1970

MCKI69 McKinney, J.M., "A Survey of Analytical Time Sharing Models", Computing Surveys, Vol.1, No.2, pp 115-116, 1969

MOOR71 Moore, C.G. III, "Network Models for Large-Scale Time-Sharing Systems" Tech Report. No.70-1, Dept. of Ind. Eng. Univ. of Michigan, April, 1971

MINT74 Muntz, R.R. and Wong, J., "Efficient Computational Procedures for Closed Queuing Networks with Product Form Solution", Modelling and Measuring, Note 17, UCCA, 1974

MUNT75 Muntz, R.R., "Analytic Modelling of Interactive Systems", Proc. IEEE, Vol.63, pp 946-953, 1975

NEUT68 Neuts, M.F., "Two Queues in Series with Finite Intermediate Waithing Room", Journal of Applied Probability", Vol.5, pp 123-142, 1968

REIC57 Reich, E., "Waiting Times When Queues are in Tandem", Ann. Math. Stats., Vol.28, pp 768-773, 1957

REIS73 Reiser, M. amd Kobayashi, H., "Recursive Algorithms for General Queuing Networks with Exponential Servers", IBM Research Report RC4254, March 1973.

REIS74 Reiser, M. and Kobayashi, H., "Accuracry of the Diffusion Approximation for Queuing Systems" IBM Journal of Research and Development, Vol.18, No.2, pp 111-124, 1974

REIS75 Reiser, M. and Kobayashi, H., "Queuing Networks with Multiple Closed Chains: Theory and Computational Algorithms", IBM Journal of Research and Development, Vol.19, No.3, pp 283-294, 1975

REIS75a Reiser, M., "QNET4 Users Guide", IBM Res. Report RC5842, IBM Watson Research Center, 1975

REIS76 Reiser, M., "Interactive Modelling of Computer Systems", IBM System Journal, Vol.4, 1976

REIS76a Reiser, M. "Numerical Methods in Seperable Queuing Networks", IBM Research Report RC5842, Feb. 1976, p 38.

ROSE76 Rose, C.A., "Validation of a Queuing Model with Classes of Customers", Proceeding of the International Symposium on Computer Performance Modelling and Evaluation, Harvard University, Cambridge, Mass, 1976

ROSE76a Rosen, S., "Digital Computers: History" in Encyclopedia of Computer Science, edited by Ralston, A. amd Meek, C.L., Petrocelli/Charter, New York, pp 474-487, 1976

SAUE75 Sauer, C.H., "Simulation Analysis of Generalised Queuing Networks", 1975 Summer Simulation Conference, pp 75-81, 1975

SAUE77 Sauer, C.H. and Woo, L.S., "Hybrid Analysis/Simulation: Distributed Networks", IBM Res. Report, IBM Watson Research Center, 1977

SCHE65 Scherr, A.L., "An Analysis of Time-Shared Computer Systems", MIT Press, Cambridge, Mass., 1965

SCHW75 Schwetman, H.D., "Workload Characterization: Why? What? How?", Technical Report GJ-41289, Purdue, University, Illinois, 1976

SEKI72 Sekino, A., "Performance Evaluation of Multi Programmed Timed Shared Computer Systems", Ph.D Dissertation, MIT, Cambridge, Mass., 1972

SHAR69 Sharpe, W.R., "The Economics of Computers", Columbia University Press, New York, 1969.

SREE74 Sreenivasan, K. and Kleinman, A.J., "On the Construction of a Representative Synthetic Workload", C.ACM 17, No.3, March, 1974

SVOB76 Svobodova, L., "Computer Performance Measurement and Evaluation Methods", Analysis and Applications Elsevier, New York, 1976

WALL66 Wallace, V.L. and Rosenberg, R.S., "Markovian Models and Numerical Analysis of Computer System Behaviour", AFIPS SJCC, pp 141-148, 1966

WHIT75 White, J.A., Schnidt, J.W. and Bennett, J.W., "Analysis of Queuing Systems", Academic Press, New York, 1975

WILL76 Williams, A.C. and Bhajdiwad, R.A., "A Fenerating Function Approach to Queuing Network Analysis of Multiprogrammed Computers", Networks, Vol.6, No.1, January 1976.

WYSZ74 Wyszewianski, R.J., "Feedback Queues in the Modelling of Computer Systems: A Survey", NTIS AD-782 360/2WC, May 1974.