ASQ MANUAL

Tom W. Keller

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-27 October 1973

ASQ MANUAL

TOM W. KELLER

OCTOBER 1973 TR-27

USER'S GUIDE TO A PROGRAM FOR THE AUTOMATIC ANALYSIS OF QUEUEING NETWORK MODELS

TOM W. KELLER
DEPT. OF COMPUTER SCIENCES
UNIV. OF TEXAS AT AUSTIN
AUSTIN, TEXAS 78712

ACKNOWLEDGMENTS

Charles Hoch is responsible for the programming of the current version of ASQ. His months of effort have made ASQ more accessable to the user and the FORTRAN code transportable to other machines.

This work was supported in part by NSF Grant GJ-1084.

CONTENTS

I. INTRODUCTION

This document is a user's manual to ASQ, a program for the rapid and inexpensive analysis of certain queueing network models. ASQ (for Arithmetic Solutions to Queues) is the implementation of a technique of queueing network analysis utilizing the concept of a local balance [1-5]. This guide does not define all the classes of queueing networks to which this technique can be applied, nor does it attempt to teach the user how to model his particular problem as a queueing network. It does attempt to provide the analyst with access to a tool for describing and analyzing queueing network models.

In contrast to many manuals' approach of using formal and elegant but frequently indecipherable descriptions of syntax, it was decided to use examples as the primary means of describing the use of the ASQ system--supplemented where necessary by syntax-defining graphs.

The rest of this section defines the models which ASQ can analyze, some syntatic details holding for the entire system, and notation common to the rest of the manual. Section II is devoted to how the user describes a model to the program. Section III concerns the commands by which steady-state characteristics of the model are obtained, and how a model may be changed in the course of a run. Section IV is composed of examples of ASQ runs.

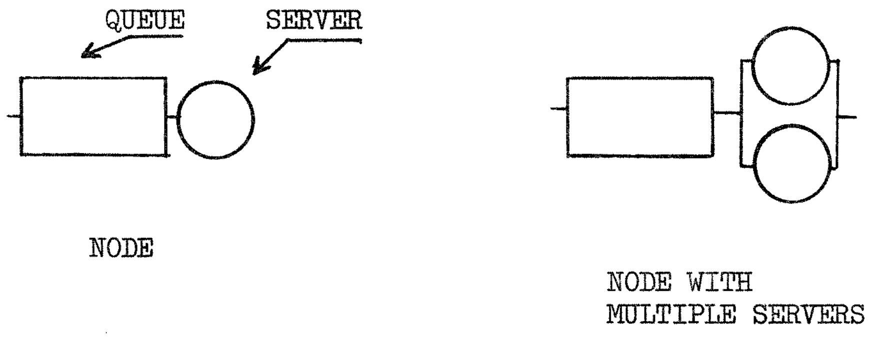
1.1 QUEUEING NETWORKS

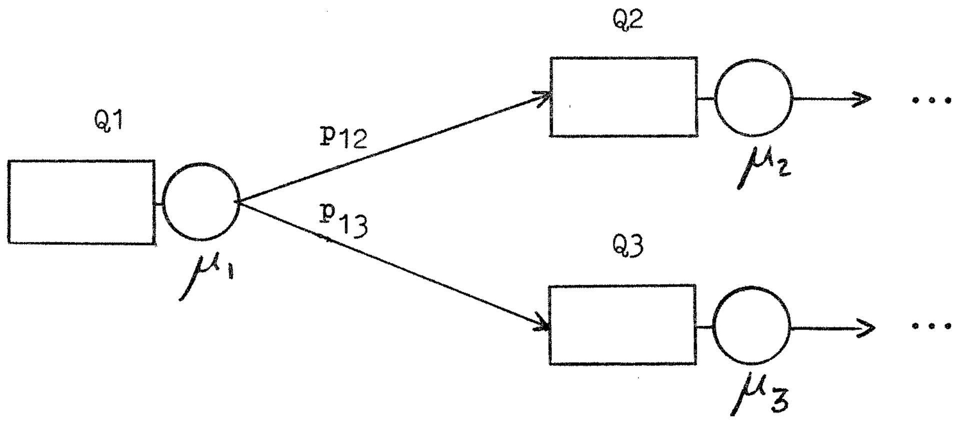
A <u>queueing network</u> is a system of interconnected servers and queues. A <u>server</u> (device, processor) services a <u>job</u> (customer, task) which must wait for service in the queue associated with that server. For ease in exposition, a server and its queue will be called a <u>node</u>. Upon completion of service at a device a job leaves that device's queue and enters the queue of another device, there to await service. Job(s) receiving service still remain in the server's queue.

Jobs move from queue to queue in a probabilistic fashion. This is reflected in a set of branching probabilities. The probability that a job enters node j upon completion of service at node i is the branching probability p_{ij}. The amount of time a customer is served at a device is governed by a probability distribution, the service distribution for that device. The manner in which customers enter, wait, and are served in a queue is the queueing discipline associated with that queue. An example of a queueing discipline commonly encountered is the First Come First Served (FCFS) discipline.

A queueing network is closed if jobs may neither enter nor leave the network, thus making the number of jobs in a closed network a constant. A queueing network in which a job enters the network precisely when another job exits can be considered closed, as the number of jobs in the system remains constant. The constant number of jobs is called the degree of multiprogramming.

A queueing network is open if it is fed by external Poisson sources and emptied by sinks. Jobs are emitted from a source at random intervals but at a fixed average rate. Jobs leave the network through the sinks. Branching probabilities define the probabilistic paths a job may take in its traversal of the network. The network may be connected such that a job can receive service many times or only once at any given server in its traversal of the network.





μ_1: mean service rate of Q1

p_{12}: branching probability - the probability that a job goes to node Q2 for service upon completing service at node Q1

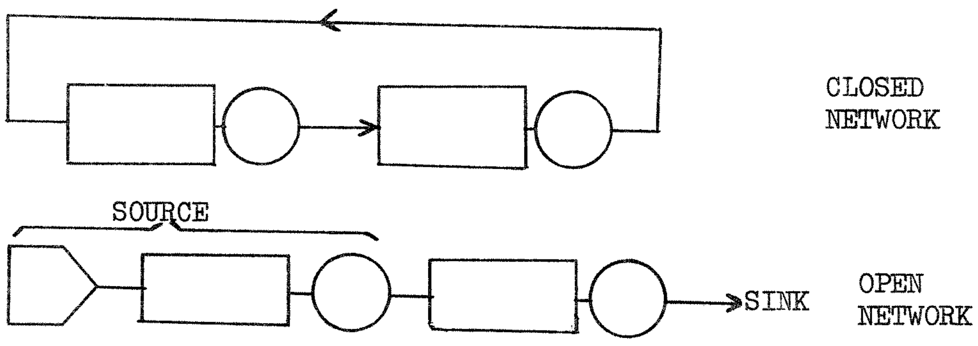


FIGURE 1.0: Legend for Queueing Network Figures Encountered Throughout the Manual

Queues in the network may be of finite or infinite capacity and belong to any of the following queueing disciplines:

First Come First Served (FCFS)

> Jobs enter the queue and receive service in the order of entry. The device satisfies a single job's service need (i.e. completes its service request) before initiating service on other job requests.

Processor Sharing (PS)

> The PS discipline is the limiting case of the zero over-head Round Robin Fixed Quantum discipline as the time quantum becomes arbitrarily small. All jobs in the queue receive service simultaneously. If n jobs are being serviced by the device then each job receives service at 1/nth the rate afforded a single job by the device, $n > 0$.

Last Come First Served Preemptive Resume (LCFSPR)

> The queue is a stack. The job most recently entering the queue receives service to completion of its service request or until it is preempted by another job entering the queue. When a job is preempted it is "pushed down" to await service.

The rate at which servers process jobs is determined by their service distributions. Any differentiable service distribution is permissable for devices in PS and LCFSPR disciplined nodes. The service distribution for FCFS disciplined nodes must be exponential. In either case, a service distribution is defined to ASQ by its mean service rate. The mean service rate of a node can be a function of the number of jobs in the node's queue. This case is specified to ASQ by listing the different mean service rates for a node, where

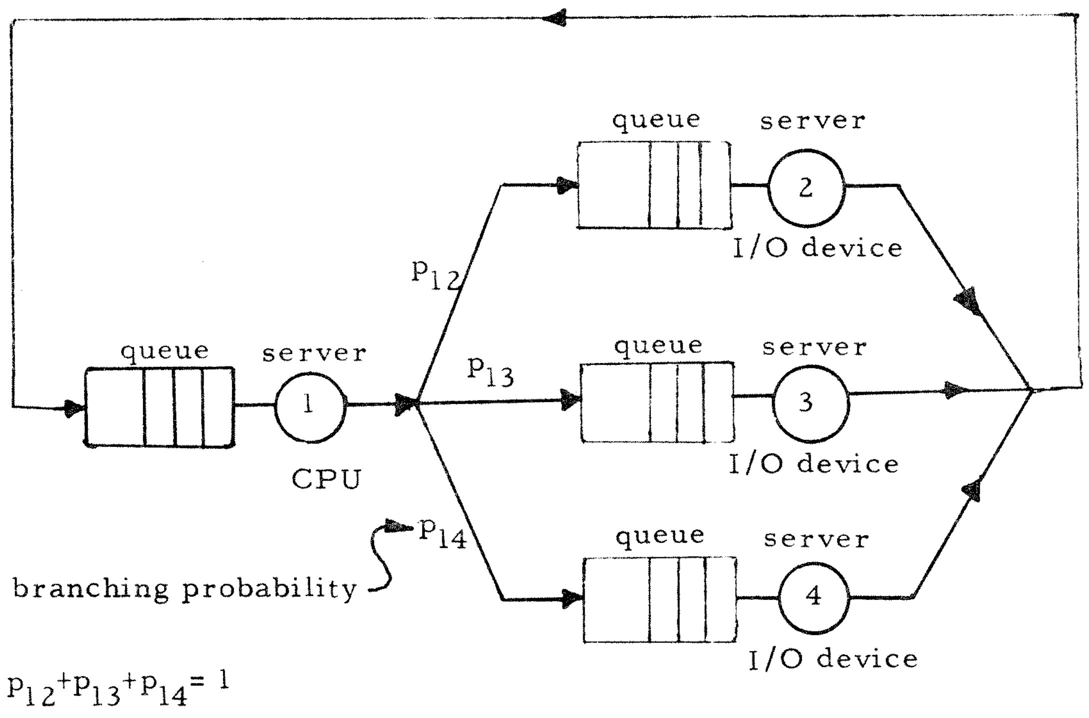


FIGURE 1.1 A CLASSICAL MODEL OF A MULTIPROGRAMMED COMPUTER

the first rate is that when the queue holds 1 job, the second for 2 jobs, ... , and the k^{th} rate for k or more jobs in the queue. An example of this case is a FCFS queue served by two identical devices, each with service rate λ . The mean rate at which the node processes jobs is λ when only one job is in the queue, as one device is empty, but 2 λ when two or more jobs occupy the queue.

An example of an application of closed queueing networks is in the modelling of a multiprogrammed computer system. Jobs continually circulate through the network, receiving service at the CPU and I/O device alternately. Figure 1.1 illustrates such a model. The network is labeled according to the conventions described.

1.2 JOB TYPING

Jobs in the network may be partitioned into types, where each type is a class of equivalent jobs. Jobs belong to the same type if they observe the same branching probabilities and service distributions throughout the network. Branching probabilities and mean service rates in the network are specified for each job type. An example where job typing might be useful is in a multiprogrammed computer with a job mix of CPU-bound and I/O-bound jobs. We can label the CPU-bound jobs type 1 and the I/O-bound type 2. The CPU-bound jobs will have larger mean CPU-burst times as compared to their I/O burst times and vice-versa for the I/O-bound jobs. We would then specify two mean service rates for all devices, the first for the type 1 jobs and the second for type 2 jobs. Figure 1.2 illustrates a simple model of a multiprogrammed computer system with two identical I/O devices and one CPU. If all jobs in the system were equivalent then there would be only one type of jobs in the network and we would describe job behavior by specifying μ , λ , and p . However, if we wished to consider the above two types of jobs in the system then we would specify μ_1, μ_2, λ_1, λ_2, p_1 , and p_2 where the parameter's index specifies its value for either type 1 or type 2 jobs.

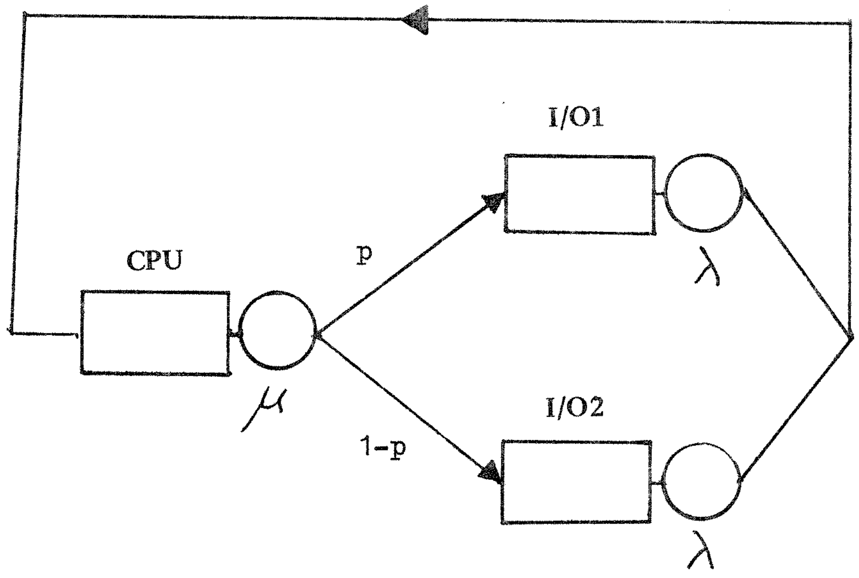
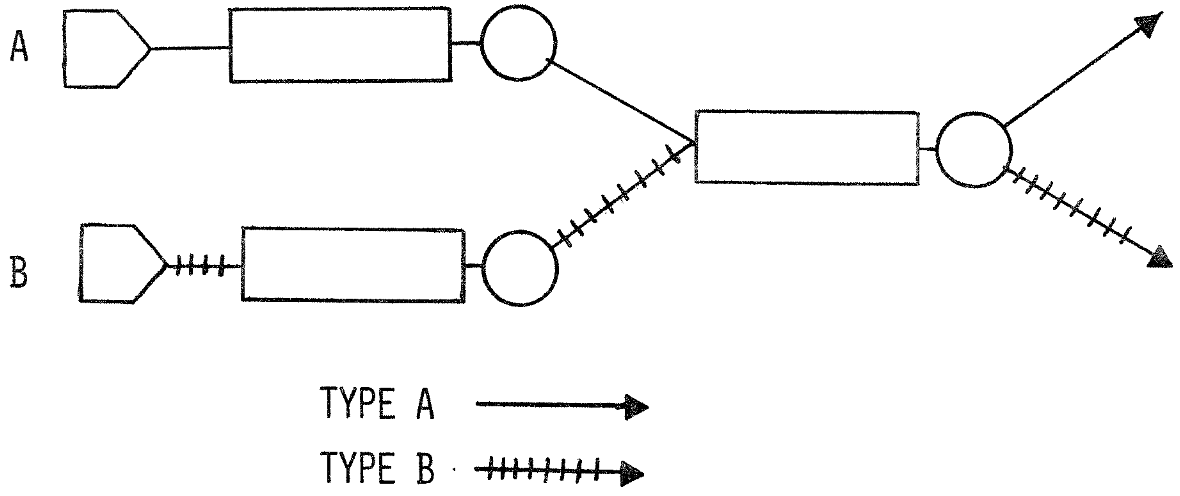


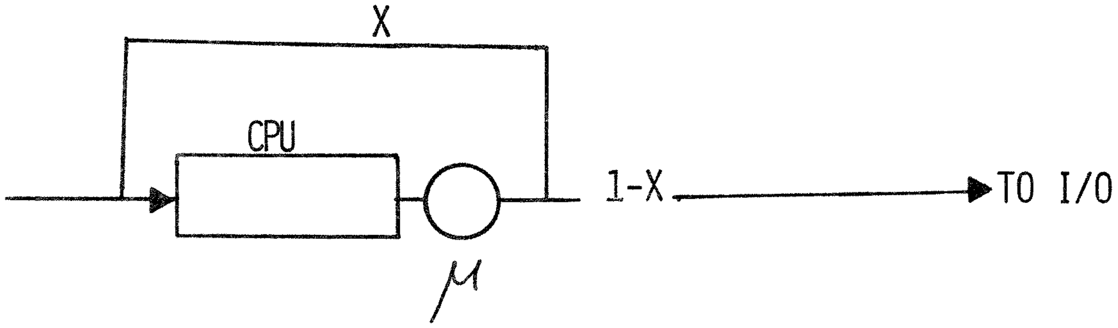
Figure 1.2: Simple Model of a Multiprogrammed Computer

Typical values might be μ_1 = 4 jobs/second, μ_2 = 500 jobs/second, λ_1 = 20 jobs/second, λ_2 = 5 jobs/second, $p_1 = p_2 = \frac{1}{2}$.

Figure 1.3 illustrates how different paths through a network may be specified for two job types. In the lower network fragment a job may generate another CPU request rather than an I/O request with a probability dependent upon job type.

Jobs are allowed to probabilistically change their type when traversing a branch between nodes. If this option is exercised by the user then the network is said to support <u>dependent</u> job typing. The number of jobs in each type is not constant for dependent job typing, so for closed networks the degree of multiprogramming is specified for





TYPE A: CPU BOUND X = .8 μ = 20 JOBS / SEC

TYPE B: I/O BOUND X = .1 μ = 500 JOBS / SEC

FIGURE 1.3: JOB TYPING

the total number of jobs, regardless of job type. If jobs are not allowed to change type in the network then the network is said to support independent job typing, and a degree of multiprogramming for each job type must be specified for a closed network.

Figure 1.3 illustrates how different paths through a network may be allowed for two independent job types. In the lower network fragment a job may generate another CPU request (re-enter the CPU queue) rather than an I/O request with a probability dependent upon job type.

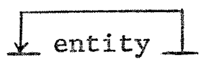
1.3 NOTATION

An attempt has been made to steer clear of the Backus Normal Form when describing syntax. Instead, an equivalent graphical description is used, supplemented with many examples. It is felt that this approach is superior when rapid orientation by a user is required.

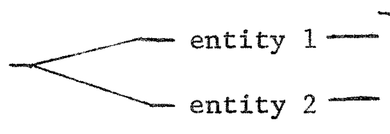
The notation is as follows:

-- All upper case words are required when the format is used.

-- Lower case words are generic terms which must be replaced by appropriate names or values.



entity is to be repeated one or more times



entity 1 | entity 2

either entity 1 or entity 2 is chosen but not both. If entity 2 is missing then entity 1 is optional.

1.4 USER INPUT FORMAT

ASQ accepts format free input. Thus an arbitrary number of blanks may reside between syntatic units. ASQ scans user input card by card (or line by line) from left to right. Entries may be spaced one to the card or may entries may be made across the card. It is strongly recommended that the same spacing format which is used in the examples be followed by the user (one command per card, one edge per card, etc.) due to the nature of the echo-print for batch runs. In the case of a series of commands, all on one card, ASQ would echo print the card and then print the results of executing all the commands below. This can lead to the user confusing which results belong to which command. In short, ASQ does not label results for the user. The lines of input themselves suffice for labels.

1.4.1 Representation of Numbers

Real numbers (syntatic type <u>real</u>) may or may not contain a decimal point. Integer numbers may not contain a decimal point. A negative number being input will eventually lead to a program error. No provision is currently made for symbolic input. Scientific notation in the FORTRAN E format is acceptable. Also, real fractions may be input in the form (num_1/num_2) where num_1 and num_2 are real or integer numbers. ASQ evaluates num_1/num_2 to the appropriate real number. This form is convenient when inputting branching probabilities and service rates. Examples of some acceptable forms of real numbers are:

```
1.34          1
0.37          5
 .37          1.52E5
(1/5)         1.23E-02
(1/9.5)
```

1.4.2 Representation of Labels

Labels are the names of nodes and branching points. Only legal FORTRAN variables are permissable as labels.

1.5 BATCH ASQ VS. INTERACTIVE ASQ

The interactive code with a few trivial modifications suffices for the batch version. The differences are in how files are handled and the batch version echo-print of user input. In the interactive version ASQ prompts the user for the appropriate input. Thus, when ASQ expects the rates for FCFS nodes to be input the program prompts the user with the message:

```
INPUT RATES FOR NODES WITH FCFS DISCIPLINE
```

No promptings are possible to the batch user. For this reason batch input decks must be carefully assembled, making certain the items required for the network description are input in the correct order with correct syntax.

II. THE ORIGINAL NETWORK DESCRIPTION

The queueing network is originally specified to ASQ in 10 steps:

1) The file upon which the network description resides
2) The number of job types
3) The number of nodes plus the number of branching points
4) The number of nodes
5) Whether the network is open or closed
6) The edges
7) The nodes which are also sources if the network is open
8) FCFS nodes and their rates
9) PS and LCFSPR nodes and their rates and
10) The degree(s) of multiprogramming if the network is closed.

Step 7 is omitted if the network is closed and step 10 is omitted if the network is open. The syntax of the original specification follows, in the order in which the items must be input.

2.1 The file upon which the network description resides.

A response of "SAME" for interactive users results in ASQ expecting the network description to be entered interactively. A response of "OTHER" from interactive users results in ASQ expecting the network description to reside on some other file. Setting a control card macro parameter to some value netwkfile , and replying with the string "OTHER" will result in ASQ expecting the network description to reside on file netwkfile .

Similarly, batch users should set this entry to "SAME" if the network description is on the input file, that is, the description resides on cards immediately following the card punched "SAME" . The entry "OTHER" tells ASQ to read the network description from file

<netwkfile> .

2.2 The (integer) number of job types. If the network is typeless then all jobs are equivalent and the number of types is 1.

2.3 The total (integer) number of nodes and branching points in the network. Nodes and branching points will be called "point" in our discussion. A "node" is a single queue and its server(s). A "branching point" is a point from which a job, in its traversal of the network, may select a path but not queue at or receive service.

2.4 The (integer) number of nodes in the network.

2.5 YES or NO

YES ≡ the network is open

NO ≡ the network is closed

2.6 The edges.

An edge is defined as an ordered triple, (point_i prob_{ij} point_j), where prob_{ij} is a real number specifying the branching probability that a job travels to point_j upon exiting point_i. prob_{ij} must be less than or equal to 1.0. node_i and node_j are labels. Any allowable FORTRAN variable is an acceptable label. Thus CPU, IO, Q1, Q15, NODE12, and Z are all acceptable labels. An edge is specified to the program by a line of input of the form:

$$\text{point}_1 \;\; \text{prob}_{ij} \;\; \text{point}_j \;;$$

where the components of the edge are separated by one or more blanks. An edge is delimited by the character ";". A string of edges specifies the entire network topology to the program. The string of edges is terminated by special character "→". Sinks in open networks are unlabeled and their existence is assumed when the sum of branching probabilities from a point is less than 1.0. A branch to a sink from the point is then created by ASQ to enforce the requirement that the sum of branching probabilities from a point equal 1.0. Figures 2.1 and 2.2 are examples of two networks and their edges.

A branching point is considered a point in the edges input merely for

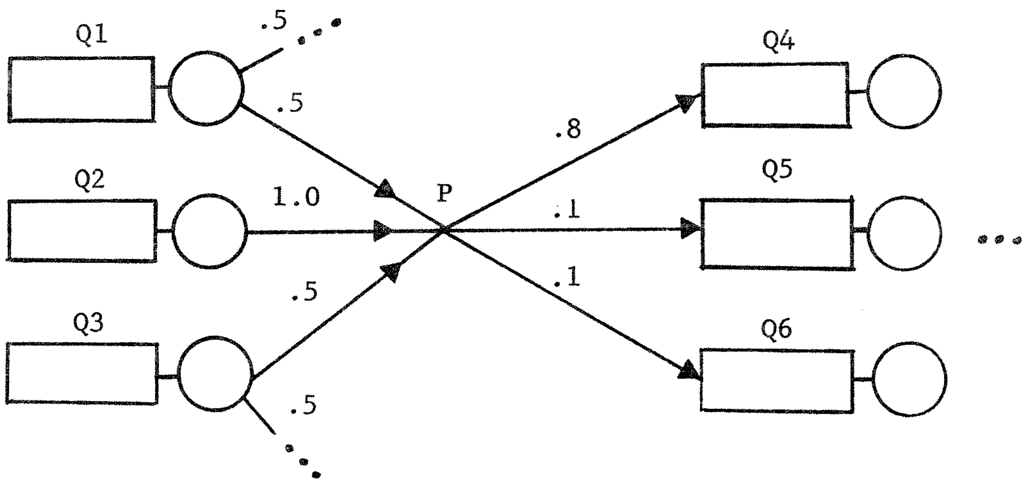
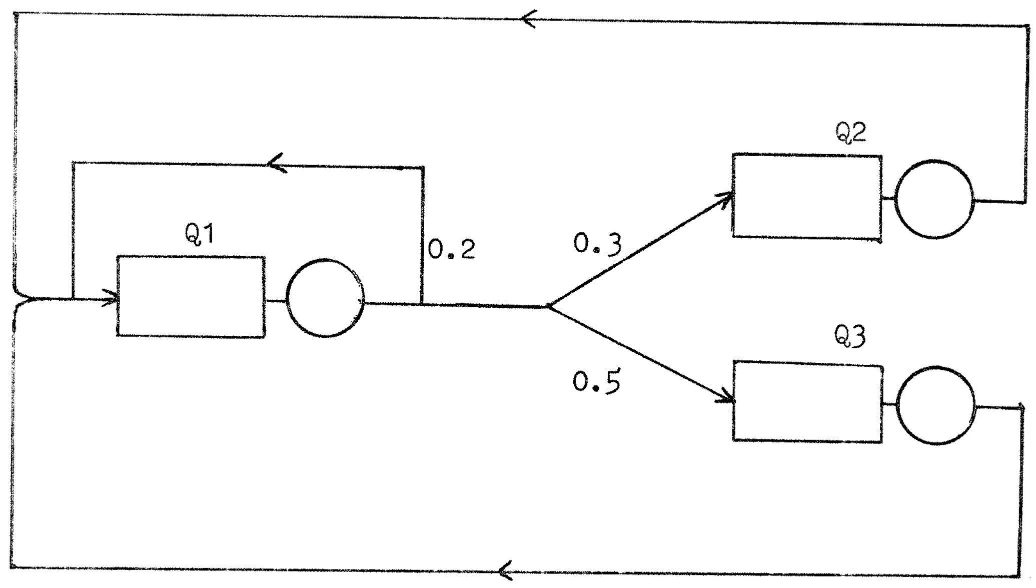
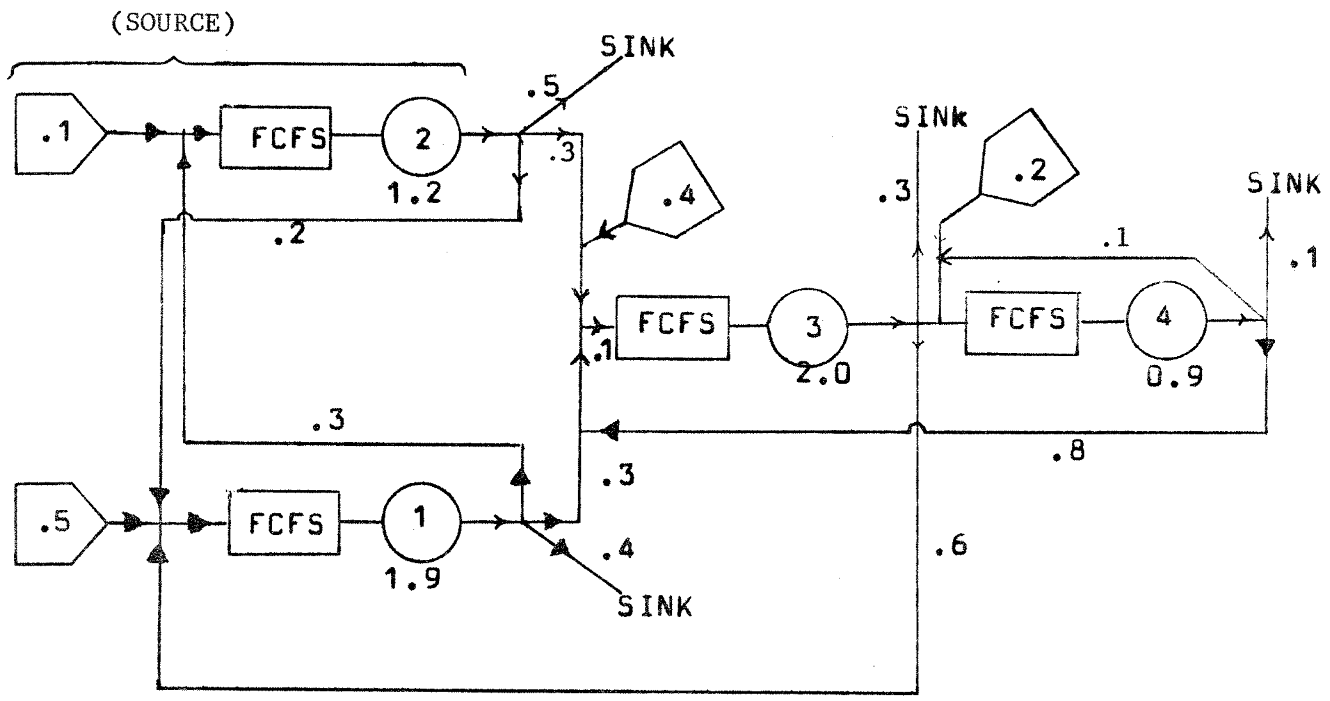


FIGURE 2.3



```
Q1  0.2  Q1 ;
Q1  0.3  Q2 ;
Q1  0.5  Q3 ;      EDGES
Q2  1.0  Q1 ;
Q3  1.0  Q1 ;
```

FIGURE 2.1: AN EXAMPLE OF A CLOSED NETWORK AND ITS EDGES



AN OPEN NETWORK

```
Q1        0.3   Q2 ;   }
Q1        0.3   Q3 ;   }
Q2        0.2   Q1 ;   }
Q2        0.3   Q3 ;   }
Q3        0.6   Q1 ;   }  EDGES
Q3        0.1   Q4 ;   }
Q4        0.8   Q3 ;   }
Q4        0.1   Q4 ;   }
→

Q1         .5 ;        }
Q2         .1 ;        }  SOURCES
Q3         .4 ;        }
Q4         .2 ;        }
→
```

FIGURE 2.2 AN OPEN NETWORK AND ITS EDGES AND SOURCES

user convenience. Consider the partial network in figure 2.3 Since P is a point then the set of edges shown is

```
Q1    0.5    P;
Q2    1.0    P;
Q3    0.5    P;
P     0.8    Q4;
P     0.1    Q5;
P     0.1    Q6;
```

which is a simpler set to specify than the equivalent:

```
Q1    0.4    Q4;
Q1    0.05   Q5;
Q1    0.05   Q6;
Q2    0.8    Q4;
Q2    0.1    Q5;
Q2    0.1    Q6;
Q3    0.4    Q4;
Q3    0.05   Q5;
Q3    0.05   Q6;
```

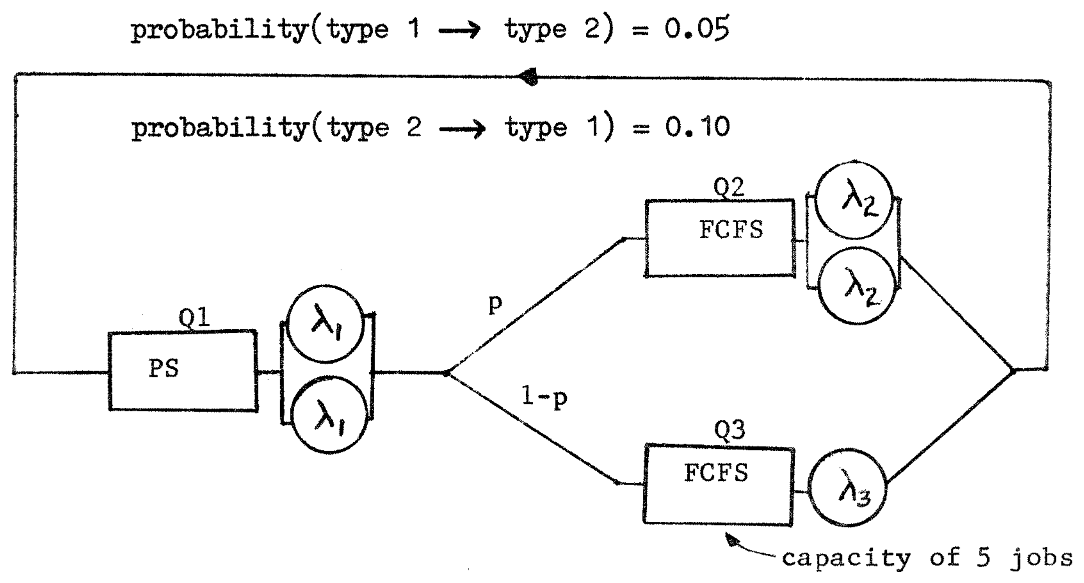
2.6.1 Edges for Networks with Job Typing

If the network has more than one job type then a set of edges for each job type must be specified to ASQ. This is because a different network topology can exist for each type of job. Each job type has an identifying integer type#, where type# = 1, 2, ..., L for L job types. The edges for each job type type# are input sequentially, type 1 first, type 2 second, etc. until all the edges are input by type. The format is identical to the typeless (1 job type case) except that

```
TYPE type#
```

is input before each set of edges, where type# is the type identifying integer.

Two forms of job typing, independent and dependent, exist. Branching probabilities for dependent typed networks specify not only the probability that a job transits to some other queue upon a service completion, but


probability(type 1 ⟶ type 2) = 0.05
probability(type 2 ⟶ type 1) = 0.10
Q1
PS
λ1
λ1
p
1-p
Q2
FCFS
λ2
λ2
Q3
FCFS
λ3
capacity of 5 jobs

TYPE 1 JOBS

$\lambda_1 = 3.1$ $\lambda_2 = 10.2$ $\lambda_3 = 8.7$ $p = 0.8$

TYPE 2 JOBS

$\lambda_1 = 2.7$ $\lambda_2 = 10.2$ $\lambda_3 = 8.7$ $p = 0.7$

Figure 2.4: A CLOSED NETWORK WITH DEPENDENT JOB TYPING

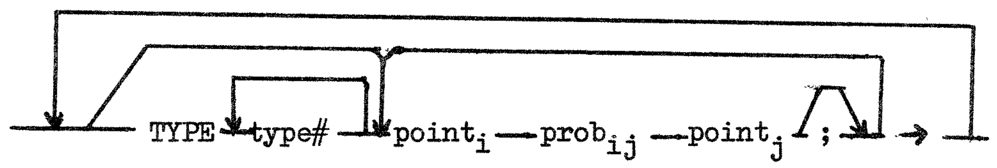


Figure 2.41: Syntax of Section 2.6 Input

also that the job changes type during that transition. An integer type# following a colon in an edge specifies to which job type the job will change. If the colon is absent in the edge then no type transition is assumed for that edge. Thus for the network of figure 2.4 the edges are specified by

```
TYPE 1
Q1    0.8    Q2;
Q1    0.2    Q3;
Q2  :2  .05  Q1;
Q2      .95  Q1;
Q3     1.0   Q1;
→
TYPE 2
Q1    0.7    Q2;
Q1    0.3    Q3;
Q2  :1  .10  Q1;
Q2      .90  Q1;
Q3     1.0   Q1;
→
```

If two or more job types have identical sets of edges, then

TYPE type#$_1$ type#$_2$ ... type#$_m$

preceding the common edge set will specify the common edge set as the type type#$_1$ set, also the type type#$_2$, edge set, etc. for the m job types with identical network topologies. Caution: one and only one edge set may be specified for each job type. Thus TYPE 3, for example, cannot appear more than once in the input of the edges.

2.7 The External Poisson Sources and Their Rates

An external Poisson source feeds jobs into an open network at some fixed average rate but with the interval between jobs being random. In an open network an arrival rate for incoming jobs is associated with every node. Those nodes at which jobs first enter the network are called "sources". The sources and their fixed average rates are specified by a string of (node rate) pairs, separated by blanks and delimited by semicolons.

The character "→" delimits the set of nodes and source rates. Rates are real numbers. Inputting an integer rate will result in an error. For figure 2.2 the sources are

```
Q1    0.5;
Q2    0.1;
Q3    0.4;
Q4    0.2;
→
```

When an open network accomodates more than one type of job, the set of sources emitting each job type is specified by a

```
TYPE  type#
```

being input before the set of sources emitting type type# jobs. The → delimiter is used only when all sources for all job types have been input. A unique set of sources is associated with each job type.

The user must remember that an open queueing network will not reach steady-state equilibrium if the rate of jobs entering some node equals or exceeds that node's processing capability. For example, a node with a job arrival rate of 4.0 and a single average service rate 3.0 is not allowed. An error message will result if the average rate of jobs entering a node exceeds the first service rate specified for that node. It may be the case that succeeding service rates for the node (see section 2.6) will extend the processing capability of the node so that the steady-state can be achieved. For this reason the user may ignore the error message. If the steady-state is not achievable, the program will terminate upon execution of the first evaluation command.

This section is omitted if the network is closed.

2.8 FCFS Nodes and Their Rates

This section specifies the average service rates for FCFS disciplined nodes. The service rate(s) for a FCFS node cannot be a function of job type. Thus the service rate(s) for a FCFS node is specified only once in this section. If no FCFS nodes appear in the network then the entry for this section is merely the delimiter "$\rightarrow$".

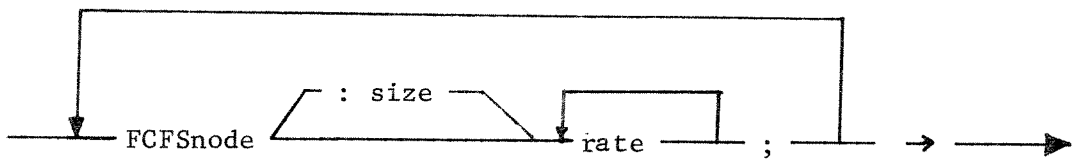


Figure 2.5 Syntax of Section 2.6 Input

A node's queue may be of finite capacity, that is, there exists a maximum number of jobs that the queue can hold. This capacity, or "size", is specified by a : ⟨integer⟩ following the node label. The integer number is the capacity. If the :⟨integer⟩ is omitted then the queue for that node is assumed to be of infinite capacity.

The average service rate for a node may be a function of the number of jobs in the queue of the node. This case is specified by a sequence of real numbers following the node label, representing the average service rates of the node when 1 job, 2 jobs, 3 jobs, ... occupy the queue. J elements in the sequence correspond to J different service rates for the

J cases of 1 job in the queue, 2 jobs in the queue, ..., to J or more jobs occupying the queue. When more than J jobs occupy the node the average service rate is the last number in the sequence.

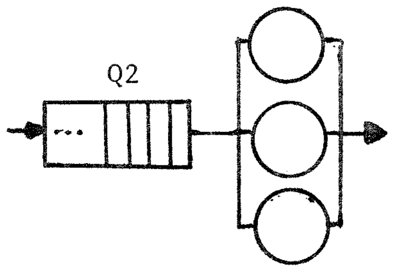


Figure 2.6

For example, consider the node in figure 2.6. Node Q2 processes jobs with 3 identical devices, each with some service rate, say 0.5. The service discipline is FCFS. Only 1 job can occupy a device and only one device can service a job at a time (no multitasking in this example). Thus the average service rate of node Q2 for 1 job in the queue is 0.5, for 2 jobs 1.0, and for 3 or more jobs is 1.5. This is specified by

```
Q2  0.5  1.0  1.5 ;
```

Another example might be a server which processes increasing numbers of jobs simultaneously (multitasking), but with decreasing effeciency because of interference or overhead. In such a case the entry for the processor L2 might be

```
L2  0.5  0.4  0.2  0.1 ;
```

The input for this section for the queueing network in figure 2.4 would be

```
Q2  10.2  20.4 ;
Q3   8.7 ;
→
```

2.9 PS and LCFSPR Nodes and Their Rates

This section specifies the average service rates for PS and LCFSPR disciplined nodes. Since service rates for PS and LCFSPR nodes may be a function of job type (not the case for FCFS nodes), a set of nodes and their respective service rates are specified for each job type, in order of type identifying integer. Thus the service rates for PS and LCFSPR nodes which serve type 1 jobs are first input, then the rates for nodes serving type 2 jobs, etc. A node can serve more than one type of job , and must serve at least one type of job . If <u>no</u> PS or LCFSPR nodes appear in the network then the entry for this section is merely the delimiter "→ ".

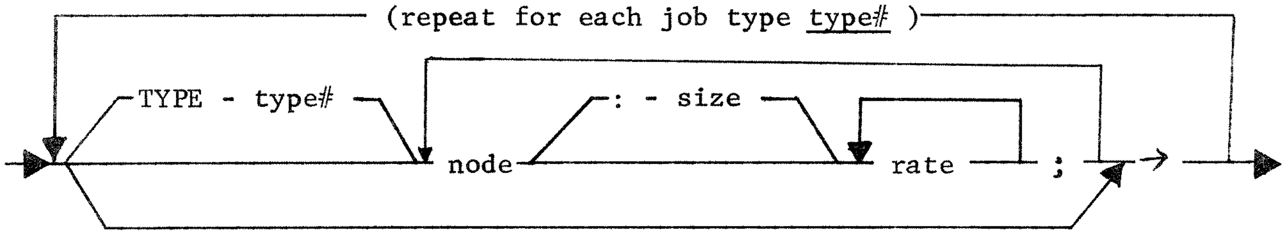


Figure 2.7 Syntax of Section 2.7 Input

Although the rate sequence for a PS or LCFSPR node may be different for different job types, the number of rates comprising the sequence must be the same for each job type. Thus if in the example in figure 2.5 PS node Q2 accomodated jobs of both type 1 and of type 2, with the service rate for each device being 0.5 for type 1 jobs but 0.6 for type 2 jobs, then the entry:

```
TYPE 1
⋮
Q2  0.5  1.0  1.5 ;
⋮
TYPE 2
⋮
Q2  0.6  1.2 ;
⋮
```

is not allowed, as 3 rates are specified for type 1 jobs but only 2 rates for type 2. If this was not an input error but was actually the representation of the node's behavior (say, for example, only 2 devices were available for type 2 jobs) then the above syntactic restriction could easily be bypassed by introducting the artifice of repeating the last rate in the shorter rate sequence to bring it up to the correct length. Thus if the last line of the example were:

```
Q2  0.6  1.2  1.2 ;
```

no error would occur.

Likewise, a queue cannot have different capacities for different job types. The last : integer associated with a node label, regardless of job type, is the queue capacity of that node for all job types. So

```
⋮
Q2  :8  1.3 ;
⋮
Q2  :7  1.6 ;
⋮
```

results in 7 being assigned to the queue capacity for node Q2 for all job types.

Consider the queueing network in figure 2.4. The rates for sections 2.6 and 2.7 would be specified by:

```
Q2  10.2  20.4 ;        )
Q3  :5  8.7  17.4 ;     }   FCFS nodes
→                       )
TYPE 1                  )
Q1  3.1  6.2 ;          }
→                       }   PS nodes
TYPE 2                  }
Q1  2.7  5.4 ;          }
→                       )
```

2.10 The Degree(s) of Multiprogramming

Syntax for typeless (1 job type) and dependent job typing:

M where M is an integer.

Syntax for independent typed networks:

$M_1\ M_2\ M_3 \ldots M_L$ where M_i is an integer and L is the number of job types.

M_i is the degree of multiprogramming for job type i, for independent job typing. The number of jobs of each dependent job type in a network with dependent job typing may vary but the total number (M) remains constant.

This section is omitted if the network is open.

2.11 The Template

Once the network is input to the program, ASQ issues a response defining the template. The template is of importance to the user only if he wishes to know specific steady-state probabilities. The order in which the node labels appear in the template defines the correspondence between labels and the integers in the state vectors. The template is defined by ASQ to be the order of appearance of the node labels in the edges.

III. COMMANDS

All user input to ASQ once the queueing network is specified is in the form of the following commands. The commands are partitioned into four groups. The first group (EVAL) is the network evaluation group which returns to the user information about the network at steady-state equilibrium. The second group (QUERY) interrogates the network representation data base. The third group (CHANGE) allows the user to change the current network by resetting values in the network representation data base. The fourth group (TABLE) returns results by repeatedly executing a member of the first group while varying some network parameter over a range of values.

3.1 Network Evaluation Commands

3.1.1 SSPB vector

SSPB returns the steady-state probability of state _vector_. The state vector is specified by a string of integers both seperated and delimited by blanks.

```
Examples:   SSPB 1 1 3 0
            SSPB 3 2 0 4 2 2 0 0 3
```

The number of integers in the vector must equal the number of nodes in the network. Each integer specifies the number of jobs occupying some queue when the network is in a particular state. The position of **an** integer in the vector string corresponds to the position of the node label in the template (see section 2.11).

3.1.2 SGPB node integer

SGPB returns the steady-state probability of there being _integer_ or more jobs in _node_.

Example: SGPB Q3 3 returns the steady-state probability of there being 3 or more jobs in node Q3.

3.1.3 SEPB node integer

SEPB returns the steady-state probability of there being exactly _integer_ number of jobs in _node_.

3.1.4 IDLE node

IDLE returns the fraction of time in the steady-state that no jobs occupy _node_.

3.1.5 UTIL node

UTIL returns the fraction of time in the steady-state that 1 or more jobs occupy <u>node</u>.

3.1.6 QLEN node

QLEN returns the mean number of jobs occupying <u>node</u> at steady-state.

3.1.7 TPUT node

TPUT returns the throughput of <u>node</u> (the mean rate at which jobs complete service and exit <u>node</u>).

3.1.8 WAIT node

WAIT returns the mean time interval between a job entering <u>node</u> and exiting <u>node</u> after completing service at the steady-state.

3.1.9 NORM

NORM returns the sum of all steady-state probabilities before normalization, the "normalization constant."

3.2 NETWORK QUERY COMMANDS

3.2.1 JOB

JOB, for a closed network, returns the total number of jobs of each type in the network. A response of 4 3 2 would indicate 4 jobs of type 1, 3 jobs of type 2, and 2 jobs of type 3 circulating through the network.

3.2.2 PRNETWK

PRNETWK describes the network in terms of the internal variables used by ASQ. Its use results in an "ASQ DUMP".

3.3 NETWORK CHANGE COMMANDS

The CHANGE command allows the user to restructure the topology and other characteristics of the network. The total number of jobs in a closed network may be changed. Any subset of the edges or the rates of the network may be changed. The degree(s) of multiprogramming may be changed. However, only the edges or rates or degree of multiprogramming for a single job type may be changed during one execution of the CHANGE command. The capacity of any number of queues in the network may be changed during one execution of CHANGE.

3.3.1 CHANGE EDGES

This command allows any subset of the edges for a single job type to be changed. New paths may be created between existing nodes. Any edges not specified in the text of the command are left unchanged.

```
          CHANGE EDGES : Q1  0.3  Q2;
                         Q3  0.9  Q2;
                         →
```

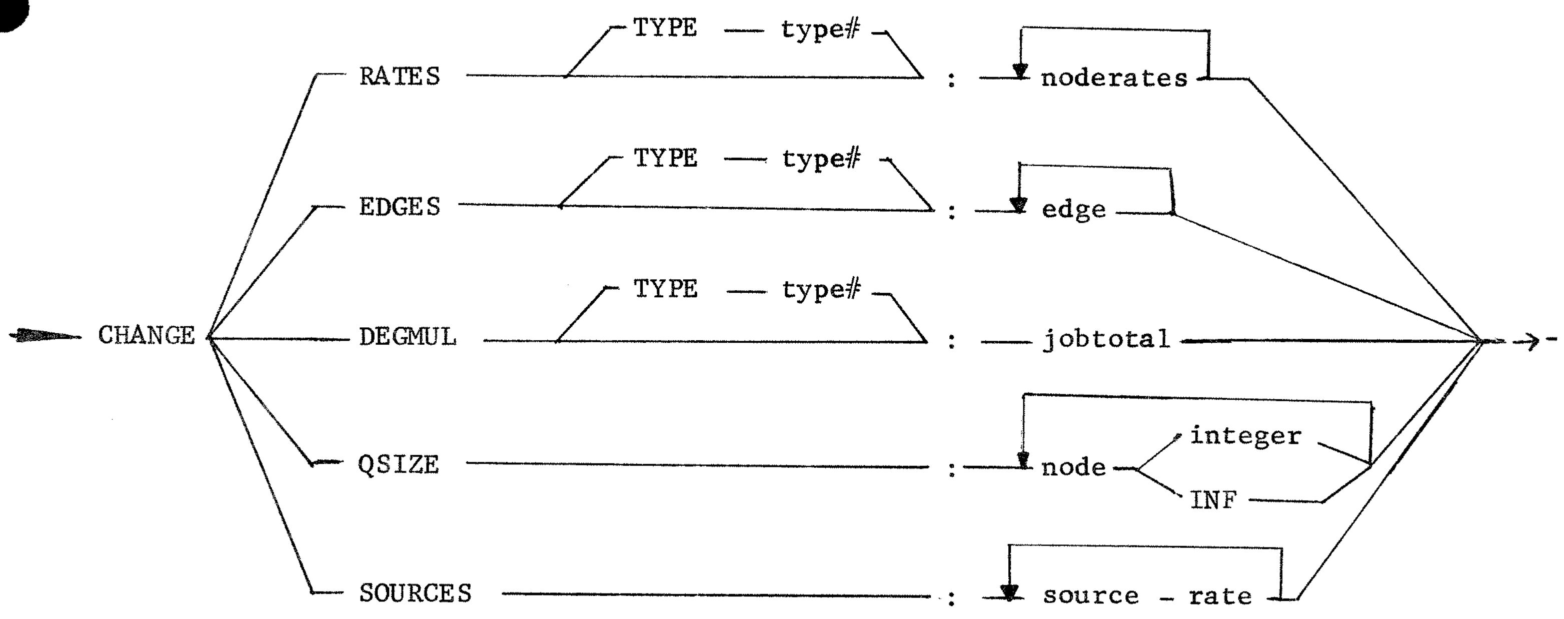
for a network with no job typing results in the branching probability from node Q1 to node Q2 being set to 0.3 and the branching probability from node Q3 to node Q2 being set to 0.9.

```
          CHANGE EDGES TYPE 2 : Q4  0.2  Q5; →
```

results in the branching probability for type 2 jobs from node Q4 to node Q5 to be set to 0.2. A path between nodes may be deleted by setting its probability to 0.0.

```
          CHANGE EDGES TYPE 1 : Q1 0.0 Q3; →
```

destroys the path from node Q1 to node Q3 for type 1 jobs. A path can be created in an equivalent manner, but only if both nodes were specified in the original network description. Nodes can neither be created nor



jobtotal ::= integer

type# ::= integer

noderates ::= node servicerate ;

servicerate ::= real

edge ::= point : type # branchingprobability —point— ;

branchingprobability ::= real# (real# ≤ 1.0)

Figure 3.1 Syntax of the CHANGE Command

destroyed, only the paths between them.*

3.3.2 CHANGE RATES

This command allows the average service rates for any number of existing nodes to be reset, with the restriction that only one type may be specified for each execution of the CHANGE RATES command.

```
CHANGE RATES : Q1 1.3 ; Q5 0.9 ;  →
```

for a network with no job typing results in the service rate of node Q1 being set to 1.3 and the average service rate of node Q5 being set to 0.9.

```
CHANGE RATES : Q3 0.9 1.2 1.5 ;
               Q1 0.2 1.2 5.3 ;
               →
```

for a network with no job typing results in the service rates for nodes Q3 and Q1 to be reset. The average service rate of Q3's server is set to 0.9 for 1 job occupying the node, 1.2 for 2 jobs, and 1.5 for 3 or more jobs ; similarly for node Q1.

```
CHANGE RATES TYPE 2 : Q5 0.9; Q7 1.43; CPU 1000.0 ; →
```

results in the average service rates for nodes Q5, Q7, and CPU to be reset for type 2 jobs. The service rates for other job types for these nodes are not changed.

The service rate for a node may be a function of job type or the number of jobs in the queue, or both: if it is the case that service rates are specified by both job type and the number of jobs in the PS or LCFSPR queue, then the number of rates specified for each type must be the same. Thus

*This restriction will be removed when certain program modifications are made.

```
CHANGE RATES TYPE 2  :  Q1 0.9 0.8; →
CHANGE RATES TYPE 1  :  Q1 0.3; →
```

is not allowed, as the number of rates following the node label must be equal for each job type.

Note: If the original network description had a larger number of rates (say M) for a node than the number specified in the CHANGE RATES command (say L), then those M - L rates unspecified by CHANGE are set to the Lth rate given in the CHANGE command. Thus if

```
Q2 1.1 2.1 3.5 ;
```

appeared in the original description and

```
CHANGE RATES : Q2 1.1 2.0 ; →
```

was given, then the resulting rates of node Q2 would be

```
1.1 2.0 2.0 .
```

3.3.3 CHANGE QSIZE

The CHANGE QSIZE command allows the user to reset the queue capacity of one or more nodes in the network. The entity INF is used to specify infinite queue capacity. In the original network description, if no queue capacity was specified, INF was assumed.

```
CHANGE QSIZE  :  I01    10
                 I02    15
                 I03    INF
                 CPU    100
                 →
```

results in the queue capacity of node I01 being set to 10, of I03 being set to infinity, etc.

3.3.4 CHANGE DEGMUL

This command is applicable only to closed networks, and resets the total number of jobs (degree of multiprogramming).

```
CHANGE DEGMUL  :  7 →
```

results in the degree of multiprogramming for a typeless closed network being reset to 7.

```
CHANGE DEGMUL TYPE 3  :  4 →
```

results in the degree of multiprogramming for type 3 jobs being set to 4.

3.3.5 CHANGE SOURCES

The CHANGE SOURCES command allows the user to reset the source rates of one or more nodes in the network.

```
CHANGE SOURCES  :  Q1  14.7
                   Q2   7.3
```

results in the source rate of node Q1 being set to 14.7 and of node Q2 to 7.3.

3.3.6 Limitations of the CHANGE Command

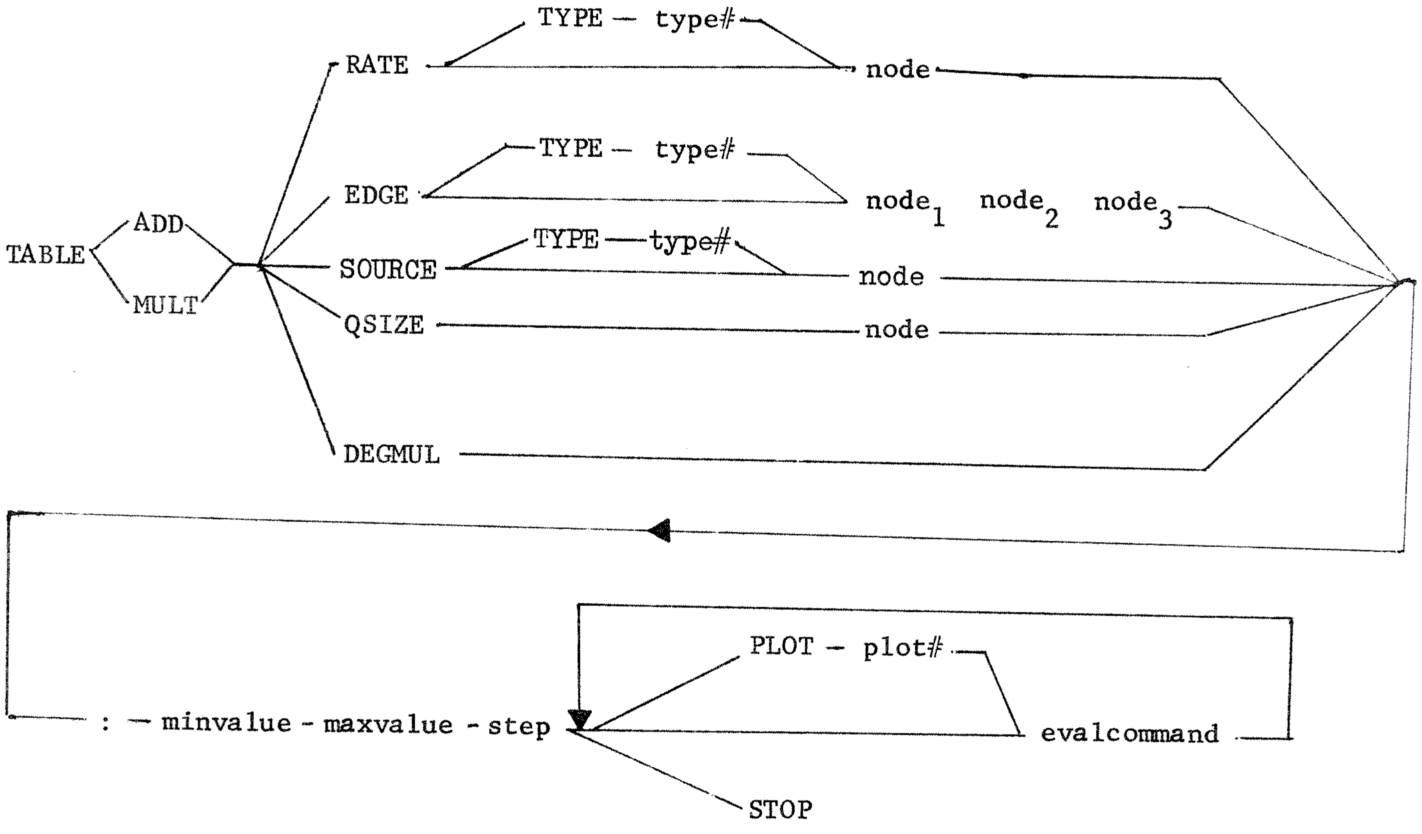
There are some properties of a network that cannot at present be changed once that network is originally specified to the program in the network description. Open networks may not be changed to closed, and vice-versa. The number of job types in a network may not be changed, although the number of types can be virtually decreased by setting all branching probabilities of one or more types to 0.0. A dependent typed network cannot be changed to independent, and vice-versa. The rate of some node cannot be set to 0.0, even if it is isolated by 0.0 branching probabilities. A "source" may be deleted from an open network by setting a node's source rate to 0.0. Nodes may neither be deleted from nor added to the network.

Care must be exercised by the user in setting or changing branching probabilities in an open network. If the sum of the branching probabilities from a point is less than 1.0, the program assumes the difference to be a branch to a sink. If the sum is greater than 1.0 the program issues an error message.

3.4 TABLE command

TABLE allows the user to obtain the results of executing one of the EVAL commands while iterating through various values of some network parameter. For example, the user might obtain the values of the utilization of some node while varying its average service rate. Four types of network parameters may be varied: the average service rate (s) of a node or the average emission rate of a Poisson source, the branching probability between two nodes, the queue capacity of a node, or the total number of jobs in a closed network (DEGMUL). In the text of a TABLE command a <u>minvalue</u>, <u>maxvalue</u>, and <u>step</u> are specified for one of the above network parameters. In the course of a TABLE execution the parameter is initialized to <u>minvalue</u>. An EVAL command is executed. The parameter is either incremented by <u>step</u> or multiplied by <u>step</u> (depending upon whether ADD or MULT is specified in the COMMAND text). The EVAL command is repeatedly executed for the increasing parameter until the parameter exceeds <u>maxvalue</u>. The results of executing the EVAL command versus the network parameter value are printed. The user may specify a sequence of EVAL commands for which the network parameter repeats its range. The STOP command terminates the TABLE execution. The network parameter is returned to the value it had before TABLE was executed.

Figure 3.2 defines the syntax of the TABLE text. Which network parameter to be varied is decided by a choice between RATE, EDGE, QSIZE, or DEGMUL. Since a node's service rate or the branching probability between nodes may also be given by job type in the original network specification, a provision is made to specify the job type identifying integer for RATE or EDGE. If RATE is specified for a source, the source's Poisson rate is the only parameter varied. If RATE is specified for a node with a single service rate, the rate is varied. However, if J different service rates were



type#	::=	integer	
plot#	::=	integer	
minvalue maxvalue step	::=	integer	(if QSIZE or DEGMUL specified)
minvalue maxvalue step	::=	real	(if RATE or EDGE specified)

evalcommand ::= SSPB vector | SGPB vector | SEPB vector
IDLE node | UTIL node | QLEN node
TPUT node | WAIT node

minvalue $\leq$ maxvalue plot# $<$1000

FIGURE 3.2 SYNTAX OF THE TABLE COMMAND

originally assigned to the node, say r_1, r_2, ... r_J, then r_1 is chosen as the network parameter and the r_2', r_3', ... r_J' are redefined by $r_i' = p_i \cdot r_1'$, where $p_i' = r_i / r_1$, during the course of a TABLE execution.

The fact that the sum of the branching probabilities of a node's outgoing edges must sum to 1.0 accounts for the $\underline{\text{node}}_1$ $\underline{\text{node}}_2$ $\underline{\text{node}}_3$ specification which must be made if an edge is varied. The branching probability of the $\underline{\text{node}}_1$ to $\underline{\text{node}}_2$ edge is the network parameter. As this probability is increased the branching probability of the ($\text{node}_1\text{node}_3$) edge is decreased accordingly. Consider the partial network

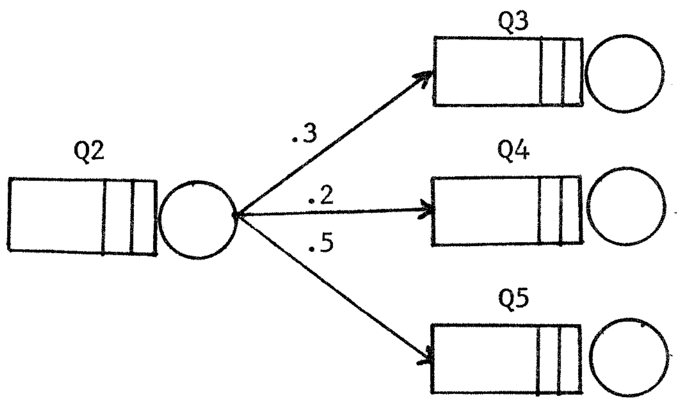


Figure 3.3 A Partial Network

```
with edges    Q2  .3  Q3
              Q2  .2  Q4
              Q2  .5  Q5
if            TABLE ADD EDGE Q2 Q4 Q5:  .1  .7  .1
              UTIL  Q2
              STOP
```

is given then the branching probability of the (Q2 Q4) edge is varied from .1 to .7 in increments of .1 while UTIL Q2 is repeatedly executed. The branching probability of edge (Q2 Q5) is adjusted from .6 to 0.0 in steps of 0.1 to keep the sum of both branching probabilities equal to 0.7. The edge (Q2 Q3) is unaffected. If <u>maxvalue</u> exceeds the sum of the two indicated branching probabilities then TABLE stops its iteration when the sum of the two exceeds the sum of the original two probabilities, as probability would no longer be "conserved". For example, if <u>maxvalue</u> for the above figure was 0.9 then at the last iteration the sum of branching probabilities from node Q2 would be 1.2.

In addition to outputting a table of parameter values versus EVAL command results, TABLE also gives the user the option of a printer plot. If the plot option is exercised no more than 100 values will be generated during the parameter iteration. A one page plot is produced on the plot file for every PLOT appearing in the TABLE text. The plot option is available to every EVAL command appearing in the TABLE text. No heading for a plot is presently available except a three digit integer. In interactive ASQ the user must record what "plot n" represents himself. In batch ASQ a copy of the input text exists on the file designated for plots for the user to reference.

The degree of multiprogramming (jobtotal) may be varied only for networks with no job typing.

```
  PLOT NO.    3
  HORIZONTAL UNIT = 2.778E-02     VERTICAL UNIT = 3.879E-02
= INDICATES MORE THAN ONE POINT IS PLOTTED IN THAT SPACE.
            I----I----I----I----I----I----I----I----I----I----I----I
 1.025E+00 H                                                  *   *H
 9.864E-01 H                                        *  *   *       H
 9.476E-01 H                                *   *                  H
 9.088E-01 H                             *                         H
 8.700E-01 H                         *                             H
 8.312E-01 H                      *                                H
 7.924E-01 H                  *                                    H
 7.536E-01 H              *                                        H
 7.148E-01 H           *                                           H
 6.760E-01 H                                                       H
 6.372E-01 H       *                                               H
 5.984E-01 H                                                       H
 5.596E-01 H    *                                                  H
 5.208E-01 H                                                       H
 4.820E-01 H*                                                      H
            I----I----I----I----I----I----I----I----I----I----I----I
         5.000E-01    I      1.056E+00    I      1.611E+00    I
            I      7.778E-01    I      1.333E+00    I
```

```
  PLOT NO.    4
  HORIZONTAL UNIT = 2.778E-02     VERTICAL UNIT = 3.225E-02
= INDICATES MORE THAN ONE POINT IS PLOTTED IN THAT SPACE.
            I----I----I----I----I----I----I----I----I----I----I----I
 9.641E-01 H*                                                      H
 9.318E-01 H    *                                                  H
 8.996E-01 H       *                                               H
 8.673E-01 H           *                                           H
 8.351E-01 H              *                                        H
 8.028E-01 H                  *                                    H
 7.706E-01 H                      *                                H
 7.383E-01 H                         *                             H
 7.061E-01 H                             *                         H
 6.738E-01 H                                *                      H
 6.416E-01 H                                    *                  H
 6.093E-01 H                                        *              H
 5.771E-01 H                                           *           H
 5.448E-01 H                                               *  *    H
 5.126E-01 H                                                      *H
            I----I----I----I----I----I----I----I----I----I----I----I
         5.000E-01    I      1.056E+00    I      1.611E+00    I
            I      7.778E-01    I      1.333E+00    I
```

Figure 3.4: Two Examples of "PLOT" Output From the TABLE Command

3.5 OTHER COMMANDS

3.5.1 END

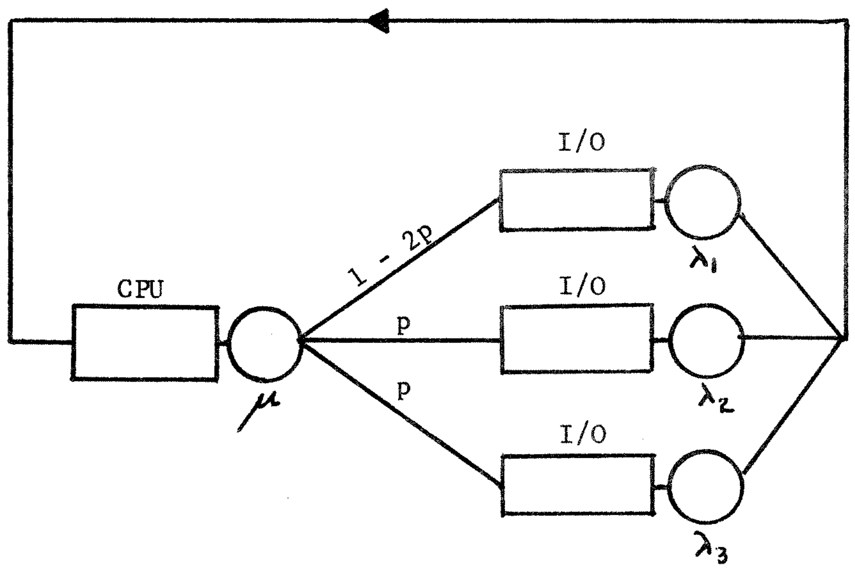
END terminates the ASQ session. It must appear as the last command in a batch deck.

3.5.2 NEW

NEW destroys the current network representation. The user must input a new description of a network immediatly following the command "NEW". In an interactive session NEW may be used at any point in the description of the edges and rates (say after input errors) to allow the user to describe the network again from scratch.

IV. EXAMPLES

Example 1.



Determine the throughput and utilizations for all devices in the above central server model of a multiprogrammed computer system with 3 I/O devices. Assume level of multiprogramming = 4.

CPU discipline = PS   I/O discipline = FCFS

$\lambda_1 = 1.0$ $\lambda_2 = \lambda_3 = 0.25$ are the rates of the I/O devices.

$\mu = 3.0$ is the rate of the CPU .

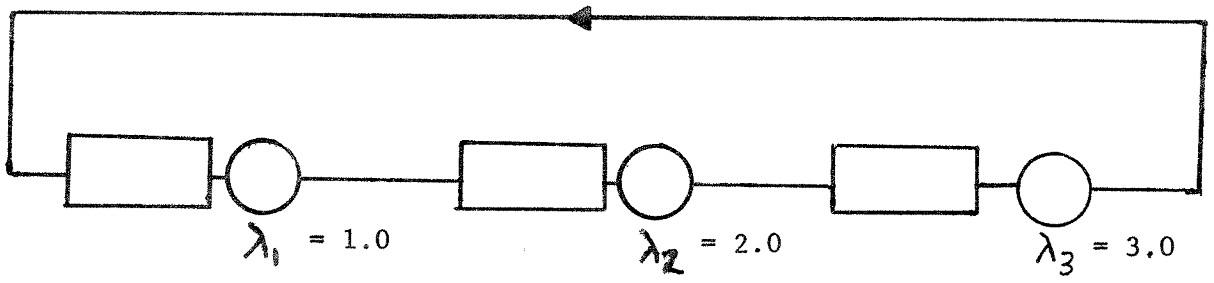
Determine solutions for p = 0.1, 0.3, and 0.167 .

Output file after batch run (equivalent to interactive session):

```
NETWK DSCPTN RESIDES ≡SAME≡/≡OTHER≡ FILE AS COMMANDS
          SAME
INPUT NUMBER OF JOB TYPES   (1 OR MORE)
          1
INPUT NO. OF NODES + BRANCHING POINTS
          4
INPUT NO. OF NODES
          4
WILL YOU HAVE EXTERNAL POISSON SOURCES≥
          NO
INPUT EDGES
          CPU .8 IO1 ;
          CPU .1 IO2 ;
          CPU .1 IO3 ;
          IO1 1. CPU ;
          IO2 1. CPU ;
          IO3 1. CPU ;
          ↱
INPUT RATES FOR NODES WITH FCFS       DISCIPLINE
          IO1 1. ;
          IO2 .25 ;
          IO3 .25 ;
          ↱
INPUT RATES FOR NODES WITH PS OR LCFS DISCIPLINE
          CPU 3.0 ;
          ↱
INPUT DEGREE(S) OF MULTIPROGRAMMING
          4



THE TEMPLATE IS:
IO1         IO2         IO3         CPU
          TPUT CPU
 1.081E+00

          TPUT IO1
 8.647E-01

          TPUT IO2
 1.081E-01

          UTIL CPU
 3.603E-01

          UTIL IO1
 8.647E-01

          UTIL IO2
 4.323E-01
```

```
            CHANGE EDGES:  CPU 0.4 IO1
                           CPU 0.3 IO2
                           CPU 0.3 IO3  ↱
            TPUT CPU
6.337E-01

            TPUT IO1
2.535E-01

            TPUT IO2
1.901E-01

            UTIL CPU
2.112E-01

            UTIL IO1
2.535E-01

            UTIL IO2
7.604E-01

            CHANGE EDGES:  CPU (2/3) IO1
                           CPU (1/6) IO2
                           CPU (1/6) IO3  ↱
            TPUT CPU
9.487E-01

            TPUT IO1
6.325E-01

            TPUT IO2
1.581E-01

            UTIL CPU
3.162E-01

            UTIL IO1
6.325E-01

            UTIL IO2
6.325E-01



            END
```

Example 2. Determine throughput, utilizations, and expected queue lengths for



Determine solutions for N = 1,2,3,4 where N is the degree of multiprogramming.

Assume the queueing discipline for the first device is PS, for others FCFS.

Output file after batch run (equivalent to interactive session):

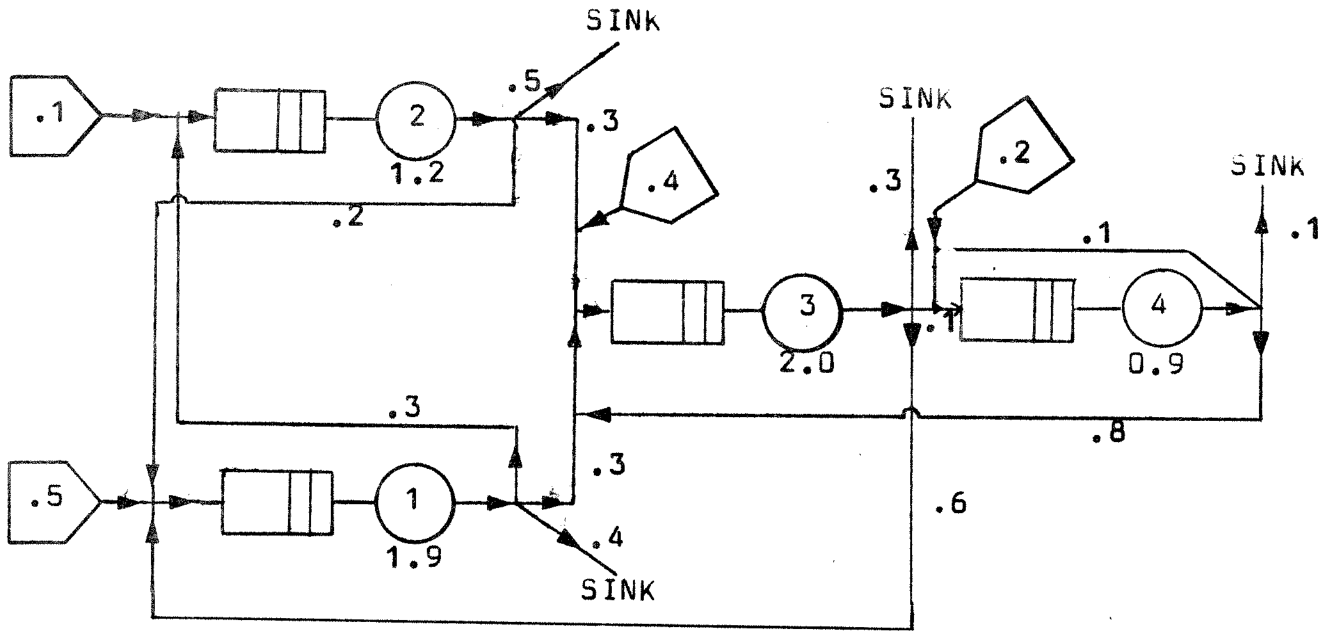
```
NETWK DESCPTN RESIDES ≡SAME≡/≡OTHER≡ FILE AS COMMANDS
         SAME
INPUT NUMBER OF JOB TYPES  (1 OR MORE)
         1
INPUT NO. OF NODES + BRANCHING POINTS
         3
INPUT NO. OF NODES
         3
WILL YOU HAVE EXTERNAL POISSON SOURCES≥
         NO
INPUT EDGES
         Q1 1. Q2 ;
         Q2 1. Q3 ;
         Q3 1. Q1 ;
         ↱
INPUT RATES FOR NODES WITH FCFS       DISCIPLINE
         Q2 2.  ;
         Q3 3.  ;
         ↱
INPUT RATES FOR NODES WITH PS OR LCFS DISCIPLINE
         Q1 1.  ;
         ↱
INPUT DEGREE(S) OF MULTIPROGRAMMING
         1
THE TEMPLATE IS:
Q2          Q3          Q1
```

```
          TABLE ADD DEGMUL: 1 4 1 TPUT Q1
     1         5.4545E-01
     2         7.7647E-01
     3         8.8696E-01
     4         9.4237E-01
                                   TPUT Q2
     1         5.4545E-01
     2         7.7647E-01
     3         8.8696E-01
     4         9.4237E-01
                                   TPUT Q3
     1         5.4545E-01
     2         7.7647E-01
     3         8.8696E-01
     4         9.4237E-01
                                   UTIL Q1
     1         5.4545E-01
     2         7.7647E-01
     3         8.8696E-01
     4         9.4237E-01
                                   UTIL Q2
     1         2.7273E-01
     2         3.8824E-01
     3         4.4348E-01
     4         4.7118E-01
                                   UTIL Q3
     1         1.8182E-01
     2         2.5882E-01
     3         2.9565E-01
     4         3.1412E-01
                                   QLEN Q1
     1         5.4545E-01
     2         1.2000E+00
     3         1.9513E+00
     4         2.7812E+00
                                   QLEN Q2
     1         2.7273E-01
     2         4.9412E-01


     3         6.6261E-01
     4         7.8339E-01
                                   QLEN Q3
     1         1.8182E-01
     2         3.0588E-01
     3         3.8609E-01
     4         4.3540E-01
                                   STOP
END ITERATION
      END
```

Input file for example 2 (in an interactive session the user's input):

```
SAME
1
3
3
NO
Q1 1. Q2 ;
Q2 1. Q3 ;
Q3 1. Q1 ;
↱
Q2 2.  ;
Q3 3.  ;
↱
Q1 1.  ;
↱
1
TABLE ADD DEGMUL : 1 4 1 TPUT Q1
                         TPUT Q2
                         TPUT Q3
                         UTIL Q1
                         UTIL Q2
                         UTIL Q3
                         QLEN Q1
                         QLEN Q2
                         QLEN Q3
                         STOP
END
```

Example 3. An Open Network



```
NETWK DSCPTN RESIDES =SAME=/=OTHER= FILE AS COMMANDS
          SAME
INPUT NUMBER OF JOB TYPES  (1 OR MORE)
          1
INPUT NO. OF NODES + BRANCHING POINTS
          4
INPUT NO. OF NODES
          4
WILL YOU HAVE EXTERNAL POISSON SOURCES≥
          YES
INPUT EDGES
          Q1 .3 Q2 ;
          Q1 .3 Q3 ;
          Q2 .2 Q1 ;
          Q2 .3 Q3 ;
          Q3 .6 Q1 ;
          Q3 .1 Q4 ;
          Q4 .1 Q4 ;
          Q4 .8 Q3 ;
          ↱
INPUT NODES WITH SOURCES AND THEIR RATES
          Q1 .5
          Q2 .1
          Q3 .4
          Q4 .2
          ↱
```

```
INPUT RATES FOR NODES WITH FCFS        DISCIPLINE
          Q1 1.9  ;
          Q2 1.2  ;
          Q3 2.0  ;
          Q4 0.9  ;
          ↱
INPUT RATES FOR NODES WITH PS OR LCFS DISCIPLINE
          ↱
THE TEMPLATE IS:
Q1          Q2          Q3          Q4
          TPUT Q1
 1.347E+00

          TPUT Q2
 5.041E-01

          TPUT Q3
 1.244E+00

          TPUT Q4
 3.604E-01

          UTIL Q1
 7.090E-01

          UTIL Q2
 4.201E-01

          UTIL Q3
 6.218E-01

          UTIL Q4
 4.005E-01

          QLEN Q1
 2.436E+00

          QLEN Q2
 7.244E-01

          QLEN Q3
 1.644E+00

          QLEN Q4
 6.679E-01
```

```
          TABLE ADD RATE Q1 : 1.75 5.0 0.25
                 UTIL Q1
1.750E+00   7.697E-01
2.000E+00   6.735E-01
2.250E+00   5.987E-01
2.500E+00   5.388E-01
2.750E+00   4.898E-01
3.000E+00   4.490E-01
3.250E+00   4.145E-01
3.500E+00   3.849E-01
3.750E+00   3.592E-01
4.000E+00   3.368E-01
4.250E+00   3.169E-01
4.500E+00   2.993E-01
4.750E+00   2.836E-01
5.000E+00   2.694E-01
                 QLEN Q1
1.750E+00   3.343E+00




2.000E+00   2.063E+00
2.250E+00   1.492E+00
2.500E+00   1.168E+00
2.750E+00   9.601E-01
3.000E+00   8.149E-01
3.250E+00   7.078E-01
3.500E+00   6.257E-01
3.750E+00   5.606E-01
4.000E+00   5.077E-01
4.250E+00   4.640E-01
4.500E+00   4.272E-01
4.750E+00   3.958E-01
5.000E+00   3.687E-01
          STOP
          END
```

Input file for example 3 (user's input):

```
SAME
1
4
4
YES
Q1 .3 Q2 ;
Q1 .3 Q3 ;
Q2 .2 Q1 ;
Q2 .3 Q3 ;
Q3 .6 Q1 ;
Q3 .1 Q4 ;
Q4 .1 Q4 ;
Q4 .8 Q3 ;
↱
Q1 .5
Q2 .1
Q3 .4
Q4 .2
↱
Q1 1.9 ;
Q2 1.2 ;
Q3 2.0 ;
Q4 0.9 ;
↱
↱
TPUT Q1
TPUT Q2
TPUT Q3
TPUT Q4
UTIL Q1
UTIL Q2
UTIL Q3
UTIL Q4
QLEN Q1
QLEN Q2
QLEN Q3
QLEN Q4
TABLE ADD RATE Q1 : 1.75 5.0 0.25
      UTIL Q1
      QLEN Q1
STOP
END
```

V. ASQ CONTROL CARDS FOR THE UT-2D SYSTEM

ASQ may be run in interactive or batch mode. In either case, four files are always required: ASQABS, which is an absolute binary file; ASQRUNA, a control card macro; an input file ; an output file. Batch runs assume INPUT and OUTPUT for the latter two files, unless otherwise specified. Interactive runs assume TTY and again TTY for the latter two files, unless otherwise specified. For both batch and interactive runs ASQ is invoked by the two control cards:

```
READPF 4375 ASQRUNA ASQABS.
ASQRUNA <options> .
```

Three options are available to the user. The first specifies whether ASQ is to be invoked in batch or interactive mode. The second specifies upon which file the original network description will reside. The third specifies upon which file the plots will be output. The options are:

option	parameter value	result
MODE	BATCH	input on file INPUT, output on file OUTPUT
	absent	input,output on file TTY
NETWORK DESCRIPTION	NET = <filename>	original network description resides on file <filename>
	absent	original network description resides on TAPE3 if "OTHER" specified as per section 2.1
PLOT OUTPUT	PLOT = <filename>	plots outputted onto file <filename>
	absent	plots outputted to OUTPUT if batch run, TTY if interactive

The options are provided for user convenience. The NET option allows the interactive user to keep his network description on some file (default valueTAPE3), rather than requiring him to manually input the description at each interactive session. It may be more convenient for the user to change the network description via EDITOR or BRUTE before invoking ASQ rather than to use the CHANGE command in an ASQ session. Likewise, it may be desirable to the user to keep plot output on a seperate file during an ASQ session.

5.1 Examples of Batch Decks

```
CSIN123, JOHN DOE.
ABC=PASSWORD.
READPF 4375 ASQRUNA ASQABS.          control cards
ASQRUNA BATCH.
7/8/9
TTY
1
4
4
NO                              original
CPU .8 I01 ;                    network description
CPU .1 I02 ;
:
4
TPUT CPU
TPUT I01
:                               commands
UTIL I01
UTIL I02
END
EØF
```

```
CSIN123, JOHN DOE.
ABC=PASSWORD.
READPF 4375 ASQRUNA ASQABS.          control cards
ASQRUNA BATCH.
7/8/9
TAPE3
TPUT CPU
TPUT I01
:                               commands
UTIL I02
END
EØF
```

REFERENCES

1. K.M. Chandy, T.W. Keller and J.C. Browne, "Design Automation and Queueing Networks," Proc. Ninth Annual Design Automation Conference, 9, pp. 357 - 367 (June 1972)

2. K.M. Chandy, "The Analysis and Solutions for General Queueing Networks," Proc. Sixth Annual Princeton Conference on Information Sciences, Princeton Univ., Princeton, N.J. (March 1972).

3. F. Baskett and R.R. Muntz, "Queueing Network Models with Different Classes of Customers," Proc. COMPCON 72, San Francisco (Sept. 1972).

4. T.W. Keller, D.F. Towsley, K.M. Chandy, J.C. Browne, "A Tool for Network Design, The Automatic Analysis of Stochastic Models of Computer Networks," Proc. COMPCON 73, San Francisco (Feb. 1973).

5. T. W. Keller, "A Program for the Automatic Analysis of Queueing Network Models", TR-6, Dept. of Computer Sciences, University of Texas, Austin, Tx. (December 1972).