STATE UNIVERSITY OF NEW YORK AT

STONY BROOK

CEAS TECHNICAL REPORT 630

Recursive Solutions for Discrete Time Queues with Applications to High Speed Switching

H.-Y. Huang and T.G. Robertazzi

June 19, 1992

Recursive Solutions for Discrete Time Queues with Applications to High Speed Switching Fabrics

H.-Y. Huang
T.G. Robertazzi, Senior Member IEEE

Dept. of Electrical Engineering,
SUNY at Stony Brook,
Stony Brook N.Y. 11794

Key Words: Queues, Discrete Time, Recursive Solution, Manhattan Street Network, Switching Fabric

Abstract

Fast recursive solutions for the equilibrium state probabilities of discrete time queuing systems are presented. A form of boundary balancing is used to arrive at their solutions. Cases examined include the Geom/Geom/K/N queue, a queue with a varying number of servers and the Manhattan Street metro- politan area network.

1 Introduction

The trend of modern communication is to integrate a variety of traffic services into a single transmission facility. In such an environment, all transmissions are in digital form regardless of its original nature. This is particularly true of proposed ATM architectures.

In a digital network, usually a group of data bits are treated as a single entity and given a name such as "packet", "cell" etc.. This group of data is either a part of, or the entire message that came from some user and is destined for some other user of the network. Grouping puts the routing information common to all the data contained in the group together in the header of the group. By manipulating the header of the group, all nodes of the network thru which the group of data passes can then successfully route the data to its destination. Such systems are often operated on a slotted time basis. The slots in these systems corresponds to a fixed duration of time during which this group of data can be transferred. In this paper "cell" will be used to denote this group of data, in accordance with terminology used in ATM technology. As a result of this slotted nature, this kind of system is best described in discrete time.

Consider a slotted network. Unless the load of a network is extremely light, there will be times when newly arrived messages are blocked because all of the network resources available to the node, where the message is originated, are being occupied at that moment. To remedy this situation, input buffers are needed to store arriving cells before they are passed to the switching mechanism of the node. Input buffers temporarily smooth out load fluctuations and thus keep the blocking probability down. Because of this fluctuation in network load there will also be times when more than one network resource is free. If during that slot, there is more than one message waiting in the input buffer, then instead of just sending one message on to the network, one can send more. This idea is known as bulk service.

Two different queuing models are discussed in this paper. One is a discrete time queue with multiple servers. If the arrival process is Bernoulli and the service discipline is geometric, then the queue posseses the memory less property and is the discrete time analog of the continuous time M/M/K/N

queue. The other queue model is a discrete time queue with varying number of servers. Again the arrival process is assumed to be Bernoulli but the service discipline is now deterministic with a service time of exactly one slot. An example of this kind of queue is the Manhattan Street Network which will also be discussed. Using the Bernoulli process enables us to describe the state of the input buffer by a discrete time Markov chain and use the Markovian property to develop the state equations.

Because of the multiple service property, it is impossible to find a product form solution for the state probabilities. But as will be seen shortly, these two queuing models have exactly the same structure as the type A structure mentioned in [9], so fast recursive solution can be developed for the calculation of the equilibrium state probabilities. The direct procedure for calculating the non-product form equilibrium state probabilities, solving N simultaneous linear (global balance) equations, requires time on the order of $O(N^3)$. The recursive procedure presented here requires time $O(N)$ and is extremely simple to compute.

In section 2 recursive equations for the determination of equilibrium state probabilities for a Geom/Geom/K/N queuing system will be presented. Section 3 contains recursive equations for the equilibrium state probabilities of a discrete time queuing system with a varying number of available servers. A special case of this, input buffer in a deflection routing Manhattan Street Network (MSN), is discussed in section 4.

2 Discrete Time Queuing System with Multiple Servers

Consider a general discrete time queue shown in Fig. 1, where p is the arrival probability of a customer at a server during a time slot and s is the service completion probability of a customer during a slot. Consequently this corresponds to Bernoulli arrival process and geometric service time. Also define N as the capacity of the queuing system and K as the number of servers. This is a Geom/Geom/K/N queue, which is the analog of M/M/K/N queue in discrete time. The state of the queue is defined to be the number of cus-

tomers in the queue (including the ones in the servers), denoted by random variable X_n (where n denotes nth slot). Shown in Fig. 2 is the state transition diagram of this queue. Note this is exactly the type A structure mentioned in [9].

The goal of the presented analysis is to find the equilibrium probabilities of queue states given p and s, and from these other important statistics of the queue, such as mean delay, blocking probability etc..

Here equilibrium probability of a queue with k customers is denoted as: $P_k = \lim_{n\to\infty} Pr\{X_n = k\}$. When K is larger than one, finding the algebraic form of P_k is a tedious job. Thus when the numerical result of the queue performance in terms of different parameters of the queue is needed, a numerical solution is more attractive. As proposed in [9], a recursive solution is proposed here. This is arrived at by checking boundaries between adjacent states and equating the flow of probability flux from left to right across the boundary to the flow from right to left [2], [4], [1], [8]. This balancing can be referred to as "boundary balance". Define,

$$\begin{aligned} a_{i,j} &= \lim_{n\to\infty} Pr\{X_{n+1} = j \mid X_n = i\}, \quad \forall\, i,j \in [0,N] \\ \binom{n}{k} &= \begin{cases} 0 & n < k \text{ or } k < 0 \\ 1 & k = 0 \\ \frac{n!}{k!(n-k)!} & otherwise \end{cases} \\ [n,K]^* &= min(n,K) \end{aligned}$$

In the usual case, a customer can arrive at the queue with probability p, and after entering a server, it can finish the service in i time slots with probability $s(1-s)^{i-1}$. This means that it is possible for a customer to enter and leave the queue in the same slot. To find the transition probabilities, all possible combinations of customers arrivals and departures have to be considered. Two sets of equations is obtained. The one step transition probabilities from state n to $n-l$ (except from N to N) are,

$$\begin{aligned} a_{n,n-l} = {} & p \binom{[n+1,K]^*}{l+1} s^{l+1}(1-s)^{[n+1,K]^*-l-1} \\ & + (1-p)\binom{[n,K]^*}{l} s^l (1-s)^{[n,K]^*-l} \\ & -1 \le l \le K, \quad \forall n \ne N \end{aligned}$$

and the transition probability from state N to N is,

$$a_{N,N} = p[\binom{K}{1} s(1-s)^{K-1} + (1-s)^K] + (1-p)(1-s)^K$$

Now if a customer can not enter and leave the queue in the same slot there will be a different set of state transition probabilities. This delay can be called the synchronization time. One example where this case might arise is when time is required to process a header in a network node. The one step transition probabilities from state n to $n-l$ (except from N to N) in this case are,

$$\begin{aligned} a_{n,n-l} =& p\binom{[n,K]^*}{l+1} s^{l+1}(1-s)^{[n,K]^*-l-1} \\ &+ (1-p)\binom{[n,K]^*}{l} s^l(1-s)^{[n,K]^*-l} \\ & -1 \le l \le K, \quad \forall n \neq N \end{aligned}$$

and the transition probability from state N to N is,

$$a_{N,N} = p[\binom{K}{1} s(1-s)^{K-1} + (1-s)^K] + (1-p)(1-s)^K$$

According to the probability flow balance relation [4], when the queue is in equilibrium , the probability flows crossing the boundary from state i to $i-1$ will equal the flow from state $i-1$ to state i (Fig. 3). Expressed mathematically,

$$\boxed{a_{i-1,i}P_{i-1} = \sum_{k=i}^{K+i-1} \sum_{j=k-K}^{i-1} a_{k,j}P_k \ , \quad i = N, \ldots, 1}$$

From the balance equation, the probability of the lower states can be determined from that of the higher states. Thus once the probability of highest state (queue full) is known, all other states can be found recursively. Using the property of probability conservation,

$$\sum_{k=0}^{N} P_k = 1$$

we can arbitrarily assign P_N and find all other P_n's subject to this P_N and then normalize their sum. The equilibrium probability is then obtained. Written out explicitly,

1. Let $P_N = 1.0$
2. Initialize $a_{i,j}$'s.
3. $i = N - 1$
4. $P_i = \frac{1}{a_{i,i+1}} \sum_{k=i+1}^{i+K} \sum_{j=k-K}^{i} a_{k,j} P_k$
5. $i = i - 1$
6. repeat step 4 and 5 until $i < 0$
7. Find $\sum P_i$.
8. Divide all P_i's acquired in step 1 and 4 by sum of step 7, this is the desired result.

One thing one has to be very careful about during numerical evaluation is the problem of overflow or underflow. This might occur if the queue size is large, and because of the repeated multiplications due to recursive calculations, the computer arithmetic operation might overflow or underflow at times. To get over this potential problem, one can use scaling techniques. That is during the recursion, software can automatically scale up or scale down the intermediate result.

With P_k's known, we can proceed to find other statistics that are of greater interest.

$$
\begin{aligned}
\gamma &= \textit{Throughput} \\
&= Pr\{\textit{a new customer enters the queue (per slot)}\} \\
&= Pr\{\textit{new customer arrival and no blocking}\} \\
&= Pr\{\textit{new customer arrival}\} \cdot Pr\{\textit{no blocking}\} \\
&= p(1 - P_b) \\
P_b &= Pr\{\textit{arriving customer being blocked}\} \\
&= Pr\{\textit{queue full and no service completion} \mid \\
&\qquad \textit{customer arrival}\} \\
&= Pr\{\textit{queue full and no service completion}\}
\end{aligned}
$$

$$
\begin{aligned}
&= Pr\{queue\ full\} \cdot Pr\{no\ service\ completion\} \\
&= P_N s_0 \quad (for\ this\ type\ of\ queue\ s_0 = (1-s)^K) \\
\overline{L} &= Average\ queue\ length = \lim_{n\to\infty} E\{X_n\} \\
&= \sum_{k=1}^{N} k P_k \\
\overline{W} &= Average\ waiting\ time\ (slots) \\
&= \frac{\overline{L}}{\gamma}\ (Little's\ formula)
\end{aligned}
$$

3 Discrete Time Queuing System with Varying Number of Servers

Assume a buffer is placed before some servers (e.g. communication link) which are also fed by other buffers (Fig. 4), and the sharing rule is determined by some control mechanism so that no "collision" will occur. Then because of the sharing of servers the number of servers available to this buffer varies with time. A network node with a buffer on each incoming trunk, and where the queued packets compete for the outgoing trunks according to some priority rule is an example. Now consider certain "slotted" communication systems. Here transiting cells (cells passing thru the node on the way to some remote destination) are given priority in access to servers over locally originating traffic (traffic whose source is at the node). Thus in such a discrete time queue, at each time epoch (defined to be the start of a time slot), the cell in the local input buffer sees a varying number of empty servers.

In analyzing such a queue, the state (X_n) can be defined to be the number of cells in the buffer at time slot n. These cells are either being serviced by the servers or waiting to be serviced. It is assumed in here that the cell arrival process is Benoulli, and the cell service time is one time slot. It is also assumed that the server availability probability is independent of the state of the queue. With these assumptions, it is clear that the state of the queue is Markovian.

Consider a queue of the above type. Assume the capacity of the queue is N, and K servers are in this queuing system. Also assume L is the maximum

number of cells from this queue allowed to be served at a single time slot. Usually $L = K$, but there might be cases where for some reason (e.g. cost) the number of distributed queues are limited to L ($L < K$). In this case at most L cells are allowed to enter the server simultaneously.

Define $s_k = Pr\{k \text{ servers available to this queue}\}$ (on per slot basis) and which can be of any kind of distribution, providing $\sum_{k=0}^{K} s_k = 1$. In the following it is assumed that s_k is determined either analytically, numerically or thru experimental data. For $N \geq L$ ($N \geq 1$, $L \leq K$), the one step state transition probability ($a_{i,j}$) from state i to j are discussed below. Note that, since only a single arrival and multiple departures of up to L can occur during a time slot, any state transitions violating these conditions corresponds to a null event. As in previous section, again two cases is examined. First, assume synchronization time (defined previously) is required. [11]

1. If $i < j$, due to a single arrival, only $j = i + 1$ is possible. Depending on i, two different cases exits,

   (a) if $i = 0$ then only one event can happen,

   - one arrival, regardless of server availability.

   $$a_{0,1} = p$$

   (b) if $i \neq 0, i < N$ then only one event can happen,

   - one arrival, no departure.

   $$a_{i,i+1} = ps_0$$

2. If $i = j$, depending on i, four different cases exits,

   (a) if $i = 0$ then only one event can happen,

   - no arrival.

   $$a_{0,0} = 1 - p$$

   (b) if $i = 1$ then two events can happen,

- no arrival, no departure.
- one arrival, one departure.

$$a_{1,1} = (1-p)s_0 + p\sum_{k=1}^{K} s_k$$

The summation finds the probability of at least 1 server being available.

(c) if $1 < i < N$ then two events can happen,

- no arrival, no departure.
- one arrival, one departure.

$$a_{i,i} = (1-p)s_0 + ps_1$$

(d) if $i = N$ then three events can happen,

- no arrival, no departure.
- one arrival, one departure.
- one arrival, no departure (arrival cleared when buffer is full).

$$a_{N,N} = (1-p)s_0 + p[s_0 + s_1] = s_0 + ps_1$$

3. If $i > j$, depending on j, three different cases exits,

(a) if $j = 0$ then only one event can happen,

- no arrival, i departure.

$$a_{i,0} = (1-p)\sum_{k=i}^{K} s_k, \quad 1 \leq i \leq L$$

(b) if $j = 1$ then two events can happen,

- no arrival, $i-1$ departures.
- one arrival, i departures.

$$a_{i,1} = (1-p)s_{i-1} + p\sum_{k=i}^{K} s_k, \quad i \leq L$$

$$a_{L+1,1} = (1-p)\sum_{k=L}^{K} s_k, \quad i = L+1$$

(c) if $j > 1$ then two events can happen,

- no arrival, $i-j$ departures.
- one arrival, $i-j+1$ departures.

$$a_{i,j} = \begin{cases} (1-p)s_{i-j} + ps_{i-j+1}, & i-j < L-1 \\ (1-p)s_{i-j} + p\sum_{k=L}^{K} s_k, & i-j = L-1 \\ (1-p)\sum_{k=L}^{K} s_k, & i-j = L \end{cases}$$

All other transition probabilities not mentioned here correspond to null events and the corresponding transition probabilities are simply zero. There are in total $(L+1) + N(L+2) - \frac{L(L+1)}{2}$ non-zero transition probabilities with only $3L+3$ distinct values, so the computational cost for finding these transition probabilities is small.

Now assume there is no synchronization time.

1. If $i < j$, due to a single arrival, only $j = i+1$ is possible. For $i < N$ only one event can happen,

   - one arrival, no departure.

   $$a_{i,i+1} = ps_0$$

2. If $i = j$, depending on i, three different cases exits,

   (a) if $i = 0$ then two events can happen,

   - no arrival.
   - one arrival, one departure.

$$a_{0,0} = (1-p) + p(1-s_0) = 1 - ps_0$$

(b) if $1 \leq i < N$ then two events can happen,

- no arrival, no departure.
- one arrival, one departure.

$$a_{i,i} = (1-p)s_0 + ps_1$$

(c) if $i = N$ then three events can happen,

- no arrival, no departure.
- one arrival, one departure.
- one arrival, no departure (arrival cleared when buffer is full).

$$a_{N,N} = (1-p)s_0 + p[s_0 + s_1] = s_0 + ps_1$$

3. If $i > j$, depending on j, two different cases exits,

(a) if $j = 0$ then two events can happen,

- no arrival, i departures.
- one arrival, $i+1$ departures.

$$\begin{aligned} a_{i,0} &= (1-p)\sum_{k=i}^{K} s_k + p\sum_{k=i+1}^{K} s_k \\ &= (\sum_{k=i}^{K} s_k) - ps_i, \quad 1 \leq i \leq L \end{aligned}$$

(b) if $j \geq 1$ then two events can happen,

- no arrival, $i-j$ departures.
- one arrival, $i-j+1$ departures.

$$a_{i,j} = \begin{cases} (1-p)s_{i-j} + ps_{i-j+1}, & i-j < L-1 \\ (1-p)s_{i-j} + p\sum_{k=L}^{K} s_k, & i-j = L-1 \\ (1-p)\sum_{k=L}^{K} s_k, & i-j = L \end{cases}$$

Again all other transition probabilities not mentioned here correspond to null events and the corresponding transition probabilities are simply zero. In this case $\frac{(2N+1)(2+L)-L^2}{2}$ transition probabilities are non-zero with only $2L+4$ distinct values.

Since the state transition diagram for this queue is similar to the Geom/Geom/K/N queue, we can expect that the probability flow balance equations are also similar (Actually they are the same if $L = K$):

$$\boxed{a_{i-1,i}P_{i-1} = \sum_{k=i}^{i+L-1} \sum_{j=k-L}^{i-1} a_{k,j}P_k \ , \quad i = N, N-1, \ldots, 1}$$

To find P_n for all n, one can use the same recursive algorithm mentioned in the last section with a change in step 4 to:

$$P_i = \frac{1}{a_{i,i+1}} \sum_{k=i+1}^{i+L} \sum_{j=k-L}^{i} a_{k,j}P_k$$

4 Input Buffer for the Manhattan Street Network

The Manhattan Street Network (MSN) is a type of metropolitan area network proposed by N.F. Maxemchuk [5, 6] of AT&T Bell Laboratories in 1985. It is a two connected regular mesh network (Fig. 5). For a NxN MSN, the N nodes in the same row or the same column are connected in a unidirectional loop. Adjacent rows and adjacent columns alternate direction like the streets of Manhattan.

Each node in a MSN has two input links and two output links connecting it to other nodes of the network. Locally generated traffic enters the node

thru a local link. Assume that there are two packet buffers in the node to temporarily store the incoming packets for processing. Thus one filled packet buffer corresponds to one occupied output link. At any given time slot at most two packets can arrive at the same node on the input links and leave for other nodes through the two output links. There is thus no internal blocking in the nodes. Deflection routing is used in the MSN. That is if two packets in a node's packet buffers prefer the same output link, one packet is sent to the preferred link and the other is "deflected" to the non-preferred link.

New traffic can only enter a node if at least one of the node's two packet buffers is not occupied by a transiting packet. So if both packet buffers are occupied, local traffic is blocked. Thus input buffers are necessary to reduce this blocking probability [7], [3]. A properly sized input buffer will reduce the blocking probability and maximizing the throughput with respect to a specific routing strategy.

From the structure and operation of the MSN described above, the input buffer can be viewed as a discrete time queue with varying number of servers. Here the servers are the packet buffers in the node. Thus the method described in the last section can be applied here.

In applying the results of previous section to the MSN, two cases could arise (in MSN, $K = 2$). The probabilistic equations governing the input buffer state probabilities for these two cases are listed below. It is assumed that s_k is binomially distributed, i.e.

$$\begin{aligned} s_k &= Pr\{k \text{ packet buffers empty (per slot)}\} \\ &= \binom{K}{k} \mathrm{s}^{\mathrm{k}}(1-\mathrm{s})^{\mathrm{K-k}} \end{aligned}$$

If at most one node packet buffer is allowed to be accessed by the local traffic during a slot, then the state transition diagram appears in Fig 6 and the boundary balance equations are,

$$\begin{aligned} P_1 &= \frac{p}{(1-s_0)(1-p)} P_0 \\ P_2 &= \frac{ps_0}{(1-s_0)(1-p)} P_1 \\ &\vdots \quad \vdots \end{aligned}$$

$$P_N = \frac{ps_0}{(1-s_0)(1-p)} P_{N-1}$$

$$P_k = \frac{p^k s_0^{k-1}}{(1-s_0)^k (1-p)^k} P_0, \quad k = 1, \ldots, N$$

where,

$$P_0 = \frac{(1-s_0-p)(1-p)^N (1-s_0)^N}{(1-s_0)^{N+1}(1-p)^N - p^{N+1} s_0^N}$$

This is in fact a one dimensional product form solution for the state probabilities.

For the case where both node packet buffers are allowed to be accessed by the local traffic during a slot, the state transition diagram is as Fig 7 and the boundary balance equations are,

$$\begin{aligned}
pP_0 &= (1-p)(1-s_0)P_1 + (1-p)s_2 P_2 \\
ps_0 P_1 &= [ps_2 + (1-p)s_1 + (1-p)s_2]P_2 + [(1-p)s_2]P_3 \\
&\vdots \quad \vdots \\
ps_0 P_i &= [ps_2 + (1-p)s_1 + (1-p)s_2]P_{i+1} + (1-p)s_2 P_{i+2} \\
&\vdots \quad \vdots \\
ps_0 P_{N-2} &= [ps_2 + (1-p)s_1 + (1-p)s_2]P_{N-1} + [(1-p)s_2]P_N \\
ps_0 P_{N-1} &= [ps_2 + (1-p)s_1 + (1-p)s_2]P_N
\end{aligned}$$

Solving explicitly one has:

$$\begin{aligned}
P_N &= 1.0 \\
P_{N-1} &= \frac{ps_2 + (1-p)s_1 + (1-p)s_2}{ps_0} P_N \\
P_i &= \frac{[ps_2 + (1-p)s_1 + (1-p)s_2]P_{i+1} + (1-p)s_2 P_{i+2}}{ps_0}, \\
&\qquad\qquad i = N-2, N-3, \ldots, 1 \\
P_0 &= \frac{(1-p)(1-s_0)P_1 + (1-p)s_2 P_2}{p}
\end{aligned}$$

this is followed by a normalization.

5 Conclusion

Some comparison of the effectiveness of recursive solution is listed in the following table. The algorithms were implemented using MATLAB. The performance is measured in terms of flops (floating point operations). In the table the "iterative method" corresponds to solving matrix equation $[\pi] = [\pi][P]$ iteratively, while the direct method corresponds to solving the matrix equation by the Gauss elimination method.

N	10	10	10	20
K	2	2	2	4
s	0.05	0.1	0.5	0.5
p	0.8	0.5	0.5	0.5
recursive	1396	1396	1396	5318
iterative	9503	30755	10262	17681
direct	4348	3576	3576	17263

The performance improvements shown here is conservative as $N = 10, 20$ is a relatively small buffer size. As N is increased the improvement can be expected to be even more dramatic. Note,

$$[P] = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N,0} & a_{N,1} & \cdots & a_{N,N} \end{pmatrix}$$

$$[\pi] = \begin{pmatrix} P_0 & P_1 & \cdots & P_N \end{pmatrix}$$

The ability to recursively solve for the equilibrium probabilities very efficiently is promising for such uses as numerical performance evaluation and real time network management. We note in closing that it would be straight forward to apply this technique to discrete time queues with a state dependent arrival probability and to a single server discrete time queue with batch arrivals. æ

Acknowledgement

This work was supported in part by the National Science Foundation under grant no. NCR-8703689 and in part by the SDIO/IST and administrated by the U.S. Office of Naval Research under grant no. N00014-85-K0610

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs N.J., 1987.

[2] D. Gross and C.M. Harris, *Fundamentals of Queueing Theory*, Wiley, New York, 2nd edition, 1985.

[3] H.-Y. Huang and T.G. Robertazzi, ''Performance Evaluation of the Manhattan Street Network with Input Buffers", *Proceedings of the International Conference on Communications '92*, Chicago, Ill., June 1992.

[4] L. Kleinrock, *Queueing Systems, Vol. 1: Theory*, Wiley, New York, 1975.

[5] N.F. Maxemchuk, ''The Manhattan Street Network", *Proceedings of IEEE Globecom '85*, 1985, pp. 255-261.

[6] N.F. Maxemchuk, ''Regular Mesh Topologies in Local and Metropolitan Area Networks", *AT&T Technical Journal*, Vol. 64, No. 7, Sept. 1985, pp. 1659-1685.

[7] T.G. Robertazzi and A.A. Lazar, ''Deflection Strategies for the Manhattan Street Network", *Proceedings of the International Conference on Communications '91*, Denver Co., June 1991. A revised version appears in *SUNY at Stony Brook College of Engineering and Applied Science Technical Report #603*, April 24, 1991. Available from T. Robertazzi.

[8] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley, Reading, Mass., 1987.

[9] I.Y. Wang and T.G. Robertazzi, ''Recursive Computation of Steady State Probabilities of Non-product Form Queueing Networks Associated with Computer Network Models", *IEEE Transactions on Communications*, Vol. 38, No. 1, Jan. 1990, pp. 115-117.

[10] H.-Y. Huang and T.G. Robertazzi, ''Recursive Solutions for Discrete Time Queues with Applications to High Speed Switching Fabrics", *Proceedings of the 1992 Conference on Information Sciences and Systems*, Princeton N.J., March 1992.

[11] J. Brassil and R. Cruz, ''Nonuniform Traffic in the Manhattan Street Network", *Proceedings of the International Conference on Communications '91*, Denver Co., June 1991, pp. 1647-1651.

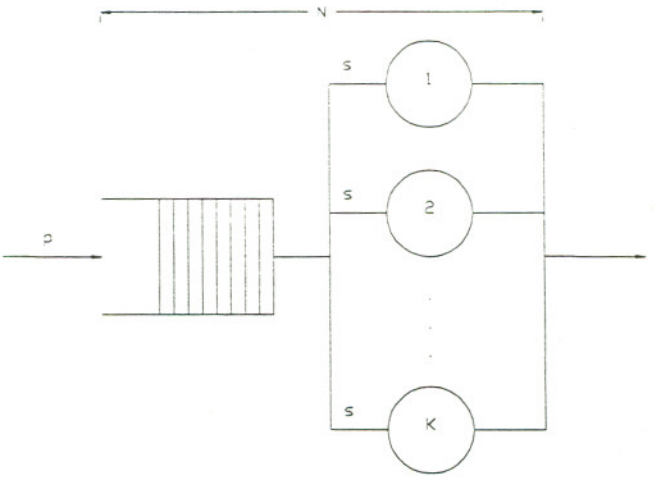

Figure 1: A Discrete Time Geom/Geom/K Queuing System

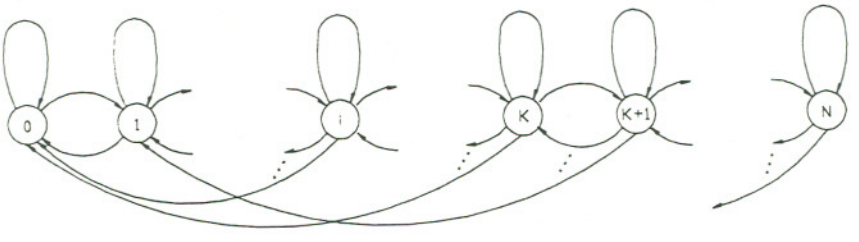

Figure 2: State Transition Diagram of Geom/Geom/K Queuing System

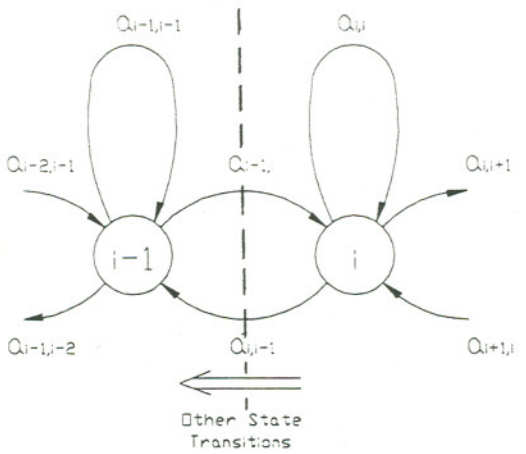

Figure 3: Boundary for Boundary Balance Relation

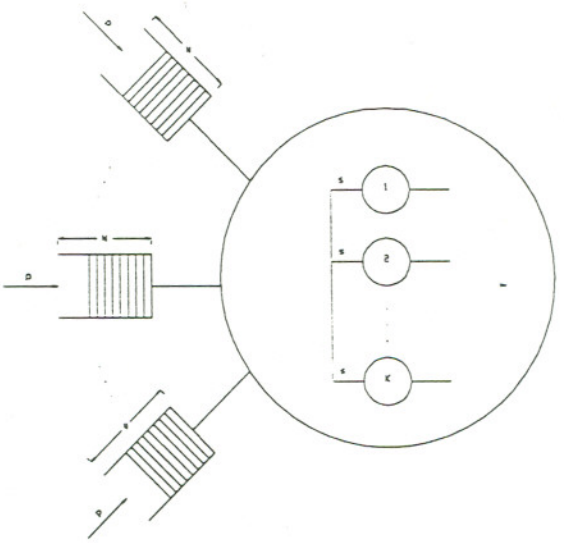

Figure 4: Discrete Time Queue with Varying Number of Servers

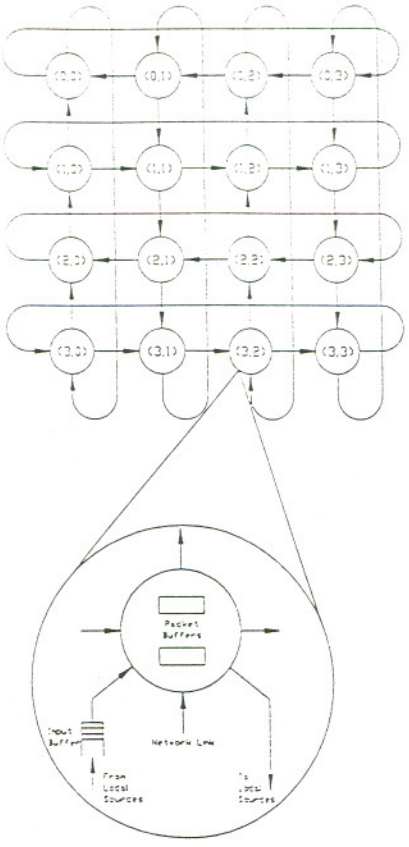

Figure 5: 4x4 MSN

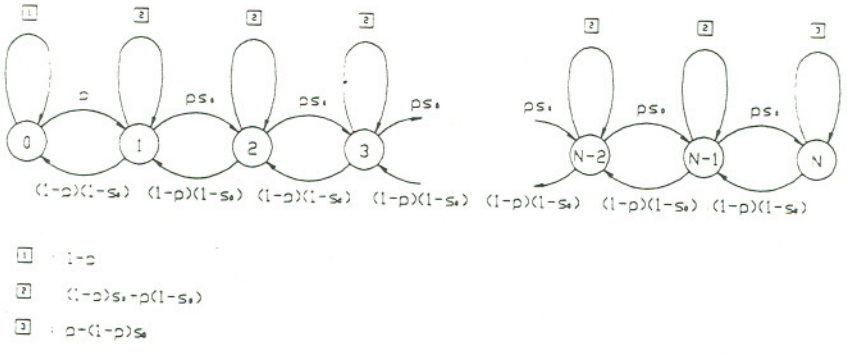

Figure 6: State Transition Diagram for MSN with L=1

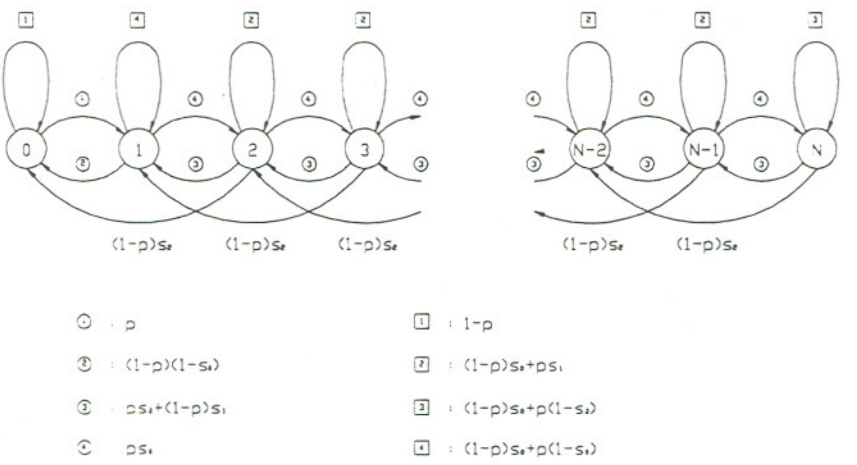

Figure 7: State Transition Diagram for MSN with L=2