Queueing Theory Analysis of Greedy Routing on Arrays and Tori

Mor Harchol[*] Paul E. Black[†]

June 9, 1993

Abstract

We apply queueing theory to derive the probability distribution on the queue buildup associated with greedy routing on an $n \times n$ array and an $n \times n$ torus of processors. We assume packets continuously arrive at each node of the array or torus with Poisson rate λ and have random destinations. We assume an edge may be traversed by only one packet at a time and the time to traverse an edge is exponentially distributed with mean 1.

To analyze the queue size in steady-state, we formulate both these problems as equivalent Jackson queueing network models. With this model, determining the probability distribution on the queue size at each node involves solving $O(n^4)$ simultaneous linear equations. However, we eliminate the need to solve these simultaneous equations by deriving a very simple formula for the total arrival rates and for the expected queue sizes in the case of greedy routing.

This simple formula shows that in the case of the $n \times n$ array, the expected queue size at a node increases as the Euclidean distance of the node from the center of the array decreases. Furthermore, in the case of the $n \times n$ torus, the probability distribution on the queue size is identical for every node.

We also translate our results about queue sizes into results about the average packet delay.

[*]Computer Science Division, UC Berkeley, CA 94720. Supported by National Physical Science Consortium (NPSC) Fellowship.

[†]Computer Science Division, UC Berkeley, CA 94720.

1 Introduction

• PROBLEM JUSTIFICATION

An array of processors is one of the most commonly used communication networks because it has a very simple layout which uses an almost minimal number of wires, and which is also very easy to enlarge. A torus of processors has the same cost benefits, with the added benefit that the nodes are indistinguishable, thereby making it easier to analyze.

The most common type of oblivious routing on arrays or tori of processors is the greedy routing algorithm which sends a packet first to its correct column and only then to its correct row. We investigate the problem of greedily routing packets which arrive continuously at the nodes of the array or torus at Poisson rate λ and have random destinations. This problem is important since it comprises the first half of any randomized routing algorithm.

Since a wire (edge) of the array (or torus) may be used by only one packet at a time, packets naturally get delayed at the processor nodes of the array (torus). It is therefore important in building arrays (tori) of processors that we create appropriate sized buffers at each node of the array (torus) to hold packets which are delayed (in queue). In this paper we determine the size of these necessary buffers.

• PREVIOUS HISTORY

Previous work in this area includes Leighton's work [Leighton,92] which examines the array problem except that:

- Packets are prioritized at the queues in terms of Farthest First, rather than our method which uses FCFS (first come first serve).
- Chernoff Bounds rather than queueing theory is the method of analysis
- Leighton only derives limits on the tail end of the distribution, rather than the whole distribution.

Leighton's work does not include an analysis of tori.

More closely related work is a paper by Stamoulis and Tsitsiklis [Stamoulis, Tsitsiklis,91]. This work uses queueing theory, however the network analyzed is a hypercube (rather than an array), and the authors are concerned more with the problem of average delay of a single packet as it moves through the network, rather than with an analysis of queue size at each node of the network.

• SYNOPSIS OF PAPER

In this paper we solve the problem of determining the queue size at each node of the $n \times n$ array (torus) in steady-state by converting the problem into a Jackson Queueing Network Model which can then be analyzed via queueing theory. The queueing theory analysis requires solving $O(n^4)$ simultaneous linear equations to determine the steady-state total arrival rate at each server, which can then be plugged into the queueing formula to determine a probability distribution on the queue size at each server. One very interesting observation made in this paper is that when the routing algorithm is greedy a very simple formula can be used to determine the total arrival rates for the nodes of the array (torus), making it unnecessary to solve all the simultaneous equations. This greatly simplifies the process of determining the probability distribution on the queue size at each server.

Another consequence of this simple formula is that in the case of an $n \times n$ array, it can be used to prove that the total arrival rate increases as we look at nodes closer and closer to the center of the array. Observing that the expected queue size at a node is directly proportional to the total arrival rate at the node, we are then able to prove that **the expected queue size at a node of the array increases as we look at nodes closer to the center of the array**.

In the case of an $n \times n$ torus, the simple formula for the total arrival rate at each node is constant for all the nodes of the torus. This leads to the conclusion that the **the expected queue size is the same at all the nodes of the torus**.

• OUTLINE OF CHAPTERS

Since the array is more difficult to analyze than the torus, we devote most of the paper to talking exclusively about the array problem. In the last section, we then show how the same type of analysis can be used to solve the torus problem.

Section 2 contains a detailed problem definition, for the case of the $n \times n$ array.

In Section 3 we provide background material on the Jackson Queueing Network Model, and also provide the formula for the probability distribution on the queue size at the servers of this network model.

In Section 4 we firstly show how to cast our original problem into the framework of the Jackson Queueing Network model. Having formulated our problem in terms of a Jackson Queueing Network, we next illustrate the simultaneous equations which must be solved to determine the total arrival rate at each server (which in turn are used to determine the queue size at the server).

Section 5 obviates the need to solve the simultaneous equations of Section 4 by developing a very simple formula for determining the total arrival rates and a very simple formula for the expected queue size at each node. This simple formula shows that the expected queue size at a node of the array increases as the node's Euclidean distance from the center of the array decreases.

Section 6 illustrates the use of this simple formula to immediately determine the probability distribution on the queue sizes of the 5×5 array problem. It also illustrates the general properties proved in Section 5 for the 5×5 array problem.

In Section 7, we apply the same analysis used for an array to determine the probability distribution on the queue size at the processors of the $n \times n$ torus. It turns out that since the torus has no distinguished nodes, the probability distribution on the queue size is the same for every processor node in the torus.

In Section 8 we give an upper bound on the arrival rate λ. We also derive a formula for the average delay of a packet and we compute expected queue sizes for intermediate λ and very high λ.

In Section 9, we summarize our results, and in Section 10 we discuss several alternative useful analyses which could be implemented using the same Jackson Queueing Network Theorem.

In *Appendix A*, we give a more formal derivation for the arrival rate formula Section 5, rather than the more intuitive proof provided in Section 5.

Appendix B looks at an alternative way we could have set up the Jackson Queueing Model for our routing problem on the array, in which we use a different way of classifying the packets.

Lastly, *Appendix C* illustrates the $O(n^4)$ simultaneous equations for the case of the 3×3 array.

2 Problem Definition

Our network is an $n \times n$ array of processors, as shown in Figure 1. New packets arrive at processor $P_{(i,j)}$ at Poisson rate λ, where $0 < \lambda < 1$.

Each packet is assigned a **random destination** in the array. A packet contains a destination field and a data field.

When a new packet arrives, it is routed to its destination via the following **greedy routing algorithm**: First, the packet is routed to its correct column and next to its correct row. If two packets require the same edge, contention is resolved via **First-Come-First-Served** (FCFS).

The time it takes for a packet to move through an edge is exponentially distributed with mean 1. Only one packet may be on an edge at a time.

Our goal is to compute the probability distribution on the queue size at each processor, when the network is in steady-state.

The first step in solving the above problem is to convert it into a common queueing network model. We next analyze the distribution on the queue sizes in the model, which gives us the distribution on the queue sizes in our original problem. In the next section we describe the queueing network model we'll be using.

3 Multiple-Job-Class Open Jackson Queueing Network Model

In this section we describe the Queueing Network Model we will be using in this paper. The model we use allows packets to be associated with a class or type. There are simpler queueing models in which the packets aren't typed, however, as we will see in Section 3 we will need this more extensive model to handle our routing problem of Section 1.

The Queueing Network Model we use [Buzacott,Shanthikumar,93] assumes there are **m servers** with one processor per server. There are **r classes**, or types of packets. Packets of class l arrive at server i from outside the network at a Poisson rate $\mathbf{r_i^{(l)}}$. This model allows packets to change their class as they move from server to server. A packet of class l at server i next moves to server j and becomes of class k type with probability $\mathbf{p_{ij}^{(l)(k)}}$. (The queueing network model assumes a complete directed graph connecting the servers. We can model a network with fewer edges, by simply making some of the edge probabilities zero.) A packet at server i may also leave the network, with some probability, rather than continuing to another server. Lastly the service rate at server i is exponentially distributed with rate $\mu_{\mathbf{i}}$.

We will use the notation $\mathbf{n}_{\mathbf{i}}^{(\mathbf{l})}$ to denote the number of packets at server i of class l, and $\mathbf{n}_{\mathbf{i}} = \sum_{l=1}^{r} n_i^{(l)}$ to denote the total number of packets at server i.

Theorem 1 (Buzacott,Shanthikumar,93) *When the queueing network is in steady state,*

$$p_i(n_i^{(1)}, \ldots, n_i^{(r)}) = \binom{n_i}{n_i^{(1)}, \ldots, n_i^{(r)}} \prod_{l=1}^{r} \left(\frac{\hat{\lambda}_i^{(l)}}{\hat{\lambda}_i} \right)^{n_i^{(l)}} (1 - \rho_i)\rho_i^{n_i} \tag{1}$$

$$\begin{aligned} \textit{where } \rho_i &= \frac{\hat{\lambda}_i}{\mu_i} && (2) \\ \hat{\lambda}_i &= \sum_{l=1}^{r} \hat{\lambda}_i^{(l)} \\ \hat{\lambda}_i^{(l)} &= r_i^{(l)} + \sum_{j=1}^{m} \sum_{k=1}^{r} p_{ji}^{(k)(l)} \hat{\lambda}_j^{(k)} && (3) \end{aligned}$$

The proof of this theorem is given in [Buzacott,Shanthikumar,93].

The above theorem tells us how to compute the joint probability of having, for example, $n_i^{(1)}$ packets of type 1 at server i and $n_i^{(2)}$ packets of type 2 at server i ..., and $n_i^{(r)}$ packets of type r at server i. It says we must first solve $m \cdot r$ simultaneous linear equations to obtain $\hat{\lambda}_i^{(l)}$ for $l = 1, \ldots, r$ and $i = 1, \ldots, m$. Then we plug these $\hat{\lambda}_i^{(l)}$'s into the above formula for $p_i(n_i^{(1)}, \ldots, n_i^{(r)})$.

Note that $\hat{\lambda}_i^{(l)}$ represents the rate at which packets of class l flow into server i, including both those packets which arrive from outside as well as packets arriving from other servers.

Now, suppose we want to know the probability that there are n_i packets at server i. By definition, $\mathbf{p}_{\mathbf{i}}(\mathbf{n}_{\mathbf{i}})$, the probability that there are n_i packets at server i, is the sum of Equation 1 over all values of $n_i^{(1)}, n_i^{(2)}, \ldots, n_i^{(r)}$ such that $n_i^{(1)} + n_i^{(2)} + \ldots + n_i^{(r)} = n_i$.

Corollary 2

$$p_i(n_i) = (1 - \rho_i)\rho_i^{n_i}$$

where ρ_i is defined as above in Theorem 1.

Proof: Note that the expression for $p_i(n_i^{(1)}, \ldots, n_i^{(r)})$ begins with a multinomial probability, and observe that by definition $\sum_{l=1}^{r} \left(\frac{\hat{\lambda}_i^{(l)}}{\hat{\lambda}_i} \right) = 1$. ■

Lastly let N_i be a random variable representing the number of packets at server i. Since by Corollary 2 N_i has a distribution which is geometric times a factor ρ_i, we have:

$$
\begin{aligned}
E[N_i] &= \frac{\rho_i}{1-\rho_i} \\
var(N_i) &= \frac{\rho_i}{(1-\rho_i)^2}
\end{aligned}
\tag{4}
$$

4 Modeling the Routing Problem on an Array as a Jackson Queueing Network

In this section we show how to formulate the problem of greedy routing on an $n \times n$ array in terms of the queueing network model introduced in Section 2 so that we may apply Theorem 1 to determine the probability distribution on the queue sizes. We build up to the exact formulation with some discussion.

A first attempt might be to let each of the n^2 nodes of the array be a server, where the servers are connected only by those edges which are in the array. (All other edges between processors have 0 probabilities associated with them.) Now the probability that a packet moves from server (i,j) to server (i',j') is either 1 or 0, depending on the destination of the packet. **If we now make the destination of the packet be the "class" associated with the packet**, the probability that a packet moves from server (i,j) to server (i',j') depends only on (i,j), (i',j'), and the class of the packet, as required by the queueing network model of Section 2. (Note that if our queueing network model didn't allow for classes, the process of moving from server to server would not be Markovian). Observe that in this formulation, the class of a packet does not change as the packet moves between servers.

The above formulation doesn't quite work, however. Suppose, for example we model the service rate of each server as 1. Queuing theory assumes one queue at each server. However, suppose that 2 packets arrive at the same node and want exit the node in different directions. The array allows both packets to leave the node since they won't conflict with each other. In the current queueing model formulation, however, one packet would first have to wait for the other packet to finish. Setting the service to 4, the number of outgoing edges, indicates that the node can send four packets out on one edge which isn't right, either.

We can fix this problem by realizing that congestion in the array is an edge problem not a node problem. Therefore we associate a server with each outgoing *edge*, rather than each node of the array, as shown in Figure 2.

Rows and columns are numbered from 0 to $n-1$ with $0,0$ being in the upper, lefthand corner. We use the notation $\mathbf{P_{i,j,L}}$ to denote the left processor at row i, column j. Likewise $\mathbf{P_{i,j,R}}$ $\mathbf{P_{i,j,U}}$ $\mathbf{P_{i,j,D}}$ respectively denote the right, up and down (i,j) processors. Lastly we let $\mathbf{P_{i,j,C}}$ denote the center (i,j) processor. We will refer to processors $P_{i,j,L}, P_{i,j,R}, P_{i,j,U}$, and $P_{i,j,D}$ respectively as the left, right, up, and down petals of the flower (i,j), and to $P_{i,j,C}$ as the center processor of flower (i,j).

Each petal processor has its own queue and sits on an edge. Packets on queue at a petal processor may be thought of as waiting to use that edge. Only one packet at a time can use the edge, and the time spent by a packet on an edge is on average one time unit (the service rate of every petal processor is 1).

When a packet originates at a node of the original $n \times n$ array, we model it as originating on the petal (of the corresponding flower) which corresponds to the first edge the packet must use to get to its destination. The packet then moves from the petal of one flower to a petal of another flower, etc., until it reaches its destination, at which point it moves to the center processor of its destination flower and leaves the system. We set the service rate for the center processors to be ∞.

The edges into one processor in our queueing network are shown in Figure 3. For clarity edges to and from the central processor (that is, into and out of the system) and edges from the petals to other nodes are not shown. In the greedy algorithm a packet which moves upward will only move upward or leave the system subsequently. So the only edge into the flower from the Up petal below goes to the Up petal of the middle flower. Likewise the only edge from the Down petal above is to the Down petal in the middle.

Packets traveling left or right may continue in the same direction or may turn up or down (or they may leave the system — not shown). So there are edges from the left and right petals of the flowers on the side to the Up, Down, and Left and Right petals in the middle.

We are now ready to formulate the routing problem on an $n \times n$ array as a Jackson Queueing Network. Given an $n \times n$ array of processors $P_{(i,j)}$ with grid connections, such that:

- new packets arrive at $P_{(i,j)}$ from outside the system at rate λ,
- Each packet is assigned a random destination when it first arrives.
- The packet is routed to its destination via the Greedy algorithm.
- The rate at which a packet traverses an edge is exponential rate 1.
- Only one packet may traverse an edge at a time.
- Edge contention is resolved using FCFS

We analyze the queue sizes at the nodes of the above array by looking at the following queueing network model:

- The number of servers, $\mathbf{m}$, is $5n^2$, denoted by $P_{i,j,R}$, $P_{i,j,L}$, $P_{i,j,U}$, $P_{i,j,D}$, $P_{i,j,C}$, for $i = 0, \ldots, n-1$ and $j = 0, \ldots, n-1$.
- The number of classes, $\mathbf{r}$, is n^2, one for each possible destination.
- Packets never change class. A packet of class d (that is, for destination d) at server P_{ijS} next moves to server $P_{i'j'S'}$ with probability $\mathbf{p^{(d)}_{ijS,i'j'S'}}$. It's value is 1 if the greedy algorithm routes a packet at server $i_{i,j,S}$ with destination d to server $i_{i',j',S'}$, and 0 otherwise.
- Packets of class d arrive at server $i_{i,j,S}$ with Poisson rate $\mathbf{r^{d}_{i,j,S}} = c \cdot \frac{q}{n^2}$, where $q =$ the number of possible destinations a packet at server $P_{i,j,S}$ might be headed for via the greedy algorithm.

- $\mu_\mathbf{i}$, the service rate at server i, is 1 for all petal servers and ∞ for the center servers.

For the above queueing network model, the system of linear equations specified in Corollary 2 from Section 2 become:

$$\begin{aligned}
\hat{\lambda}_i^{(d)} &= r_i^{(d)} + \sum_{j=1}^{m} p_{ji}^{(d)} \hat{\lambda}_j^{(d)} \qquad (5)\\
\hat{\lambda}_i &= \sum_{d=1}^{r} \hat{\lambda}_i^{(d)} \\
\rho_i &= \frac{\hat{\lambda}_i}{\mu_i} = \hat{\lambda}_i \\
p_i(n_i) &= (1-\rho_i)\rho_i^{n_i}
\end{aligned}$$

Thus to calculate $\mathbf{p_i(n_i)}$, the probability of having n_i packets at server i, we can first solve the simultaneous equations for all the $\hat{\lambda}_i^{(d)}$'s and then sum them to obtain the $\hat{\lambda}_i$'s. Since the service rate, μ_i is 1, the $\hat{\lambda}_i$'s are the ρ_i's which then give us $p_i(n_i)$ for each i and any n_i we choose. Also the number of packets has a geometric distribution.

The expected number of packets queued at a node is the sum of the expected number of packets queued at each petal of the associated flower.

$$E[N_{rc}] = \sum_{S \in R,U,L,D} E[N_{rcS}]$$

The number of simultaneous linear equations generated by Equation 5 is $r \cdot m = 5n^4$. Solving a system of $5n^4$ linear equations in $5n^4$ unknowns seems daunting. For general networks with feedback paths there are mutually dependent variables. **However in the model of a greedy routing algorithms, no packet path has a loop, so no variables are mutually dependent. This, along with other features of greedy routing makes a general analytic solution of the system of equations for arbitrary sized arrays feasible**. As an example, Appendix C shows the simultaneous linear equations which result for the case of a 3×3 array.

In the next section, we propose, however, an easier way to obtain the $\hat{\lambda}_i$'s directly by combinatorial analysis.

5 A Simple Method For Determining Queue Size on Array

Recall from Section 2 that $\hat{\lambda}_i$ represents the total rate at which packets arrive at server i from both outside the system as well as from other servers. In the observation below, we give an equivalent interpretation to $\hat{\lambda}_i$ which also allows it to be computed quickly.

Observation 3 *For oblivious routing schemes, the value $\hat{\lambda}_i$ has a simple, intuitive meaning: it is the number of paths through node i weighted by the frequency of use of each path. If the frequencies of use of all paths are the same, $\hat{\lambda}_i$ is just the number of paths times the frequency of use.*

Theorem 4 *The total arrival rate of packets at petal node $P_{r,c,S}$ is*

$$\begin{aligned}
\hat{\lambda}_{P_{r,c,R}} &= \frac{\lambda}{n}(col(P)+1)(n-col(P)-1) \\
\hat{\lambda}_{P_{r,c,L}} &= \frac{\lambda}{n}(n-col(P))col(P) \\
\hat{\lambda}_{P_{r,c,U}} &= \frac{\lambda}{n}(n-row(P))row(P) \\
\hat{\lambda}_{P_{r,c,D}} &= \frac{\lambda}{n}(row(P)+1)(n-row(P)-1)
\end{aligned}$$

Proof: Here we present an combinatorial proof of these equations. A derivation from the original defining equations is given in Appendix A.

Consider the number of paths through some right petal $P_{r,c,R}$. All paths through that petal must have a destination in a flower to the right of it. There are $n(n-col(P)-1)$ destinations that paths might have. See Figure 4. Additionally since the algorithm routes packets to the correct column before changing rows, only packets which arrive from outside at the $col(P)$ flowers to the left on the same row, plus those arriving from the outside at the flower $P_{r,c}$, go through the petal. Thus there are a total of $(col(P)+1)n(n-col(P)-1)$ paths through the petal. Since the arrival rate at each flower from outside is λ and each of n^2 destinations is equally likely, the arrival rate at any right petal $P_{r,c,R}$ is $\frac{\lambda}{n}(col(P)+1)(n-col(P)-1)$.

The arguments for petals in other directions is similar. ■

The above theorem gives us an easier way to compute all the $\hat{\lambda}_i$'s and thereby the $p_i(n_i)$'s. Note that the above theorem assumes (as we have assumed throughout this paper) that the arrival rate from outside to the nodes in the original $n \times n$ array is the same for every node.

Theorem 4 doesn't mention the arrival rate at the center server of each flower. This can safely be ignored. Since any packet arriving at the center node leaves the system, the arrival rate can't influence the arrival rate at any other petal. Also since packets leave the system immediately, that is the service rate $\mu_{P_{rcC}}$ is ∞, no queue ever forms:

$$\rho_{P_{rcC}} = \frac{\hat{\lambda}_{P_{rcC}}}{\infty} = 0$$

and

$$E[N_{P_{rcC}}] = \frac{\rho_{P_{rcC}}}{1-\rho_{P_{rcC}}} = 0$$

Theorem 5 *The expected total number of packets in queue at a flower* (r,c) *is*

$$E[N_{(r,c)}] = \frac{n}{\lambda}\left(\frac{1}{b+x^2-x}+\frac{1}{b+(x+1)^2-(x+1)}+\frac{1}{b+y^2-y}+\frac{1}{b+(y+1)^2-(y+1)}\right)-4$$

where $b=\frac{n}{\lambda}-\frac{n^2-1}{4}$ *and* x *and* y *are the horizontal and vertical distance of* (r,c) *from the center of the array.*

Proof: Recall we number rows and columns $0,\ldots,n-1$. The center of the array is at $(\frac{n-1}{2},\frac{n-1}{2})$. Let $x=col(P)-\frac{n-1}{2}$ and $y=row(P)-\frac{n-1}{2}$, the x and y offsets of the node from the center of the array. So $col(P)=x+\frac{n-1}{2}$ and $row(P)=y+\frac{n-1}{2}$. Note that when n is even, the center of the array as well as the offsets are fractions.

Rewriting the formulas from Theorem 4 in terms of x and y gives

$$\begin{aligned}
\hat{\lambda}_{P_{r,c,R}} &= \frac{\lambda}{n}\left(\frac{n-1}{2}+(x+1)\right)\left(\frac{n+1}{2}-(x+1)\right)\\
\hat{\lambda}_{P_{r,c,L}} &= \frac{\lambda}{n}\left(\frac{n-1}{2}+x\right)\left(\frac{n+1}{2}-x\right)\\
\hat{\lambda}_{P_{r,c,U}} &= \frac{\lambda}{n}\left(\frac{n-1}{2}+y\right)\left(\frac{n+1}{2}-y\right)\\
\hat{\lambda}_{P_{r,c,D}} &= \frac{\lambda}{n}\left(\frac{n-1}{2}+(y+1)\right)\left(\frac{n+1}{2}-(y+1)\right)
\end{aligned}$$

Using Equation 4 we have

$$\begin{aligned}
E[N_{P_{r,c,R}}] &= \frac{\frac{\lambda}{n}\left(\frac{n-1}{2}+(x+1)\right)\left(\frac{n+1}{2}-(x+1)\right)}{1-\frac{\lambda}{n}\left(\frac{n-1}{2}+(x+1)\right)\left(\frac{n+1}{2}-(x+1)\right)}\\
&= \frac{\frac{n^2-1}{4}-(x+1)^2+(x+1)}{\left(\frac{n}{\lambda}-\frac{n^2-1}{4}\right)+(x+1)^2-(x+1)}\\
E[N_{P_{r,c,L}}] &= \frac{\frac{\lambda}{n}\left(\frac{n-1}{2}+x\right)\left(\frac{n+1}{2}-x\right)}{1-\frac{\lambda}{n}\left(\frac{n-1}{2}+x\right)\left(\frac{n+1}{2}-x\right)}\\
&= \frac{\frac{n^2-1}{4}-x^2+x}{\left(\frac{n}{\lambda}-\frac{n^2-1}{4}\right)+x^2-x}\\
E[N_{P_{r,c,U}}] &= \frac{\frac{\lambda}{n}\left(\frac{n-1}{2}+y\right)\left(\frac{n+1}{2}-y\right)}{1-\frac{\lambda}{n}\left(\frac{n-1}{2}+y\right)\left(\frac{n+1}{2}-y\right)}\\
&= \frac{\frac{n^2-1}{4}-y^2+y}{\left(\frac{n}{\lambda}-\frac{n^2-1}{4}\right)+y^2-y}\\
E[N_{P_{r,c,D}}] &= \frac{\frac{\lambda}{n}\left(\frac{n-1}{2}+(y+1)\right)\left(\frac{n+1}{2}-(y+1)\right)}{1-\frac{\lambda}{n}\left(\frac{n-1}{2}+(y+1)\right)\left(\frac{n+1}{2}-(y+1)\right)}
\end{aligned}$$

$$= \frac{\frac{n^2-1}{4} - (y+1)^2 + (y+1)}{\left(\frac{n}{\lambda} - \frac{n^2-1}{4}\right) + (y+1)^2 - (y+1)}$$

Let $a = \frac{n^2-1}{4}$ and $b = \frac{n}{\lambda} - \frac{n^2-1}{4}$. We see that

$$\begin{aligned}
E[N_{P_{r,c,R}}] &= \frac{a - (x+1)^2 + (x+1)}{b + (x+1)^2 - (x+1)} \\
E[N_{P_{r,c,L}}] &= \frac{a - x^2 + x}{b + x^2 - x} \\
E[N_{P_{r,c,U}}] &= \frac{a - y^2 + y}{b + y^2 - y} \\
E[N_{P_{r,c,D}}] &= \frac{a - (y+1)^2 + (y+1)}{b + (y+1)^2 - (y+1)}
\end{aligned}$$

The expected value of the queue at a node is the sum of expected values of queues at each petal, so

$$\begin{aligned}
E[N_{r,c}] &= \sum_S E[N_{P_{r,c,S}}] \\
&= \frac{a - (x+1)^2 + (x+1)}{b + (x+1)^2 - (x+1)} + \frac{a - x^2 + x}{b + x^2 - x} + \frac{a - y^2 + y}{b + y^2 - y} + \frac{a - (y+1)^2 + (y+1)}{b + (y+1)^2 - (y+1)} \\
&= \frac{a}{b + (x+1)^2 - (x+1)} - \frac{(x+1)^2 + (x+1)}{b + (x+1)^2 - (x+1)} + \frac{a}{b + x^2 - x} - \frac{x^2 + x}{b + x^2 - x} \\
&\quad + \frac{a}{b + y^2 - y} - \frac{y^2 + y}{b + y^2 - y} + \frac{a}{b + (y+1)^2 - (y+1)} - \frac{(y+1)^2 + (y+1)}{b + (y+1)^2 - (y+1)} \\
&= \frac{a}{b + (x+1)^2 - (x+1)} + \frac{b}{b + (x+1)^2 - (x+1)} + \frac{a}{b + x^2 - x} + \frac{b}{b + x^2 - x} \\
&\quad + \frac{a}{b + y^2 - y} + \frac{b}{b + y^2 - y} + \frac{a}{b + (y+1)^2 - (y+1)} + \frac{b}{b + (y+1)^2 - (y+1)} - 4 \\
&= (a+b)\left(\frac{1}{b + (x+1)^2 - (x+1)} + \frac{1}{b + x^2 - x} + \frac{1}{b + y^2 - y} + \frac{1}{b + (y+1)^2 - (y+1)}\right) - 4 \\
&= \frac{n}{\lambda}\left(\frac{1}{b + (x+1)^2 - (x+1)} + \frac{1}{b + x^2 - x} + \frac{1}{b + y^2 - y} + \frac{1}{b + (y+1)^2 - (y+1)}\right) - 4
\end{aligned}$$

■

In the next section we will show an example of how the formulas derived in theorem 4 together with Corollary 2 are used to quickly derive the exact probability distribution on the queue sizes in the case of the 5×5 array. We will also observe the phenomenon of Theorem 5, when analyzing the 5×5 array.

6 An Example: Numerical Results

In this section we compute the probability distribution on the queue sizes of the nodes (i, j) of the 5×5 array of processors where $i = 0, \ldots, 4$ and $j = 0, \ldots, 4$. To do this we look at the associated

Jackson Queueing Network as described in Section 4, and compute the probability distribution on the queue size of each petal processor in the Jackson Network. Then, for each flower, (i,j), we sum up the queue size of each of its petals to obtain the queue size for the flower, which in turn is the queue size of node (i,j) in the original 5×5 array of processors.

To compute the probability distribution on the queue size of each petal processor in the associated Jackson Queueing Network, we use Equation 4 (see Section 3), however rather than solving simultaneous equations, we derive the $\hat{\lambda}_i$'s directly by the formulas in Theorem 5.

Figure 5a shows the $\hat{\lambda}_i$ for each petal server i as derived using Theorem 4. We assume $\lambda = \frac{5}{12}$, since it makes the numbers nice. Since $\mu_i = 1$ for all petal servers i, $\rho_i = \hat{\lambda}_i$. Corollary 2 then tells us that the probability that there are n_i packets in queue at petal server i, $p_i(n_i)$, is equal to $(1-\rho_i)\rho_i^{n_i}$. We have not drawn the probability distribution on the queue size for each petal, however, it is clear that the distribution is geometric and therefore has an exponential shape. (Note that since the queue sizes on the petals of a flower are not independent we can't simply combine these probability distributions, however we can sum their expected values).

In Figure 5b, we show $E[N_i]$, the expected number of packets in queue at i, for each petal server i. Lastly, in Figure 5c, for each flower (i,j), we total the expected number of packets in queue at each of its petals, to obtain the expected number of packets in queue at node (i,j) of the original 5×5 array. Observe that Figure 5c clearly illustrates the phenomenon described in Theorem 5.

7 A Simple Method For Determining Queue Size on A Torus

In this section we study the queue build-up on an $n \times n$ torus with bidirectional edges. A 5×5 torus is illustrated in Figure 6. Recall that for the $n \times n$ array, the expected queue size at each node decreases as we moved farther from the center of the array. Since the torus has no distinguished nodes, it seems reasonable that the expected queue size at each node of a torus should be the same. In this section we show this to be the case.

The assumptions are that packets arrive at each node of the torus at Poisson rate λ. The packets have random destinations. We assume the packets take the *greedy* path to their destinations. On the torus, this means that a packet first moves within its row to the correct destination column by taking the shortest route to the column (either left or right $\leq \frac{n}{2}$ steps). Then, the packet moves with that column to its destination again by taking the shortest route (either up or down $\leq \frac{n}{2}$ steps). If n is even, destinations exactly $\frac{n}{2}$ nodes away (that is, equally close either direction) are routed up or to the right.

We also assume only one packet may traverse an edge at a time, and the time to traverse an edge is exponentially distributed with mean 1. Therefore queues build up on the edges of the torus. To analyze the queue size, we will convert the torus into a Jackson Queueing Network by replacing each node of the torus by a flower with 4 petal servers, one for each edge incident to the node. This is the same transformation we made on the $n \times n$ array. As in the case of the array, we assume the service rate at each petal server is exponentially distributed with mean 1. Also as in the case of the array, we associate a class with each packet, which is the packet's final destination node.

To apply the queueing theory formulas of section 3, we need to compute $\hat{\lambda}_i$, the total arrival rate into server i, for each server i in the network. We could of course do this by solving the $O(n^4)$ simultaneous linear equations for the $\hat{\lambda}_i^{(d)}$'s as described in section 4, and then summing the $\hat{\lambda}_i^{(d)}$'s to obtain $\hat{\lambda}_i$.

A far simpler idea is to use observation 3 to obtain a version of theorem 4 for the torus. We present this theorem below. Observe that the total arrival rate for a petal is independent of the petal's row and column, as we expected. First we present the average distance and arrival rates for a ring.

Lemma 6 *For a ring with n nodes, the average distance to any destination on a ring is*

$$D_{ave} = \begin{cases} \frac{n-\frac{1}{n}}{4} & \text{if } n \text{ is odd} \\ \frac{n}{4} & \text{if } n \text{ is even} \end{cases}$$

Proof: We don't show the derivation here. ■

Lemma 7 *For a ring with n nodes, where n is even, the total arrival rate of packets at petal node P_S is*

$$\begin{aligned} \hat{\lambda}_{P_R} &= \frac{\lambda}{8}(n+2) \\ \hat{\lambda}_{P_L} &= \frac{\lambda}{8}(n-2) \end{aligned}$$

If n is odd, the total arrival rate of packets at petal node P_S is

$$\hat{\lambda}_{P_R} = \hat{\lambda}_{P_L} = \frac{\lambda}{8}(n - \frac{1}{n})$$

Proof: • RING: n: even

$$\begin{aligned} \hat{\lambda}_{P_R} &= \text{(frequency of any path)(no. paths through } i_R) \\ &= \frac{\lambda}{n}(\frac{n}{2} + (\frac{n}{2} - 1) + \cdots + 1) \\ &= \frac{\lambda}{n}(1 + 2 + 3 + \cdots + \frac{n}{2}) \\ &= \frac{\lambda}{8}(n+2) \end{aligned}$$

$$\begin{aligned} \hat{\lambda}_{P_L} &= \text{(frequency of any path)(no. paths through } i_L) \\ &= \frac{\lambda}{n}((\frac{n}{2} - 1) + (\frac{n}{2} - 2) + \cdots + 1) \\ &= \frac{\lambda}{n}(1 + 2 + 3 + \cdots + (\frac{n}{2} - 1)) \\ &= \frac{\lambda}{8}(n-2) \end{aligned}$$

• RING: n: odd

$$\begin{aligned}
\hat{\lambda}_{P_R} &= \hat{\lambda}_{P_L} \\
&= \frac{\lambda}{n}(\frac{(n-1)}{2} + (\frac{(n-1)}{2} - 1) + \cdots + 1) \\
&= \frac{\lambda}{n}(1 + 2 + 3 + \cdots + \frac{(n-1)}{2}) \\
&= \frac{\lambda}{8}(n - \frac{1}{n})
\end{aligned}$$

∎

Theorem 8 *For an $n \times n$ torus, where n is even, the total arrival rate of packets at petal node P_S is*

$$\begin{aligned}
\hat{\lambda}_{P_R} &= \frac{\lambda}{8}(n+2) \\
\hat{\lambda}_{P_L} &= \frac{\lambda}{8}(n-2) \\
\hat{\lambda}_{P_U} &= \frac{\lambda}{8}(n+2) \\
\hat{\lambda}_{P_D} &= \frac{\lambda}{8}(n-2)
\end{aligned}$$

Regardless of the direction, where n is odd the total arrival rate of packets at petal node P_S is

$$\hat{\lambda}_{P_S} = \frac{\lambda}{8}(n - \frac{1}{n})$$

Proof:

To compute $\hat{\lambda}_{P_S}$, we see by Observation 3 that

$$\hat{\lambda}_{P_S} = \text{(frequency of any path) (number of paths through } P_S\text{)}$$

• TORUS: n: even

When n is even, the maximum distance travelled Up or Right is $\frac{n}{2}$, while the maximum distance travelled Down or Left is $\frac{n}{2} - 1$.

Figure 7a illustrates computing the number of greedy paths through P_U. Computing the number of greedy paths through P_D is the same process, except that now the maximum distance travelled is $\frac{n}{2} - 1$.

$$\hat{\lambda}_{P_U} = \text{(frequency of any path)(no. greedy paths through } P_U\text{)}$$

$$\begin{aligned}
&= \frac{\lambda}{n^2}(n \cdot \frac{n}{2} + n \cdot (\frac{n}{2} - 1) + \cdots + n \cdot 1) \\
&= \frac{\lambda}{n^2} \cdot n \cdot (1 + 2 + 3 + \cdots + \frac{n}{2}) \\
&= \frac{\lambda}{8}(n + 2)
\end{aligned}$$

$$\begin{aligned}
\hat{\lambda}_{P_D} &= \text{(frequency of any path)(no. greedy paths through } P_D) \\
&= \frac{\lambda}{n^2}(n \cdot (\frac{n}{2} - 1) + n \cdot (\frac{n}{2} - 2) + \cdots + n \cdot 1) \\
&= \frac{\lambda}{n^2} \cdot n \cdot (1 + 2 + 3 + \cdots + (\frac{n}{2} - 1)) \\
&= \frac{\lambda}{8}(n - 2)
\end{aligned}$$

Figure 7*b* illustrates computing the number of greedy paths through P_R. Computing the number of greedy paths through P_L is the same process, except that now the maximum distance travelled is $\frac{n}{2} - 1$.

$$\begin{aligned}
\hat{\lambda}_{P_R} &= \text{(frequency of any path)(no. greedy paths through } P_R) \\
&= \frac{\lambda}{n^2}(\frac{n}{2} \cdot n + (\frac{n}{2} - 1) \cdot n + \cdots + 1 \cdot n) \\
&= \frac{\lambda}{n^2} \cdot n \cdot (1 + 2 + 3 + \cdots + \frac{n}{2}) \\
&= \frac{\lambda}{8}(n + 2)
\end{aligned}$$

$$\begin{aligned}
\hat{\lambda}_{P_L} &= \text{(frequency of any path)(no. greedy paths through } P_L) \\
&= \frac{\lambda}{n^2}((\frac{n}{2} - 1) \cdot n + (\frac{n}{2} - 2) \cdot n + \cdots + 1 \cdot n) \\
&= \frac{\lambda}{n^2} \cdot n \cdot (1 + 2 + 3 + \cdots + (\frac{n}{2} - 1)) \\
&= \frac{\lambda}{8}(n - 2)
\end{aligned}$$

- TORUS: n: odd

When n is odd, the maximum distance travelled in any direction is $\frac{n-1}{2}$. Computing $\hat{\lambda}_{P_U}$ is the same as computing $\hat{\lambda}_{P_D}$ and computing $\hat{\lambda}_{P_R}$ is the same as computing $\hat{\lambda}_{P_L}$.

$$\hat{\lambda}_{P_U} = \hat{\lambda}_{P_D} = \text{(frequency of any path)(no. greedy paths through } P_D)$$

$$
\begin{aligned}
&= \frac{\lambda}{n^2}(n \cdot \frac{n-1}{2} + n \cdot (\frac{n-1}{2} - 1) + \cdots + n \cdot 1) \\
&= \frac{\lambda}{n^2} \cdot n \cdot (1 + 2 + 3 + \cdots + \frac{n-1}{2}) \\
&= \frac{\lambda}{8}(n - \frac{1}{n})
\end{aligned}
$$

The computation of $\hat{\lambda}_{P_L}$ is similar.

■

It is interesting to observe that the total arrival rates computed above for the torus are the same as the total arrival rates in the case of a ring.

Using the $\hat{\lambda}_{P_S}$'s, we can now compute the expected queue size at each node from equation 4. If n is even, the expected queue size at node i is

$$E[N_i] = \frac{2(n+2)}{\frac{8}{\lambda} - (n+2)} + \frac{2(n-2)}{\frac{8}{\lambda} - (n-2)}$$

If n is odd, the expected queue size is

$$E[N_i] = \frac{n^2 - 1}{\frac{8n}{\lambda} - (n^2 + 1)}$$

8 Bounds on λ, Queue Sizes, and Delay Times for Arrays

Up until this point, we never discussed what values of λ (arrival rate at each node from outside) were plausible, i.e., allowed the network to reach steady state. In this section, we give upper bounds for λ. We will then compute the queue sizes at the nodes of the array for specific values of λ, including 99% of maximum and half of maximum. Lastly, we will use Little's Formula to translate our knowledge about queue sizes into formulas for packet delay times.

8.1 Bounds on λ

Theorem 9 *In order for the array network to reach steady state, we must have*

$$\lambda < \frac{4}{n}$$

Proof: Recall the formula for expected queue size at i_S (node i, petal S), given in Equation 4.

$$E[N_{i_S}] = \frac{\rho_{i_S}}{1 - \rho_{i_S}} = \frac{\hat{\lambda}_{i_S}}{1 - \hat{\lambda}_{i_S}}$$

where the $\hat{\lambda}_{i_S}$'s are as defined in Theorem 4. Thus the expected queue size at i_S becomes infinite as $\hat{\lambda}_{i_S}$ approaches 1.

The expected queue size is largest for the left or upper petal of the center node of the array, i.e., node $(\frac{n}{2},\frac{n}{2})$. By Theorem 4,

$$\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},L}} = \frac{\lambda}{n}\cdot\frac{n}{2}\cdot\frac{n}{2} = \lambda\frac{n}{4}$$

To ensure that the queue size is finite, we require $\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},L}} < 1$, which implies $\lambda < \frac{4}{n}$. ■

8.2 Average Delay For Array

In this subsection we derive the average delay of packets.

Little's Formula [Buzacott,Shanthikumar,93] states

$$\overline{N} = \lambda\overline{T}$$

where $\overline{N}$ is the average number of jobs in queue and $\overline{T}$ is the average delay of a job and λ is the arrival rate of jobs. Solving for the average delay

$$\overline{T} = \frac{\overline{N}}{\lambda}$$

The arrival rate of jobs *in the system* is $n^2\lambda$ (a different λ) since there are n^2 nodes in the system. We solve here for n odd. Let $q = 4b-1 = n(\frac{4}{\lambda}-n)$.

$$\begin{aligned}
\overline{T} &= \frac{1}{\lambda}\overline{N} \\
&= \frac{1}{n^2\lambda}\sum_{x=\frac{-(n-1)}{2}}^{\frac{n-1}{2}}\sum_{y=\frac{-(n-1)}{2}}^{\frac{n-1}{2}} E[N_{x,y}] \\
&\approx \frac{1}{n^2\lambda}\int_{\frac{-n}{2}}^{\frac{n}{2}}\int_{\frac{-n}{2}}^{\frac{n}{2}}\left(\frac{1}{b+x^2-x}+\frac{1}{b+(x+1)^2-(x+1)}+\frac{1}{b+y^2-y}+\frac{1}{b+(y+1)^2-(y+1)}-4\right)dxdy \\
&= \frac{1}{n^2\lambda}\left(\frac{2}{\sqrt{q}}\left(y\tan^{-1}\frac{2x-1}{\sqrt{q}}+y\tan^{-1}\frac{2x+1}{\sqrt{q}}+x\tan^{-1}\frac{2y-1}{\sqrt{q}}+x\tan^{-1}\frac{2y+1}{\sqrt{q}}\right)-4xy\right)\Bigg|_{x,y=-\frac{n}{2}}^{x,y=\frac{n}{2}} \\
&= \frac{4}{n\lambda}\left(\frac{2}{\sqrt{q}}\left(\tan^{-1}\frac{n-1}{\sqrt{q}}+\tan^{-1}\frac{n+1}{\sqrt{q}}\right)-n\right)
\end{aligned}$$

8.3 Average Delay for Torus

In this section we compute the average delay of a packet in the case of a torus. As in the previous section, we simply use Little's Formula. Now, however the computations are much simpler because the expected queue size at each node is the same.

We assume n:odd

$$
\begin{aligned}
\overline{T} &= \frac{1}{n^2\lambda}\overline{N} \\
&= \frac{1}{n^2\lambda}n^2 \cdot \frac{n^2-1}{\frac{8n}{\lambda}-(n^2+1)} \\
&= \frac{n^2-1}{2n-\frac{\lambda}{4}(n^2-1)}
\end{aligned}
$$

8.4 Queue Sizes

To get a feel for the difference in queue size over nodes of the array, we will compare the expected queue size at the center of the array (where it's highest, by theorem 5) with the expected queue size at the corner of the array (where it is lowest). We do this first for the case where λ is half of the maximum arrival rate, namely $\lambda = \frac{2}{n}$.

$$
\begin{aligned}
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},R}} &= \frac{1}{2}-\frac{2}{n^2} \\
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},L}} &= \frac{1}{2} \\
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},D}} &= \frac{1}{2}-\frac{2}{n^2} \\
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},U}} &= \frac{1}{2}
\end{aligned}
$$

$$
\begin{aligned}
E[N_{P_{\frac{n}{2},\frac{n}{2},R}}] &= \frac{n^2-4}{n^2+4} \xrightarrow[n\to\infty]{} 1 \\
E[N_{P_{\frac{n}{2},\frac{n}{2},L}}] &= 1 \\
E[N_{P_{\frac{n}{2},\frac{n}{2},D}}] &= \frac{n^2-4}{n^2+4} \xrightarrow[n\to\infty]{} 1 \\
E[N_{P_{\frac{n}{2},\frac{n}{2},U}}] &= 1
\end{aligned}
$$

$$
\begin{aligned}
\hat{\lambda}_{P_{0,0,R}} &= \frac{2(n-1)}{n^2} \\
\hat{\lambda}_{P_{0,0,L}} &= 0 \\
\hat{\lambda}_{P_{0,0,D}} &= \frac{2(n-1)}{n^2} \\
\hat{\lambda}_{P_{0,0,U}} &= 0
\end{aligned}
$$

$$
\begin{aligned}
E[N_{P_{0,0,R}}] &= \frac{2(n-1)}{n^2-2(n-1)} \underset{n\to\infty}{\to} 0 \\
E[N_{P_{0,0,L}}] &= 0 \\
E[N_{P_{0,0,D}}] &= \frac{2(n-1)}{n^2-2(n-1)} \underset{n\to\infty}{\to} 0 \\
E[N_{P_{0,0,U}}] &= 0
\end{aligned}
$$

So for $\lambda = \frac{2}{n}$, and for large n, the highest expected queue size is 4 and the lowest queue size is 0. Now let's look at what happens when the arrival rate is almost at the maximum, $\lambda = .99 \cdot \frac{4}{n}$.

$$
\begin{aligned}
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},R}} &= .99(1-\frac{4}{n^2}) \\
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},L}} &= .99 \\
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},D}} &= .99(1-\frac{4}{n^2}) \\
\hat{\lambda}_{P_{\frac{n}{2},\frac{n}{2},U}} &= .99
\end{aligned}
$$

$$
\begin{aligned}
E[N_{P_{\frac{n}{2},\frac{n}{2},R}}] &= \frac{.99n^2-3.96}{.01n^2+3.96} \underset{n\to\infty}{\to} 99 \\
E[N_{P_{\frac{n}{2},\frac{n}{2},L}}] &= 99 \\
E[N_{P_{\frac{n}{2},\frac{n}{2},D}}] &= \frac{.99n^2-3.96}{.01n^2+3.96} \underset{n\to\infty}{\to} 99 \\
E[N_{P_{\frac{n}{2},\frac{n}{2},U}}] &= 99
\end{aligned}
$$

$$
\begin{aligned}
\hat{\lambda}_{P_{0,0,R}} &= 3.96 \cdot \frac{n-1}{n^2} \\
\hat{\lambda}_{P_{0,0,L}} &= 0 \\
\hat{\lambda}_{P_{0,0,D}} &= 3.96 \cdot \frac{n-1}{n^2} \\
\hat{\lambda}_{P_{0,0,U}} &= 0
\end{aligned}
$$

$$
E[N_{P_{0,0,R}}] = \frac{3.96(n-2)}{n^2-3.96(n-2)} \underset{n\to\infty}{\to} 0
$$

$$
\begin{aligned}
E[N_{P_{0,0,L}}] &= 0 \\
E[N_{P_{0,0,D}}] &= \frac{3.96(n-2)}{n^2 - 3.96(n-2)} \xrightarrow[n\to\infty]{} 0 \\
E[N_{P_{0,0,U}}] &= 0
\end{aligned}
$$

9 Conclusion

This paper combines ideas from the areas of communication networks, queueing theory, and combinatorics to analyze the queue buildup at the nodes of an $n \times n$ array and an $n\times$n torus during greedy routing.

The three main contributions of the paper are:

- A way to formulate the problem of greedy routing on an array or torus as a Jackson Queueing Network model.
- A very simple method for computing the probability distribution on the queue size and delay in the Jackson Queueing Network with greedy routing.
- A theorem showing that the expected queue size is greater for nodes closest to the center of the array.

10 Future Extensions

There are numerous possible extensions to the work in this paper.

The techniques of this paper can be used for analyzing queue size on different commonly used networks, such as the shuffle-exchange network. Alternatively, the same networks can be analyzed using dfferent oblivious routing algorithms.

Another idea is to use the same Jackson Queueing Network Model, but examine only the queue buildup of packets whose destination is, say, the center node. Note that it is easy to look at the queues formed by just one class. The usefulness of such a study is in avoiding bottlenecks when writing routing algorithms.

Lastly, in this paper we have only dealt with queue size in steady state. The rate of queue build up when the network is overloaded or the clearing of packets after an overload can be computed by casting the same balance equations used to derive Theorem 1 as differential equations and solving them. For a start, see [Buzacott,Shanthikumar,93].

11 Acknowledgements

We thank J. George Shanthikumar, Sheldon Ross, and Abhiram Ranade for their help and suggestions in developing these results.

A Derivation of Theorem 4

In this appendix we give a rigorous derivation of Theorem 4. For brevity we indicate the row of processor P_{rcS} by P_r, the column by P_c, and the side which it is on (Right, Left, Up, or Down) by P_S. To begin the derivation, we define the probability of a packet moving from one petal to another.

$$p^{(d)}_{j_{r,c,s}\, i_{r,c,s}} = \begin{cases} 1 & \begin{cases} \text{if } i_s = R \text{ and } d_c > i_c \text{ and } j_s = R \text{ and } j_{r,c-1} = i_{r,c} \\ \text{if } i_s = U \text{ and } d_c = i_c \text{ and } d_r < i_r \text{ and } \begin{cases} j_s = R \text{ and } j_{r,c-1} = i_{r,c}, \text{ or} \\ j_s = U \text{ and } j_{r+1,c} = i_{r,c}, \text{ or} \\ j_s = L \text{ and } j_{r,c+1} = i_{r,c} \end{cases} \\ \text{if } i_s = L \text{ and } d_c < i_c \text{ and } j_s = L \text{ and } j_{r,c+1} = i_{r,c} \\ \text{if } i_s = D \text{ and } d_c = i_c \text{ and } d_r > i_r \text{ and } \begin{cases} j_s = R \text{ and } j_{r,c-1} = i_{r,c}, \text{ or} \\ j_s = L \text{ and } j_{r,c+1} = i_{r,c}, \text{ or} \\ j_s = D \text{ and } j_{r-1,c} = i_{r,c} \end{cases} \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Of course, $p^{(d)}_{ji}$ is 0 if $j_{r,c,s}$ is outside the array, that is, if $j_r < 0$, $j_r \geq n$, $j_c < 0$, or $j_c \geq n$. We ignore the probability of moving to the center processor since it has no effect on queue sizes.

Applying the above definitions, the general system of linear equations specified in Equation 5 simplifies to

$$\begin{aligned}
\hat\lambda^{(d)}_{P_{r,c,R}} &= r^{(d)}_{P_{r,c,R}} + p^{(d)}_{P_{r,c-1,R}P_{r,c,R}} \hat\lambda^{(d)}_{P_{r,c-1,R}} \\
\hat\lambda^{(d)}_{P_{r,c,U}} &= r^{(d)}_{P_{r,c,U}} + p^{(d)}_{P_{r,c-1,R}P_{r,c,U}} \hat\lambda^{(d)}_{P_{r,c-1,R}} + p^{(d)}_{P_{r+1,c,U}P_{r,c,U}} \hat\lambda^{(d)}_{P_{r+1,c,U}} + p^{(d)}_{P_{r,c+1,L}P_{r,c,U}} \hat\lambda^{(d)}_{P_{r,c+1,L}} \\
\hat\lambda^{(d)}_{P_{r,c,L}} &= r^{(d)}_{P_{r,c,L}} + p^{(d)}_{P_{r,c+1,L}P_{r,c,L}} \hat\lambda^{(d)}_{P_{r,c+1,L}} \\
\hat\lambda^{(d)}_{P_{r,c,D}} &= r^{(d)}_{P_{r,c,D}} + p^{(d)}_{P_{r,c-1,R}P_{r,c,D}} \hat\lambda^{(d)}_{P_{r,c-1,R}} + p^{(d)}_{P_{r,c+1,L}P_{r,c,D}} \hat\lambda^{(d)}_{P_{r,c+1,L}} + p^{(d)}_{P_{r-1,c,D}P_{r,c,D}} \hat\lambda^{(d)}_{P_{r-1,c,D}}
\end{aligned}$$

Again nodes "beyond" the edge and center nodes do not contribute and are not counted. Ignoring the equations at the edge nodes, we have only $4n^2$ equations to solve, each with an average of 2 unknowns.

The arrival rates at the petals from the outside is

$$r^{(d_{r,c})}_{P_{r,c,S}} = \begin{cases} \frac{\lambda}{N} & \begin{cases} \text{if } P_S = R \text{ and } d_c > P_c, \text{ or} \\ \text{if } P_S = U \text{ and } d_c = P_c \text{ and } d_r < P_r, \text{ or} \\ \text{if } P_S = L \text{ and } d_c < P_c, \text{ or} \\ \text{if } P_S = D \text{ and } d_c = P_c \text{ and } d_r > P_r \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Let's begin by writing the equations for Right chains beginning in the left flowers of the array.

$$\begin{aligned}
\hat{\lambda}^{(d)}_{P_{r,0,R}} &= r^{(d)}_{P_{r,0,R}} \\
\hat{\lambda}^{(d)}_{P_{r,c,R}} &= r^{(d)}_{P_{r,c,R}} + p^{(d)}_{P_{r,c-1,R}P_{r,c,R}} \hat{\lambda}^{(d)}_{P_{r,c-1,R}}
\end{aligned}$$

Since each $\hat{\lambda}^{(d)}_{ir,c,R}$ only depends on the $\hat{\lambda}^{(d)}_{P_{r,c,R}}$ values to the left of it in the same row, we can easily compute their values. To begin, we specialize the equations for the destination

$$\begin{aligned}
\hat{\lambda}^{(d)}_{P_{r,0,R}} &= \begin{cases} \frac{\lambda}{N} & \text{if } d_c > 0 \\ 0 & \text{otherwise} \end{cases} \\
\hat{\lambda}^{(d)}_{P_{r,c,R}} &= \left\{ \begin{array}{ll} \frac{\lambda}{N} & \text{if } d_c > P_c \\ 0 & \text{otherwise} \end{array} \right\} + p^{(d)}_{P_{r,c-1,R}P_{r,c,R}} \hat{\lambda}^{(d)}_{P_{r,c-1,R}}
\end{aligned}$$

Expanding this for the next column

$$\begin{aligned}
\hat{\lambda}^{(d)}_{P_{r,1,R}} &= \left\{ \begin{array}{ll} \frac{\lambda}{N} & \text{if } d_c > 1 \\ 0 & \text{otherwise} \end{array} \right\} + p^{(d)}_{P_{r,0,R}P_{r,1,R}} \hat{\lambda}^{(d)}_{P_{r,0,R}} \\
&= \left\{ \begin{array}{ll} \frac{\lambda}{N} & \text{if } d_c > 1 \\ 0 & \text{otherwise} \end{array} \right\} + p^{(d)}_{P_{r,0,R}P_{r,1,R}} \begin{cases} \frac{\lambda}{N} & \text{if } d_c > 0 \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} 2\frac{\lambda}{N} & \text{if } d_c > 1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

And in general

$$\hat{\lambda}^{(d)}_{P_{r,c,R}} = \begin{cases} (P_c + 1)\frac{\lambda}{N} & \text{if } d_c > P_c \\ 0 & \text{otherwise} \end{cases}$$

Similarly

$$\begin{aligned}
\hat{\lambda}^{(d)}_{P_{r,c,L}} &= \begin{cases} (n - P_c)\frac{\lambda}{N} & \text{if } d_c < P_c \\ 0 & \text{otherwise} \end{cases} \\
\hat{\lambda}^{(d)}_{P_{r,c,U}} &= \begin{cases} n(n - P_r)\frac{\lambda}{N} & \text{if } d_c = P_c \text{ and } d_r < P_r \\ 0 & \text{otherwise} \end{cases} \\
\hat{\lambda}^{(d)}_{P_{r,c,D}} &= \begin{cases} n(P_r + 1)\frac{\lambda}{N} & \text{if } d_c = P_c \text{ and } d_r > P_r \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

Next we need to sum over all the classes. The results are the equations given in Theorem 4

$$\begin{aligned}
\hat{\lambda}_{P_{r,c,R}} &= \frac{\lambda}{n}(P_c + 1)(n - P_c - 1) \\
\hat{\lambda}_{P_{r,c,L}} &= \frac{\lambda}{n}(n - P_c)P_c \\
\hat{\lambda}_{P_{r,c,U}} &= \frac{\lambda}{n}(n - P_r)P_r \\
\hat{\lambda}_{P_{r,c,D}} &= \frac{\lambda}{n}(P_r + 1)(n - P_r - 1)
\end{aligned}$$

B Alternative Queueing Network Model Design for Formulating the Routing Problem on Array

There are many other ways in which greedy routing on an $n \times n$ array can be formulated in terms of the queueing network model. We discuss one possibility here. This formulation uses only 5 classes and one server per node, leading to $5n^2$ simultaneous equations for the $\hat{\lambda}_i^{(l)}$'s rather than the $5n^4$ simultaneous, but independent, equations we currently solve. In this model, the class of a packet represents what direction the packet arrived from (left, right, up, or down, or outside). The probabilities $p_{ji}^{(d)}$ are now reals between 0 and 1 relating to the average distribution of packets from that node. Since this formulation induces loops, some variables are mutually dependent.

C Simultaneous Equations for the 3×3 Array Problem

Although solving an arbitrary system of 4×81 linear equations in 4×81 unknowns specified by Equation 5 is daunting, the equations derived from our model have a straight forward solution. Since no packet's path ever returns to the same node, there are no dependencies loops among the variables in the system of linear equations.

Recall that the general solution given in Theorem 4 allows one to find the $\hat{\lambda}$'s *without* writing out and solving the system of linear equations. However these serve as a check that the results from the general solution are, indeed, correct.

We use coding in the symbols to reduce the number of equations which need to be written out. The symbol $\hat{\lambda}_{00R}^{([012]0)}$ represents the destination classes 00, 10, and 20, that is $\hat{\lambda}_{00R}^{(00)}$, $\hat{\lambda}_{00R}^{(10)}$, and $\hat{\lambda}_{00R}^{(20)}$. For instance the first two statements represent nine equations.

$$\begin{aligned}
\hat{\lambda}_{00R}^{([012]0)} &= 0 \\
\hat{\lambda}_{00R}^{([012][12])} &= \frac{\lambda}{9}
\end{aligned}$$

$$\begin{aligned}
\hat{\lambda}_{00D}^{\neg([12]0)} &= 0 \\
\hat{\lambda}_{00D}^{([12]0)} &= \frac{\lambda}{9} + \hat{\lambda}_{01L}^{([12]0)}
\end{aligned}$$

$$\begin{aligned}
\hat{\lambda}_{01R}^{\neg([012]2)} &= 0 \\
\hat{\lambda}_{01R}^{([012]2)} &= \frac{\lambda}{9} + \hat{\lambda}_{00R}^{([012]2)}
\end{aligned}$$

$$\begin{aligned}
\hat{\lambda}_{01L}^{\neg([012]0)} &= 0 \\
\hat{\lambda}_{01L}^{([012]0)} &= \frac{\lambda}{9} + \hat{\lambda}_{02L}^{([012]0)}
\end{aligned}$$

$$\begin{aligned}
\hat{\lambda}_{01D}^{\neg([12]1)} &= 0 \\
\hat{\lambda}_{01D}^{([12]1)} &= \frac{\lambda}{9} + \hat{\lambda}_{00R}^{([12]1)} + \hat{\lambda}_{02L}^{([12]1)}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{([012]2)}_{02L} &= 0\\
\hat\lambda^{([012][01])}_{02L} &= \tfrac{\lambda}{9}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg([12]2)}_{02D} &= 0\\
\hat\lambda^{([12]2)}_{02D} &= \tfrac{\lambda}{9} + \hat\lambda^{([12]2)}_{01R}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{([012]0)}_{10R} &= 0\\
\hat\lambda^{([012][12])}_{10R} &= \tfrac{\lambda}{9}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg(00)}_{10U} &= 0\\
\hat\lambda^{(00)}_{10U} &= \tfrac{\lambda}{9} + \hat\lambda^{(00)}_{11L} + \hat\lambda^{(00)}_{20U}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg(20)}_{10D} &= 0\\
\hat\lambda^{(20)}_{10D} &= \tfrac{\lambda}{9} + \hat\lambda^{(20)}_{11L} + \hat\lambda^{(20)}_{00D}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg([012]2)}_{11R} &= 0\\
\hat\lambda^{([012]2)}_{11R} &= \tfrac{\lambda}{9} + \hat\lambda^{([012]2)}_{10R}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg(01)}_{11U} &= 0\\
\hat\lambda^{(01)}_{11U} &= \tfrac{\lambda}{9} + \hat\lambda^{(01)}_{10R} + \hat\lambda^{(01)}_{21U} + \hat\lambda^{(01)}_{12L}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg([012]0)}_{11L} &= 0\\
\hat\lambda^{([012]0)}_{11L} &= \tfrac{\lambda}{9} + \hat\lambda^{([012]0)}_{12L}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg(21)}_{11D} &= 0\\
\hat\lambda^{(21)}_{11D} &= \tfrac{\lambda}{9} + \hat\lambda^{(21)}_{10R} + \hat\lambda^{(21)}_{01D} + \hat\lambda^{(21)}_{12L}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg(02)}_{12U} &= 0\\
\hat\lambda^{(02)}_{12U} &= \tfrac{\lambda}{9} + \hat\lambda^{(02)}_{11R} + \hat\lambda^{(02)}_{22U}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{([012]2)}_{12L} &= 0\\
\hat\lambda^{([012][01])}_{12L} &= \tfrac{\lambda}{9}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg(22)}_{12D} &= 0\\
\hat\lambda^{(22)}_{12D} &= \tfrac{\lambda}{9} + \hat\lambda^{(22)}_{11R} + \hat\lambda^{(22)}_{02D}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{([012]0)}_{20R} &= 0\\
\hat\lambda^{([012][12])}_{20R} &= \tfrac{\lambda}{9}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda^{\neg([01]0)}_{20U} &= 0\\
\hat\lambda^{([01]0)}_{20U} &= \tfrac{\lambda}{9} + \hat\lambda^{([01]0)}_{21L}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda_{21R}^{\neg([012]2)} &= 0\\
\hat\lambda_{21R}^{([012]2)} &= \tfrac{\lambda}{9} + \hat\lambda_{20R}^{([012]2)}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda_{21U}^{\neg([01]1)} &= 0\\
\hat\lambda_{21U}^{([01]1)} &= \tfrac{\lambda}{9} + \hat\lambda_{20R}^{([01]1)} + \hat\lambda_{22L}^{([01]1)}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda_{21L}^{\neg([012]0)} &= 0\\
\hat\lambda_{21L}^{([012]0)} &= \tfrac{\lambda}{9} + \hat\lambda_{22L}^{([012]0)}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda_{22U}^{\neg([01]2)} &= 0\\
\hat\lambda_{22U}^{([01]2)} &= \tfrac{\lambda}{9} + \hat\lambda_{21R}^{([01]2)}
\end{aligned}$$

$$\begin{aligned}
\hat\lambda_{22L}^{([012]2)} &= 0\\
\hat\lambda_{22L}^{([012][01])} &= \tfrac{\lambda}{9}
\end{aligned}$$

Again in coded form, the solution to all the variables is

$$\begin{array}{lll lll lll}
\hat\lambda_{00R}^{([012][12])} &=& \frac{\lambda}{9} & \hat\lambda_{10R}^{([012][12])} &=& \frac{\lambda}{9} & \hat\lambda_{20R}^{([012][12])} &=& \frac{\lambda}{9}\\
 & & & \hat\lambda_{10U}^{(00)} &=& 6\frac{\lambda}{9} & \hat\lambda_{20U}^{([01]0)} &=& 3\frac{\lambda}{9}\\
\hat\lambda_{00D}^{([12]0)} &=& 3\frac{\lambda}{9} & \hat\lambda_{10D}^{(20)} &=& 6\frac{\lambda}{9} & & & \\
\hat\lambda_{01R}^{([012]2)} &=& 2\frac{\lambda}{9} & \hat\lambda_{11R}^{([012]2)} &=& 2\frac{\lambda}{9} & \hat\lambda_{21R}^{([012]2)} &=& 2\frac{\lambda}{9}\\
 & & & \hat\lambda_{11U}^{(01)} &=& 6\frac{\lambda}{9} & \hat\lambda_{21U}^{([01]1)} &=& 3\frac{\lambda}{9}\\
\hat\lambda_{01L}^{([012]0)} &=& 2\frac{\lambda}{9} & \hat\lambda_{11L}^{([012]0)} &=& 2\frac{\lambda}{9} & \hat\lambda_{21L}^{([012]0)} &=& 2\frac{\lambda}{9}\\
\hat\lambda_{01D}^{([12]1)} &=& 3\frac{\lambda}{9} & \hat\lambda_{11D}^{(21)} &=& 6\frac{\lambda}{9} & & & \\
 & & & \hat\lambda_{12U}^{(02)} &=& 6\frac{\lambda}{9} & \hat\lambda_{22U}^{([01]2)} &=& 3\frac{\lambda}{9}\\
\hat\lambda_{02L}^{([012][01])} &=& \frac{\lambda}{9} & \hat\lambda_{12L}^{([012][01])} &=& \frac{\lambda}{9} & \hat\lambda_{22L}^{([012][01])} &=& \frac{\lambda}{9}\\
\hat\lambda_{02D}^{([12]2)} &=& 3\frac{\lambda}{9} & \hat\lambda_{12D}^{(22)} &=& 6\frac{\lambda}{9} & & &
\end{array}$$

For the 3×3 array, the sum of $\hat\lambda_{P_{r,c,S}}^{(d)}$'s over all destinations is the same: $\frac{2\lambda}{3}$. The expected values $E[N_{P_{r,c,S}}]$ are all $\frac{2\lambda}{3-2\lambda}$.

$$\begin{aligned}
E[N_{00}] &= \tfrac{4\lambda}{3-2\lambda}\\
E[N_{01}] &= \tfrac{6\lambda}{3-2\lambda}\\
E[N_{02}] &= \tfrac{4\lambda}{3-2\lambda}\\
E[N_{10}] &= \tfrac{6\lambda}{3-2\lambda}\\
E[N_{11}] &= \tfrac{8\lambda}{3-2\lambda}\\
E[N_{12}] &= \tfrac{6\lambda}{3-2\lambda}\\
E[N_{20}] &= \tfrac{4\lambda}{3-2\lambda}\\
E[N_{21}] &= \tfrac{6\lambda}{3-2\lambda}\\
E[N_{22}] &= \tfrac{4\lambda}{3-2\lambda}
\end{aligned}$$

Example: Say $\lambda = \frac{1}{2}$. The expected queue lengths are

1	1.5	1
1.5	2	1.5
1	1.5	1

Example: Say $\lambda = \frac{1}{4}$. The expected queue lengths are

.4	.6	.4
.6	.8	.6
.4	.6	.4

Note that the expected queue size grows significantly toward the center of the array. Theorem 5, shows that this is true for arrays of all sizes.

D Average Delay in an Array by Summing Delays at Nodes

Substituting $E[N_i]$ for $\overline{N}$ and $\hat{\lambda}_i$ for λ, we get

$$\begin{aligned} \overline{T_i} &= \frac{\hat{\lambda}_i}{\mu_i - \hat{\lambda}_i} \frac{1}{\hat{\lambda}_i} \\ &= \frac{1}{\mu_i - \hat{\lambda}_i} \end{aligned}$$

The delay along any path is the sum of the delay at each petal through which the packet passes from the source, S, to the destination, D.

$$\overline{T_{path}} = \sum_{i=S}^{D} \frac{1}{\mu_i - \hat{\lambda}_i}$$

Since we set all service rates μ_i to 1, we have

$$\overline{T_{path}} = \sum_{i=S}^{D} \frac{1}{1 - \hat{\lambda}_i}$$

We can compute the average delay, A, by summing the delay of paths from all sources to all destinations and dividing by the number of paths.

$$A = \frac{1}{n^4} \sum_{S=0}^{n^2} \sum_{D=0}^{n^2} \sum_{i=S}^{D} \frac{1}{1 - \hat{\lambda}_i}$$

Since we are summing, we can seperate the computations for rows and columns.

$$A = \frac{1}{n^4} \sum_{S=0}^{n^2} \sum_{D=0}^{n^2} (\text{total row delay}_{S,D}) + \frac{1}{n^4} \sum_{S=0}^{n^2} \sum_{D=0}^{n^2} (\text{total column delay}_{S,D})$$

From Theorem 4 we see that the row and column formulas are equivalent. Therefore we only need make one of the calculations and double it. We arbitrarily choose rows.

$$A = \frac{2}{n^4} \sum_{S=0}^{n^2} \sum_{D=0}^{n^2} (\text{total row delay}_{S,D})$$

Notice that the columns no longer matter in the above formula. Therefore we only need to do the calculation for one row and multiply by the number of columns, n.

$$\begin{aligned} A &= \frac{2}{n^4}\; n \sum_{S_{row}=0}^{n-1} n \sum_{D_{row}=0}^{n-1} (\text{total row delay}_{S,D}) \\ &= \frac{2}{n^2} \sum_{S_{row}=0}^{n-1} \sum_{D_{row}=0}^{n-1} (\text{total row delay}_{S,D}) \end{aligned}$$

Plugging in the actual formulas from Theorem 4 we get

$$\begin{aligned} A &= \frac{2}{n^2} \sum_{S_{row}=0}^{n-1} \sum_{D_{row}=0}^{n-1} \sum_{i=S}^{D} \begin{cases} \frac{1}{1-\frac{\lambda}{n}i(n-i)} & \text{if } D_{row} < S_{row} \\ 0 & \text{if } D_{row} = S_{row} \\ \frac{1}{1-\frac{\lambda}{n}(i+1)(n-i-1)} & \text{if } D_{row} > S_{row} \end{cases} \\ &= \frac{2}{n^2} \left(\sum_{S_{row}=1}^{n-1} \sum_{D_{row}=0}^{S_{row}-1} \sum_{i=D_{row}+1}^{S_{row}} \frac{1}{1-\frac{\lambda}{n}i(n-i)} \right. \\ &\quad \left. + \sum_{S_{row}=0}^{n-2} \sum_{D_{row}=S_{row}+1}^{n-1} \sum_{i=S_{row}}^{D_{row}-1} \frac{1}{1-\frac{\lambda}{n}(i+1)(n-i-1)} \right) \end{aligned}$$

Note that the expected delay at a petal node doesn't depend on the source or the destination. Therefore we can simply multiply the expected delay at each petal node by the number of paths through it and sum them all.

$$A = \frac{2}{n^2} \left(\sum_{i=0}^{n-1} \frac{i(n-i)}{1-\frac{\lambda}{n}i(n-i)} + \sum_{i=0}^{n-1} \frac{(i+1)(n-i-1)}{1-\frac{\lambda}{n}(i+1)(n-i-1)} \right)$$

Behold! The same expression appears in the numerator and the denominator. Is this reasonable? Remember from Observation 3 that $\hat{\lambda}_i$ is just the number of paths through a petal times a weight. Thus it is reasonable to have the same expression in the numerator and denominator.

We have not been able to reduce the expression further. We note that both forms are similar to

$$\int \frac{x\,dx}{a+bx} = \frac{x}{b} - \frac{a}{b^2}\log(a+bx)$$

except for the missing dx, which is:

$$\frac{d\,i(n-i)}{di} = n - 2i$$

$$\frac{d\,(i+1)(n-i-1)}{di} = n - 2i - 2$$

References

[Buzacott,Shanthikumar,93] John A. Buzacott and J. George Shanthikumar. *Stochastic Models of Manufacturing Systems*. Prentice Hall, NJ, 1993.

[Leighton,92] F. Thomas Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*. Morgan Kaufmann Publishers, CA, 1992.

[Ross,83] Sheldon M. Ross. *Stochastic Processes*. John Wiley and Sons, NY,1983.

[Stamoulis,Tsitsiklis,91] George D. Stamoulis and John N. Tsitsiklis. *The Efficiency of Greedy Routing in Hypercubes and Butterflies*. Journal of the ACM, 1991.