Queueing Theory Analysis of Greedy Routing on Square Arrays

Mor Harchol* Paul E. Black

May 20, 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 of processors. We assume packets continuously arrive at each node of the array 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 Poisson distributed with mean 1.

To analyze the queue size in steady-state, we formulate the problem into an equivalent Jackson queueing network model. It turns out that determining the probability distribution on the queue size at each node is then just a matter of solving $O(n^4)$ simultaneous linear equations which determine the total arrival rate at each node and then plugging these arrival rates into a short formula for the probability distribution given by the queueing theory. However, we even eliminate the need to solve these simultaneous equations by deriving a very simple formula for the total arrival rates in the case of greedy routing.

Lastly, we use this simple formula to prove that the expected queue size at a node of the $n \times n$ array increases as the Euclidean distance of the node from the center of the array decreases.

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

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.

The most common type of oblivious routing on arrays 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 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 may be used by only one packet at a time, packets naturally get delayed at the processor nodes of the array. It is therefore important in building arrays of processors that we create appropriate sized buffers at each node of the array 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 same 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.

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 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, 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 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 $n \times n$ 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**.

• OUTLINE OF CHAPTERS

Section 2 contains a detailed problem definition.

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 arrival rates. Section 5 next proves that the total arrival rate at a node increases as we look at nodes closer to the center of the array, and furthermore, that total arrival rate is directly related to expected queue size. Section 5 concludes with a theorem stating 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 summarize our results, and in Section 8 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 Jakcson 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 Poisson 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 $\mathbf{r_i^{(l)}}$. One of the nice features of this model is that it 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 $\mu_{\mathbf{i}}$.

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

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

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

$$\begin{aligned} \textit{where } \rho_i &= \frac{\hat{\lambda}_i}{\mu_i} \\ \hat{\lambda}_i &= \sum_{l=1}^{r} \hat{\lambda}_i^{(l)} \end{aligned}$$

$$and\ \hat{\lambda}_i^{(l)} \quad = \quad r_i^{(l)} + \sum_{j=1}^{m} \sum_{k=1}^{r} p_{ji}^{(k)(l)} \hat{\lambda}_j^{(k)} \tag{2}$$

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, ..., 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, $\mathbf{p_i(n_i)}$, the probability that there are n_i packets at server i, is the sum of Expression 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 $\mathbf{N_i}$ be a random variable representing the number of packets at server i. Then, since N_i has a distribution which is geometric times a factor ρ_i, we have:

$$E[N_i] = \frac{\rho_i}{1 - \rho_i}$$

$$var(N_i) = \frac{\rho_i}{(1 - \rho_i)^2}$$

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 into the central processor, edges from the petals to other nodes, and paths into and out of the system 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 Poisson 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:

$$\begin{aligned}
\hat{\lambda}_i^{(d)} &= r_i^{(d)} + \sum_{j=1}^{m} p_{ji}^{(d)} \hat{\lambda}_j^{(d)} \qquad (3) \\
\hat{\lambda}_i &= \sum_{d=1}^{r} \hat{\lambda}_i^{(d)} \\
\rho_i &= \frac{\hat{\lambda}_i}{\mu_i} = \hat{\lambda}_i \\
p_i(n_i) &= (1-\rho_i)\rho_i^{n_i}
\end{aligned}$$

Thus to calculate $\mathbf{p_i(n_i)}$, the probability of having n_i packets at server i, we need to 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.

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 3 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, a far easier way to obtain the $\hat{\lambda}_i$'s directly which doesn't require solving simultaneous equations for the $\hat{\lambda}_i^{(d)}$'s.

5 A Simple Method For Determining Queue Size

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 *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. For any oblivious routing scheme the number of paths through a node can be easily computed.*

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 informal proof of these equations. A precise 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. If that is not the case, we can still determine the probability distribution on the queue sizes, however now we must solve the $5n^4$ simultaneous equations.

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), $E[N_{(r,c)}]$, decreases with the flower's Euclidean distance from the center of the array.*

Proof: The following is a proof sketch. The center of the array is at $(\frac{n}{2}, \frac{n}{2})$. Let $x = c - \frac{n}{2}$ and $y = r - \frac{n}{2}$, the x and y offsets of the node from the center of the array. We will show that as $\sqrt{x^2 + y^2}$, the distance from the center of the array, increases, the expected queue size decreases.

From Theorem 4 we see that $\hat{\lambda}_{P_{r,c,R}} = \hat{\lambda}_{P_{r,c,L}} \approx c(n - c)$, and $\hat{\lambda}_{P_{r,c,U}} = \hat{\lambda}_{P_{r,c,D}} \approx r(n - r)$. Rewriting in terms of x and y we have $\hat{\lambda}_{P_{r,c,R}} = \hat{\lambda}_{P_{r,c,L}} \approx (\frac{n}{2} + x)(\frac{n}{2} - x) = \frac{n^2}{4} - x^2$, and $\hat{\lambda}_{P_{r,c,U}} = \hat{\lambda}_{P_{r,c,D}} \approx (\frac{n}{2} + y)(\frac{n}{2} - y) = \frac{n^2}{4} - y^2$.

The expected queue size at each petal is proportional to the arrival rate, $\hat{\lambda}_{P_{r,c,S}}$, at the petal. The expected queue size at the node is the sum of the expected queue sizes at each petal. Therefore the expected queue size at the node is directly proportional to the arrival rate at the petals.

$$\begin{aligned} \text{E[queue size at node } r, c] \quad & \alpha \quad \frac{n^2}{2} - (x^2 + y^2) \\ & \alpha \quad -(x^2 + y^2) \end{aligned}$$

Therefore the greater the value of $x^2 + y^2$, the smaller the expected queue at the node. Since the Euclidean distance from the center of the array is $\sqrt{x^2 + y^2}$, the queue size at a node is inversely proportional to it's Euclidean distance. ■

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.

Before looking at the example in the next section, we mention that 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. One possibility is discussed in Appendix B. The formulation in Appendix B 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 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.

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 simply use Corollary 2 as suggested by (3) (see Section 4), however rather than solving simultaneous equations, we derive the $\hat{\lambda}_i$'s directly by the formulae in Theorem 3.

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 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 during greedy routing.

The three main contributions of the paper are:

- A way to formulate the problem of greedy routing on an array as a Jackson Queueing Network model.
- A very simple method for computing the probability distribution on the queue size 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.

8 Future Extensions

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

One area of future work is to redo all the analysis in this paper for the case of a torus, rather than an $n \times n$ array, since the torus has no distinguished nodes. We conjecture this will result in all nodes having the same size queues.

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.

Thirdly, the techniques of this paper may be applied to studying other routing algorithms.

Observe also that any conclusion reached about queue size can be used to derive conclusions about expected packet delay, since packet delay and queue size are related via Little's Formula.

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].

9 Acknowledgements

We thank J. George Shanthikumar and Sheldon Ross 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 3 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

(We have done all the work for this appendix, but have left the write-up out due to time constraints. Please see us if you have questions).

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 3 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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.