Queueing Analysis of Oblivious Packet-Routing Networks

Mor Harchol-Balter* Paul E. Black†

March 23, 1995

Abstract

We consider the problem of determining the probability distribution on the queue sizes in a general oblivious packet-routing network. We assume packets continuously arrive at each node of the network according to a Poisson process with rate λ. We also assume that an edge may be traversed by only one packet at a time, and the time to traverse an edge is an exponentially distributed random variable with mean 1.

We show that the queueing-theoretic solution to the problem requires solving a large system of simultaneous equations.

We present a simple combinatorial formula which represents the solution to the system of queueing equations. This combinatorial formula is especially simple and insightful in the case of greedy routing to random destinations.

We use the formula to obtain results including: the probability distribution on the queue sizes, the expected queue sizes, and the expected packet delay (all as a function of λ) in the case of an array network and a torus network with greedy randomized routing.

*Computer Science Division, UC Berkeley, CA 94720. Supported by National Physical Science Consortium (NPSC) Fellowship and by NSF grant number CCR-9201092

†Computer Science Division, UC Berkeley, CA 94720.

1 Introduction

1.1 Setup

Definition 1 *An* **oblivious routing scheme** $\mathcal{R}$ *specifies for any (source, destination) pair exactly one acyclic path from the source to the destination.*

Let $\mathcal{R}$ be any oblivious packet routing scheme.

Let $\mathcal{N}$ be any network with the following properties:

- $\mathcal{N}$ consists of m processors with directed wires between some pairs of processors.
- New packets arrive at processor i according to a Poisson Process with rate r_i. That is, the time between arrivals is exponential with rate r_i.
- A packet contains a destination field and a data field.
- Packets are routed from their source (origination processor in $\mathcal{N}$) to their destination processor according to $\mathcal{R}$.
- The time it takes for a packet to move through an edge has distribution $\mathcal{D}$ and mean 1.
- Only one packet may be on a particular directed edge at a time. If two packets require the same edge, contention is resolved via First-Come-First-Serve (FCFS).

We consider the problem of determining the queue buildup on the edges of $\mathcal{N}$.

1.2 Previous History

One approach to the above problem is to convert $\mathcal{N}$ with routing scheme $\mathcal{R}$ into a Jackson Queueing Network $\mathcal{Q}$. The nodes of $\mathcal{Q}$ correspond to the edges of $\mathcal{N}$, and we can apply queueing formulae to $\mathcal{Q}$ to obtain the exact probability distribution on the queue sizes at the nodes of $\mathcal{Q}$. The queueing theory approach gives us the results we want, however it requires that $\mathcal{D}$ be exponential, whereas in most real-world networks $\mathcal{D}$ is constant.

Alternatively, one can assume $\mathcal{D}$ is constant, equal to 1. Now, however, the problem of determining the queue sizes is much more difficult. Leighton [12] has addressed the problem in the case where $\mathcal{N}$ is a $\sqrt{N} \times \sqrt{N}$ array or torus, each packet has a random destination, and $r_i = \lambda$, $\forall i$ His analysis requires complex probabilistic reasoning, yet his results aren't nearly as detailed in nature as results typically obtained using queueing theory[1].

[1] For both array networks and torus networks, Leighton proves that if the arrival rate of packets is at most 99% of network capacity, then in any window of T steps, the maximum observed queue size is $O\left(1 + \frac{\log T}{\log N}\right)$ with probability $1 - O(\frac{1}{TN})$.

Since analyses which use queueing theory obtain so much more detailed results, there has been a lot of work done on converting the case where $\mathcal{D}$ is constant to the case where $\mathcal{D}$ is exponential. One example is Erlang's "Method of Stages", described in [9], [1], which involves decomposing a constant-distributed service time into a collection of exponentially distributed service times. Stamoulis and Tsitsklis [18] use this method to analyze the routing problem on the hypercube in the case where $\mathcal{D}$ is constant. Mitra and Cieslak [14] use Kuehn's Method to approximate a general-distributed edge-traversal time by an exponentially-distributed edge-traversal time in order to obtain bounds on queue sizes for the Omega network. Several other commonly used methods for converting between the case where $\mathcal{D}$ is constant and the case where $\mathcal{D}$ is exponential are described in [9].

1.3 Synopsis of Paper

It is clear from the previous discussion that even solving the problem in Section 1.1 for the case where $\mathcal{D}$ is exponential is important, since solutions in that case imply solutions to the case where $\mathcal{D}$ is constant.

The analysis in this paper is only for the case where $\mathcal{D}$ is exponential. Therefore, queueing theory is applicable and we can convert $\mathcal{N}$ with routing scheme $\mathcal{R}$ into a Jackson Queueing Network $\mathcal{Q}$ and compute the probability distribution on the queue size at the nodes of $\mathcal{Q}$. **A major practical drawback to using queueing theory when m, the number of processors in $\mathcal{N}$, is large is that determining the queue size at the nodes of $\mathcal{Q}$ requires first setting up and solving as many as $O(m^4)$ simultaneous equations.** This system of equations determines the total arrival rate (rate of flow into a node from outside the network as well as from the node's neighbors) at each node of $\mathcal{Q}$.

Much research has been done specifically into how to solve the system of simultaneous equations that arise from queueing theory. Wallace [20] presents an extensive survey of methods used to solve the system of equations for the total arrival rate at each node. All the methods however still require a significant amount of work and some give only approximate solutions. Also, even if one is interested only in the queue size at a particular node, one must solve the entire system of equations.

In this paper we eliminate the need to set up and solve the system of equations associated with using queueing theory. We derive a simple combinatorial formula for the total arrival rate at each node. In the special case of randomized routing (as defined by [19]) where packets each have random destinations and $r_i = \lambda, \forall i$, our combinatorial formula states that:

$$(\text{Total arrival rate into a node of } \mathcal{Q}) = \frac{\lambda}{m} \cdot (\text{no. paths through the node consistent with } \mathcal{R})$$

where a path is consistent with $\mathcal{R}$ if it is a path specified by $\mathcal{R}$.

This formula seems so intuitive that we are surprised none of our references, including the following major queueing texts [17], [21], [9] [1], [2] and the foundational queueing papers [7], [8], [6] make this observation. We believe the reason that this combinatorial relation hasn't been noticed before is that queueing theory, although extremely popular for communication and scheduling problems (see for example [10] and [15]), hasn't been applied nearly as much to randomized routing on networks.

Besides its importance from a computational point of view, the above combinatorial formula is very important because of the insight it gives us into the queue lengths. The queue size at a node of $\mathcal{Q}$ increases with the total arrival rate at the node. Since the total arrival rate at a node is proportional to the number of paths consistent with $\mathcal{R}$ through the node, we see that the queue size is greatest for nodes which have a lot of $\mathcal{R}$-consistent paths through them.

Although our results apply to any network in which packets are routed via an oblivious routing scheme, as examples we look at the array network and the torus network and analyze them in the case with random destination and where $\mathcal{R}$ is greedy routing. We choose this setup specifically because it is the same scenario as analyzed by Leighton[2]. [12]

Since for an array the number of greedy paths through a node increases as the Euclidean distance of the node from the center of the array decreases, we see by our combinatorial formula that the queue sizes in the array increase as we look at nodes closer to the center of the array. In the case of a torus, the nodes are indistinguishable, so the number of greedy paths through each node is the same, and therefore so is the queue size. For both the array and torus, we use the combinatorial formula to compute the probability distribution on the queue sizes, the expected queue sizes, and the average packet delay, as a function of λ.

1.4 Outline

In Section 2 we give a brief tutorial on queueing theory. We will not use any queueing theory beyond what is contained in this section. In Section 3 we show how to convert any network $\mathcal{N}$, of the type described in Section 1.1 with exponential edge-traversal times, having any associated oblivious routing scheme into a Jackson Queueing Network, so that we can apply the formulae from queueing theory. Section 4 relates total arrival rate into a node to the number of $\mathcal{R}$-consistent paths through the node. Section 5 and Section 6 apply our result to arrays and tori respectively. For the omitted proofs, see [4] and [5].

2 Multiple-Job-Class Open Jackson Queueing Network Model

The Queueing Network Model we use [2], [6] 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 $\mathbf{r_i^{(l)}}$. A packet of class l at server i next moves to server j with probability $\mathbf{p_{ij}^{(l)}}$. (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 time at server i is exponentially distributed with rate $\mu_{\mathbf{i}}$.

[2] Although [11] and [16] also derive results on queue sizes for the array, they only analyze permutation routing.

We will use the notation $\mathbf{n_i}$ to denote the number of packets at server i.

Theorem 2 *[2] When the queueing network is in steady state, and if $\mu_i = 1$, $\forall i$, then the probability that there are n_i packets queued at server i , $p_i(n_i)$, is given by*

$$p_i(n_i) = (1 - \hat{\lambda}_i)\hat{\lambda}_i^{n_i} \tag{1}$$

$$\begin{aligned} \text{where } \hat{\lambda}_i &= \sum_{l=1}^{r} \hat{\lambda}_i^{(l)} \\ \hat{\lambda}_i^{(l)} &= r_i^{(l)} + \sum_{j=1}^{m} p_{ji}^{(l)} \hat{\lambda}_j^{(l)} \end{aligned} \tag{2}$$

Here $\hat{\lambda}_{\mathbf{i}}$ represents the total arrival rate of packets into server i from both outside the network and from neighboring servers, and $\hat{\lambda}_{\mathbf{i}}^{(\mathbf{l})}$ is the total arrival rate into server i of class l packets. Equations 2 are known as the balance equations, since they balance the rate at which packets enter and leave a server.

Observe that determining $p_i(n_i)$ requires solving $O(m \cdot r)$ simultaneous equations for the $\hat{\lambda}_i^{(l)}$'s.

Lastly let N_i be a random variable representing the number of packets at server i. Since by Theorem 2, N_i has a distribution which is geometric times a factor $\hat{\lambda}_i$, we have:

$$\begin{aligned} E[N_i] &= \frac{\hat{\lambda}_i}{1 - \hat{\lambda}_i} \\ var(N_i) &= \frac{\hat{\lambda}_i}{(1 - \hat{\lambda}_i)^2} \end{aligned} \tag{3}$$

$E[N_i]$ can also be used to derive the average delay of packets. Recall Little's Formula states $\overline{N} = A\overline{T}$ where $\overline{N}$ is the average number of jobs in queue in the entire system and $\overline{T}$ is the average delay of a job and A is the arrival rate of jobs into the system.

For the Jackson Queueing Network above, $\overline{N} = \sum_i E[N_i]$ and $A = \lambda m$, so

$$\overline{T} = \frac{\sum_i E[N_i]}{\lambda m} \tag{4}$$

3 Modeling Oblivious Packet Routing on a Network as a Jackson Queueing Network

Definition 3 *An* **oblivious routing scheme** *$\mathcal{R}$ specifies for any (source, destination) pair exactly one path from the source to the destination. We assume every path specified is acyclic.*

We assume we are given a network $\mathcal{N}$ with the following properties:

- $\mathcal{N}$ consists of m processors with directed wires between some pairs of processors.
- New packets arrive at processor i according to a Poisson Process with rate r_i.
- A packet contains a destination field and a data field.
- Each packet is routed from its source (origination processor in $\mathcal{N}$) to its destination processor according to an oblivious routing scheme $\mathcal{R}$ (independently of other packets).
- The time it takes for a packet to traverse an edge is exponentially distributed with mean 1.
- Only one packet may traverse a particular directed edge at a time. If two packets require the same edge, contention is resolved via First-Come-First-Serve (FCFS).

In this section we show how to convert any network $\mathcal{N}$ as described above with an associated oblivious packet routing scheme into a Jackson Queueing Network, $\mathcal{Q}$. This allows us to use the formulae of Section 2 to determine the exact probability distribution on the queue buildup at each edge of $\mathcal{N}$.

Since congestion takes place on the edges of $\mathcal{N}$, rather than the nodes, we create one server in $\mathcal{Q}$ for each directed edge in $\mathcal{N}$ and set $\mu_i = 1$ for all servers i in $\mathcal{Q}$. When a packet originates at a processor node of $\mathcal{N}$, we model it as originating at the server of $\mathcal{Q}$ which corresponds to the first edge it must traverse according to $\mathcal{R}$. Observe that $\mathcal{Q}$ may have as many as $O(m^2)$ servers.

We now must determine p_{ij} for $\mathcal{Q}$, that is the probability that a packet at server i next moves to server j, so that the routing algorithm is modeled by the p_{ij}'s. For our general setup, p_{ij} may not be defined, and is certainly difficult to compute. Therefore, we will associate a class, $(source, destination)$, with each packet and compute $p_{ij}^{(s,d)}$, which is very clearly defined.

$$p_{ij}^{(s,d)} = \begin{cases} 1 & \text{if } j \text{ follows } i \text{ in the path specified by } \mathcal{R} \text{ from } s \text{ to } d \\ 0 & \text{otherwise} \end{cases}$$

Note that $r_s^{(s,d)}$, the rate at which packets headed for d arrive at server s, is also specified by $\mathcal{N}$.

Lastly, we create a server in $\mathcal{Q}$ corresponding to each node in $\mathcal{N}$. We call these destination servers. Destination servers have service rate equal to infinity. Packets only enter destination servers if they have reached their destination. Note that since queues never form at the destination servers, these servers may be omitted from the queueing analysis completely.

We have now defined $\mathcal{Q}$ to simulate $\mathcal{N}$ with routing algorithm $\mathcal{R}$. By Theorem 2, we can now calculate the $p_i(n_i)$, the probability of having n_i packets at server i of $\mathcal{Q}$, by first solving the system of simultaneous equations equations for the $\hat{\lambda}_i^{(s,d)}$'s and then summing those to obtain the $\hat{\lambda}_i$'s. However, since the number of classes is m^2 and the number of servers in $\mathcal{Q}$ is $O(m^2)$, **we must solve $O(m^4)$ simultaneous equations just to obtain $\hat{\lambda}_i$ and therefore $p_i(n_i)$.** Note that system of equations is linear and has 0/1 coefficients, however, as pointed out by [20], solving the equations is still a major task when m is large.

In the next section, we present a simple combinatorial formula for $\hat{\lambda}_i$ in $\mathcal{Q}$ which obviates the need to set up and solve the $O(m^4)$ simultaneous equations, and provides insight into the solution.

4 Total Arrival Rate at $i \longleftrightarrow \#$ $\mathcal{R}$-consistent Paths through i

Definition 4 *Let $\mathcal{R}$ be an oblivious routing scheme. A directed path starting at s and ending at d is* **consistent with** $\mathcal{R}$*, or* $\mathcal{R}$**-consistent***, if it is the path specified by $\mathcal{R}$ from s to d.*

Theorem 5 *Let $\mathcal{Q}$ be the queueing network associated with a network $\mathcal{N}$ of the type described in Section 3, having an oblivious routing scheme $\mathcal{R}$. Then the value $\hat{\lambda}_i$ has a simple intuitive meaning: It is the sum of the frequency of use of each $\mathcal{R}$-consistent path through i. If the frequencies of use of all paths are the same, $\hat{\lambda}_i$ is just the frequency of use of a path times the number of $\mathcal{R}$-consistent paths through i.*

Proof: For the sake of the proof, assume that each packet has a class (s, d) associated with it, where s is the packet's source (server at which it entered the network) and d is the packet's destination. Then

$$\hat{\lambda}_i = \sum_{(s,d)} \hat{\lambda}_i^{(s,d)}$$

where $\hat{\lambda}_i^{(s,d)}$ is the total arrival rate of packets into server i which have source s and destination d.

Since packets are routed in the network according to $\mathcal{R}$, and since $\mathcal{R}$ is oblivious, then for any (s, d) and any i, if the path from s to d specified by $\mathcal{R}$ passes through i, then $\hat{\lambda}_i^{(s,d)} = r_s^{(s,d)}$, where $r_s^{(s,d)}$ is the rate at which packets arrive at s (from outside) headed for d. If the path from s to d specified by $\mathcal{R}$ doesn't pass through i, then $\hat{\lambda}_i^{(s,d)} = 0$. So

$$\begin{aligned}
\hat{\lambda}_i &= \sum_{(s,d)} \hat{\lambda}_i^{(s,d)} \\
&= \sum_{\substack{(s,d) \text{ s.t. } i \text{ is on} \\ s \text{ to } d \text{ path} \\ \text{specified by } \mathcal{R}}} r_s^{(s,d)} \\
&= \sum_{\substack{(s,d) \text{ s.t. } i \text{ is on} \\ s \text{ to } d \text{ path} \\ \text{specified by } \mathcal{R}}} \text{rate at which path in } \mathcal{R} \text{ from } s \text{ to } d \text{ is used} \\
&= \sum_{\substack{\mathcal{R}\text{-consistent} \\ \text{paths through } i}} \text{rate at which that path is used}
\end{aligned}$$

If each path has the same frequency of use, then

$$\hat{\lambda}_i = (\text{no. paths through } i \text{ consistent with } \mathcal{R})(\text{frequency of use of a path})$$

■

Theorem 6 *Let $\mathcal{N}$ be a network of the type described in Section 3, for which $r_i = \lambda$, $\forall i$, and packets have random destinations. Let $\mathcal{R}$ be an oblivious routing scheme for $\mathcal{N}$. Let $\mathcal{Q}$ be the queueing network associated with $\mathcal{N}$ and $\mathcal{R}$. Then*

$$\hat{\lambda}_i = \frac{\lambda}{m}(\textit{no. paths through } i \textit{ consistent with } \mathcal{R})$$

Proof: Each path is traversed with frequency $\frac{\lambda}{m}$. ■

5 Example 1: Greedy Randomized Routing on Arrays

Let $\mathcal{N}$ be an $n \times n$ network of processors arranged as a doubly-directed array, where

- New packets arrive at processor i of $\mathcal{N}$ according to a Poisson Process with rate λ.
- A packet contains a destination field which is a **random processor in the array** and a data field. (This assumption comprises the first half of any randomized routing algorithm [19]).
- The time it takes for a packet to traverse an edge is exponentially distributed with mean 1.
- Only one packet may traverse a particular directed edge at a time. If two packets require the same edge, contention is resolved via First-Come-First-Serve (FCFS).

Let $\mathcal{R}$ be the following **greedy routing algorithm** for $\mathcal{N}$: First the packet is routed to the correct column (as specified by its destination) and then to its correct row. Observe that $\mathcal{R}$ is an oblivious routing scheme.

Then by the method of Section 3, we can model $\mathcal{N}$ with routing algorithm $\mathcal{R}$ by a Jackson Queueing Network, $\mathcal{Q}$. We don't even need to specify the parameters of $\mathcal{Q}$, but rather we can immediately jump to Theorem 6 and compute $\hat{\lambda}_i$ in $\mathcal{Q}$. By Theorem 6,

$$\hat{\lambda}_i = (\text{number of greedy paths through } i) \cdot \frac{\lambda}{n^2}$$

We introduce some notation. Let the rows and columns of $\mathcal{N}$ be numbered from 0 to $n-1$ with $(0,0)$ being in the upper, lefthand corner. Recall that there is a server in $\mathcal{Q}$ for each node and directed edge of $\mathcal{N}$. There are 4 edges directed out of node (i,j) in $\mathcal{N}$. We use the notation $P_{ijL}, P_{ijR}, P_{ijU}, P_{ijD}$ to denote the 4 servers in $\mathcal{Q}$ corresponding to the 4 directed edges out of processor (i,j) in $\mathcal{N}$. We use the notation P_{ijC} to denote the destination (i,j) server. (L:left, R:right, D:down, U:up, C:center). We use the notation P_{ijS} as shorthand for $\{P_{ijS} : S \in \{R, L, U, D\}\}$.

Theorem 7 *The total arrival rate of packets at server $P_{r,c,S}$ in $\mathcal{Q}$ is*

$$\hat{\lambda}_{P_{row,col,R}} \quad = \quad \frac{\lambda}{n}(col+1)(n-col-1)$$

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

Proof: We determine the number of greedy paths through $P_{r,c,R}$. All greedy paths through $P_{r,c,R}$ must have a destination to the right of (r,c). There are $n(n-c-1)$ such possible destinations. Additionally since the algorithm routes packets to the correct column before changing rows, the only possible path sources are the $c+1$ sources $(r,0),(r,1),\ldots,(r,c)$. Thus there are a total of $(c+1)n(n-c-1)$ paths through $P_{r,c,R}$. So $\hat{\lambda}_{P_{r,c,R}}$ is $\frac{\lambda}{n^2}(c+1)n(n-c-1)$.

The arguments for $P_{r,c,L}, P_{r,c,U}, P_{r,c,D}$ are similar. ■

Given the total arrival rates, we can now easily compute the probability distribution on the number of packets queued at each server of $\mathcal{Q}$, using Formula 1 . For example,

$$Pr[k \text{ packets queued at } P_{r,c,R}] = (\frac{\lambda}{n}(c+1)(n-c+1))^k \cdot (1-\frac{\lambda}{n}(c+1)(n-c+1))$$

Theorem 8 *We define $N_{(r,c)}$ to be the sum of the number of packets queued at $P_{r,c,R}, P_{r,c,L}, P_{r,c,U}, P_{r,c,D}$.*

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

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: The proof uses Formula 3 plus some algebraic manipulation (see Appendix). ■

Speaking very loosely, Theorem 8 tells us that given a node with horizontal distance x and vertical distance y from the array center, the expected sum of the queue lengths on edges out of the node is proportional to $\frac{1}{x^2}+\frac{1}{y^2}$. Note the relationship to the Euclidean distance of the node from the center of the array.

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

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

Proof: Recall from formula 3, the expected queue size at server i of $\mathcal{Q}$, $E[N_i] = \frac{\hat{\lambda}_i}{1-\hat{\lambda}_i}$. This becomes infinite when $\hat{\lambda}_i = 1$. Since by Theorem 7, $\hat{\lambda}_i \leq \frac{\lambda}{n}\cdot\frac{n}{2}\cdot\frac{n}{2}$, we require $\lambda < \frac{4}{n}$. ■

This bound agrees with that derived from bisection arguments by Leighton [12].

We know from Theorem 7 that the maximum queue size occurs at the array center. What is the probability distribution on the maximum queue size in the array?

Theorem 10 *If the arrival rate λ is p fraction of the maximum capacity, i.e. $\lambda = \frac{4p}{n}$, then*

$$Pr[k \text{ packets at } P_{\frac{n}{2},\frac{n}{2},L}] = p^k(1-p)$$

Proof: From Theorem 7 and Formula 1. ∎

Observe that Theorem 10 is approximately true for $P_{\frac{n}{2},\frac{n}{2},R}$, $P_{\frac{n}{2},\frac{n}{2},U}$, $P_{\frac{n}{2},\frac{n}{2},D}$ as well.

Theorem 11 [3] *If the arrival rate λ is p fraction of the maximum capacity, i.e. $\lambda = \frac{4p}{n}$, then*

$$E[N_{\frac{n}{2},\frac{n}{2}}] = \frac{4n^2}{(1-p)n^2+p} \underset{n\to\infty}{\longrightarrow} \frac{4}{1-p}$$

Lastly, we apply Formula 4 to determine the average delay, $\overline{T}$, of a packet in $\mathcal{Q}$. Since the arrival rate of packets into the *system* is $n^2\lambda$,

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

6 Example 2: Greedy Randomized Routing on a Torus

Let $\mathcal{N}$ be an $n \times n$ network of processors arranged as a doubly-directed torus, having the same properties as the array network of Section 5.

Let $\mathcal{R}$ be the following **greedy routing algorithm**: 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 within that column to its destination again by taking the shortest route (either up or down $\leq \frac{n}{2}$ steps). If n is even, for destinations exactly $\frac{n}{2}$ nodes away (that is, equally close either direction) Up and Right have preference.

[3] We ran simulations for arrays of size 3×3, 4×4, 5×5, and 10×10, using many different values of λ. In steady state, our simulations agreed to within a couple percent with our theoretical results. Except for very small λ, the expected queue size at the center of the array in the case of exponential edge-traversal times was always higher than for constant edge-traversal times, but usually by less than 10% and never by more than 40%.

We now state without proof results for the torus network, of the same type as we proved for the array network. The proofs are analogous, and are further simplified by the fact that the nodes of a torus are indistinguishable.

Theorem 12 *For an $n \times n$ torus, where n is even, the total arrival rate of packets at $P_{r,c,S}$ in Q is*

$$\begin{aligned} \hat{\lambda}_{P_{r,c,R}} &= \frac{\lambda}{8}(n+2) \\ \hat{\lambda}_{P_{r,c,L}} &= \frac{\lambda}{8}(n-2) \\ \hat{\lambda}_{P_{r,c,U}} &= \frac{\lambda}{8}(n+2) \\ \hat{\lambda}_{P_{r,c,D}} &= \frac{\lambda}{8}(n-2) \end{aligned}$$

$$E[N_{(r,c)}] = \frac{2(n+2)}{\frac{8}{\lambda} - (n+2)} + \frac{2(n-2)}{\frac{8}{\lambda} - (n-2)}$$

If n is odd the total arrival rate of packets at $P_{r,c,S}$ in Q is

$$\hat{\lambda}_{P_{r,c,S}} = \frac{\lambda}{8}(n - \frac{1}{n})$$

$$E[N_{(r,c)}] = \frac{n^2-1}{\frac{8n}{\lambda} - (n^2+1)}$$

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.

Theorem 13 *In order for the network to reach steady state, we must have*

$$\lambda < \frac{8}{n}$$

Theorem 14 *If the arrival rate λ is p fraction of the maximum capacity, i.e. $\lambda = \frac{8p}{n}$, then (assuming n is odd)*

$$Pr[k \textit{ packets at } P_{r,c,S}] = (1 - p + \frac{p}{n^2})p^k(1 - \frac{1}{n^2})^k$$

Theorem 15 *If the arrival rate λ is p fraction of the maximum capacity, i.e. $\lambda = \frac{8p}{n}$, then (assuming n is odd)*

$$E[N_{r,c}] = \frac{p(n^2-1)}{n^2(1-p) - p} \xrightarrow[n\to\infty]{} \frac{p}{1-p}$$

We compute the average delay of a packet in the case of a torus, (assuming n is odd).

$$\overline{T} = \frac{1}{n^2\lambda}\overline{N} = \frac{n^2-1}{8n - \lambda(n^2-1)}$$

Acknowledgements

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

References

[1] Søren Asmussen. *Applied Probability and Queues.* John Wiley, 1987.

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

[3] Edmundo de Souza e Silva and Richard R. Muntz. *Queueing Networks: Solutions and Applications.* Stochastic Analysis of Computer and Communication Systems, North-Holland, 1990.

[4] Mor Harchol and Paul Black *Queueing Theory Analysis of Greedy Routing on Square Arrays.* Tech. Report Number csd-93-746, Computer Science Dept., U.C. Berkeley, Berkeley, CA, May 1993.

[5] Mor Harchol and Paul Black *Queueing Theory Analysis of Greedy Routing on Arrays and Tori.* Tech. Report Number csd-93-756, Computer Science Dept., U.C. Berkeley, Berkeley, CA, June 1993.

[6] James R. Jackson *Jobshop-Like Queueing Systems.* Management Science, vol 10, 1963, pp 131-142.

[7] F. P. Kelly. *Networks of Queues With Customers of Different Types.* Journal of Applied Probability, vol 12, 1975, pp 542-554.

[8] F. P. Kelly. *Networks of Queues.* Advances in Applied Probability, vol 8, 1976, pp 416-432.

[9] Leonard Kleinrock. *Queueing Systems. Volume I: Theory.* John Wiley, 1975.

[10] Sunil Kumar and P. R. Kumar. *Performance Bounds for Queueing Networks and Scheduling Policies.* Preprint.

[11] F. Thomas Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes.* Morgan Kaufmann, 1992, pp. 172 - 173.

[12] F. Thomas Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes.* Morgan Kaufmann, 1992, pp. 173-178.

[13] K. C. Madan. *On a Certain Type of Network of Queues.* Journal of Applied Probability, vol 12, 1975, pp 195-200.

[14] Debasis Mitra and R. Cieslak. *Randomized Parallel Communications.* Proc. of the 1986 International Conference on Parallel Processing, pp 224-230.

[15] J. Perkins and P. R. Kumar. *Stable, Distributed, Real-Time Scheduling of Flexible Manufacturing/Assembly/Disassembly Systems.* Preprint.

[16] Sanguthevar Rajasekaran and Thanasis Tsantilas. *Optimal Routing Algorithms for Mesh-Connected Processor Arrays.* Algorithmica, vol 8, pp 21-38.

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

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

[19] L. G. Valiant and G. J. Brebner. *Universal Schemes for Parallel Communication.* Thirteenth Annual ACM Symposium on Theory of Computing, 1981, pp 263-277.

[20] Victor L. Wallace *Algebraic Techniques for Numerical Solution of Queueing Networks.* Mathematical Methods in Queueing Theory (Proc. Conf., Western Michigan University, Kalamazoo, Mich., 1973) in Lecture Notes in Economics and Mathematical Systems, vol 98, 1974, pp 295-305.

[21] Jean Walrand, U.C. Berkeley, Dept. of Engineering. Personal Communication.

A Proof of Theorem 8

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 7 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 3 and setting $a = \frac{n^2-1}{4}$ and $b = \frac{n}{\lambda} - a$, we have

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

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

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

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

■