Dynamic Deflection Routing on Arrays

(Preliminary version)

Andrei Broder* Eli Upfal†

Abstract

We study the performance of a simple one-bend packet routing algorithm on arrays with no buffering in the routing switches, under a stochastic model in which new packets are continuously generated at each node at random times and with random destinations. We prove that on the two dimension torus network our algorithm is stable for an arrival rate that is within a constant factor of the hardware bandwidth. Furthermore, we show that in the steady state the expected time a packet spends in the system is optimal (up to a constant factor). Sharper results (in terms of the constants) are obtained for the ring (dimension one torus).

1 Introduction

Most theoretical work on communication networks has focused on batch, or static routing: A set of packets is injected into the system at time 0, and the routing algorithm is measured by the time it takes to deliver all these packets to their destinations, assuming that no more packets are injected into the system in the meantime. This communication paradigm leads to a reach and interesting theory (see Leighton [12] for an extensive survey) but rarely reflects the practical reality of communication networks. Most real-life networks operate in a *dynamic* mode whereby new packets are continuously injected into the system. Each processor usually controls only the rate at which it injects its own packets and has only a limited knowledge of the global state.

This situation is better modeled by a stochastic paradigm whereby packets are injected to the system according to some distribution, and the routing algorithm is evaluated according to its long term behavior. In particular, quantities of interest are the maximum arrival rate for which the system is stable (that is, arrival rate that ensures that the expected number of packets waiting in queues does not grow with time), and the expected time a packet spends in the system in the steady state.

One might assume that queuing theory would provide ready answers, at least for the simplest topologies and algorithms, but this is not the case: most of the work in queuing theory is based on independence assumptions (such as between successive processing times). These assumptions do not model accurately the routing process on a communication networks where there are complex and hard-to-analyze interactions between packets.

Several recent articles do address the dynamic routing problem, in the context of packet routing on arrays [11, 6, 14], and on the hypercube and the butterfly [16]. The analysis in all these works requires unbounded queues in the routing switches (though some works give high probability bound on the size of the queue used [11, 6]). Unbounded queues allow the application of some tools from queuing theory (see [4, 5]) and helps bounding the correlation between events in the system, thus simplifying the analysis at the cost of a less realistic model.

Here we address the problem of dynamic routing with bounded or no buffers in the routing switches. This paradigm is a better model for current network technologies in which routing switches are built with either very small or no buffers at all. We are not aware of any previous work that presents a rigorous analyzes of a dynamic routing problem on a network with bounded buffers.

Our routing model resembles the deflection (hot-potato) routing model that has been studied in a number of theory papers in the context of batch routing [1, 9, 10, 3]. A node in that model consists of a pro-

*Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, USA. E-mail: broder@pa.dec.com.

†IBM Almaden Research Center, San Jose, CA 95120, USA, and Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel. Work at the Weizmann Institute supported in part by the Norman D. Cohen Professorial Chair of Computer Science, a MINERVA grant, and a grant from the Israeli Academy of Science. E-mail: eli@wisdom.weizmann.ac.il

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copy-right notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.

STOC'96, Philadelphia PA,USA
© 1996 ACM 0-89791-785-5/96/05..$3.50

cessor and a routing switch. The processor has a queue in which it stores the packets it generates. The routing switches, have no buffers to store packets. All packets that reach a routing switch at a given step must leave the switch at the next step. If more than one packet needs to leave the switch through the same edge, all but one of the packets are deflected through other outgoing edges. Thus, packets are always moving, but some packets may temporarily move further away from their destinations. This model suffices for batch routing. To accurately model dynamic routing we need to augment the model with some 'flow-control' mechanism that exists in most routing networks. In 'real-life' routing algorithms a processor keeps a copy of the packet it sent, until it verifies that the packets was received. If no acknowledgment is received within a reasonable amount of time the packet is retransmitted. On the other hand, packets not delivered during some pre-specified time interval are removed. This feature helps the network recover from overloading. We indirectly model this feature by allowing a packet that is not delivered within a given number of steps to return to its sender. The sending processor must send it again at a later time (to the same destination). Parallel machines such as the HEP multiprocessor [15], and high speed communication networks [17] use various forms of deflection routing.

Our main results is an algorithm for routing on the two dimensional torus. The routes chosen are the simplest one-bend routes – the gist of the algorithm is in the choices a processor makes regarding when to insert a packet into the network, and what to do when the packet fails to reach its intended destination.

Theorem 1 *Consider an $n \times n$ 2-dimension torus, with no buffers in the routing switches. Assume that at each step, at each node independently, a new packet with a random destination is inserted with probability λ/n. There is constant $\lambda_0 > 0$ such that our algorithm is stable for any constant $\lambda \leq \lambda_0$, and the expected time a packet spends in the system is $O(n)$.*

Since the expected minimum number of edges that a packet with a random destination must traverse is $\Omega(n)$, our result is optimal up to constant factors with respect to both parameters.

In section 7 we analyze the special case of routing on a ring of n processors with no intermediate queues. Packets already in the ring have priority over new packets. A node moves a new packet to its routing buffer whenever it is empty, and the packet moves on the ring until it reaches its destination. We prove that this procedure can sustain optimal injection rate.

2 Routing on a Two Dimension Torus

2.1 The Model

We consider an $n \times n$ torus network. Each node consists of a processor and a communication switch, each edge represents a bidirectional communication link between communication switches. The network is synchronized. In each step at most two packets can traverse a link, one in each direction. At the beginning of each time step a switch receives packets along its incoming links, at most one packet per link. At the end of the time step the switch sends every incoming packet along some outgoing link, at most one packet per link. There are no buffers at the communication switches: once a packet enters the communication network it continuously moves until it reaches its destination or returns to its sender.

Packets are generated within the processors. A processor can inject a packet into the communication switch when the switch has a free outgoing link, that is only in a step in which the switch did not receive a packet through every one of its four incoming links. We assume that the processor (not the switch!) has a queue that stores packets generated within that processor.

We measure the performance of the network in a stochastic model in which at each step, within each processor, one new packet is generated with probability p, and the generation events at different times and different processors are independent. We refer to p as the arrival rate of the system. Let Z_t denote the total number of packets at the system at time t. We say that the system is stable if Z_t has a limit distribution as $t \to \infty$.

For the analysis it is helpful the view the routing system on the torus as a set of $4n^2$ *containers*. At each time step there are four containers in each switch, the UP, DOWN, LEFT, and RIGHT containers. In each routing step all the UP containers in the system move one step up, all the the DOWN containers move one step down, etc. At any time a container may be empty, or it may contain one packet. Before each routing step the switch can move packets between the four containers it currently has, as long as every packet ends up in one container, and no container contains more than one packet. To inject a new packet into the system the processor needs to put the packet into an empty container currently at its switch.

3 Notation

For simplicity we consider an algorithm whereby a packet first goes into RIGHT container, then it goes into a DOWN container. (Using all four links improves only the constants.)

The n^2 locations on the grid are denoted $M_{i,j}$ for

$i, j \in [0, \ldots, n-1]$. We make the conventions that operations involving these i, j indices are always done mod n.

The RIGHT (resp. DOWN) containers are denoted $H_{i,j}$ (resp. $V_{i,j}$). These notations are fixed, that is, at time t the container $H_{i,j}$ is in position $M_{i,j+t}$ and $V_{i,j}$ is in position $M_{i+t,j}$.

We further define a subset of the RIGHT containers as diagonal Δ_j, via

$$\Delta_{j,i} = H_{i,j-i},$$

that is, Δ_j consists of $\{H_{0,j}, H_{1,j-1}, \ldots, H_{n-1,j-n+1} = H_{n-1,j+1}\}$. The reason for defining diagonals is as follows: consider $V_{i,j}$. At time t it will be in position $M_{i+t,j}$ where it will meet the RIGHT container $H_{i+t,j-t} = \Delta_{i+j,j-t}$. So $V_{i,j}$ only meets the RIGHT containers that belong to Δ_{i+j}, these are the only containers that "feed" it.

4 Description of the algorithm

Our algorithm tries to deliver a packet by a one-bend path, first along the row within which it was generated, then along the column of its destination. The algorithm consists of three stages:

1. Packets generated at $M_{i,j}$ are stored in a FIFO queue. Once a packet is at the top of the queue it waits for an indicator flag, $f_{i,j}$, to become 0. When this happens, the packet becomes *current* and the flag is set back to 1. Eventually the flag will be reset to 0 by the processor at $M_{i,j}$ after the current packet is certainly delivered. This ensures that for every (i, j) there is at most one current packet at time t, denoted $m_{i,j}(t)$.

2. Once current, the packet waits for a random amount of time, geometrically distributed with parameter θ_1, until it becomes *active*.

3. Once active, the packet repeatedly waits n steps, waits for a random H-container, tosses a coin with probability of success θ_2, boards the H-container, and transfers to the corresponding V-container, given its column destination. If the toss is unsuccessful or if either container is full, this stage is repeated until successful. (If the V-container is full, the packet returns to its start point in the H-container it came.)

The precise description of the algorithm is given in Figure 1. Note that system time "flows" only during the execution of **wait** statements.

```
begin
        wait until at head of the queue.
        wait until $f_{i,j} = 0$.
        Set $f_{i,j} \leftarrow 1$ and become *current*.
        while not *active* do
                wait 1 step.
                With probability $\theta_1/n$
                  become *active*.
        od
        Pick a random destination $M_{i',j'}$.
    A:  wait $n$ steps.
        Choose $k$ uniformly at random in
          $[0, n-1]$.
        wait for the container $H_{i,k}$ to
          arrive at $M_{i,j}$.
        With probability $1 - \theta_2$ goto A.
        if $H_{i,k}$ is not empty then goto A.
        else
                Board $H_{i,k}$.
                wait until $H_{i,k}$ arrives in
                  column $j'$. (There it meets
                  $V_{i-j'+k,j'}$.)
                if $V_{i-j'+k,j'}$ is empty then
                        Transfer to $V_{i-j'+k,j'}$.
                        wait until $V_{i-j'+k,j'}$
                          arrives in row $i'$, then
                          leave.
                        After $H_{i,k}$ arrives back
                          to $M_{i,j}$ (empty or
                          with another packet)
                          wait $n$ steps and set
                          $f_{i,j} \leftarrow 0$.
                        done
                else
                        wait until $H_{i,k}$ returns
                          to $M_{i,j}$.
                        Leave $H_{i,k}$.
                        goto A.
                fi
        fi
end
```

Figure 1: Algorithm for packets at $M_{i,j}$ with destination $M_{i',j'}$.

5 Analysis of the algorithm

Specific constants have been chosen for convenience, we made no attempt to optimize them, and, in general, we only claim that inequalities dependent on n hold for n sufficiently large. For the purpose of the analysis we take $\theta_1 = 1/2000$ and $\theta_2 = 1/5$.

To simplify the analysis we shall initially assume that at all times all queues are not empty. We shall see that even under this pessimistic scenario, every packet, once at the top of its queue, is "usually" (the meaning of this will be made precise later) delivered within $O(n)$ time.

We say that the system is in a *normal* state at time t if for all j no more than $n/8$ active packets at time t have destination in column j. Otherwise we say that the system is in an *abnormal* state.

We say that an event $\mathcal{E}$ occurs with extremely high probability (**wehp**) if there exists a constant $\alpha > 0$ such that $\mathbf{Pr}(\neg\mathcal{E}) < e^{-\alpha n}$.

The analysis has three main components:

- In an interval of length ℓn the number of newly active packets with destination in column j is **wehp** bound by a constant times ℓn where the constant can be made as small as desired by choosing θ_1 appropriately (Theorem 2).
- If the system is in a normal state then every active packet has a constant probability of being delivered within $6n$ steps (Theorem 3) and thus if the system is in a normal state at time t it will be **wehp** in a normal state at time $t + 6n$ (Corollary 1) since the number of newly active packets is bounded as above.
- If the system is in an abnormal state then the number of packets that compete for a popular destination **wehp** decrease every $5n$ steps by at least a constant times n. Thus **wehp** within $O(n^2)$ the system returns to a normal state (Theorem 4).

In section 6 we show how the facts above imply Theorem 1.

Theorem 2 *For every time t, the number of newly active packets in interval $[t, t+\ell)$, with destination in column j denoted $N_j(t, t+\ell)$ satisfies*

$$\mathbf{E}(N_j(t, t+\ell)) \leq n^2 \cdot \frac{\theta_1}{n} \cdot \frac{1}{n} \cdot \ell n = \theta_1 \ell n,$$

and **wehp**

$$N_j(t, t+\ell) \leq \frac{10}{9}\theta_1 \ell n.$$

Proof: There are n^2 locations. At every step the probability that a new packet becomes active at a particular location is less than θ_1/n, and the probability that it chooses j as a destination is $1/n$ independently of all other events. □

Before considering the next theorem we need to establish a series of facts and lemmas about the algorithm depicted in Figure 1.

Fact 1 *There is an interval of at least $2n$ between the time a packet leaves an H-container and another packet from the same node enters an H-container (line 21 + line 4 + line 10 in the algorithm).*

Fact 2 *There is an interval of at least n between the time a packet leaves a H-container and the same packet enters again an H-container (line 10 in the algorithm) and thus the number of H-containers a packet can enter in an interval of length ℓn is $\lceil \ell/2 \rceil$.*

Lemma 1 *Let $\mathcal{E}_{i,j}(t)$ be the event that there exist a time r with $t+n \leq r < t+2n$ such that $m_{i,j}(r)$ occupies some H-container. For every t*

$$\mathbf{Pr}(\mathcal{E}_{i,j}(t)) \leq \mathbf{Pr}(X_{i,j}(t) = 1),$$

where $X_{i,j}(t)$ are independent Bernoulli variables, with

$$\mathbf{Pr}(X_{i,j}(t) = 1) = 2\theta_2 = \frac{2}{5}$$

Proof: In view of facts 1 and 2, between $t+n$ and $t+2n$ there is at most one packet from $M_{i,j}$ that might occupy an H-container. Furthermore this packet must succeed (that is, not return to A) at line 13 in the algorithm at some time r with $t < r < t+2n$ and it has at most two opportunities to try. □

Lemma 2 *The probability that $H_{i,k}$ is full when it arrives at $M_{i,j}$ (line 11) is less than 1/2.*

Proof: Let t be the time when $m_{i,j}$ is at A (start of line 10) in the algorithm. By Lemma 1 the probability that a packet from (i, j') occupies an H-container at any time between $t+n$ and $t+2n$ is independently less than $2/5$. Therefore **wehp** no more than, say, $9n/20$ H-containers in row i are occupied at any time between $t+n$ and $t+2n$. Since k is chosen uniformly at random, the probability that $H_{i,k}$ is full when it arrives at $M_{i,j}$ is less that $9/20+$ an exponentially small amount. □

Fact 3 *If a packet is active at time t by time $t+3n$ is either delivered or has to execute line 13 at least once.*

Theorem 3 *If a packet $m_{i,j}$ is active at time t and the system is in a normal state then the probability that it is delivered by time $t+6n$ is at least $\theta_3 \geq 1/20$.*

Proof sketch: The packet $m_{i,j}$ will be either delivered or has to execute line 13 before $t+3n$. (By fact 3.) Assume that it succeeds (that is, does not return to A) the first time it executes line 13 after t. (This happens with probability θ_2.) Let k be its random choice in line 11. Assume that $H_{i,k}$ at line 13 is empty. By Lemma 2 this happens *unconditionally* with probability greater than 1/2. We now need to evaluate the probability that $V_{i-j'+k,j'}$ is empty at line 16.

Let $s(k)$ be the time when $H_{i,k}$ meets $V_{i-j'+k,j'}$ under the scenario above. Observe that if $V_{i-j'+k,j'}$ is not empty then the diagonal D_{i+k} contains a packet with column destination j' at time $s(k)$ (call such a diagonal *busy*) and thus plainly if D_{i+k} did not contain a packet with column destination j' at any time between t and $s(k)$ (call such a diagonal *free*) then container $V_{i-j'+k,j'}$ will be empty when needed.

Now we need to estimate how many busy diagonals could be. First observe that $s(k) < t + 5n$. By hypothesis the number of packets with column destination j' at time t is at most $n/8$ and by Theorem 2 the number of newly active packets in interval $[t, t + 5n)$, with destination in column j' is **wehp** less than $\frac{50}{9}\theta_1 n$. By fact 2 each such packet can affect at most 3 diagonals in the interval of interest, therefore the total number of busy diagonals is **wehp** less than

$$\frac{3n}{8} + \frac{50\theta_1 n}{3} \leq \frac{7n}{16}.$$

Hence the probability that a randomly chosen diagonal is free is at least 1/2.

Now since k is picked uniformly at random, D_{i+k} is uniformly distributed over the diagonals. Hence we claim that the probability that $V_{i-j'+k,j'}$ is empty at line 16 is at least 1/2. (Observe that the fact that $H_{i,k}$ was empty and the fact that D_{i+k} is free are positively correlated.) We conclude that the probability of the entire scenario at least $\frac{\theta_1}{4} = \frac{1}{20}$. □

Corollary 1 *There exists a constant $\alpha > 0$ such that if the system is in a normal state at time t, then with probability at least $1 - e^{-\alpha n}$ the system will be in a normal state at time $t + 6n$.*

Proof: First observe that by Theorem 2 **wehp**

$$N_j(t, t + 6n) \leq \frac{60}{9}\theta_1 n < \frac{n}{200}.$$

Let $A_j(t)$ be the number of active packets at time t with destination in column j. There are two cases to consider: If $A_j(t) < n/9$ then clearly **wehp** A_j will be less than $n/8$ during the entire interval. If $n/9 \leq A_j(t) \leq n/8$, then by Theorem 3 **wehp** $n/200$ of these packets will be delivered by time $t + 6n$. □

Theorem 4 *There are constants $\beta > 0$ and $\gamma > 1$ such that if the system is in a abnormal state at time t, then with probability at least $1 - e^{-\beta n}$ the system will be in a normal state at time $t + \gamma n^2$.*

Proof sketch: Suppose that $A_j(t) > n/9$. Then it can be shown that **wehp** at least $n/100$ of these packets will attempt to board a V-container before $t+4n$. Since each packet attempts to board a random diagonal, the number of "busy" diagonals (see proof of Theorem 3) will be **wehp** at least $n/120$. From each such busy diagonal at least one packet will be delivered before $t + 5n$. On the other hand **wehp** $N_j(t, t + 5n) < \frac{n}{240}$. Therefore **wehp** for every j

$$A_j(t + 5n) \leq A_j(t) - \frac{n}{240}, \qquad \text{if } A_j \geq \frac{n}{9}$$

$$A_j(t + 5n) \leq \frac{n}{8}, \qquad \text{if } A_j \leq \frac{n}{9}.$$

Since $A_j(t) \leq n^2$, we conclude that **wehp**

$$A_j(t + 1200n^2) \leq \frac{n}{8}$$

for all j. □

Theorem 5 *Assume that the system is in a normal state at time t. The probability that the system is in abnormal state at any fixed time $t' > t + 6n$ is bounded by $e^{-\mu n}$, for a constant $\mu > 0$.*

Proof sketch: First assume that $t' \leq t + \gamma n^2$. Then the result follows from Corollary 1. Otherwise condition on the situation at $t' - \gamma n^2$. If normal then apply again Corollary 1; if abnormal then apply first Theorem 4. □

6 Analysis of the actual queues

In this section we consider a single queue and view the rest of the network as a "black box" server $\mathcal{S}$. We view the service time of a particular message as the interval between the time $f_{i,j}$ becomes 1 in line 4 and the time $f_{i,j}$ becomes 0 again in line 21.

We use the series of theorems proven above to bound the behavior of this server. The serving time of $\mathcal{S}$ is not independent of the past; hence, in order to apply standard queuing theorems we shall construct a server $\mathcal{S}'$ such that for every request $\mathcal{S}'$ has a longer service time than the original $\mathcal{S}$ but the service times for $\mathcal{S}'$ are i.i.d. variables.

This new server $\mathcal{S}'$ is best represented by a state diagram as depicted in figure 2. The meaning of a label p, t on an arrow is that the corresponding transition happens with probability p and the elapsed time due to it is t.

Theorem 6 *The expected length of a given queue at a given time is $O(1)$. The expected time that a packet spends in the system is $O(n)$.*

Proof sketch: Define the following random variables:

1. Y_1 is a random variable geometrically distributed with parameter θ_1/n; it counts the number of steps server $\mathcal{S}'$ stays in the Start state, which stochastically dominates the number of steps from the time a packet becomes current until it becomes active.

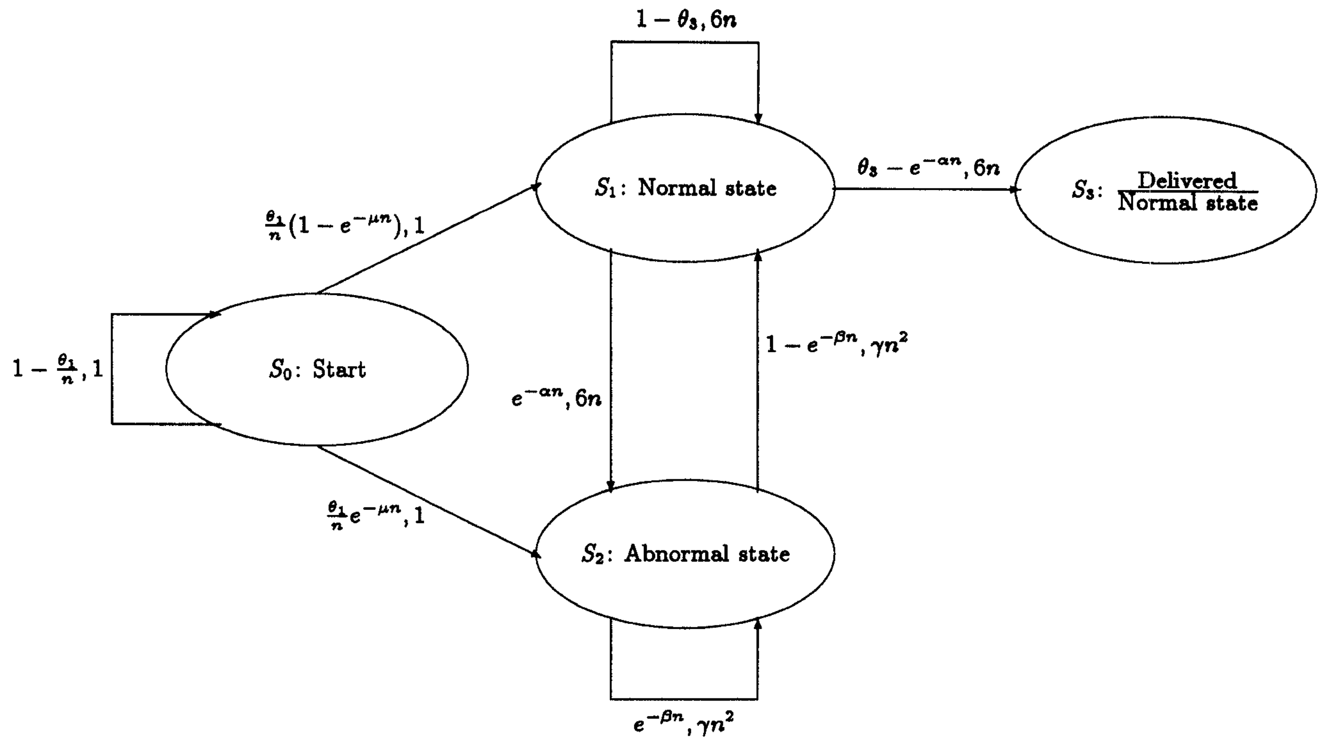
$1-\theta_3, 6n$
S_1: Normal state
$\theta_3 - e^{-\alpha n}, 6n$
S_3: $\frac{\text{Delivered}}{\text{Normal state}}$
$\frac{\theta_1}{n}(1-e^{-\mu n}), 1$
$1-\frac{\theta_1}{n}, 1$
S_0: Start
$e^{-\alpha n}, 6n$
$1-e^{-\beta n}, \gamma n^2$
$\frac{\theta_1}{n}e^{-\mu n}, 1$
S_2: Abnormal state
$e^{-\beta n}, \gamma n^2$

Figure 2: State diagram for server S'.

2. Y_2 is geometrically distributed with parameter $\theta_3 - e^{-\alpha n}$. It counts the number of times server S' enters the state S_1.

3. Y_3 is a Bernoulli variable: if S' eventually moves from the state S_0 to state S_2 then $Y_3 = 1$; otherwise (that is, S' eventually moves to S_1) $Y_3 = 0$.

4. Y_4 is geometrically distributed with parameter $1 - e^{-\beta n}$. Assume that $Y_3 = 1$, then Y_4 counts the number of time S' visits state S_2 before reaching state S_1.

5. The variables $Y_{5,i}$ are i.i.d. Bernoulli variables: $Y_{5,i} = 1$ if S' moves to state S_2 after the ith time it entered state S_1, else $Y_{5,i} = 0$.

6. The variables $Y_{6,i}$ are i.i.d. variables, geometrically distributed with parameter $1-e^{-\beta n}$. Assume that S' moved to state S_2 after the ith time it entered states S_1. $Y_{6,i}$ counts the number of times the server returns to state S_2 before moving back to state S_1.

The number of steps server S' spends from the Start state to the Delivered state is given by

$$X = Y_1 + \gamma n^2 Y_3 Y_4 + \sum_{i=1}^{Y_2}(6n + \gamma n^2 Y_{5,i} Y_{6,i}),$$

and X stochastically bounds the number of steps from the time a packet reaches the head of the queue to the time it is delivered.

We define an $M/G/1$ queue that always contains more packets than the queue of a given node v. The inter-arrival time of the $M/G/1$ queue is the same as that of the queue of node v, that is, it is distributed geometrically with parameter λ/n. The service time of the queue has the distribution of the random variable X.

It is easy to verify that

$$\mathbf{E}(Y_1) = \frac{n}{\theta_1}, \quad \mathbf{var}(Y_1) = (\frac{n}{\theta_1})^2,$$

$$\mathbf{E}(Y_2) = \frac{1}{\theta_3 - e^{-\alpha n}}, \quad \mathbf{var}(Y_2) = (\frac{1}{\theta_3 - e^{-\alpha n}})^2,$$

$$\mathbf{E}(Y_3 Y_4) = \mathbf{E}(Y_{5,i} Y_{6,i}) = \frac{e^{-\alpha n}}{1 - e^{-\beta n}},$$

and

$$\mathbf{var}(Y_3 Y_4) = \mathbf{var}(Y_{5,i} Y_{6,i}) \le \frac{2e^{-\alpha n}}{(1 - e^{-\beta n})^2}.$$

Since the random variables Y_1, Y_2, Y_3, Y_4, $Y_{5,i}$ and $Y_{6,i}$ are independent we compute:

$$\mathbf{E}(X) = \mathbf{E}(Y_1) + \gamma n^2\, \mathbf{E}(Y_3 Y_4) + \mathbf{E}(Y_2)(6n + \gamma n^2\, \mathbf{E}(Y_{5,1} Y_{6,1}))$$

$$= \frac{n}{\theta_1} + \gamma n^2 \frac{e^{-\mu n}}{1-e^{-\beta n}} + \frac{1}{\theta_3 - e^{-\alpha n}}(6n + \gamma n^2 \frac{e^{-\alpha n}}{1-e^{-\beta n}})$$
$$= \Theta(n).$$

If Z_i are i.i.d. random variables, independent of T, then

$$\mathbf{var}(\sum_{i=1}^{T} Z_i) = \mathbf{E}(T)\,\mathbf{var}(Z_1) + (\mathbf{E}(Z_1))^2\,\mathbf{var}(T).$$

Thus,

$$\mathbf{var}(X) = \mathbf{var}(Y_1) + \gamma^2 n^4\,\mathbf{var}(Y_3 Y_4) +$$
$$\mathbf{var}(Y_2)(6n + \gamma n^2\,\mathbf{E}(Y_{5,1}Y_{6,1}))^2 +$$
$$\mathbf{E}(Y_2)\gamma^2 n^4\,\mathbf{var}(Y_{5,1}Y_{6,1}) = O(n^2).$$

Let $\rho = \frac{\lambda}{n}\mathbf{E}(X)$. Applying the Pollaczek-Khinchin mean-value formula [8, page 187] we conclude that the expected number of packets in the queue is

$$\mathbf{E}(N) = \rho + \rho^2 \frac{1 + \mathbf{var}(X)/\mathbf{E}(X)^2}{2(1-\rho)} = O(1),$$

for any λ such that $\frac{\lambda}{n}\mathbf{E}(X) < 1$. Applying Little's principle the expected number of steps a packet spends in the system is given by

$$\frac{\mathbf{E}(N)}{\lambda/n} = O(n). \qquad \square$$

7 Deflection Routing on the Ring

Consider an n-node directed ring. Each node has one incoming link and one outgoing link. There are n containers moving over the ring from node to next node such that at each time step each node has one container. A node can move a packet from its queue to a container if the container it currently holds is empty. Since the expected distance a packet needs to travel on the ring is $n/2$, the process cannot be stable for an injection rate higher than $2/n$. The following theorem proves that this bound is tight.

Theorem 7 *Routing with no intermediate buffers on an n node directed ring is stable for any injection rate* $\frac{\lambda}{n}$, $\lambda < 2$.

Proof: Consider the queue at a given node v. We analyze the performance of the queue as an $M/G/n$ queuing process. The arrival rate to the queue is Poisson with an average of $\frac{\lambda}{n}$ arrivals per step. View every container as a server of the queue at v. Clearly, it services the queue whenever it delivers a packet to v or arrives empty to v. The expected time a packet occupies a container is $n/2$. Every time a container becomes empty it either gets a packet that with probability $\frac{1}{n}$ has destination v, or it continues empty to the next node. Thus, the expected service time of each container with respect to the queue of node v is bounded by $\frac{n^2}{2}$. Let $\bar{x}$ denote the average arrival time to queue v, and let $\bar{t}$ be the average service time of a server. It was shown in [7] that $\rho = \frac{\bar{x}}{m\bar{t}} < 1$ is a sufficient condition for stability for any $G/G/m$ queue. Thus, the queue of v is stable for any $\lambda < 2$. $\square$

General methods for analyzing $M/G/n$ queues give only an $O(n^2)$ bound for the expected time a packet spends in the system, for $\lambda < 2$. The following theorem proves an $O(n)$ bound for $\lambda < 1$, that is, injection rate up to $1/n$.

Theorem 8 *The expected time a packet spends in the system in routing with no intermediate queues on an n node ring is $O(n)$ for any injection rate λ/n, with* $\lambda < 1$.

Proof: A packet never occupies a container for more than $n-1$ steps, thus when a container reaches node v it is either empty, or it holds a packet that it received in the last $n-1$ steps. That packet has destination v with probability at least $1/n$, therefore the queue is serviced in each step with probability at least $1/n$. Viewed as an $M/M/1$ queue, the expected number of packets in the queue is $\frac{\lambda}{1-\lambda}$, the expected service time is n, and applying Little's theorem we get that the expected time a packet spends in the system is $\frac{n}{2} + \frac{\lambda n}{(1-\lambda)}$. $\square$

References

[1] A. Bar-Noy, P. Raghavan, B. Schuster, and H. Tamaki. Fast deflection routing for packets and worms. *Proceedings of 12th ACM Symposium on Principles of Distributed Computing*, 1993.

[2] H. Chernoff. A measure for asymptotic efficiency of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23:493–507, 1952.

[3] U. Feige and P. Raghavan. Exact analysis of hot potato routing. *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pp. 553–562, 1992.

[4] M. Harcol-Balter and P. Black. Queuing analysis of oblivious packet routing networks. *Proceedings of the 5th Annual ACM-SIAM Symp. on Discrete Algorithms.* Pages 583–592, 1994.

[5] M. Harcol-Balter and D. Wolf. Bounding delays in packet-routing networks. *Proceedings of the 27th Annual ACM Symp. on Theory of Computing,* 1995, pp. 248–257.

[6] N. Kahale and T. Leighton. Greedy dynamic routing on arrays. *Proceedings of the 6th Annual ACM-SIAM Symp. on Discrete Algorithms.* Pages 558–566, 1995.

[7] J. Keifer and J. Wolfowitz. On the theory of queues with many servers. *Transaction of the American Math. Society,* 78:1–18, 1955.

[8] L. Kleinrock. *Queuing Systems Volume I: Theory,* John Wiley, New York, NY, 1975.

[9] A.G. Greenberg and B. Hajek. Deflection routing in hypercube networks. *IEEE Transaction on Communications,* 1992.

[10] B. Hajek. Bounds on evacuation time for deflection routing. *Distributed Computing,* 5:1-6, 1991.

[11] T. Leighton. Average case analysis of greedy routing algorithms on arrays. *Proceedings of the Second Annual ACM Symp. on Parallel Algorithms and Architectures.* Pages 2–10, 1990.

[12] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan-Kaufmann, San Mateo, CA 1992.

[13] N.F. Maxemchuk. Comparisons of deflection and store and forward techniques in the Manhattan street and shuffle exchange networks. *Proceedings of IEEE INFOCOM,* pages 800–809, 1989.

[14] M. Mitzenmacher. Bounds on the greedy algorithms for array networks. *Proceedings of the 6th Annual ACM Symp. on Parallel Algorithms and Architectures.* Pages 346–353, 1994.

[15] B. Smith. Architecture and applications of the HEP multiprocessor computer system. *Proceedings of Real Time Signal Processing IV,* pages 241–248, 1981.

[16] G.D. Stamoulis and J.N. Tsitsiklis. The efficiency of greedy routing in hypercubes and butterflies. *Proceedings of the 6th Annual ACM Symp. on Parallel Algorithms and Architectures.* Pages 346–353, 1994.

[17] T. Tzymanski. An analysis of "Hot Potato" routing in a fiber optic packet switches hypercube. *Proceedings IEEE INFOCOM,* pages 918–925, 1990.