The Performance of Simple Routing Algorithms that Drop Packets

Suprakash Datta Ramesh Sitaraman *
Department of Computer Science,
Univ. of Massachusetts,
Amherst, MA 01003, USA.
{datta,ramesh}@cs.umass.edu

Abstract

Several modern high-speed networks implement routing algorithms that resolve contention for resources such as buffer space by dropping (i.e., deleting) packets. In this paper, we analyze the performance of such routing algorithms for the commonly-used butterfly network. We assume that each switch of the butterfly has a buffer that can hold a bounded number of packets, and any packet attempting to enter a switch with a full buffer is simply dropped from the network. We study three significant metrics that characterize routing performance: expected throughput of the network, packet loss rate, and expected delay of a packet. Our main results are analytic expressions for these three performance metrics in terms of the network-size, size of the buffer at each switch, and the packet arrival rate. Our analyses for the throughput and packet loss rate hold for any non-predictive queuing protocol, including simple, often-implemented protocols such as first-in first-out (FIFO) and fixed-priority scheduling. Our delay expressions hold for the FIFO protocol. Several facts of interest to a network designer fall out of our analysis. Further, our results provide quantitative insights into how the three performance metrics tradeoff against each other. Also, we present simulation results to bolster the results of our analysis. Finally, we outline preliminary results for routing on other networks such as the crossbar.

1 Introduction

Many commercial and experimental parallel computers use regular interconnection networks such as the butterfly to route packets between processors [Got87, PBG+87]. Several proposed designs for the switching fabric of scalable high-speed ATM networks also use the butterfly and other closely-related banyan networks [RCG94]. Typically, a packet is routed from its source to its destination along a path in the network consisting of a sequence of switches. Further, each switch has a buffer that can hold a bounded number packets, and each switch uses a simple queueing protocol such as first-in first-out (FIFO) to determine which packets to transmit at each step.

An important distinguishing feature of a network routing algorithm is the manner in which contention for resources at a switch is resolved. In most traditional parallel networks, if a packet wishes to enter a switch that has no buffer space available, or if a packet wishes to use a link that is busy, the packet is simply blocked, until a time the requested resource becomes available. However, several ATM implementations [RCG94] and high-speed switches like NEC's ATOM switch [NMT+91] simply *drop* (i.e., delete) the packet that contends for an unavailable resource. For instance, a packet attempting to enter a switch with a full buffer is dropped from the network.

There are several advantages to dropping packets to resolve contention. In traditional parallel networks where packets can be blocked, the delay of a packet can be large in the worst-case. It is estimated that in the Intel Paragon with 1024 nodes, the delay of a packet can theoretically be as large as several days [Int91]!. However, networks that drop packets to resolve contention never get congested due to blocked packets, and the delay of each packet that successfully reaches its destination is easily bounded in the worst-case. This is an important concern in the design of parallel networks for delay-sensitive, real-time applications like process control and multimedia. Further, several types of network traffic, for instance real-time audio and video, can tolerate some amount of packet loss without a significant degradation in quality, but they are intolerant to delay. Dropping packets is a natural contention-resolution mechanism for such traffic.

Despite the existence of several practical networks that utilize packet dropping, much remains to be known about the performance of these networks from a theoretical standpoint. In this paper, we study routing on a butterfly network where contention is resolved by dropping packets. We restrict our attention to a simple, easily-implementable class of queueing protocols that includes protocols such as First-in First-Out (FIFO). We focus on three performance metrics: expected throughput, packet loss rate, and expected delay. We derive precise quantitative expressions for these performance metrics in terms of the network-size, the size of the buffer at each switch, and the packet arrival rate. While our primary results pertain to the butterfly network, similar techniques can be applied to other networks (such as the crossbar). Our work helps answer a number of questions that are important to a network designer. For instance, what is the size of the buffers required at a switch for a specific expected throughput, packet loss rate, or expected delay?

*The authors are supported in part by NSF Grant CCR-94-10077.

SPAA '97 Newport, Rhode Island USA

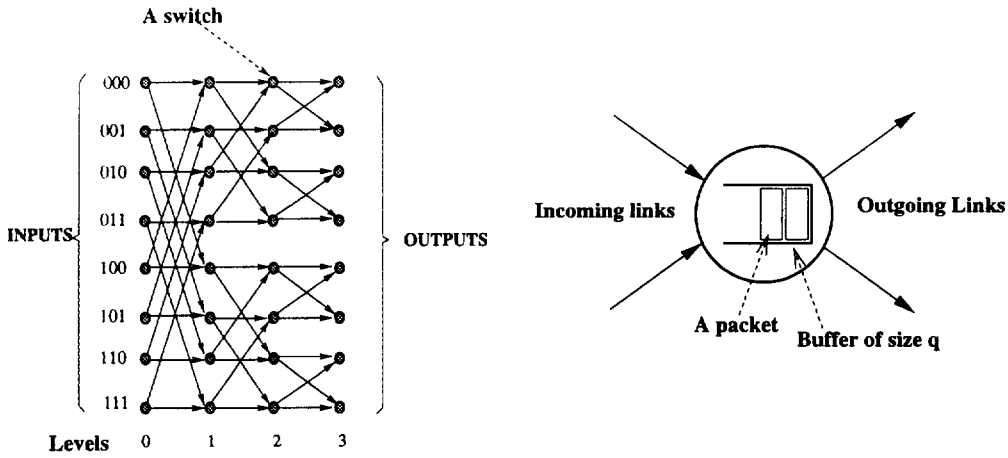

Figure 1: An 8-input butterfly network, and the internal structure of a switch.

Or, how does packet loss rate tradeoff with expected delay? We also provide empirical simulation results to bolster our analysis.

1.1 Our Model

In this section, we describe our network and traffic model, and define our performance metrics.

Network Definition. An N-input *butterfly* has $(\log N + 1)N$ nodes arranged in $\log N + 1$ levels[1] (See Figure 1). We use the term *size* of the butterfly network to denote the number of inputs it has. Each edge is an uni-directional communication link, directed from the node in the smaller numbered level to the node in the larger numbered level. Each node of the butterfly is a 2×2 switch with two incoming and two outgoing links. We use the terms node and switch interchangeably in the rest of the paper. Each node has a distinct label (w, i), where i is the *level* of the node ($0 \leq i \leq \log N$) and w is a $\log N$-bit binary number that denotes the *row* of the node. Two nodes (w, i) and (w', i') are linked by an edge if and only if $i' = i + 1$ and either w and w' are identical (straight edge) or w and w' differ in precisely bit i' (cross edge). The switches on level 0 are called *inputs*, and the those on level $\log N$ are called *outputs*. In a butterfly network, packets are typically sent from the inputs on level 0 to the outputs on level $\log N$. In a butterfly network there is an unique path of length $\log N$ between any input and any output. A packet enters the network at an input node, and traverses the unique path from its input node to its output node, passing through $\log N + 1$ switches en route.

Switch Model. We assume that each switch of the butterfly network has a single buffer that can hold a fixed number of packets denoted by q (See Figure 1). All packets have the same size, and a packet can be transmitted across a link in unit-time. The packets enter and leave the switches at discrete time steps governed by a common clock. At the beginning of each time step, each switch selects a *single* packet from its buffer using a *non-predictive queueing protocol* [Lei92, Section 3.4.4], i.e., each switch selects a packet from its buffer *without examining the destinations of any of the packets in its buffer*. Note that many easily implementable as well as conceptually simple protocols like FIFO and fixed-priority scheduling are non-predictive. Then, each switch forwards the selected packet to a switch in the next level of the butterfly. Following this, each switch receives packets sent to it in the previous time step, and places them in its buffer. A buffer is said to be *full*, if it contains q packets. A packet wishing to enter a switch with a full buffer is simply dropped. Note that if two packets simultaneously enter a switch with $q-1$ packets in its buffer, one of the two packets is dropped, while the other packet is placed in the buffer.

Traffic Model. We assume that packets arrive at each input with a geometric interarrival distribution[2] with arrival rate λ, $0 < \lambda \leq 1$. In other words, the probability that a packet arrives at a specific input, at a specific time step is λ. Further, the event that a packet arrives at a specific input at a specific time step is independent of all other packet arrivals at other inputs or at other times. Each packet chooses its output destination randomly, such that each output is equally likely to be chosen. Further, each packet chooses its destination independent of all other packets. (Note that we do not model the retransmission of dropped packets. Such retransmission may or may not be required in practice depending on the type of traffic.)

Performance Metrics. We are concerned with three important performance metrics: *expected throughput*, *packet loss rate*, and *expected delay*. Other metrics like jitter are relevant for some classes of traffic, but we will not consider them in this paper. Expected throughput is the expected number of packets that reach their destinations in a time step at steady state. Packet loss rate is the probability that a packet is dropped by a switch on its path, at steady state. Expected delay is the expected value of the total number of time steps that a packet waits in the buffers of the switches along its path, before reaching its destination, at steady state. Note that the expectation is taken only over packets that successfully reach their destinations, excluding packets that get dropped. Expected latency is the expected value of the total time taken by a packet to go from its source to its destination. Note that expected latency is simply the expected delay plus the time steps in which the packet moves from one switch to the next, i.e., for an N-input butterfly network, expected latency is expected delay plus $\log N$.

[1] All logarithms in this paper are base 2.

[2] Geometric distribution is a memoryless distribution and is the discrete analog of the Poisson distribution.

1.2 Previous work

The study of routing on interconnection networks has a long history dating back to Batcher [Bat68]. We refer to Leighton's book[Lei92] or survey [Lei92a] for a comprehensive treatment. In this section, we describe only work closely-related to this paper.

Several papers deal with *static* routing in butterfly networks, where all the packets that need to be routed are present at time zero, and no new packets arrive when the routing is in progress. The first results devise butterfly routing algorithms that require buffers of unbounded size at each switch [Val82, VB81, Ale82, Upf84]. Subsequent papers improve on these results to obtain butterfly routing algorithms that require only buffers of bounded size at each switch [Pip84, LMRR94, Ran91, MS92].

The study of *dynamic* routing in butterfly networks, where packets arrive at the inputs in an on-line fashion according to a statistical distribution, has received recent attention. Dynamic packet routing models routing on real networks more closely than static routing. Stamoulis and Tsitsiklis [ST91] study dynamic butterfly routing with unbounded buffers at each switch. Broder, Frieze, and Upfal [BFU96] generalize static routing results to provide dynamic butterfly routing algorithms that utilize buffers of bounded size at each switch. All the papers mentioned so far deal with butterfly networks that block packets to resolve contention, and there is no packet loss. Recent papers on universal store-and forward packet routing also provide new results for butterfly routing [MadHV95, CadHSV96, VS96].

There are several papers that study the expected throughput of static circuit-switching on the butterfly, where a circuit attempting to use a saturated link is simply dropped [Pat81, KS83, MS92, Lei92]. Koch [Koc88] showed that if each input of an N-input butterfly routes to a random output, the expected number of inputs that succeed in locking down circuits to their output destinations is $\Theta(N/\log^{\frac{1}{q}} N)$, where q is the maximum number of circuits that any link of the butterfly can support. Koch's circuit-switching result can be used to derive a lower bound on the expected throughput of static routing on butterfly networks that drop packets [MS92]. Specifically, if one packet is routed from each input of an N-input butterfly network to a random output, and the buffer-size at each switch is q, the expected number of packets that reach their destination is $\Omega(N/\log^{\frac{1}{q}} N)$[MS92]. (The throughput analysis in Section 2.5 suggests that this lower bound is not tight, and that the expected throughput is $\Theta(N/\log^{\frac{1}{2q-1}} N)$.)

The first major step in the analysis of dynamic routing in butterfly networks that drop packets was taken in [RMDL96]. Their paper shows that the expected throughput of an N-input butterfly network is $\Theta(N/\sqrt{\log N})$, when a packet destined for a random output arrives at each input at each time step (i.e., $\lambda = 1$), and when each switch can buffer at most one packet at each of its incoming links. Unlike our switch model which utilizes a single shared buffer of arbitrary size q, their work assumes that unit-sized buffers are placed at the incoming links. We utilize some techniques developed in their paper and in the earlier work of Koch [Koc88] in our throughput analysis for arbitrary λ and q.

While all of the above papers perform exact analyses, several papers provide approximate analyses of butterfly routing [SS89, Mer91, Jen83, TRH91, Tur93, YLL90, GP93, GP94]. Typically, simplifying assumptions, such as assuming statistical independence of dependent events, are made to facilitate analysis. The results of the approximate analysis are often validated numerically or via simulations.

1.3 Our Results

Our paper provides the first exact analysis of the expected throughput, packet loss rate, and expected delay of dynamic routing on an N-input butterfly network for arbitrary arrival rate $0 < \lambda \leq 1$, and buffer-size $q > 0$. Our main results are that if packets are routed on an N-input butterfly using a non-predictive queueing protocol, the expected throughput of the network is

$$\frac{N}{\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1} + o(q\log N)\right)^{\frac{1}{2q-1}}},$$

the packet loss rate is

$$1 - \frac{1}{\lambda\left(\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1} + o(q\log N)\right)^{\frac{1}{2q-1}}\right)},$$

where $\Lambda = (2-\lambda)/\lambda$. For the FIFO queueing protocol, the expected delay of a packet is zero, when $q = 1$, and is

$$\Theta\left(\frac{1}{q-1}\left(\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1}\right)^{1-\frac{1}{2q-1}} - \Lambda^{2q-2}\right)\right)$$

when $q > 1$, where $\Lambda = (2-\lambda)/\lambda$.

Several facts of interest to a network designer fall out of our analysis. For instance, consider the case when the network is heavily-loaded, say with $1 - 1/2q \leq \lambda \leq 1$. In this case, the expected throughput is $\Theta(N/\log^{1/(2q-1)} N)$, packet loss rate is $1 - \Theta(1/\log^{1/(2q-1)} N)$, and expected delay is $\Theta(\log^{1-1/(2q-1)} N)$. Therefore, if we require that the expected throughput be a constant fraction of the network-bandwidth[3], or that the packet loss rate be a constant fraction less than 1, the buffer-size q must be $\Omega(\log\log N)$. Further, when the network is heavily-loaded, our analysis predicts that the *expected* delay is $o(\log N)$, for any fixed constant q. Note that the *maximum* delay of a packet is larger and is $(q-1)\log N$, since each packet can be delayed by $q-1$ other packets at a switch, and each packet traverses $\log N$ switches excluding its input switch.

Our results also provide quantitative insights on how the expected throughput, packet loss rate, and expected delay tradeoff against each other. For instance if we require the packet loss rate to be small, or the expected throughput to be large, we must increase the buffer-size q, which in turn increases the expected delay. If the network is heavily-loaded, say with $1 - 1/2q \leq \lambda \leq 1$, our analysis shows that the expected throughput initially increases super-linearly with buffer-size q, i.e., doubling q can more than double the expected throughput. This increase in expected throughput tapers off for higher values of q. This suggests that it may be cost-effective for a network designer to add extra buffers when q is small, provided the network is heavily loaded. However, when the network is lightly-loaded, say with $\lambda = \Theta(1/\log N)$, the expected throughput is $\Theta(N/\log N)$ and both expected throughput and packet loss rate are less sensitive to variations in the buffer-size q.

[3] We define network-bandwidth to be the maximum number of packets the network can deliver in a single time step. By our definition, the bandwidth of an N-input butterfly is N.

1.4 Outline of the paper

The paper is organized as follows. In Section 2, we derive expressions for the expected throughput (Section 2.5), packet loss rate (Section 2.5), and expected delay (Section 2.6) for routing on a butterfly network. In Section 3, we provide simulation results to bolster our analytical results. Finally, in Section 4, we present concluding remarks. The details of several proofs appear in the appendices.

2 Performance analysis

In this section, we derive expressions for the expected throughput, packet loss rate, and expected delay as functions of the network-size N, arrival rate λ, and the buffer-size q.

2.1 Modeling the routing network as a markov chain

Lemma 2.1.1 *The process of routing packets using a non-predictive queueing protocol on an N-input butterfly with buffer-size $q > 0$ and packet arrival rate $0 < \lambda \leq 1$ can be modeled as a finite-state markov chain MC.*

Proof: The states of the markov chain MC are defined as follows. Let $s_1, s_2, \cdots s_{(\log N+1)N}$ be an arbitrary ordering of the switches in the network. We represent each state σ of MC as a $(\log N + 1)N$-tuple,$\langle b_1, b_2, \cdots, b_{(\log N+1)N}\rangle$, where b_j, $0 \leq b_j \leq q$, is the number of packets in switch s_j of the network. Initially, there are no packets in the network, i.e., the network is in the *zero-state* that is defined to be the tuple $\langle 0, 0, \cdots, 0\rangle$. The set of states S of markov chain MC is simply all the states that are reachable with non-zero probability from the zero-state. Clearly, S is a finite set, since $|S|$ is at most $(q+1)^{(\log N+1)N}$.

Let X_t represent the state of the network at time t. We show that the random process $\{X_t\}$ has the markovian property, i.e., the future behavior of the random process depends on the current state only, and not on the sequence of states it took to reach the current state. That is,

$$\Pr(X_{t+1} = \sigma_{t+1} | X_t = \sigma_t, X = \sigma_{t-1}, \cdots, X_0 = \sigma_0) = \Pr(X_{t+1} = \sigma_{t+1} | X_t = \sigma_t). \quad (1)$$

Note that any packet selected by a switch s at time t uses one of the two outgoing links of s with probability 1/2, independent of the previous history of the packet. This follows from the fact that the switches choose packets in a non-predictive fashion without examining their destinations, and each chosen packet is either successfully transmitted or dropped. Further, new packets arrive at each input using the memory-less geometric interarrival distribution. Thus, Equation 1 holds. □

Caveat: Note that the markovian property fails to hold when the selected packet can get blocked due to link contention [RMDL96] or buffer contention[MS92]. In such cases, the random process can only be approximated by a markov chain.

2.2 Steady-state behavior of the markov chain

Let $\sigma_1, \sigma_2, \cdots, \sigma_{|S|}$ be an arbitrary ordering of the states in S, where S is the set of states in markov chain MC. Let Π^t denote the state probability vector of the markov chain MC at time t, i.e., $\Pi^t = [\pi_1^t, \pi_2^t, \cdots, \pi_{|S|}^t]$, where π_j^t is the probability that markov chain is in state σ_j at time t. Since we are interested in the steady-state behavior of the network, we must show that the state probability vector Π^t converges to a stationary steady-state vector Π, i.e.,

$$\lim_{t\to\infty} \Pi^t = \Pi$$

Lemma 2.2.1 *The state probability vector of the markov chain MC converges to a steady state stationary vector Π.*

Proof: To show that the state probability vector of the finite-state markov chain MC converges to a steady state vector Π, we must show that MC has exactly one ergodic class $\mathcal{C}$ of states, i.e., MC has exactly one class $\mathcal{C}$ that is recurrent and aperiodic [Gal96].

We define the state of MC in which each switch of the network has exactly one packet to be the *one-state*. Let $\mathcal{C}$ be the set of states that are reachable with non-zero probability from the one-state in a finite number of steps. We show that $\mathcal{C}$ is the only class of states of MC that is recurrent and aperiodic.

We show that class $\mathcal{C}$ is recurrent as follows. First, note that the one-state is reachable from any state σ_i in MC in a finite number of steps with non-zero probability. Since $\lambda > 0$, there is a non-zero probability that one packet arrives at each input of the network at a particular time step, and each such packet routes to its destination using only the straight edges of the network. If the network is started at any state σ_i in MC, after at most $q(\log N + 1)$ such time steps, the network will be in the one-state, since packets that use only the straight edges have non-intersecting paths. Next, note that any state σ_j in $\mathcal{C}$ is reachable with non-zero probability from the one-state, by the definition of $\mathcal{C}$. Thus, for every pair of states σ_i and σ_j in $\mathcal{C}$, there exists an integer k such that the probability that the markov chain goes from state σ_i to state σ_j in k time steps is non-zero, i.e., class $\mathcal{C}$ is recurrent. Note that $\mathcal{C}$ is the only recurrent class because any state $\sigma \notin \mathcal{C}$ is transient, since the one-state is reachable from σ with non-zero probability but not vice-versa.

We now show that the class $\mathcal{C}$ is aperiodic. Suppose MC is in a state σ in $\mathcal{C}$ at time 0. Let k be an integer such that the probability that MC returns to state σ at time k after passing through the one-state is non-zero. Such a k exists because σ is in the recurrent class $\mathcal{C}$. Since the markov chain MC can remain in the one-state for any finite number of steps with non-zero probability, for all $t \geq k$, the probability that the markov chain MC returns to state σ at time t is non-zero. Thus, every state σ in $\mathcal{C}$ is aperiodic, i.e., the class $\mathcal{C}$ is aperiodic. □

2.3 A recurrence relation for steady-state probabilities

Now that we have determined that the state probability vector of MC converges to a steady-state vector Π, we characterize the steady-state probabilities via recurrence relations. Note that any two switches within the same level of the butterfly are statistically identical. This follows from the symmetric structure of the butterfly, and the symmetric traffic assumptions. Let $p(i,j)$, $0 \leq i \leq \log N$ and $0 \leq j \leq q$, denote the probability that a switch at level i has j packets in its buffer at the *end* of a time step at steady state. In other words,

$$p(i,j) = \lim_{t\to\infty} p^t(i,j),$$

where $p^t(i,j)$ is the probability that a switch at level i has j packets at the end of time step t, assuming that the markov chain MC starts in the zero-state at time 0.

Let $I_{i,j}$, $0 \leq i \leq \log N$, $j = 0, 1$, or 2, represent the steady-state probability that a switch at level i receives exactly j packets from switches at level $i-1$ in a specific time step. For $3 \leq j < q$,

$$p(i,j) = p(i,j-1)I_{i,2} + p(i,j)I_{i,1} + p(i,j+1)I_{i,0}. \quad (2)$$

The above equation holds because, in steady state, if a switch s at level i has j packets, $3 \leq j < q$, in the current time step, one of the following events happened.

- Switch s had $j-1$ packets in the previous time step, the switch sent out one packet and received two new packets in the current time step.
- Switch s had j packets in the previous time step, and the switch sent out one packet, and received one new packet in the current time step.
- Switch s had $j+1$ packets in the previous time step, and the switch sent out one packet, and received no new packets in the current time step.

Using similar arguments, the following hold.

$$\begin{aligned} p(i,0) &= p(i,0)I_{i,0} + p(i,1)I_{i,0} && (3)\\ p(i,1) &= p(i,0)I_{i,1} + p(i,1)I_{i,1} + p(i,2)I_{i,0} && (4)\\ p(i,2) &= p(i,0)I_{i,2} + p(i,1)I_{i,2} + p(i,2)I_{i,1} \\ &\quad + p(i,3)I_{i,0} && (5)\\ p(i,q) &= p(i,q-1)I_{i,2} + p(i,q)I_{i,1} + p(i,q)I_{i,2} && (6) \end{aligned}$$

Lemma 2.3.1 *The steady state probability $p(i,0)$ can be expressed in terms of $I_{i,0}$ and $I_{i,2}$ as follows.*

$$p(i,0) = \frac{I_{i,0} - I_{i,2}}{1 - \left(\frac{I_{i,2}}{I_{i,0}}\right)^q}$$

Proof: Using Equation 6, we obtain

$$p(i,q) = p(i,q-1)\left(\frac{I_{i,2}}{I_{i,0}}\right) \quad (7)$$

Inductively, for any $3 \leq j < q$, we can use Equation 2 to show that

$$p(i,j) = p(i,j-1)\frac{I_{i,2}}{I_{i,0}} \quad (8)$$

Using Equations 7, 8, and 5, we can show that for any $2 \leq j \leq q$,

$$p(i,j) = (p(i,0) + p(i,1))\left(\frac{I_{i,2}}{I_{i,0}}\right)^{j-1} \quad (9)$$

The lemma follows by using Equations 9 and 3, and equating the sum of the steady-state probabilities to 1. □

Now, we express the steady state probabilities $I_{i,0}$, and $I_{i,2}$, in terms of $p(i-1,0)$ as follows.

$$\begin{aligned} I_{i,0} &= \frac{1}{4}(1 - p(i-1,0))^2 + (p(i-1,0))^2 \\ &\quad + 2.\frac{1}{2}p(i-1,0)(1 - p(i-1,0)) && (10)\\ I_{i,2} &= \frac{1}{4}(1 - p(i-1,0))^2 && (11) \end{aligned}$$

Equation 10 holds because, in steady state, if switch s at level i receives no packets in a specific time step from the two switches s' and s'' at level $i-1$ connected to s, one of the following events happened.

- Both switches s' and s'' send a packet in the previous time step, but neither packet is sent to s. Note that a switch sends a packet in a time step if its buffer is non-empty. Further, events concerning switches s' and s'' are independent, because s' and s'' contain traffic generated in non-intersecting sub-butterflies of the network. Hence, the probabilities of these events can be multiplied together.
- Both switches s' and s'' are empty in the previous time step. Hence, neither switch sends a packet.
- Exactly one of the two switches s' and s'' is non-empty in the previous time step, but no packet is sent to s.

Equation 11 can be justified in a similar fashion.

Lemma 2.3.2 *The following recurrence relation holds. For $i \geq 1$,*

$$p(i,0) = \frac{p(i-1,0)(1 + p(i-1,0))^{2q}}{(1 + p(i-1,0))^{2q} - (1 - p(i-1,0))^{2q}}$$

and $p(0,0) = 1 - \lambda$.

Proof: The base case of the recurrence follows from the fact that a switch at level 0, i.e., an input, is empty in a time step if and only if no packet arrives in that time step. From our traffic assumptions, the probability that no packet arrives at an input at a given time step is $1 - \lambda$. To obtain the recurrence, for $i \geq 1$, we use Equations 10 and 11 to substitute for $I_{i,0}$ and $I_{i,2}$ respectively in the expression for $p(i,0)$ in Lemma 2.3.1. □

2.4 Solving the recurrence relation

To solve the non-linear recurrence in Lemma 2.3.2, we need the following theorem due to de Bruijn [dB58, Koc88, RMDL96].

Theorem 2.4.1 *Let a_i be a sequence of real numbers satisfying*

$$a_{i+1} = a_i - ca_i^{q+1} + O(a_i^{q+2})$$

where $q > 0$, $c > 0$, and $\lim_{i\to\infty} a_i = 0$. Then,

$$a_i = \left(cqi + \frac{1}{a_0^q} + o(qi)\right)^{-\frac{1}{q}}$$

We solve the non-linear recurrence in Lemma 2.3.2 as follows.

Theorem 2.4.2 *For $q > 0$, $i \geq 0$, and $0 < \lambda \leq 1$,*

$$p(i,0) = 1 - \left(\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)i}{2} + \Lambda^{2q-1} + o(qi)\right)^{\frac{1}{2q-1}}\right)^{-1},$$

where $\Lambda = (2-\lambda)/\lambda$.

Sketch of proof: Let $m_i = \frac{1-p(i,0)}{1+p(i,0)}$, $m_0 = 1/\Lambda$. Clearly,

$$\lim_{i\to\infty} m_i = \lim_{i\to\infty} \frac{1 - p(i,0)}{1 + p(i,0)} = \frac{1-1}{1+1} = 0$$

Writing the recurrence in Lemma 2.3.2 in terms of m_i and simplifying, we get

$$m_i = m_{i-1} - \frac{1}{2}m_{i-1}^{2q} + O(m_{i-1}^{2q+1})$$

Using Theorem 2.4.1, we solve the recurrence for m_i to obtain the following.

$$m_i = \left(\frac{(2q-1)i}{2} + \Lambda^{2q-1} + o(qi) \right)^{-\frac{1}{2q-1}}$$

Substituting $m_i = \frac{1-p(i,0)}{1+p(i,0)}$ in the above equation yields the required expression for $p(i,0)$. See Appendix A for a detailed proof of this theorem. □

2.5 Expected throughput and packet loss rate

We use the expression for $p(i,0)$ to derive expressions for the expected throughput and packet loss rate.

Theorem 2.5.1 *The expected throughput of routing packets using a non-predictive queuing protocol on an N-input butterfly network, with buffer-size $q > 0$, and packet arrival rate λ, $0 < \lambda \leq 1$, is*

$$\frac{N}{\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1} + o(q \log N)\right)^{\frac{1}{2q-1}}},$$

and the packet loss rate is

$$1 - \frac{1}{\lambda\left(\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1} + o(q \log N)\right)^{\frac{1}{2q-1}}\right)},$$

where $\Lambda = (2-\lambda)/\lambda$.

Proof: Each output node sends a packet at each time step with probability $1 - p(\log N, 0)$, in the steady state. Therefore, expected throughput equals $N(1 - p(\log N, 0))$, which can be evaluated using the expression for $p(i,0)$ in Theorem 2.4.2. Further, since the expected number of packets arriving at the inputs in one time step is λN, the probability that a packet reaches its destination is expected throughput divided by λN. Therefore, the packet loss rate is $1 - \frac{\text{expected throughput}}{\lambda N}$. □

2.6 Expected delay

We will now analyze the expected delay of a packet for the FIFO protocol. As defined earlier, expected delay is the expected value of the total number of time steps that a packet waits in the buffers of the switches along its path, before reaching its output destination, at steady state. Note that the above expectation is taken only over packets that do not get dropped. Let the switch-delay of a packet at switch s be the number of steps the packet waits in the buffer at s. Since switches at the same level of the butterfly have the same statistical properties, we define D_i to be the expected switch-delay of a packet in a switch at level i, at steady state. Further, let L_i denote the expected number of packets stored in the buffer of a switch at level i, at steady state.

Lemma 2.6.1 *The expected delay of a packet equals*

$$\sum_i D_i = \sum_i \left(\frac{L_i}{1 - p(i,0)} - 1 \right)$$

Proof: The expected delay of a packet equals the sum of the expected switch-delays at each switch in the path of the packet. Since a packet uses exactly one switch from each level of the butterfly, expected delay is $\sum_i D_i$. Using Little's law[BG87], we know that the expected time spent by a packet at a switch s in level i equals the expected number of packets buffered at s divided by the rate at which packets enter (or exit) the buffer at s. Since packets enter the buffer at switch s at the rate of $(1 - p(i,0))$, the expected time spent by a packet at s is $L_i/(1 - p(i,0))$. Since each packet takes one time unit for transmission, the expected switch-delay D_i at s is $L_i/(1 - p(i,0)) - 1$. The lemma follows. □

Lemma 2.6.2 *For $q > 1$, the expected number of packets, L_i, in a switch at level i is*

$$1 - p(i,0) + \frac{1}{4}(1 - p(i-1,0))^2 + O((1 - p(i-1,0))^3)$$

Proof: See Appendix B.

Theorem 2.6.3 *For the FIFO protocol, the expected delay of a packet is zero when $q = 1$, and is*

$$\Theta\left(\frac{1}{q-1} \left(\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1} \right)^{1-\frac{1}{2q-1}} - \Lambda^{2q-2} \right) \right),$$

when $q > 1$, where N is the number of inputs in the butterfly, q is the buffer-size, $0 < \lambda \leq 1$ is the packet arrival rate, and $\Lambda = (2-\lambda)/\lambda$.

Sketch of proof: It is clear that there is no delay when $q = 1$. For $q > 1$, we express expected delay in terms of $p(i,0)$, $0 \leq i \leq \log N$, by substituting the expression for L_i in Lemma 2.6.2 in the expression for the expected delay in Lemma 2.6.1. Now, using Theorem 2.4.2, we substitute for $p(i,0)$ and bound the sum using integrals to obtain an expression for the expected delay. See Appendix C for a detailed proof of this theorem. □

3 Experimental Results

We simulated the butterfly routing algorithm utilizing the FIFO queueing protocol at each switch, for different network sizes (N), packet arrival rates (λ), and buffer sizes (q). We ran the simulations for sufficiently long periods of time, until the measurements for the expected throughput, packet loss rate, and expected delay showed only small variations. These simulation results bolster the predictions made by our analysis. Let the expected normalized throughput be the expected throughput divided by λN. We plot expected normalized throughput, packet loss rate, and expected delay as functions of the buffer size (q), the number of levels in the butterfly ($\log N + 1$), and the packet arrival rate λ. Unless stated otherwise, the network was heavily-loaded with $\lambda = 1$. Figures 2 and 3 show the variation of expected normalized throughput and expected delay with the buffer size q. As predicted by the theory, the throughput increases very rapidly for small values of q, and then tapers off for larger values of q. The expected delay also tapers off for larger values of q, though it is less pronounced.

Figures 4 and 5 show the variation of the expected normalized throughput and expected delay with the number of

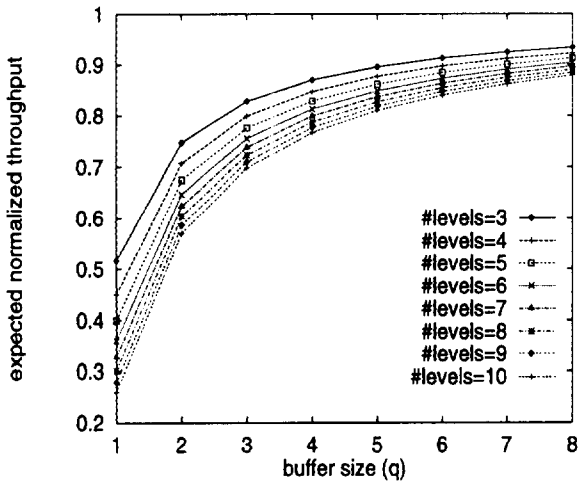

Figure 2: Expected normalized throughput vs buffer size (q) for different values of the network size (N). Note that #levels in the butterfly equals $\log N + 1$.

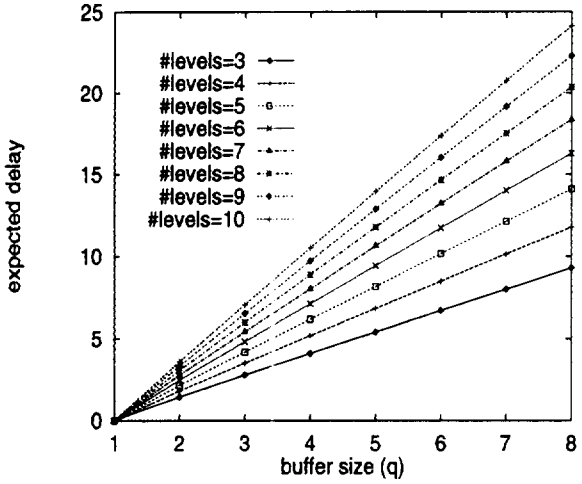

Figure 3: Expected delay vs buffer size (q) for different values of the network size (N).

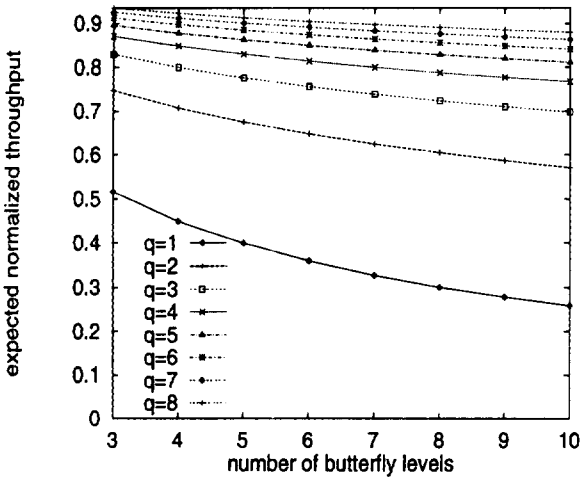

Figure 4: Expected normalized throughput vs the number of butterfly levels (log N + 1) for different values of the buffer size (q).

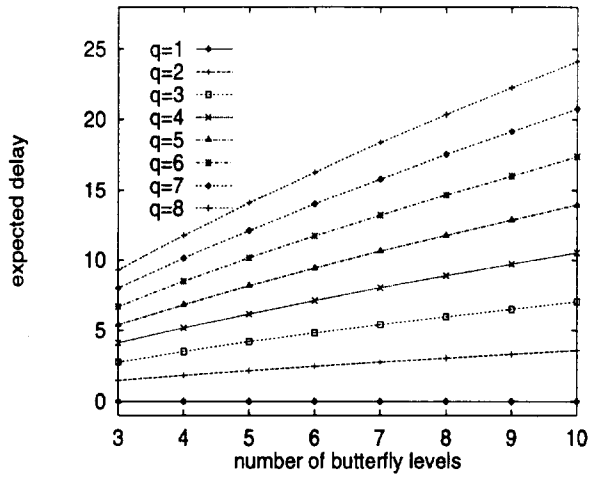

Figure 5: Expected delay vs the number of butterfly levels (log N + 1) for different values of the buffer size (q).

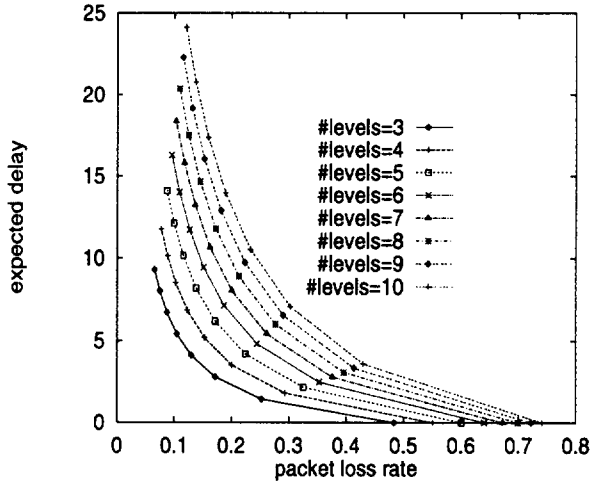

Figure 6: Tradeoff between expected delay and packet loss rate for different values of the network size (N).

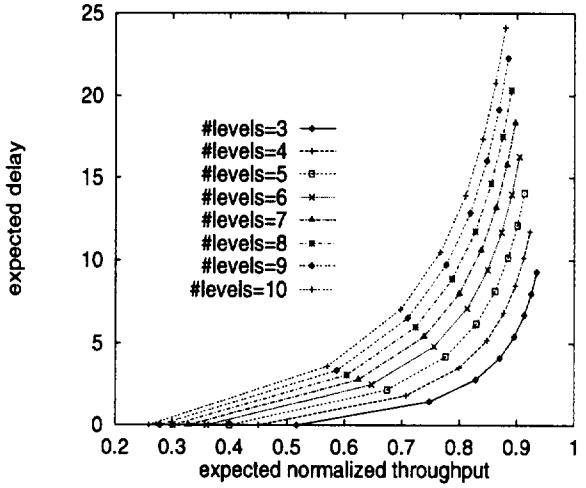

Figure 7: Variation of the expected delay with expected normalized throughput for different values of the network size (N).

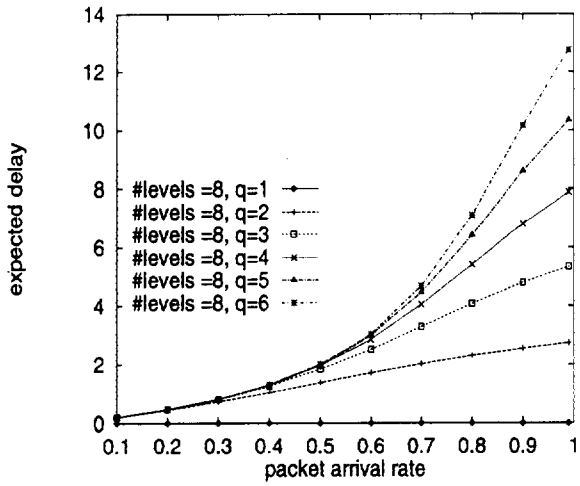

Figure 8: Variation of expected delay with the packet arrival rate (λ) for different values of the buffer-size(q).

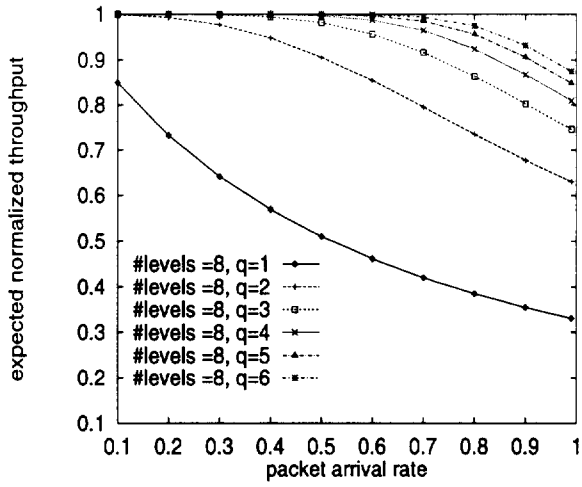

Figure 9: Variation of expected normalized throughput with the packet arrival rate (λ) for different values of the buffer-size (q).

levels in the butterfly ($\log N + 1$). The throughput goes to zero as the number of levels increase. As predicted by theory, the expected delay grows sub-linearly with the number of levels $\log N + 1$, when q is a constant. However, due to the range of N and q that we were able to simulate, the deviation of the expected delay plot from a straight line is small.

Figure 6 shows the tradeoff between expected delay and packet loss rate. Different regions of this tradeoff are suitable for different types of traffic, depending on their relative sensitivity to packet loss and delay. For instance, delay-insensitive data traffic belong in the initial portion of the curve which corresponds to larger buffer size. Delay-sensitive real-time multimedia traffic fall in the latter potion of the curve which corresponds to smaller buffer sizes. Figure 7 shows a similar variation between expected delay and expected normalized throughput.

Finally, Figures 8 and 9 show how the expected delay increases and the expected normalized throughput decreases with an increase in the packet arrival rate λ.

4 Concluding Remarks

In some applications the *variance* of the delay, known as jitter, is more important than the expected delay itself. It is worthwhile studying higher-order moments, particularly variance, of the throughput and delay. It is of interest to extend our study to other commonly-used switching networks, such as the crossbar. An $N \times N$ crossbar consists of N inputs, N outputs, and N^2 2×2 switches (See Figure 10). Each packet takes an one-bend path from its input to its output, i.e., each packet traverses the row of its input, and turns into the column of its output. Each switch has an unit-sized buffer for storing packets that route along the row, and a buffer of size q for storing packets that route along the column (see Figure 10). Analogous to our butterfly analysis, we can define a markov chain and write recurrence relations for the probabilities. Unfortunately, currently, we can solve the recurrence in closed form only for the simple case when $q = 1$ to obtain the expected throughput to be $N(1-(1-(\lambda/N))^N)$. and the packet loss rate to be $1-\lambda^{-1}(1-(1-(\lambda/N))^N)$. A more general solution for the crossbar is open for future research.

Also of interest is the study of simple routing algorithms that are capable of routing packets belonging to more than one class of traffic. Packets in different traffic classes could have very different requirements on the routing performance of the network. A number of issues, like buffer allocation and simple queuing protocols for multiple traffic classes, are open for future theoretical investigation.

References

[Ale82] R. Aleliunas. Randomized parallel communication. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 60–72, August 1982.

[Bat68] K. Batcher. Sorting networks and their applications. In *Proceedings of AFIPS Spring Joint Computing Conference*, volume 32, pages 307–314, 1968.

[BG87] D. Bertsekas and R. Gallagher. *Data Networks*. Prentice-Hall Inc., 1987.

[BFU96] A. Z. Broder, A.M. Frieze, and E. Upfal. A general approach to dynamic packet routing with bounded buffers. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, October 1996.

[CadHSV96] Robert Cypher, Friedhelm Meyer auf der Heide, Christian Scheideler, and Berthold Vöcking. Universal algorithms for store-and-forward and wormhole routing. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 356–365, May 1996.

[CLR92] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1992.

[dB58] N.G. de Bruijn. *Asymptotic Methods in Analysis*. Interscience Publishers, 1958.

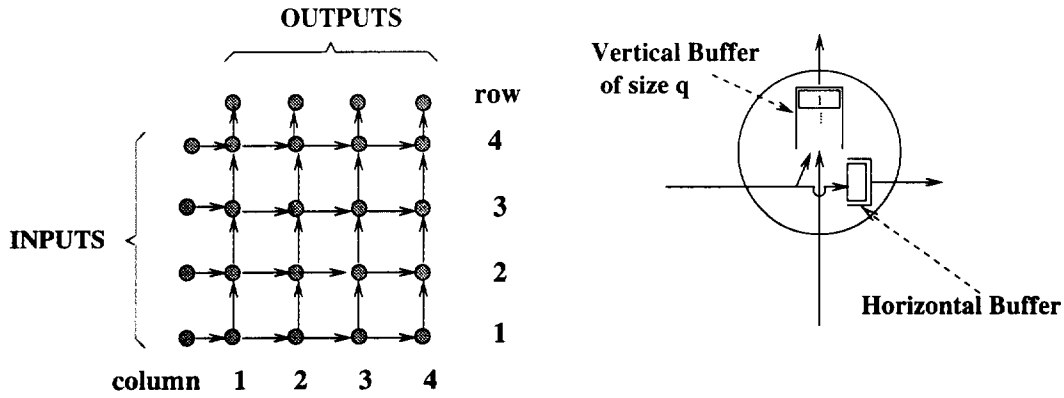

Figure 10: An 4-input crossbar network, and the internal structure of a switch.

[Gal96] R. G. Gallagher. *Discrete Stochastic Processes.* Kluwer Academic Publishers, 1996.

[Got87] A. Gottlieb. An overview of the NYU Ultracomputer project. In J. J. Dongarra, editor, *Experimental Parallel Computing Architectures*, pages 25–95. North-Holland, 1987.

[GP93] S. Gianatti and A. Pattavina. Analytical models for the performance evaluation of banyan networks with shared queueing. In *Proceedings of the IEEE GLOBECOM*, volume 2, pages 860–866, 1993.

[GP94] S. Gianatti and A. Pattavina. Performance analysis of ATM banyan networks with shared queueing - Part I: Random offered traffic, Part II: Correlated/unbalanced offered traffic. *IEEE/ACM Transactions on Networking*, 2(4):398–424, August 1994.

[Int91] Intel Corporation. Paragon XP/S Product Overview, 1991.

[Jen83] Y. Jenq. Performance analysis of a packet switch based on a single-buffered banyan network. *IEEE Journal on Selected Areas in Communications*, 1:1014–1021, December 1983.

[Koc88] R Koch. Increasing the size of a network by a constant factor can increase performance by more than a constant factor. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 221–230, October 1988.

[KS83] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Trans. on Computers*, C-32(12):1091–1098, December 1983.

[Lei92] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, San Mateo, Calif., 1992.

[Lei92a] F.T. Leighton. Methods for message routing in parallel machines. *In Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 77–96, May 1992.

[LMRR94] F. T. Leighton, B. M. Maggs, S. B. Rao, and A. G. Ranade. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, July 1994.

[MadHV95] F. Meyer auf der Heide and B. Vöcking. A packet routing protocol for arbitrary networks. In *Proceedings of the 12th STACS*, pages 291–302, 1995.

[Mer91] A. Merchant. A markov chain approximation for the analysis of banyan networks. In *Proceedings of the ACM SIGMETRICS conference on Measurement and Modeling of Computer Systems*, pages 60–67, 1991.

[MS92] B. M. Maggs and R. K. Sitaraman. Simple algorithms for routing on butterfly networks with bounded queues. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 150–161, May 1992.

[NMT+91] T. Nakata, S. Matsushita, N. Tanabe, N. Kajihara, H. Onozuka, Y. Asano, and N. Koike. Parallel programming on Cenju: A multiprocessor system for modular circuit simulation. *NEC Research & Development*, 32(3):421–429, July 1991.

[Pat81] J.H. Patel. Performance of processor-memory interconnections for multiprocessors. *IEEE Trans. on Computers*, C-30(10), 1981.

[PBG+87] G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfelder, K. P. McAuliffe, E. A. Melton, V. A. Norton, and J. Weiss. An introduction to the IBM Research Parallel Processor Prototype (RP3). In J. J. Dongarra, editor, *Experimental Parallel Computing Architectures*, pages 123–140. North-Holland, 1987.

[Pip84] N. Pippenger. Parallel communication with limited buffers. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, pages 127–136, October 1984.

[Ran91] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.

[RCG94] R. Rooholamini, V. Cherkassky, and M. Garver. Finding the right ATM switch for the market. *IEEE Computer*, April 1994.

[RMDL96] R. Rehrmann, B. Monien, R. Diekmann, and R. Lueling. On the communication through-

put of buffered multistage interconnection networks. *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, July 1996.

[SS89] T. Szymanski and S. Shaikh. Markov chain analysis of packet-switched banyans with arbitrary switch sizes, queue sizes, link multiplicities and speedups. In *Proceedings of the IEEE INFOCOM '89*, pages 960–971, April 1989.

[ST91] G. D. Stamoulis and J. N. Tsitsiklis. The efficiency of greedy routing in hypercubes and butterflies. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 248–259, July 1991.

[TRH91] T.H. Theimer, E.P. Rathgeb, and M.N. Huber. Performance analysis of buffered banyan networks. *IEEE Trans. on Communications*, 39(2):269–277, February 1991.

[Tur93] J.S. Turner. Queueing analysis of buffered switching networks. *IEEE Trans. on Communications*, pages 412–420, February 1993.

[Upf84] E. Upfal. Efficient schemes for parallel communication. *Journal of the ACM*, 31(3):507–517, July 1984.

[Val82] L. G. Valiant. A scheme for fast parallel communication. *SIAM J. Comput.*, 11(2):350–361, May 1982.

[VB81] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 263–277, May 1981.

[VS96] Berthold Vöcking and Christian Scheideler. Universal continuous routing strategies. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 142–151, June 1996.

[YLL90] H. Yoon, K.Y. Lee, and M.T. Liu. Performance analysis of multibuffered packet-switching networks in multiprocessor systems. *IEEE Trans. on Computers*, 39(3):319–327, March 1990.

A Detailed proof of Theorem 2.4.2

Theorem 2.4.2 is restated below for convenience.

Theorem A.0.4 *For $q > 0$, $i \geq 0$, and $0 < \lambda \leq 1$,*

$$p(i,0) = 1 - \left(\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)i}{2} + \Lambda^{2q-1} + o(qi)\right)^{\frac{1}{2q-1}}\right)^{-1},$$

where $\Lambda = (2-\lambda)/\lambda$.

Proof: Define m_i to be $\frac{1-p(i,0)}{1+p(i,0)}$. Since $m_i = \frac{2}{1+p(i,0)} - 1$, $p(i,0) = \frac{2}{1+m_i} - 1$. We write the recurrence in Lemma 2.3.2 in terms of m_i and simplify as follows.

$$\begin{aligned}
\frac{2}{1+m_i} - 1 &= \left(\frac{2}{1+m_{i-1}} - 1\right)\left(\frac{1}{1-m_{i-1}^{2q}}\right) \\
\frac{2}{1+m_i} &= \frac{2}{1+m_{i-1}} + \left(\frac{2}{1+m_{i-1}} - 1\right)\left(\frac{m_{i-1}^{2q}}{1-m_{i-1}^{2q}}\right) \\
&= \frac{2}{1+m_{i-1}}\left(1 + \frac{1-m_{i-1}}{2}\left(\frac{m_{i-1}^{2q}}{1-m_{i-1}^{2q}}\right)\right) \\
&= \frac{2}{1+m_{i-1}}\left(\frac{2 - m_{i-1}^{2q}(1+m_{i-1})}{2(1-m_{i-1}^{2q})}\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
1+m_i &= \frac{(1+m_{i-1})\left(1-m_{i-1}^{2q}\right)}{1 - \frac{1}{2}m_{i-1}^{2q}(1+m_{i-1})} \\
&= (1+m_{i-1})\left(1-m_{i-1}^{2q}\right) \cdot \\
&\quad \left(1 + \frac{1}{2}m_{i-1}^{2q} + O(m_{i-1}^{2q+1})\right) \\
&= 1 + m_{i-1} - \frac{1}{2}m_{i-1}^{2q} + O(m_{i-1}^{2q+1})
\end{aligned}$$

Therefore,

$$m_i = m_{i-1} - \frac{1}{2}m_{i-1}^{2q} + O(m_{i-1}^{2q+1}) \tag{12}$$

Since $p(0,0) = 1-\lambda$, $m_0 = 1/\Lambda$
Clearly,

$$\lim_{i\to\infty} m_i = \lim_{i\to\infty} \frac{1-p(i,0)}{1+p(i,0)} = \frac{1-1}{1+1} = 0$$

Using Theorem 2.4.1, we get

$$m_i = \left(\frac{(2q-1)i}{2} + \frac{1}{m_0^{2q-1}} + o(qi)\right)^{-\frac{1}{2q-1}}$$

Since $p(i,0) = \frac{2}{1+m_i} - 1$, and $m_0 = 1/\Lambda$, we get

$$p(i,0) = 1 - \left(\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)i}{2} + \Lambda^{2q-1} + o(qi)\right)^{\frac{1}{2q-1}}\right)^{-1}$$

□

B Detailed proof of Lemma 2.6.2

Lemma 2.6.2 is restated below.

Lemma B.0.5 *For $q > 1$, the expected number of packets, L_i, in a switch at level i is*

$$1 - p(i,0) + \frac{1}{4}(1-p(i-1,0))^2 + O((1-p(i-1,0))^3)$$

Proof: From Equation 10 and Equation 11 we get

$$\begin{aligned}
\frac{I_{i,2}}{I_{i,0}} &= \left(\frac{1-p(i-1,0)}{1+p(i-1,0)}\right)^2 \\
&= \left(\frac{1-p(i-1,0)}{2-(1-p(i-1,0))}\right)^2 \\
&= \frac{(1-p(i-1,0))^2}{4}\left(\frac{1}{1-0.5(1-p(i-1,0))}\right)^2 \\
&= \frac{(1-p(i-1,0))^2}{4} \cdot \\
&\quad \left(1 + (1-p(i-1,0)) + O((1-p(i-1,0))^2)\right)
\end{aligned}$$

So we have

$$\frac{I_{i,2}}{I_{i,0}} = \frac{(1-p(i-1,0))^2}{4} + \frac{(1-p(i-1,0))^3}{4} + O((1-p(i-1,0))^4) \quad (13)$$

Therefore for $j \geq 2$,

$$\left(\frac{I_{i,2}}{I_{i,0}}\right)^j = O((1-p(i-1,0))^4)$$

The expected number of packets buffered at switch at level i is given by

$$L_i = \sum_{j=1}^{q} jp(i,j)$$

$$= p(i,1) + \sum_{j=2}^{q} jp(i,j)$$

From Equation 9, we get

$$\begin{aligned} L_i &= -p(i,0) + (p(i,0)+p(i,1)) + \\ &\quad (p(i,0)+p(i,1)) \sum_{j=2}^{q} j \left(\frac{I_{i,2}}{I_{i,0}}\right)^{j-1} \\ &= -p(i,0) + \frac{\sum_{j=1}^{q} j \left(\frac{I_{i,2}}{I_{i,0}}\right)^{j-1}}{\sum_{j=1}^{q} \left(\frac{I_{i,2}}{I_{i,0}}\right)^{j-1}} \\ &= -p(i,0) + 1 + \frac{\sum_{j=1}^{q-1} j \left(\frac{I_{i,2}}{I_{i,0}}\right)^{j}}{\sum_{j=1}^{q} \left(\frac{I_{i,2}}{I_{i,0}}\right)^{j-1}} \\ &= -p(i,0) + 1 + \frac{I_{i,2}}{I_{i,0}} + O\left(\left(\frac{I_{i,2}}{I_{i,0}}\right)^2\right) \end{aligned}$$

Using Equation 13 we have,

$$\begin{aligned} L_i &= -p(i,0) + 1 + \frac{1}{4}(1-p(i-1,0))^2 + \\ &\quad \frac{1}{4}(1-p(i-1,0))^3 + O((1-p(i-1,0))^4) \\ &= 1 - p(i,0) + \frac{1}{4}(1-p(i-1,0))^2 + \\ &\quad O((1-p(i-1,0))^3) \end{aligned}$$

□

C Detailed proof of Theorem 2.6.3

Theorem 2.6.3 is restated below for convenience.

Theorem C.0.6 *For the FIFO protocol, the expected delay of a packet is zero when $q = 1$, and is*

$$\Theta\left(\frac{1}{q-1}\left(\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1}\right)^{1-\frac{1}{2q-1}} - \Lambda^{2q-2}\right)\right),$$

when $q > 1$, where N is the number of inputs in the butterfly, q is the buffer size, $0 < \lambda \leq 1$ is the packet arrival rate, and $\Lambda = (2-\lambda)/\lambda$.

Proof: Let D denote the expected delay of a packet. From Lemma 2.6.1, we have

$$D = \sum_i D_i = \sum_i \left(\frac{L_i}{1-p(i,0)} - 1\right)$$

From Lemma 2.6.2 we have

$$\begin{aligned} D_i &= \frac{L_i}{1-p(i,0)} - 1 \\ &= \frac{\frac{1}{4}(1-p(i-1,0))^2 + O((1-p(i-1,0))^3)}{1-p(i,0)} \\ &= \frac{1}{4}\left(\frac{1-p(i-1,0)}{1-p(i,0)}\right)(1-p(i-1,0)) \\ &\quad + \left(\frac{1-p(i-1,0)}{1-p(i,0)}\right) O((1-p(i-1,0))^2) \end{aligned}$$

From the expression for $(1-p(i,0))$ in Theorem 2.4.2 it follows that

$$1 \leq \frac{1-p(i-1,0)}{1-p(i,0)} \leq 2$$

So we have

$$\frac{1}{4}(1-p(i-1,0)) \leq D_i \leq \frac{1}{2}(1-p(i-1,0)) + O((1-p(i-1,0))^2)$$

Therefore

$$\begin{aligned} D &\geq \frac{1}{4}\sum_{i=1}^{\log N}(1-p(i-1,0)) \\ D &\leq \frac{1}{2}\sum_{i=1}^{\log N}(1-p(i-1,0)) + \\ &\quad O\left(\sum_{i=1}^{\log N}(1-p(i-1,0))^2\right) \end{aligned} \quad (14)$$

So,

$$D = \Theta\left(\sum_{i=1}^{\log N}(1-p(i-1,0))\right)$$

From Theorem 2.4.2

$$\sum_{i=1}^{\log N} (1-p(i-1,0)) =$$

$$\sum_{i=1}^{\log N} \left(\frac{1}{2} + \frac{1}{2}\left(\frac{(2q-1)(i-1)}{2} + \Lambda^{2q-1} + o(qi)\right)^{\frac{1}{2q-1}}\right)^{-1} \quad (15)$$

We integrate the RHS of Equation 15 with respect to i to derive the following bounds[CLR92].

$$\begin{aligned} &\frac{C}{q-1}\left(\left(\frac{(2q-1)(\log N-1)}{2} + \Lambda^{2q-1}\right)^{1-\frac{1}{2q-1}} - \Lambda^{2q-2}\right) \\ \leq\ &\sum_{i=1}^{\log N}(1-p(i-1,0)) \\ \leq\ &\frac{2}{q-1}\left(\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1}\right)^{1-\frac{1}{2q-1}} - \Lambda^{2q-2}\right) \end{aligned}$$

where C is a constant. This implies that $D =$

$$\Theta\left(\frac{1}{q-1}\left(\left(\frac{(2q-1)\log N}{2} + \Lambda^{2q-1}\right)^{1-\frac{1}{2q-1}} - \Lambda^{2q-2}\right)\right)$$

□