An Improved Analytical Model for Wormhole Routed Networks with Application to Butterfly Fat-Trees

Ronald I. Greenberg
Mathematical and Computer Sciences
Loyola University
6525 North Sheridan Road
Chicago, IL 60626
rig@math.luc.edu

Lee Guan
Electrical Engineering
University of Maryland
College Park, MD 20742
leeguan@eng.umd.edu

Abstract

A performance model for wormhole routed interconnection networks is presented and applied to the butterfly fat-tree network. Experimental results agree very closely over a wide range of load rate. Novel aspects of the model, leading to accurate and simple performance predictions, include (1) use of multiple-server queues, and (2) a general method of correcting queuing results based on Poisson arrivals to apply to wormhole routing. These ideas can also be applied to other networks.

keywords: interconnection network, wormhole routing, latency, throughput, butterfly fat-tree

1 Introduction

Many recent multicomputers have adopted *wormhole* [3] routing techniques to reduce the communication latency for fine-grained parallel programs. Several performance models have been presented for wormhole routing. Dally [2] focused on k-ary n-cube networks, and the other works have been primarily geared towards improving accuracy or simplicity of some aspects of the prior models. In particular, Draper and Ghosh [4] present a simple model that is particularly accurate for binary hypercubes. The common feature of these models is the use of results from queuing theory in an iterative fashion working backwards from message destination to message source.

None of the prior works, however, lead directly to a suitable model for the network of particular interest in this paper, the butterfly fat-tree. Fat-trees constitute an interesting class of networks due to their area-universality properties (e.g., [5, 6, 9]) and their influence on the design of actual parallel computers [1, 10].

There are several ways that modeling the butterfly fat-tree differs from modeling k-ary n-cubes. First, the butterfly fat-tree is not node-symmetric, so it does not suffice to analyze the traffic situation at a single node. Still, the butterfly fat-tree has a very regular structure, and deadlock never results when messages are routed over shortest paths. Deadlock avoidance schemes for k-ary n-cubes produce some complication in the sense that they create asymmetry among different links in the network, but they actually lead to a major simplification by fixing a specific path for any message with a given source and destination. In the butterfly fat-tree, messages often have a choice among two outgoing links from a node, necessitating the use of multiple-server queuing models.

In this paper, we first present an improved general model for analyzing wormhole routed networks in Section 2. In Section 3, we apply the model to the butterfly fat-tree network and compare to simulation results. Concluding remarks are included in Section 4.

2 A Wormhole Routing Model

This section presents a general approach to analyzing the performance of wormhole routed interconnection networks. The measures we seek to compute are average latency and throughput.

The model presented in this section is based on the following assumptions, common to other analyses: (1) Arrivals at each source node are Poissonian, and destinations are uniformly random. (2) Worms have a fixed length longer than the diameter of the network. (3) Contentions at incoming links to a node are resolved according to First-Come First-Served (FCFS) scheduling. (4) Messages arriving at destinations are immediately consumed at the rate of one flit per time step, i.e., no blocking is encountered at destinations.

2.1 Average Latency

An interconnection network consists of processing elements (PE) and routing elements (RE). In direct networks(e.g., k-ary n cubes) a node consists of both a PE and an RE. In indirect networks (e.g., tree-based networks where processors are placed at leaves) processing elements and routing elements are separate

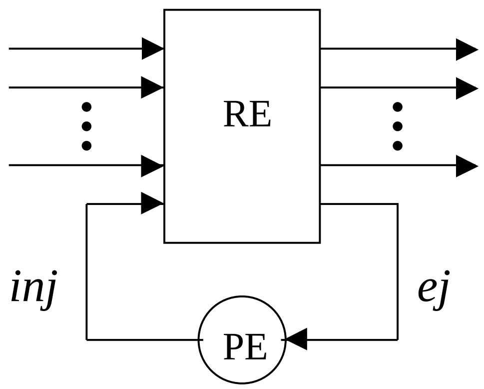

Figure 1: A general routing model. A network consists of processing elements (PE) and routing elements (RE). A PE is attached to an RE through an injecting channel and an ejecting channel.

nodes. Figure 1 shows a general routing model that can be used to represent both direct and indirect networks. a PE is always attached to an RE through an injecting channel and an ejecting channel. An RE, however, may or may not have a PE attached to it.

When a message is generated at a processing node j, it encounters the following latencies: 1) A waiting time $W_{inj,j}$ for the injecting channel. This waiting time doesn't depend on the routing scheme (store-and-forward or wormhole), and can be determined as long as the behavior of message arrival rates and the service time for the injecting channel are known. Under the Poisson arrival assumption the waiting time can be resolved using the M/G/1 model. 2) A service time $x_{inj,j}$ at the injecting channel. This is the time from the moment the first flit of the message is accepted by the injecting channel to the moment the last flit of the message has left the injecting channel. In wormhole routing, the flits of a message spread over many links on the message's path. When the head flit is blocked, the other flits of the worm are blocked in place. Under the long-worm assumption, the service time at the injecting channel includes the waiting times due to blocking at all subsequent channels. 3) An additional time to traverse the rest of the channels on the message's path. Under the assumptions that the length of the worm is longer than the diameter of the network and that there is no blocking at the destination, when the tail of the message has left the injecting channel, the head of the message must have arrived at the destination; the rest of the message will be received one flit per clock step. Therefore, it will take another $D-1$ clock steps for the entire message to be received at the destination, where D is the length of the path.

From the above analysis we can write the the latency L_j for the message injected at node j as

$$L_j = W_{inj,j} + x_{inj,j} + D - 1 \ . \tag{1}$$

Averaging over all processing nodes (and the probability distribution of message generation), the average latency $\overline{L}$ for the entire network is then

$$\overline{L} = \frac{1}{N}\sum_j \overline{L}_j = \frac{1}{N}\sum_j \left(\overline{W}_{inj,j} + \overline{x}_{inj,j}\right) + \overline{D} - 1 \ , \tag{2}$$

where N if the number of processing nodes in the network and $\overline{D}$ is the average message distance.

The service time at the injecting channel $x_{inj,j}$ depends on the service time of the subsequent channels. More precisely, the service time of a channel is the sum of the waiting time and the service time encountered at the channel immediately following it. Service times are resolved in the reverse order of the channels traversed, from the last channel (ejecting channel) backwards to the injecting channel.

2.2 Waiting Times and Service Times

At any RE, messages from an incoming channel i may be routed to outgoing channels denoted by $j = 0$, 1, etc. The service time for the incoming channel depends on the service times and waiting times at all possible outgoing channels. Denote the probability that a message from incoming channel i is routed to outgoing channel j by $R_{i|j}$, the service time for incoming channel i can then be expressed as

$$x_i^{\text{in}} = \sum_j (x_j + w_{i|j}) \cdot R_{i|j} \ , \tag{3}$$

where x_j is the service time for the outgoing channel j and $w_{i|j}$ is the waiting time for outgoing channel j of messages from incoming channel i. The above equation states that the service time at a channel depends on the service time of the subsequent channel and the waiting time for the subsequent channel.

The mean waiting times $w_{i|j}$ is caused by contention for the outgoing channel j. When a message is blocked, it must wait for the message that is holding the outgoing channel to be fully serviced. (A worm in service can not be preempted since only the head flit contains routing information.) This motivates us to take advantage of well-known queuing models that have been employed to analyze store-and-forward routing.

When an outgoing channel is treated as single server, results from the M/G/1 model [8] can be used:

$$\overline{W}_{M/G/1} = \frac{\rho\overline{x}(1 + C_b^2)}{2(1-\rho)} \ , \tag{4}$$

where $\rho = \lambda\overline{x}$ is the server utilization, λ is the rate of message arrivals destined for the outgoing channel, $\overline{x}$

is the mean service time, and $C_b^2 = \frac{\sigma_b^2}{\overline{x}^2}$, where σ_b^2 is the variance of service time distribution. In light of arguments of Draper and Ghosh [4, p. 206], we adopt the following approximation:

$$C_b^2 = \frac{(\overline{x} - s/f)^2}{\overline{x}^2} \;, \tag{5}$$

where s and f are the length of the message and the flit width respectively, so that s/f is the length of the message in flits.

Substituting for ρ and C_b^2 in Equation 4, we have

$$\overline{W}_{M/G/1} = \frac{\lambda \overline{x}^2}{2(1-\lambda\overline{x})} \cdot \left[1 + \frac{(\overline{x} - s/f)^2}{\overline{x}^2}\right] \;. \tag{6}$$

In certain situations, multiple outgoing links from a switch must be treated as one multi-server channel. This is usually due to the existence of redundant paths to increase bandwidth. Multiple-server systems with general service time distributions (M/G/m queues) are more complicated than M/G/1 queues, but we make use of an approximation of Hokstad [7] that leads to:

$$\overline{W}_{M/G/2} \;=\; \frac{\lambda^2\overline{x}^3}{2(4-\lambda^2\overline{x}^2)}(1 + C_b^2) \;. \tag{7}$$

We again use Equation 5 to approximate C_b^2, yielding:

$$\overline{W}_{M/G/2} = \frac{\lambda^2\overline{x}^3}{2(4-\lambda^2\overline{x}^2)} \cdot \left[1 + \frac{(\overline{x} - s/f)^2}{\overline{x}^2}\right] \;. \tag{8}$$

But the M/G/m model assumes independent arrivals at the inputs of a switch, all of which may block one another, which is not accurate for wormhole routing. Once an input link is occupied by a worm, there can be no more arrivals on that link until the first worm is fully serviced. Thus, once a worm arrives on a link, it only needs to wait for worms from other incoming links. Therefore, to use the M/G/m waiting time result, we multiply by a blocking probability $P_{i|j}$:

$$w_{i|j} = P_{i|j} W_j \;, \tag{9}$$

where $P_{i|j}$ should reflect the probability that m messages deemed to be in service by the M/G/m model actually emanate from m distinct incoming links other than link i. A simple approximation is

$$P_{i|j} = 1 - m\frac{\lambda_i^{\text{in}}}{\lambda_j} R_{i|j} \;, \tag{10}$$

where λ_i^{in} is the total message rate on incoming channel i, λ_j is the total message rate on outgoing channel j, and the number of servers, m, is less than the number of incoming links. When $m = 1$, the expression is exact, i.e., $P_{i|j}$ is 1 minus the probability that an arbitrary message destined for output j is from input i. For larger m, we approximately account for the probability that any of the servers holds a message from input i; if all the arrival rates on incoming links are modest relative to the rate on outgoing channel j, the probabilities of multiple arrivals from the same input in the M/G/m model are small enough to safely ignore.

By combining Equations 3, 9 and 10 we obtain the service time for messages on incoming channel i:

$$x_i^{\text{in}} = \sum_j \left[x_j + (1 - m\frac{\lambda_i^{\text{in}}}{\lambda_j} R_{i|j}) W_j\right] R_{i|j} \;. \tag{11}$$

Equation 11 is used together with Equations 6 and 8 to iteratively resolve the service times for all channels. Average latency is then determined from Equation 2.

2.3 Throughput

Throughput is another important metric of network performance. Through the above analysis, waiting times at each link on a route can be obtained, from which we can determine the service time at the source. To find the throughput, the source service time is set equal to the reciprocal of the source arrival rate [2]. At this operating point messages are being offered as fast as the network can deliver them; the network saturates and can accept no more traffic.

3 Analysis of Butterfly Fat-trees

Section 2 presented a general performance model for wormhole routed networks. We now apply the general model to the butterfly fat-tree. We start with a brief description of the network. We then determine the message rates, service time and waiting time to each channel. Latency and throughput are then resolved and compared with results from empirical simulations.

3.1 The Butterfly Fat-Tree

We use the butterfly fat-tree with N processors as shown in Figure 2. Each node is labeled by a pair of indices (l, a), where l represents the level of the node in the network and a represents the address of the node in that level. The level of a node is its distance from the leaves. At the lowest level ($l = 0$) are the N processors with addresses 0 to $N-1$. Each switch $S(l, a)$ has six ports: $parent_0$, $parent_1$, $child_0$, $child_1$, $child_2$ and $child_3$. The processors are connected to $N/4$ switches at the 1st level such that processor $P(0, a)$ is connected to the $child_{a \bmod 4}$ of switch $S(1, \lfloor a/4 \rfloor)$. At the l-th level (for $l = 1$ to $\log_4 N$) there are $N/2^{l+1}$ switches. The connections of a switch are determined by the switch's address as follows: $parent_0$ of $S(l, a)$ is connected to $child_i$ of $S(l+1, \lfloor \frac{a}{2^{l+1}} \rfloor \cdot 2^l + a \bmod 2^l)$, and $parent_1$ of $S(l, a)$ is connected to $child_i$ of $S(l+1, \lfloor \frac{a}{2^{l+1}} \rfloor \cdot 2^l + (a + 2^{l-1}) \bmod 2^l)$, where $i = \lfloor \frac{a \bmod 2^{l+1}}{2^{l-1}} \rfloor$.

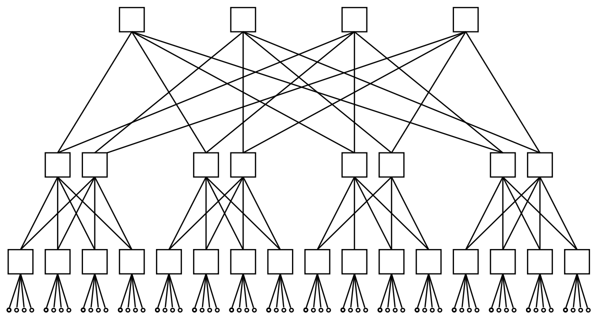

Figure 2: Butterfly Fat-Tree With 64 Processors

There is more than one shortest path between a pair of leaves in the butterfly fat-tree. More precisely, a message can take any of the two up links from a switch, if the destination is not in the subtree rooted at the switch. (There is no redundancy for down links.) When a worm needs to go up, it selects an up-link randomly, if that link is blocked, it tries the other, and if both are blocked, it waits.

3.2 Message Arrival Rates

To obtain the message arrival rates to each link, we assume the mean departure rate of a node is equal to the mean arrival rate provided that the network is not saturated [4]. Note that in a butterfly fat-tree, links that are at the same level and run in the same direction (up or down) are symmetrical, hence there is no need to distinguish among them. We can label the links and their arrival rates by a pair of indices $\langle i, j\rangle$ where i is the starting level of the link and j is the ending level of the link in the network, $0 \leq i, j \leq n$ with $n = \log_4 N$.

Assume each processor injects messages into the network at a rate of λ_0. Under steady state conditions, we have $\lambda_{0,1} = \lambda_{1,0} = \lambda_0$ for links between the processors ($l = 0$) and the first level switches ($l = 1$).

Now consider links between switches between level l and $l+1$ ($1 \leq l < n$). Since there are $N = 4^n$ processors in the system, a message may have $4^n - 1$ destinations, of which $4^l - 1$ can be reached without going up from level l. Then, the probability that a message goes up from level l, denoted $P_l^{\uparrow}$, is

$$P_l^{\uparrow} = \frac{4^n - 4^l}{4^n - 1}\ , \tag{12}$$

and the probability that a message goes down is

$$P_l^{\downarrow} = 1 - P_l^{\uparrow}\ . \tag{13}$$

$P_l^{\downarrow}$ is used later when computing service times.

The total message rates going up from level l to level $l+1$ is $P_l^{\uparrow} 4^n \lambda_0$. There are $\frac{4^n}{2^l}$ links between level l and $l+1$. The message rate to each channel going from level l to level $l+1$ is $\lambda_{l,l+1} = P_l^{\uparrow} 4^n \lambda_0/(4^n/2^l) = \lambda_0 P_l^{\uparrow} 2^l$. The message rate going downward from level $l+1$ to level l equals that going up from level l to level $l+1$ due to symmetry. In summary, we have

$$\begin{aligned} \lambda_{l,l+1} &= \lambda_0 \frac{4^n - 4^l}{4^n - 1} 2^l && (14)\\ \lambda_{l+1,l} &= \lambda_{l,l+1}\ . && (15) \end{aligned}$$

3.3 Waiting and Service Times

Since a message is received by the destination processor one flit at a clock as soon as the head flit has reached the destination, the service time for links from a level 1 switch to a processor is deterministic, i.e, the length of a worm:

$$x_{1,0} = s/f\ . \tag{16}$$

The mean waiting time $\overline{W}_{1,0}$ is determined using Equation 6, i.e.,

$$\overline{W}_{1,0} = \overline{W}_{M/G/1}(\lambda_{1,0}, x_{1,0})\ . \tag{17}$$

For any other down-going channels from level $l+1$ to level l ($1 \leq l < n$), there are 4 possible outgoing channels (the 4 children), each with the same probability (1/4). The mean service time $\overline{x}_{l+1,l}$ is determined using Equation 11:

$$\overline{x}_{l+1,l} = \overline{x}_{l,l-1} + \left(1 - \frac{1}{4}\frac{\lambda_{l+1,l}}{\lambda_{l,l-1}}\right)\overline{W}_{l,l-1}\ . \tag{18}$$

The mean waiting time $\overline{W}_{l+1,l}$ is determined using $\overline{x}_{l+1,l}$:

$$\overline{W}_{l+1,l} = \overline{W}_{M/G/1}(\lambda_{l+1,l}, \overline{x}_{l+1,l})\ . \tag{19}$$

Now consider up-going channels, starting with channel $\langle n-1, n\rangle$. There are only 3 possible outgoing channels (siblings) after traversing channel $\langle n-1, n\rangle$, each with the same probability (1/3). Therefore

$$\begin{aligned} \overline{x}_{n-1,n} &= \overline{x}_{n,n-1} + \left(1 - \frac{\lambda_{n-1,n}}{\lambda_{n,n-1}}\frac{1}{3}\right)\overline{W}_{n,n-1} \\ &= \overline{x}_{n,n-1} + \frac{2}{3}\overline{W}_{n,n-1}\ . && (20) \end{aligned}$$

The mean waiting time $\overline{W}_{n-1,n}$ is determined using the two-server model (Equation 8), i.e.,

$$\overline{W}_{n-1,n} = \overline{W}_{M/G/2}(\lambda_{n-1,n}, \overline{x}_{n-1,n})\ . \tag{21}$$

Correction:
Insert: 2

For any other up-going channels from level $l-1$ to level l ($1 \leq l < n-1$), a message may go upward from level l with probability $P_l^{\uparrow}$ or go downward with probability $P_l^{\downarrow}$. In the case that the message goes upward, there are two redundant up-going channels that are treated as one two-server channel. In the case that the message goes downward, there are three possible outgoing channels (siblings), each with the same probability (1/3). Therefore the mean service time is

$$\overline{x}_{l-1,l} = \left[\overline{x}_{l,l+1} + \left(1 - \frac{\lambda_{l-1,l}}{\lambda_{l,l+1}} P_l^{\uparrow}\right) \overline{W}_{l,l+1}\right] P_l^{\uparrow} + \left[\overline{x}_{l,l-1} + \left(1 - \frac{P_l^{\downarrow}}{3}\right) \overline{W}_{l,l-1}\right] P_l^{\downarrow} . \quad (22)$$

The mean waiting time $\overline{W}_{l-1,l}$ is determined using the two-server model (Equation 8)

Correction: Insert: 2

$$\overline{W}_{l-1,l} = \overline{W}_{M/G/2}(\lambda_{l-1,l}, \overline{x}_{l-1,l}) , \quad (23)$$

except for $l = 1$. Channel $\langle 0, 1\rangle$ is from processor to first level switch with no redundant channel; therefore, the single server model should be applied, i.e.,

$$\overline{W}_{0,1} = \overline{W}_{M/G/1}(\lambda_{0,1}, \overline{x}_{0,1}) , \quad (24)$$

3.4 Average Latency

Now we can use Equation 2 to compute the average latency. For the butterfly fat-tree, $x_{inj,j} = x_{0,1}$ and $W_{inj,j} = W_{0,1}$. Since all processors are equivalent due to symmetry, averaging over injecting channels is unnecessary. Therefore the latency is determined as

$$\overline{L} = \overline{W}_{0,1} + \overline{x}_{0,1} + (\overline{D} - 1). \quad (25)$$

3.5 Throughput

Maximum throughput is computed by setting the source service time to the reciprocal of the source arrival rate, i.e.,

$$\overline{x}_{0,1} = \frac{1}{\lambda_0} . \quad (26)$$

Source service time $\overline{x}_{0,1}$ increases as arrival rate increases, while $\frac{1}{\lambda_0}$ is a monotonically decreasing function of λ_0. Graphically, if $\overline{x}_{0,1}$ and $\frac{1}{\lambda_0}$ are plotted against arrival rate, the maximum throughput is the arrival rate at the intersection of the two curves. In practice we let source arrival rate increase (starting at a small value) until the above equation is satisfied.

3.6 Experimental Validation

The performance model for the butterfly fat-tree was validated through comparisons with simulations. Fixed length messages are used for the simulation. Latencies from the model and simulation were compared for networks with up to 1024 processing nodes. Messages of 16, 32 and 64 flits in length are studied. Figure 3 shows the result of the comparisons for average latencies with 1024 processors. The model produced accurate predictions on latency and throughput for all cases under study.

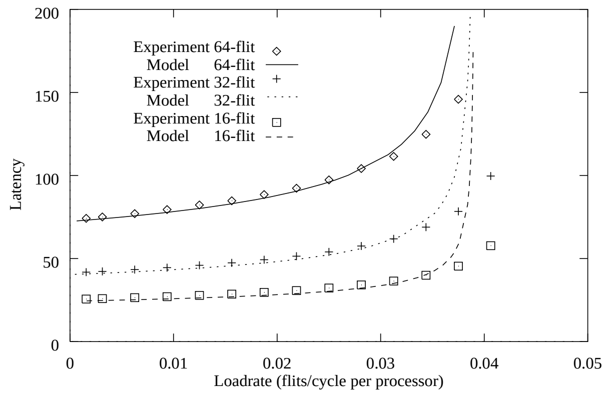

Figure 3: Comparisons of latency and throughput between model and simulation for 1024-processor

4 Conclusion

We have presented a general performance model for wormhole routed networks and applied it to the fat-tree network. Included in the process was the use of two-server queuing models, and the framework can be extended for networks that require queuing models with more than two servers

Average latency and maximum throughput for the butterfly fat-tree network were analyzed using the the model presented and validated through comparison with simulation results. The model was simple but produced very accurate predictions of performance.

References

[1] J. Beecroft et al. Meiko CS-2 interconnect Elan-Elite design. *Parallel Computing*, 20:1627–1638, Nov. 1994.

[2] W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Computers*, 39(6):775–785, June 1990.

[3] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Computers*, C-36(5):547–553, May 1987.

[4] J. T. Draper and J. Ghosh. A comprehensive analytical model for wormhole routing in multicomputer systems. *J. Par. Dist. Comp.*, 23:202–214, 1994.

[5] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In S. Micali, editor, *Randomness and Computation.* Volume 5 of *Advances in Computing Research*, pages 345–374. JAI Press, 1989.

[6] R. I. Greenberg and H.-C. Oh. Universal wormhole routing. *IEEE Tr. Par. Dist. Sys.*, 8(3):254–262, 1997.

[7] P. Hokstad. Approximations for the M/G/m queue. *Operations Research*, 26(3):510–523, 1978.

[8] L. Kleinrock. *Queueing Systems*, vol. I. J. Wiley, 1975.

[9] F. T. Leighton et al. Randomized routing and sorting on fixed-connection networks. *J. Alg*, 17(1):157–205.

[10] C. E. Leiserson et al. The network architecture of the connection machine CM-5. In *Proc. 4th ACM Symp. Parallel Alg. and Arch.*, pages 272–285, 1992.