Buffer and Delay Bounds in High Radix Interconnection Networks

Arjun Singh and William J Dally
Computer Systems Laboratory, Stanford University
{arjuns, billd}@cva.stanford.edu

***Abstract*— We apply recent results in queueing theory to propose a methodology for bounding the buffer depth and packet delay in high radix interconnection networks. While most work in interconnection networks has been focused on the throughput and average latency in such systems, few studies have been done providing statistical guarantees for buffer depth and packet delays. These parameters are key in the design and performance of a network. We present a methodology for calculating such bounds for a practical high radix network and through extensive simulations show its effectiveness for both bursty and non-bursty injection traffic. Our results suggest that modest speedups and buffer depths enable reliable networks without flow control to be constructed.**

I. Introduction

High radix Interconnection networks are widely used in supercomputer networks (Merrimac Streaming Supercomputer [3]) and for I/O interconnect (Infiniband Switch fabric [8]). Most research for such interconnection networks focuses on analyzing the throughput and average packet latency of the system. However, little work has been done towards bounding the occupancy of the buffers in the network or the delay incurred by a packet.

The buffer occupancy and the delay distributions play a key role in the design and performance of the network. Bounding the number of packets in a buffer in the network is valuable for network administration and buffer resource allocation. A statistical bound on the packet delay is essential for guaranteeing Quality of Service for delivery of packets.

The analysis of a network of deterministic service queues is a known hard problem. The difficulty in analysis primarily arises from the fact that the traffic processes do not retain their statistical properties as they traverse such a network of queues. Given a myriad of sophisticated techniques developed for analyzing a single deterministic service queue, there has been some recent work that attempts to decompose the network based on large deviations techniques [11], [10]. Most of these results are applicable in the convergence regimes, such as in the case when there are several flows passing through a queue, called the *many sources asymptotic* regime. Using the many-sources-aysmptotic, Wischik [11], [10] shows that the distribution of a traffic flow is preserved by passage through a queue with deterministic service, in the limit where the number of independent input flows to that queue increases and the service rate and buffer size increase in proportion. More recently, Eun & Shroff [4], [5] use similar convergence results to significantly simplify the analysis of such a network. In particular, they show that, if internal nodes in a network are capable of serving many flows, we can remove these nodes from consideration and the queueing behavior of other network nodes remains largely the same.

Manuscript submitted: 22 Oct. 2004. Manuscript accepted: 10 Dec. 2004. Final manuscript received: 15 Dec. 2004.

In this paper, we use the aforementioned convergence results to bound the queue depth and packet delay in high radix interconnection networks. Our simulations show that such convergence results start to kick in when the radix (degree) of the network is as small as $16 - 32$. We also use our bounds to propose a routing mechanism with almost negligible flow control overhead.

II. Many Sources Queueing Regime

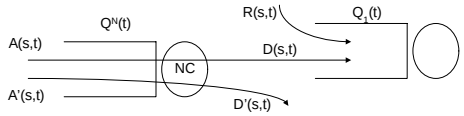

Fig. 1. An upstream queue feeding some flows into a downstream queue

In order to see how network analysis can be simplified in the many sources regime, consider the set up of two queues in Figure 1. Let there be N flows going into the upstream queue. The subset of these flows that go on into the downstream queue have a combined arrival process given by $A(s,t)$ which is the total number of packets arriving in the time interval (s,t). The remaining set of flows have arrival process $A'(s,t)$. Let the service capacity of the upstream queue be NC, i.e. the service per flow is C packets per time step. The departing flows from the first queue going into the downstream queue have a departure process given by $D(s,t)$. The downstream queue can also receive more cross traffic given by $R(s,t)$. Let the queue depth of the downstream queue at time t be $Q_1(t)$ while that for the upstream queue be $Q^N(t)$. In order to find the Prob$(Q_1 > x)$, we can simplify this scenario into 1 queue.

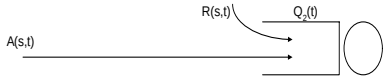

Fig. 2. Simplified scenario of the set up of two queues

Figure 2 shows a simplified scenario of Figure 1. In this figure, the effect of the upstream queue on flows $A(s,t)$ is

ignored. Let the queue depth of the downstream queue for this scenario be $Q_2(t)$. The authors of [5], [4] prove that as $N \to \infty$, Prob$(Q_1 > x)$ converges to Prob$(Q_2 > x)$ and that the speed of this convergence is exponentially fast. Hence, with a modest number of multiplexed sources, the convergence results start to hold.

III. Application to High Radix Fat Trees

In the rest of this paper, we shall apply the many sources regime results to analyzing buffer depth in a popular high radix topology — Clos [2] or fat tree networks [7]. The high radix switch queues are an appropriate application for the many sources asymptotic results. As the radix (and the number of sources) increases, the statistical properties of the flows get preserved as they traverse the network. Our simulations show that the convergence results hold for a radix as small as $16-32$.

In our experimental set up, we have simulated a specific kind of fat tree — a k-ary n tree network. A k-ary n tree network has n levels of internal switches and a total of k^n leaf nodes that can communicate with each other using these switches. There are nk^{n-1} internal switches which have $2k$ incoming ports and $2k$ outgoing ports. The internal switches have buffers inside where packets are stored and serviced to their appropriate output port. Figure 3 shows a diagram for a 2-ary 4 tree. In our simulations, we will concentrate on high radix trees where k is of the order of 16 or 32.

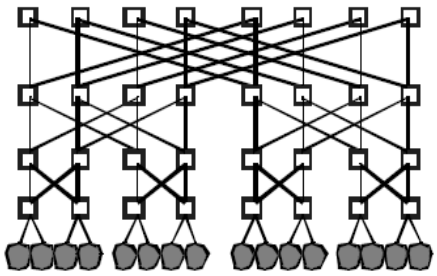

Fig. 3. A 2 ary 4 tree

Load balancing on such a fat tree is easily accomplished using Random Root Routing (RRR) in which each packet is routed to a randomly chosen root switch and then down to the desired destination resulting in a total of $2n-1$ hops for every packet. A more sophisticated approach — called Random Common Ancestor Routing (RCAR) — is to route up to a (randomly chosen) common ancestor of the source and destination leaf node and then down to the destination node. In our analysis, we focus on RRR as it enables us to treat all traffic patterns as two phases of uniformly random traffic, thus making the analysis more tractable. The analysis for RRR is also a conservative analysis for RCAR as the latter has strictly fewer packets using the resources in the upper levels of the network.

IV. Results

In this section, we first study the impact of the radix on the buffer occupancy distribution in the queues at each hop of the fat tree network. Our approach is to first increase the radix k in a k ary 3 tree network while keeping the per channel bandwidth constant. We then plot the Complementary Cumulative Distribution Functions (CCDFs) of the queue depth occupancy at each of the 5 hops of the network. We perform this experiment for non-bursty Bernoulli Uniform and bursty injection traffic. For both these injection processes, the many sources convergence results start to manifest themselves at reasonably small values of k. Using these values of k, we can analytically calculate the exact CCDF from queueing theory for each of the 5 queues, thus giving us buffer depth bounds. We then use the per hop buffer depth bound to approximate the end-to-end delay of a packet through the network by convolving the per-hop distributions obtained. This approximation gives very accurate results especially at high injection loads.

A. Increasing the radix k

In the very first experiment, we inject packets at each source according to a non-bursty Bernoulli Uniform process. Each source injects a packet with a probability p at every time step. We increase the radix k of each switch and measure the queue depth at each of the 5 hops of a k-ary 3 tree network. Figure 4 shows that the CCDFs of the queues are quite divergent for a low radix ($k = 2$) but tend to converge to almost identical plots as k is increased to 16. This is because, for a high enough radix, the statistical properties of the flows are preserved as they traverse the network. Hence, for the non-bursty injection process, a radix of 16 is a reasonable working parameter to use the convergence results.

B. Analytically obtaining the CCDF

We now describe our analytical approach for obtaining the CCDFs of the queue depths at each hop. For the 16 ary 3 tree case with non-bursty injection, it suffices to obtain the CCDF for the first hop as the other hops behave almost identically.

Let A be the random variable that represents the total traffic at each time step to the queue at the first hop. Thus, $A = \sum_{i=1}^{k} X_i$, where $X_1, X_2, \ldots X_k$ are Bernoulli IID random variables corresponding to the k sources such that if p_i is the probability that the source i will send a packet along this queue, then $X_i = 1$ with probability p_i. Also, the mean arrival rate is given by $E[A] = \sum_{i=1}^{k} p_i$.

Let the service capacity of the queue be $C = 1$ packet per time step. If the queue is stable i.e., $E[A] < C$, we can find the Probability Generating Function (PGF)[1] of the queue size $Q(z)$ using the formula derived in [6].

$$Q(z) = P(Q = 0)\frac{(z-1)A(z)}{(z - A(z))} \tag{1}$$

In our case

$$A(z) = \prod_{i=0}^{k}((1-p_i) + p_i z) \text{ and } P(Q = 0) = 1 - E[A] \tag{2}$$

[1]The PGF, G, of a random variable, X, is given by $G(z) = E(z^X) = \sum_{i=0}^{\infty} f(i)z^i$, where f is the probability mass function for X.

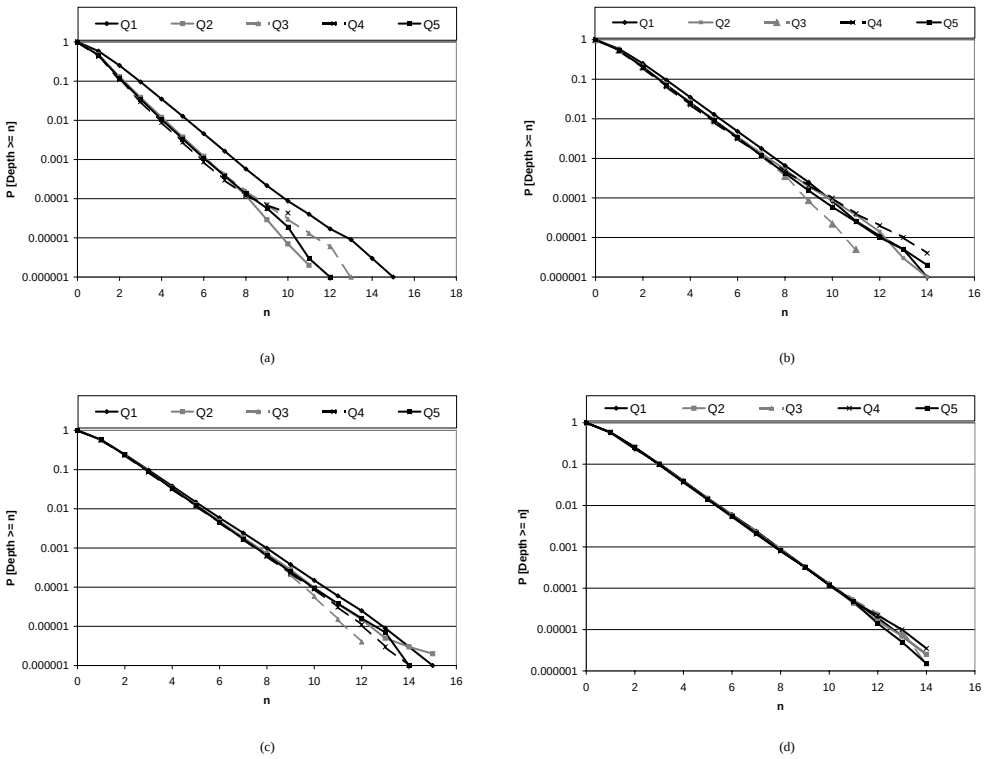

Fig. 4. Queue depth at each hop of a k-ary 3 tree for (a) k=2 (b) k=4 (c) k=8 and (d) k=16

The derivation for $P(Q = 0)$ is shown in [9]. Moreover, $p_i = p$ for all i since all sources inject traffic with the same probability. Using Equation 1 we can find out the queue size distribution in the following way :

$$P(Q = n) = \frac{1}{n!} \left[\left(\frac{d}{dz} \right)^n Q(z) \right]_{z=0} \quad (3)$$

Equation 3 can then be used to derive the CCDF for the queue depth at each hop of the 16 ary 3 tree network.

Figure 5 shows the analytically derived queue depth against the measured values obtained in the previous subsection. The queues at each hop are at a utilization of 60% i.e., $E[A]/C = 0.6$. The model matches almost exactly with the simulations. While the observed values were obtained through simulations run for 2 days, the theoretical CCDF could be derived in minutes using the Maple software [1] for solving (3). In doing so, we could quickly derive the buffer depth required for which the overflow probability would be of the order of 10^{-15}, something that was not tractable by simulations alone.

C. Approximating the delay CCDF

Having found the buffer depth distributions at each hop, our next aim is to approximate the end-to-end delay of packets traversing the network. The latency, T, incurred by a packet is the sum of two components, $T = H + D$, where H is the hop count and D is the queueing delay. For the 16 ary 3 tree case, $H = 5$. The random variable D is the queueing delay incurred due to buffering at each of the hops in the network. The per-hop delay in each queue is directly related to the occupancy of the queue and its service capacity. In order to find the distribution of D, we make the simplifying assumption that the

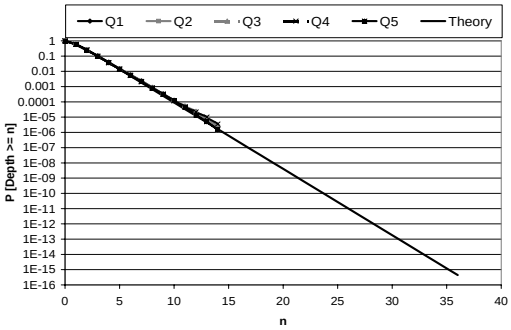

Fig. 5. Analytically derived queue depth against the measured queue depth at each hop for a 16 ary 3 tree at 0.6 load

per hop delays at each hop are independent of each other. This assumption has been shown to give accurate results in practice especially as the load is increased [12]. This is because, as the buffer load increases, the output of a queue (which is the input to the downstream queue) gets less correlated with the input.

Using this independence assumption, we can find the distribution of D by simply convolving the per-hop delay distributions calculated before, which is equivalent to simply taking the product of their PGFs. Finally, adding the constant H to D gives us the end-to-end delay distribution.

Figure 6 compares the analytical delay CCDF with the measured values for injection loads of 0.2, 0.6 and 0.8. As expected, the analytical and observed plots almost match for the highest load of 0.8. They also are reasonably good approximations for the lower loads.

It is to be noted that the radix at which reasonable con-

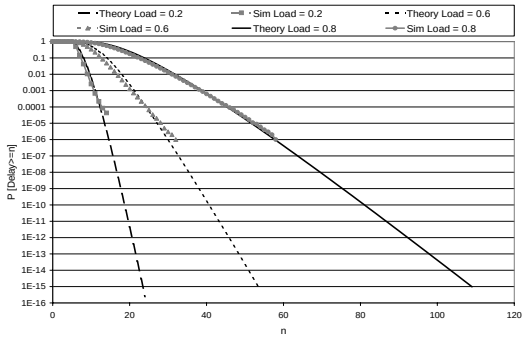

Fig. 6. End-to-end packet delay (theoretical and measured) for a 16 ary 3 tree at different injected loads

vergence holds depends on the size of the network and the injection sources. For instance, for bursty injection sources, a radix value of 32 shows convergence sufficient for the above analysis to give good results. The complete results are omitted for brevity and can be found in [9].

V. Discussion

An interesting fallout from deriving buffer depth bounds is that we can use them to make the flow control of the packets trivial. In order to not drop packets, a packet in the upstream node does not leave for the downstream node until it gets a *credit* signifying there is enough buffer space available there. If we were to send packets to the downstream node ignoring this flow control information, the packet would be dropped if there was no space in the downstream node. However, we can set our buffer depths to a reasonable size and make sure that for admissible traffic the probability of a drop due to buffer overflow is substantially lower than the probability of a drop due to a hard error. In order to do this, we must either police the injection process or provide a little internal speedup to the network.

Take for instance, the non-bursty injection process on a 16 ary 3 tree. We have successfully computed the buffer depth requirement for a particular injection load of 0.6 that has a very low probability of exceeding (of the order 10^{-15}). Repeating our calculations for different loads, we can study how the buffer depth requirements grow as the load is increased keeping the probability of overflow constant at 10^{-15}.

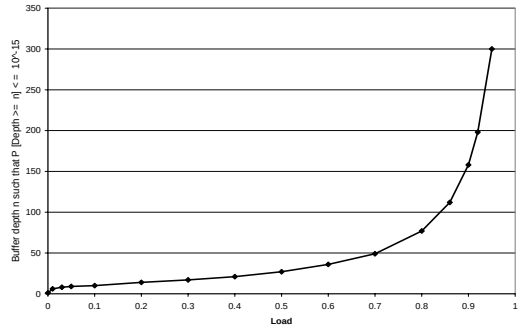

Fig. 7. Buffer depth requirement at various injection loads

Figure 7 shows that as the injected load reaches the saturation value of 1, the buffers start to grow without bound. However, at slightly less than this saturation value, say 0.9, a buffer size of 160 packets is required to ensure that the drop probability without flow control is less than 10^{-15}. Hence, either by policing the injection to make sure that the injection rate stays below 0.9 or by providing a small internal speedup of $1/0.9 = 1.11$, we can remove the flow control overhead from the routing process. A similar analysis can be carried out for other injection processes.

VI. Conclusion and Future work

In this paper, we have used recent convergence results in queueing theory to propose a methodology for bounding the buffer depth and packet delay in high radix interconnection networks. We have presented extensive simulations to show that the convergence results start to kick in for radix values as small as $16 - 32$. Finally, we use the bounds to propose a routing mechanism with negligible flow-control overhead by either policing the network at the source or introducing a small internal speedup in the network.

While delay bounds are essential for guaranteeing QOS, another useful application is in studying reordering of packets caused by load balancing algorithms. In order to deliver packets in order, packets are sorted at the destination according to their sequence numbers in reorder buffers (ROBs). The size of the ROB is also a function of the delay packets can incur over the different paths in the network leading to the same destination. Bounding the size of the ROB as a function of the delay distribution remains an open question.

References

[1] B. Char, K. Geddes, B. G.H. Gonnet, M. Monagan, and S. Watt., *Maple V Language Reference Manual*. Springer-Verlag New York, Inc., 1991.

[2] C. Clos, "A study of non-blocking switching networks," *The Bell System technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[3] W. J. Dally, P. Hanrahan, M. Erez, T. J. Knight, F. Labonte, J.-H. Ahn, N. Jayasena, U. J. Kapasi, A. Das, J. Gummaraju, and I. Buck, "Merrimac: Supercomputing with Streams," in *Proceedings of SuperComputing, SC'03*, Phoenix, Arizona, November 2003.

[4] D. Eun and N. Shroff, "Network decomposition in the many-sources regime," *Advances in Applied Probability*, vol. 36, no. 3, pp. 893–918, September 2004.

[5] D. Y. Eun and N. B. Shroff, "Simplification of network analysis in large-bandwidth systems," in *Proceedings of IEEE INFOCOM*, San Francisco, California, April 2003.

[6] L. Kleinrock, *Queueing Systems – Volume 1: Theory.* Wiley, New York, 1975, pp. 191–194, Eqn 5.85.

[7] C. Leiserson, "Fat-trees: Universal networks for hardware efficient supercomputing," *IEEE Transactions on Computer*, vol. C-34, no. 10, pp. 892–901, October 1985.

[8] G. Pfister, *An Introduction to the InfiniBand Arechitecture (http://www.infinibadta.org)*. IEEE Press, 2001.

[9] A. Singh and W. J. Dally, "Delay and buffer bounds in high radix interconnection networks," Concurrent VLSI Architecture (CVA) Technical Report (ftp://cva.stanford.edu/pub/publications/arjun_bounds.pdf), Oct 2004.

[10] D. Wischik, "Sample path large deviations for queues with many inputs," *Annals of Applied Probability*, vol. 11, no. 2, pp. 379–404, May 2001.

[11] D. J. Wischik, "The output of a switch, or, effective bandwidths for networks," *Queueing Syst. Theory Appl.*, vol. 32, no. 4, pp. 383–396, 1999.

[12] D. Yates, J. Kurose, D. Towsley, and M. Hluchy, "On per-session end-to-end delay distributions and the call admission problem for real-time applications with qos requirement," *Journal on High Speed Networks*, vol. 3, no. 4, pp. 429–458, 1994.