Throughput-Centric Routing Algorithm Design *

Brian Towles, William J. Dally, and Stephen Boyd
Department of Electrical Engineering
Stanford University
{btowles,billd}@cva.stanford.edu

ABSTRACT

The increasing application space of interconnection networks now encompasses several applications, such as packet routing and I/O interconnect, where the throughput of a routing algorithm, not just its locality, becomes an important performance metric. We show that the problem of designing oblivious routing algorithms that have high worst-case or average-case throughput can be cast as a linear program. Globally optimal solutions to these optimization problems can be efficiently found, yielding provably good oblivious routing algorithms.

Applying these techniques to k-ary 2-cube (tori) networks shows that previous routing algorithms sacrifice too much locality to achieve optimal worst-case throughput. This motivates the development of two new algorithms, IVAL and 2TURN, which improve locality to within 0.3% of optimal for an 8-ary 2-cube. Both algorithms have simple, deadlock-free implementations. Expanding the analysis of tori to average-case throughput reveals that there is a weak tradeoff between average-case and worst-case throughput. Specifically, both the IVAL and 2TURN algorithms developed for the worst-case also have good average-case throughput.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Routing and Layout*; C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures—*Interconnection architectures*

General Terms

Algorithms, Performance

Keywords

Oblivious routing, interconnection networks, multicommodity flows

*This work has been supported by an NSF Graduate Fellowship with supplement from Stanford University, under the MARCO Interconnect Focus Research Center, and by Cray, Inc. through DARPA subcontract CR02-C-0002 under U.S. government Prime Contract Number NBCHC020049.

SPAA'03, June 7–9, 2003, San Diego, California, USA.

1. INTRODUCTION

As interconnection networks are applied to throughput-centric applications, such as packet routing [1] and I/O interconnect [2], the throughput of a routing algorithm becomes an important design consideration. For example, in the packet routing application, little can be said about the incoming traffic patterns, and there is no path for backpressure to slow the flow of incoming packets. Therefore, the guaranteed throughput of the router is bounded by the worst-case throughput over all traffic patterns. In many other applications, such as multiprocessor interconnect, it is desirable to maximize the average-case throughput of the interconnection network.

In this paper, we show that the design of oblivious routing algorithms with optimal worst-case and average-case throughput can be cast as multicommodity flow (MCF) problems. Worst-case throughput can be reformulated in terms of a linear cost function and linear constraints and average-case throughput can be accurately approximated with a linear cost function, allowing both routing algorithm design problems to be expressed as linear programs. The resulting optimization problems can be solved efficiently and exactly, yielding globally optimal solutions.

To demonstrate the practical utility of optimization in designing throughput-centric routing algorithms, the techniques are applied to the problem of routing in k-ary 2-cube (tori) topologies. Previous routing algorithms and theoretical work have explored the tradeoff between locality and worst-case throughput of a routing algorithm, but using the framework developed in this paper, this tradeoff can be exactly quantified by solving a series of linear programs. An example of this tradeoff is shown in Figure 1 for an 8-ary 2-cube topology. Each point in this space represents an oblivious routing algorithm with a particular worst-case throughput and average path length (locality). The region of feasible algorithms is shown in gray and, because the linear programming approach is guaranteed to find globally optimal solutions, any algorithm outside this feasible region is provably unobtainable. Therefore, the set of routing algorithms denoted by the solid line at the edge of feasible region are Pareto optimal — for each Pareto optimal point, no oblivious routing algorithm can achieve both better worst-case throughput and better locality. Plotting the performance of existing routing algorithms in this tradeoff space reveals that these algorithms often lie far from the Pareto optimal.

Motivated by the lack of routing algorithms that have maximum worst-case throughput while achieving near-optimal locality, two new routing algorithms, IVAL and 2TURN, are presented that improve locality over Valiant's routing algorithm (VAL) [3] by 19.3% and 25.8%, respectively, with 2TURN only 0.3% from optimal. These new algorithms along with simple dimension-order routing (DOR) [4] cover the two extremes of the Pareto optimal curve and we show that *interpolated* routing algorithms can be created that lie

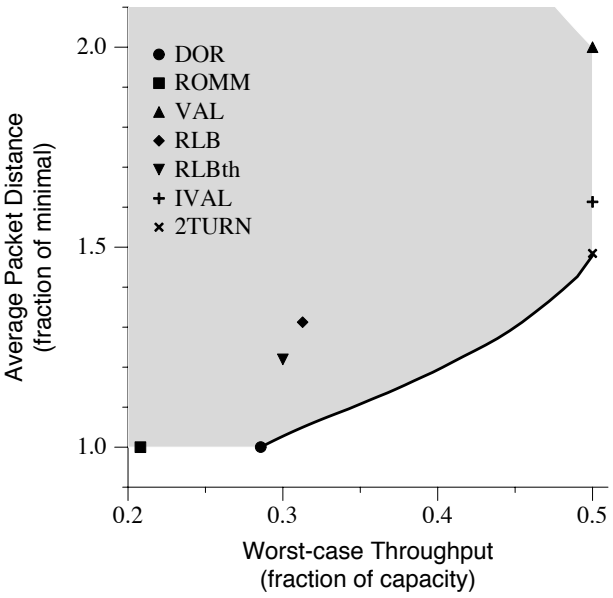

Figure 1: Tradeoff between average path length (vertical axis) and worst-case throughput (horizontal axis) on the 8-ary 2-cube. The optimal tradeoff is shown as a solid line and the region of feasible routing algorithms is gray. The performance of several routing algorithms are also shown as points in the tradeoff space.

between the extreme points. For example, by interpolating between 2TURN and DOR, it is possible to realize a routing algorithm that is within 10% of any particular Pareto optimal routing algorithm.

A similar analysis for average-case throughput again reveals that most existing algorithms lie far from the optimal tradeoff curve. However, the two new algorithms, IVAL and 2TURN, have near optimal average-case throughput with 2TURN within 10% of the maximum average-case throughput. This reveals an interesting result for k-ary 2-cubes: there is almost no tradeoff between worst-case throughput and average-case throughput. That is, an oblivious routing algorithm does not have to sacrifice average-case throughput to achieve good worst-case throughput and vice-versa. A new routing algorithm 2TURNA specifically optimized for average case performance is also introduced and lies only 8% from the maximum average-case throughput.

The remainder of this paper is organized as follows. Section 2 describes the formulation of routing algorithm design as an optimization problem along with several basic performance metrics. Throughput-centric cost functions are presented in Section 3 and the complexity of the resulting is optimization problems are addressed in Section 4. These ideas are then applied to torus networks in Section 5. Related work is discussed in Section 6 and Section 7 presents conclusions.

2. BACKGROUND

We first describe the definitions and assumptions for the networks and routing algorithms considered in this paper. Then, routing algorithm design is cast as a multicommodity flow problem. This formulation allows several basic metrics of network performance, such as throughput and latency, to be expressed.

2.1 Network Model and Definitions

The topology of each interconnection network in this paper is described by a directed graph (N, C) of N nodes (vertices) and C channels (edges). Nodes have unit injection and ejection bandwidth[1] and the channel bandwidths b_c are expressed as a multiple of the unit bandwidth.

Only *randomized, oblivious routing algorithms* are considered. For a randomized oblivious routing algorithm, the path a packet takes through the network is only a function of its source, destination, and a random variable — network state is not considered. This excludes the class of *adaptive routing algorithms* which can incorporate network state into their routing decisions. The affects of this restriction are discussed in Section 5.5 for k-ary 2-cube topologies.

To simplify the analysis and isolate our results from any particular flow-control or packet scheduling scheme, the ideal throughput of our network is determined completely by edge congestion. The system is assumed to be stable, if for every channel, the average number of packets that need to cross that channel per cycle is less than the bandwidth of the channel. If the number of packets that need to cross a channel meets or exceeds a channel's bandwidth, this channel is *saturated*. This is obviously an upper bound on the performance of any practical network. As discussed in [5], this upper bound is achievable with output queuing in each node router, large queues, a simple scheduling protocol, and a burstiness constraint on the incoming traffic process. Practical systems can typically reach 60-75% of this bound [6].

2.2 Oblivious Routing as a Flow Problem

Oblivious routing algorithm design can be captured as a multicommodity flow problem (MCF) [7][8]. First, an oblivious routing algorithm defines a probability distribution over the paths in the network for each source-destination pair — a total of N^2 distributions. To describe a routing algorithm R, we let $R(p)$ be the probability that a packet uses the path p. Then, any R represents a valid oblivious routing algorithm as long as

$$\sum_{p \in P_{s,d}} R(p) = 1 \quad \forall s, d \in N,$$
$$R(p) \geq 0 \quad \forall p \in P,$$

where P is the set of all paths and $P_{s,d}$ is the set of all paths between source s and destination d. Without eliminating any productive routing algorithms, we exclude paths that revisit channels. In the terminology of MCF's, this formulation creates a commodity for each of the N^2 source-destination pairs in the network.

Then, for a general cost function $C(R)$, the routing algorithm design problems considered in this paper can be expressed as an optimization problem:

$$\begin{array}{lll} \text{minimize} & C(R) & \\ \text{subject to} & \sum_{p \in P_{s,d}} R(p) = 1, & \forall s, d \in N \\ & R(p) \geq 0, & \forall p \in P. \end{array} \quad (1)$$

That is, find a valid, oblivious routing algorithm R that minimizes the cost function $C(R)$. In later sections, we will also add side constraints to this basic formulations to study the tradeoff between performance metrics.

[1]The constraint that each node have unit injection/ejection bandwidth can be relaxed by modeling a single logical node with several unit bandwidth nodes (e.g. a node with an injection/ejection bandwidth of two units can be replaced by two unit bandwidth nodes during optimization).

Throughout the paper, we are interested in both convex and linear cost functions in R. Informally, a convex program (CP) specifies a convex cost function to be minimized over a convex set. Because linear constraints define a convex set, the routing design problem described above is a CP when $C(R)$ is convex. Efficient techniques exist for finding globally optimal solutions to CP's [9][10]. When both the constraints and cost function are linear, the optimization problem becomes a linear program (LP). LP's are subclass of CP's and specialized LP solvers can handle larger problems than general convex optimization techniques. Thus, expressing a routing algorithm design problems as a linear program has a significant practical benefit.

2.3 Basic Metrics

Using the multicommodity flow formulation introduced in the previous section, several basic network measures can be defined. By our definition of stability, the maximum throughput a network can sustain under a given traffic pattern is determined by the channel loads — once the average load on a channel reaches the channel's capacity, that channel is saturated. The first channel to saturate determines the network's throughput.

The expected number of packets that cross a particular channel c per cycle, referred to as the load γ_c, is the sum of the loads contributed by each source-destination pair. In terms of both the traffic pattern Λ and the routing algorithm R

$$\gamma_c(R,\Lambda) = \sum_{s,d} \lambda_{s,d} \sum_{\substack{p:c\in p,\\ p\in P_{s,d}}} R(p). \tag{2}$$

Each entry of the traffic pattern $\lambda_{s,d}$ specifies the fraction of source s's injection bandwidth carrying traffic destined to node d. Since each node is defined to have unit injection and ejection bandwidth, the traffic matrix Λ is *doubly-stochastic*: the entries of each row and column of Λ sum to one.

Using the above formula for individual channel loads, the channel closest to its saturation point defines the normalized maximum channel load:

$$\gamma_{\max}(R,\Lambda) = \max_{c\in C}\left[\frac{\gamma_c(R,\Lambda)}{b_c}\right]. \tag{3}$$

This maximum channel load then defines the throughput $\Theta(R,\Lambda)$ of the network under the traffic pattern:

$$\Theta(R,\Lambda) = \gamma_{\max}(R,\Lambda)^{-1}. \tag{4}$$

So, for example, if $\Theta(R,\Lambda) = 0.5$, each node in the network can send and receive packets at up 50% of their maximum bandwidth under the traffic pattern Λ without saturating a channel.

It is important to note that both measures of channel load, γ_c and $\gamma_{\max}$, are convex in R. This is obviously the case for γ_c since it is linear in R. Because taking the point-wise maximum of a set of convex functions is convex [9], it follows that $\gamma_{\max}$ is also convex.

Although the paper focuses on the throughput properties of routing algorithms, another important metric is *locality*. The locality of an algorithm is expressed as the distance a packet travels on average, which largely determines end-to-end delay of packets at low load[2] and influences other performance metrics such as average power dissipated by a network. Increasing locality, or equivalently reducing the average packet distance, can also have a positive affect on average-case throughput (Section 5.4). We define average path length H_{avg} over all traffic patterns as

$$H_{\text{avg}}(R) = \frac{1}{|N|^2}\sum_{s,d}\sum_{p\in P_{s,d}} \mathbf{len}(p)R(p), \tag{5}$$

where $\mathbf{len}(p)$ is the length of path p through the network. It is often useful to normalize average path length and this can be accomplished by expressing the average path length as a fraction of the minimal path length, defined by the average length of the shortest paths between each source-destination pair.

3. COST FUNCTIONS

Given the basic routing algorithm design problem and metrics developed in the previous sections, we introduce three throughput-centric cost functions. The *capacity* of a network is presented first as a useful quantity for normalizing throughput results. Then our two new cost functions, worst-case and average-case throughput, are presented along with their linear formulations.

3.1 Capacity

Using the above definition of throughput, the *capacity* of a network is defined as its maximum throughput under the uniform traffic pattern U.[3] The capacity of a network can be found by solving the convex optimization problem

$$\begin{array}{lll} \text{minimize} & C_U(R) = \gamma_{\max}(R,U) & \\ \text{subject to} & \sum_{p\in P_{s,d}} R(p) = 1, & \forall s,d \in N \\ & R(p) \geq 0, & \forall p \in P. \end{array} \tag{6}$$

The result of the optimization is the minimum channel load under uniform traffic and a routing algorithm that realizes this load. Of course, the maximum throughput (capacity) is simply $\gamma_{\max}(R,U)^{-1}$. Expressing the throughput of other traffic patterns Λ as a fraction of capacity, $\Theta(R,\Lambda)/\Theta(R,U)$, allows a meaningful comparison of throughputs between different networks.

3.2 Worst-case Throughput

Worst-case throughput is an important metric for applications of interconnection networks which must maintain throughput levels under adversarial traffic. As described in this section, designing algorithms that have optimal worst-case throughput can be expressed as a linear program.

The worst-case throughput of a network is defined as its minimum throughput over all traffic patterns. As shown in [11], it is sufficient to limit the search for worst-case traffic patterns to permutations, so

$$C_{\text{wc}}(R) = \gamma_{\text{wc}}(R) = \max_{\pi\in\Pi}\gamma_{\max}(R,\pi), \tag{7}$$

where Π is the set of all permutation matrices. Then the worst-case throughput is $\Theta_{\text{wc}}(R) = \gamma_{\text{wc}}(R)^{-1}$. Maximum channel load is convex in R and because the point-wise maximum over several convex function preserves convexity, the worst-case channel load is also convex in R. Therefore, designing routing algorithms to maximize worst-case throughput can be formulated as a convex optimization problem as in the capacity case.

By introducing variables u, v, and w and several new constraints, the convex formulation of the worst-case routing problem can be converted to a linear program. The details of this reformulation are

[2]The other key factor in the latency of packets at low loads is serialization latency — the time required to inject a wide packet into a narrower channel.

[3]Under the uniform traffic pattern U, each source sends to each destination with equal probability ($u_{s,d} = 1/N$).

presented in the Appendix. The resulting linear program is:

$$\begin{array}{lll} \text{minimize} & w & \\ \text{subject to} & \sum_{p \in P_{s,d}} R(p) = 1, & \forall s,d \in N \\ & R(p) \geq 0, & \forall p \in P \\ & \sum_{\substack{p:c \in p, \\ p \in P_{s,d}}} R(p) \leq v_{d,c} - u_{s,c}, & \forall s,d \in N, c \in C \\ & \sum_{d \in N} v_{d,c} - \sum_{s \in N} u_{s,c} = b_c w, & \forall c \in C. \end{array} \tag{8}$$

The LP keeps the same basic form as the previous MCF problems and still includes the flow constraints.

The new variables and constraints are directly related to dual of the maximum-weight matching problem [7][12]. Because channel loading is linear, $\gamma_c(R,\pi)$ can be interpreted as the weight of a matching π in a bipartite graph whose edge weights are determined by R [11]. Each channel c has dual variables $u_{s,c}$ associated with each source and dual variables associated with each destination $v_{d,c}$. These dual variables are often called *potentials* in the context of minimum cost flows. Minimizing the sum of potential differences between the sources and destinations, the fourth constraint of (8), gives a maximum weight matching. The resulting value of $\gamma_{\text{wc}}(R)$ is stored in the variable w.

3.3 Average-case Throughput

Rather than ensuring a maximum worst-case throughput, systems may seek to maximize the average-case throughput of the network. As in worst-case throughput, it would be natural to define the average-case throughput of a routing algorithm as its average throughput over all traffic matrices, which in this case, corresponds to an integral over the space of all doubly-stochastic matrices. Unfortunately, even the volume of this space is difficult to evaluate [13]. Moreover, any integral or sum of the throughput over a restricted set or random sampling of traffic patterns, is neither convex nor concave in the routing function. To get a tractable definition for the average-case throughput, several approximations are necessary.[4]

Since channel load is convex in R and any sum of convex functions is also convex, the average throughput can be approximated as the reciprocal of the average maximum channel load. Therefore, our cost function for average-case channel load is

$$C_{\text{avg}}(R) = \gamma_{\text{avg}}(R) \approx \frac{1}{|X|} \sum_{\Lambda \in X} \gamma_{\max}(R,\Lambda), \tag{9}$$

where $X \subseteq \mathcal{S}$ is a random, finite subset of traffic matrices. In addition to the random sampling, the harmonic mean of channel loads implied by averaging throughputs is replaced with an arithmetic mean. As in the worst-case throughput, the optimization problem can be formulated to minimize the average maximum channel load, which is convex in R. Then the maximum average throughput is approximately the reciprocal the average maximum channel load. In practice, this approximation works well. For example, with $|X| = 100$ and $N = 64$, the approximate expression is within 5% of the original expression for average throughput for each of the routing algorithms used in the later sections of this paper.

[4]Throughput *is* quasi-concave in R, which could be exploited in optimizing for throughput. However, since calculating the average-case throughput requires approximation regardless, we adopt a good, linear approximation of throughput that results in simpler optimization problems.

4. COMPLEXITY AND SYMMETRY

It may appear that the number of variables in the MCF problems must be exponential in the size of the network because the number of paths through a network grows exponentially. However, it is well-known that instead of tracking the probability of individual flows through the network, it is sufficient to only know the total probability of all paths crossing each channel in the network (see [7], page 665). In this formulation, a single flow variable is assigned to each channel for each commodity. Since the routing algorithm design problems contain a commodity for each source-destination pair, this requires CN^2 variables and CN^2 non-negativity constraints. The flow constraints are also reformulated to express the conservation of total flow at each node: the flow injected plus the flow entering a node must equal the flow ejected and leaving a node. This requires a constraint at each node for each commodity or N^3 constraints for routing algorithm design. The linear programming formulation of the worst-case throughput problem requires $O(CN^2)$ additional constraints and the average-case throughput problem requires $O(C|X|)$ additional constraints, where $|X|$ is the size of the random sample of traffic matrices. Therefore, both the number of constraints and variables in the MCF problems can be made polynomial in the network size. Also, given the flow variables from a solution of the reformulated problem, paths can easily be recovered.

Despite their polynomial-size, the complexity of these optimization problems can quickly grow too large to be practically solved. For the LP's considered in this paper, a general purpose linear programming package is experimentally limited to a few million non-zero terms before memory requirements or solution times become prohibitive. This complexity can be greatly reduced by taking advantage of symmetry. Moreover, the validity of these symmetry reductions is a direct result of the fact that our cost functions are convex.

For example, if the network's topology is vertex-symmetric, it is sufficient to limit the search for optimal routing algorithms to those that can be described based on the relative position of the source and destination nodes. That is, $R(p)$ is only computed for all paths from a single, canonical source node to each destination. Then other values of $R(p)$ are found by mapping from an arbitrary source-destination pair to the canonical source using an automorphism defined by the vertex-symmetry of the topology. The result of exploiting this symmetry is the number of variables required to describe the routing algorithm is reduced by N to N^2 and the number of constraints is reduced to CN. A similar optimization for edge-symmetric topologies yields a reduction in the number of worst-case throughput constraints to $O(CN)$.

5. ROUTING IN TORI

In this section, the metrics and optimization techniques developed over the previous sections are applied to routing algorithm design on k-ary 2-cube (2-dimensional tori) topologies (Figure 2). Low dimensional tori are popular in implementations and many routing algorithms have been previously proposed for these topologies.

An initial set of optimization problems explore the tradeoff between locality and worst-case performance of oblivious routing algorithms. While dimension-order routing achieves an optimal worst-case for minimal algorithms, the non-minimal algorithms studied lie far from the optimal tradeoff curve. Motivated by this observation, two new routing algorithms are developed. Both algorithms have simple descriptions and practical, deadlock-free implementations and both greatly improve the locality of routing algorithms

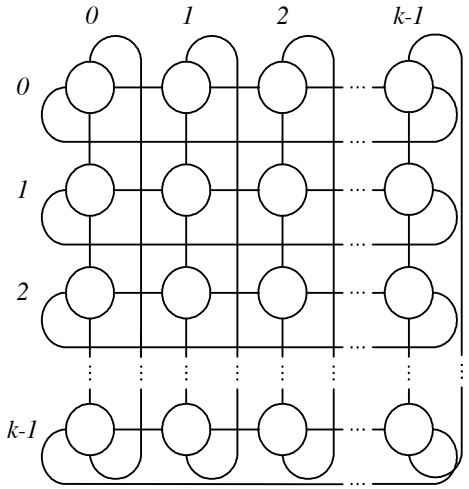

Figure 2: The k-ary 2-cube (2-dimensional torus) topology.

that achieve the maximum worst-case throughput. These new algorithms combined with dimension-order routing cover the ends of the optimal tradeoff curve, but we also show that an interpolation between these extreme points creates a set of routing algorithms that trade locality for worst-case throughput and lie within 20% of the optimal tradeoff curve — far better than existing algorithms. A study of the tradeoffs in average-case throughput versus locality reveals a similar trend to the worst-case tradeoff, but less locality must be sacrificed. More importantly is that the two near-optimal worst-case algorithms we developed have near-optimal average-case throughputs as well. This reveals an interesting result for tori: there is virtually no tradeoff between worst-case and average-case throughput. Finally, we comment on the performance of these oblivious routing algorithms relative to adaptive routing algorithms.

All routing algorithm design problems presented in the following sections are formulated as linear programs and solved with ILOG's CPLEX LP solver [14]. Because the torus is both edge- and vertex-symmetric, the overall problem size is reduced to $O(CN)$.

5.1 Locality versus Worst-case Throughput

The first tradeoff we consider for oblivious routing algorithm design is how optimizing for the worst-case affects the average distance a packet travels. From [15][16][17][18], a general trend is known for tori and other networks — greedy, oblivious algorithms that attempt to maximize locality suffer from poor worst-case performance. This trend can be quantified as a tradeoff by solving a series of routing algorithm design problems: for a particular average path length L, what is the best worst-case throughput that can be achieved?

Restating problem this as a linear program gives

$$\begin{array}{lll} \text{minimize} & C_{\text{wc}}(R) & \\ \text{subject to} & \sum_{p \in P_{s,d}} R(p) = 1, & \forall s, d \in N \\ & R(p) \geq 0, & \forall p \in P \\ & H_{\text{avg}}(R) = L & \end{array} \tag{10}$$

This optimization problem follows the basic form given Section 2.2, except for the additional linear constraint on the average distance a packet travels H_{avg}.

The result of solving a series of these optimization problems for a range of locality constraints (L) on an 8-ary 2-cube is shown in Figure 1. The horizontal axis shows worst-case throughput normalized to the capacity of the network. Average path length is shown on the vertical axis normalized to the minimal path length. Algorithms that achieve an optimal tradeoff between worst-case throughput and locality are shown as the solid line and the set of all feasible routing algorithms is shown in gray.

In addition to the tradeoff curve, several existing routing algorithms are also plotted in the tradeoff space of Figure 1. The details of the routing algorithms are described in Table 1. Both dimension-order routing (DOR) and ROMM are minimal algorithms and therefore have normalized average path lengths of one. As shown, DOR falls at one extreme of the Pareto optimal portion of the tradeoff curve, thus DOR has the best possible worst-case of any minimal routing algorithm in the 8-ary 2-cube. At the other extreme, Valiant's randomized algorithm (VAL) achieves the best possible worst-case, but requires path lengths that are twice the minimal distance. The RLB and RLBth both trade some locality for better worst-case performance. However, VAL, RLB, and RLBth all lie far from the optimal tradeoff curve.

5.2 Worst-case Optimal Routing

Although dimension-order routing achieves the optimal worst-case throughput for a minimal routing algorithm, the tradeoff space demonstrated an obvious lack of any routing algorithm that maximized locality while achieving the maximum worst-case throughput (50% of capacity). In this section, we develop two routing algorithms that approach this worst-case optimal point.

First, it is important to realize that although routing algorithms exist at each point in the feasible region of the worst-case versus locality space, these routing algorithms do not necessarily have simple closed formed descriptions. This appears to be the case for the worst-case optimal point in the 8-ary 2-cube. However, algorithms with simple descriptions do get within a small factor of the optimal performance.

The first algorithm we introduce is an optimized variant of Valiant's algorithm. For the torus, each phase of Valiant's algorithm routes minimally between any two nodes. Because of the randomization of the intermediate node, load is exactly balanced, but at a cost of a path length that is twice minimal. However, many choices of the minimal routing algorithms used in the two phases of Valiant result in some paths that revisit a node. An example of this is shown in Figure 3, where dimension order routing is used for both phases of the algorithm. As illustrated, a "loop" in the path results in extra distance traveled by the packet. Removing this loop from the path would reduce path length while maintaining the optimal worst-case — removing the loop only reduces the channel loads, therefore the worst-case throughput cannot drop. Another important observation is that the minimal routing algorithms used in each of the phases may be different as long as they both realize the capacity of the network under uniform traffic. This leads to a simple observation: the routing algorithms for the phases can be chosen to increase the number of loops that appear in the paths of packets. Since the loops can be safely removed from the paths, increasing the number of loops reduces the average distance a packet travels. Our improved version of Valiant's algorithm (IVAL) implements these ideas.

IVAL works similarly to VAL, but with a much reduced average path length. As in VAL, routing a packet in IVAL begins with choosing a random intermediate node. Then dimension order routing is used to create a path from the source to the intermediate. The second phase of the algorithm also uses DOR, but *reverses* the order of dimension traversal. So, for example, if the first phase routed in the X dimension followed by the Y dimension, the second phase routes in Y followed by X. This simple reversal of di-

Table 1: Summary of routing algorithms

DOR	Dimension-order routing [4]. Packets are routed minimally in the X dimension first, then in Y. If either direction is minimal in a dimension, routes are split evenly between both directions.
VAL	Valiant's routing algorithm [3]. In the first phase, packets are routing from the source to a randomly chosen intermediate node using a minimal routing algorithm (e.g. DOR). The second phase routes minimally from the intermediate to the destination.
ROMM	ROMM [19]. A two-phase algorithm that uses DOR for both phases, like Valiant's, but routes are kept minimal by always choosing the intermediate from the minimal quadrant.
RLB	Randomized local balance [18]. Another two-phase algorithm where the intermediate is chosen so that minimal routing occurs in the X dimension with probability $(k - \Delta_X)/k$, where Δ_X is the minimum distance in X that must be traveled and minimal routing occurs in Y with probability $(k - \Delta_Y)/k$. DOR is used for both phases.
RLBth	RLB threshold [18]. A modification of RLB where a packet is always routed minimally in X if $\Delta_X < k/4$ and minimally in Y if $\Delta_Y < k/4$.

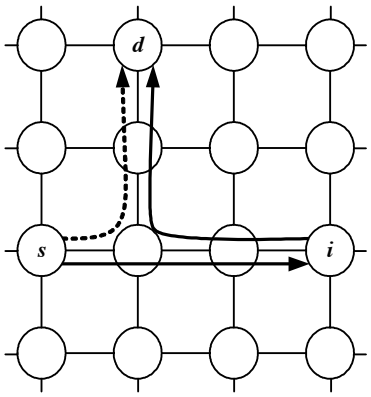

Figure 3: An example of a loop in a path used by Valiant's routing algorithm on a portion of a torus network. X first dimension-order routing is used from the source s to intermediate i and then from the intermediate to the destination d (path shown as solid). A more efficient dotted path removes the loop in the solid path and reduces path length without decreasing throughput.

mension order between the phases greatly increases the chances of loops being formed in the path. Of course, these loops are removed before packet routing begins. Taking the average over all paths shows that IVAL has a average path length of about 1.61 times minimal compared to the two times minimal paths required in Valiant — an almost 20% reduction in the average path length (Figure 1). IVAL can also be made deadlock free with a modest number of virtual channels: one set of virtual channels is used for each of the two phases of the algorithm and two virtual channels are required to break intra-dimension deadlocks within the phases, as in [20]. Therefore IVAL requires a total of four virtual channels per physical channel to avoid deadlock, the same as VAL.

Although IVAL has a simple closed-form (algorithmic) description and improves significantly over Valiant's algorithm, its average path length is still about 9.1% above the optimal point of just below 1.48 times minimal. Of course, there may exist other simple algorithms that lie within this gap, however the gap can also be reduced if the designer is willing to abandon a purely closed-form description of the routing algorithm. We introduce such an algorithm, called 2TURN, that does just that.

Instead of requiring that the routing algorithm have a closed-form description, the 2TURN algorithm only uses a closed-form description of the possible paths a packet may take through the network. As its name implies, 2TURN allows any path through the network which contains at most two turns. A turn is defined as any change from routing in one dimension to the other. Also, "u-turns" or changes of direction within dimensions are disallowed in the 2TURN algorithm. Since every path in IVAL also has at most two turns, 2TURN contains all the paths considered by IVAL. 2TURN can also use paths not available to IVAL. For example, IVAL always routes minimally after its final turn, but 2TURN has the option to route non-minimally.

Then, any path p with at most two turns $R(p) \geq 0$ and for any path q outside this set $R(q) = 0$. These linear constraints can easily be incorporated into the optimization problem to find an optimal weighting of the paths. Since the paths of 2TURN are a superset of those of IVAL, 2TURN can match IVAL's worst-case performance of half of capacity. At the same time, the average path length of 2TURN is reduced to approximately 1.48 times minimal, only 0.36% more than the optimal algorithm (Figure 1). The key advantage of 2TURN over the optimal algorithm is the fact that its paths can be described in simple terms, allowing simple deadlock analysis: 2TURN can be made deadlock free by incrementing a packet's virtual channel set after each turn from the Y to the X dimension. Because any two turn path has at most one Y to X turn, this approach requires two virtual channel sets. Again, intra-dimension deadlock requires two virtual channels, so 2TURN can also be made deadlock-free with four virtual channels per physical channel.

A further comparison of the locality of the IVAL and 2TURN algorithms versus the network radix k for several k-ary 2-cubes is shown in Figure 4. The optimal locality is also plotted for comparison. The figure reveals a significant variation in locality between odd and even radices for both the optimal and 2TURN routing algorithms. In the even cases, the difference between the optimal and IVAL is maximized and 2TURN provides the most benefit. For the $k = 4$ and $k = 6$ cases, 2TURN exactly matches the optimal. The variations and gap between optimal and IVAL both continue to decrease as the radix increases with IVAL settling to roughly 1.64 times minimal and optimal oscillating around approximately 1.52 times minimal.

5.3 Interpolated Routing Algorithms

In the previous sections we focused on the extreme points of the worst-case versus locality tradeoff curve, but there still exists a significant gap between these extreme points in the worst-case tradeoff space. This section, we introduce interpolated routing algorithms to fill this gap and provide a continuum of routing algorithms.

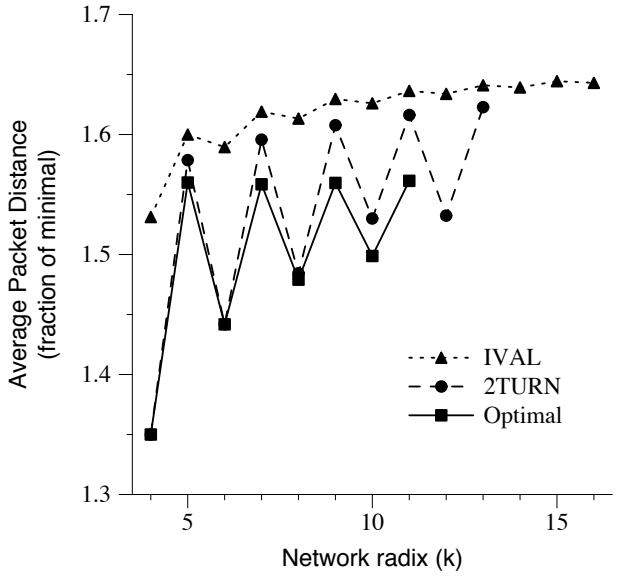

Figure 4: The average path length for several routing algorithms (IVAL and 2TURN) with optimal worst-case performance on different size k-ary 2-cubes. The optimal locality found by solving the corresponding MCF problem is also shown.

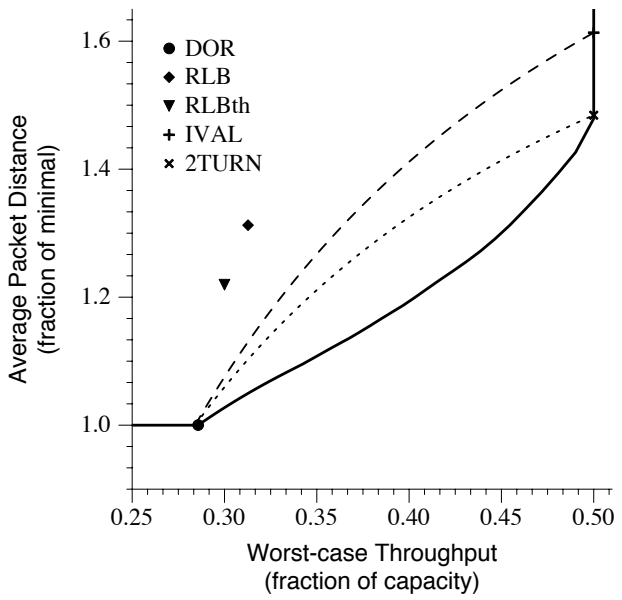

Figure 5: The performance of two interpolated routing algorithms in the tradeoff space between locality and worst-case throughput. The set of algorithms produced by interpolating between DOR and IVAL is shown as a dashed line and the algorithm produced by interpolating between DOR and 2TURN is shown as a dotted line.

Since oblivious routing algorithms can be captured in terms of a probability distribution over all paths, two existing algorithms R_1 and R_2 can be combined to create a new algorithm R':

$$R'(p) = \alpha R_1(p) + (1-\alpha)R_2(p), \quad \forall p \in P, \tag{11}$$

where $0 \leq \alpha \leq 1$. It can be easily verified that $\sum_{p \in P_{s,d}} R'(p) = 1$ for all source-destination pairs and therefore R' is a valid routing algorithm. The interpolation factor α controls the relative influence of R_1 and R_2 on R'. Intuitively, as α sweeps from one to zero, the properties of R' transition from those of R_1 to those of R_2.

Quantitatively, the interpolated routing function has an average path length of

$$H_{\text{avg}}(R') = \alpha H_{\text{avg}}(R_1) + (1-\alpha)H_{\text{avg}}(R_2). \tag{12}$$

The worst-case channel load of the interpolated function can also be calculated using the convexity of channel load:

$$\gamma_{\text{wc}}(R') \leq \alpha\gamma_{\text{wc}}(R_1) + (1-\alpha)\gamma_{\text{wc}}(R_2). \tag{13}$$

Rewriting using throughputs, the worst-case of the interpolated function is a weighted harmonic mean:

$$\Theta_{\text{wc}}(R') \geq \left(\frac{\alpha}{\Theta_{\text{wc}}(R_1)} + \frac{(1-\alpha)}{\Theta_{\text{wc}}(R_2)}\right)^{-1}. \tag{14}$$

Using these ideas, interpolated routing algorithms can be used to generate a tradeoff between two routing algorithms situated at different points in the worst-case versus locality space. For example, in the 8-ary 2-cube example from the previous section, an interpolation between DOR and IVAL can realize any routing algorithm along the dashed curve shown in Figure 5. Each point in the curve corresponds to a different value of α used in the interpolation. For this example, it also happens that the worst-case throughput of the interpolated routing algorithms is exactly equal to the lower bound derived above.[5] Similarly, an interpolation between DOR and 2TURN is shown as a dotted line in the figure.

The interpolated algorithms between DOR and IVAL are at most 17% above the optimal locality with the maximum percent difference occurring about 65% of the way between DOR and IVAL. At the same worst-case throughput as RLB, the interpolated algorithm gives a roughly 14% reduction in path length. Also, the reduction in path length over RLBth is about 12%. Interpolating between DOR and 2TURN only improves the performances, with the set of interpolated algorithms at most 10% above the optimal locality and a 19% and 15% improvement over RLB and RLBth, respectively.

The algorithms produced from both algorithms are also simple to implement. To interpolate between IVAL and DOR, for example, a packet is routed using IVAL with probability α and using DOR with probability $1-\alpha$. Also, since DOR produces paths that are a subset of the paths produced by IVAL and 2TURN, no additional virtual channels are required to avoid deadlock.

5.4 Average-case Throughput

As in the worst-case, the tradeoff between average-case throughput and locality can be found by solving the following linear program at different values for the average packet distance L:

$$\begin{array}{lll} \text{minimize} & C_{\text{avg}}(R) & \\ \text{subject to} & \sum_{p \in P_{s,d}} R(p) = 1, & \forall s,d \in N \\ & R(p) \geq 0, & \forall p \in P \\ & H_{\text{avg}}(R) = L & \end{array} \tag{15}$$

The average case throughput is approximated using a sample of 100 random traffic matrices as described in Section 3.3 ($|X| = 100$).

The resulting optimal tradeoff curve is plotted in Figure 6 and

[5] Since DOR and IVAL share a worst-case traffic permutation, the actual worst-case throughput of the interpolated routing algorithm is equal to the bound. This is true when interpolating between any routing algorithms that share a common worst-case permutation.

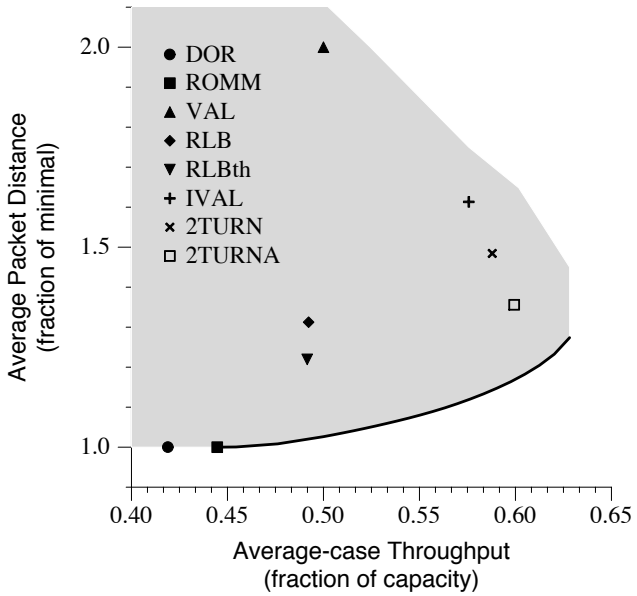

Figure 6: Tradeoff between average path length (vertical axis) and average-case throughput (horizontal axis) on the 8-ary 2-cube. The optimal tradeoff is shown as a solid line and the region of feasible routing algorithms is gray. The performance of several routing algorithms are also shown as points in the tradeoff space.

the region of feasible routing algorithms is shown as gray. The tradeoff shows a similar trend to the worst-case and achieving a higher average throughput requires a sacrifice of locality.

Previously existing algorithms generally lie far from the optimal tradeoff, with VAL offering the best average-case throughput at 50% of capacity. Computing the average throughput of the two new routing algorithms developed for the worst-case in Section 5.2, IVAL and 2TURN, reveals that they are both much closer to the maximum average-case throughput: IVAL is within 8.4% and 2TURN is within 6.4%. This also demonstrates that there is a weak tradeoff between worst-case and average-case throughput in the 8-ary 2-cube and that routing algorithms can be designed to have *both* good worst-case and good average-case throughput.

Adapting the approach used to develop the 2TURN algorithm for the worst-case, a similar algorithm 2TURNA can be designed for the average-case. 2TURNA allows all paths with at most two turns and the probabilities for each path are found by first optimizing for maximum average-case throughput, then for maximizing locality. The resulting algorithm's performance is also plotted in Figure 6. As shown, 2TURNA has an average-case throughput within 4.6% of the maximum of approximately 62.8% of capacity. 2TURNA also increases locality over IVAL and 2TURN, sitting roughly 16% above the optimal tradeoff curve in terms of average path length. Performing the same optimization as in 2TURNA, but limiting the paths to also be minimal, produces a routing algorithm that matches the performance of ROMM. Therefore, ROMM exhibits good average-case throughput over the space of simple, minimal routing algorithms.

5.5 Comments on Adaptive Routing

To this point in the paper, we have assumed all of our routing algorithms are oblivious and therefore do not incorporate network state into their routing decisions. Considering the torus, adaptivity offers no advantage in terms of worst-case throughput as shown in [21]. However, adaptivity can potentially increase locality for a given worst-case. For example, the GOAL algorithm [21] has an average path length of approximately 1.3 times minimal and has an experimental worst-case throughput of half of capacity.[6] In the average case, minimal adaptive algorithms can offer an average throughput of about 60%, roughly the same as 2TURNA, but without the sacrifice in locality [21]. However, the gains in locality offered by any adaptive algorithm could be offset by its loses due to increased per-hop latency to dynamically compute routes.

6. RELATED WORK

Many different routing algorithm design problems have been cast as multicommodity flows (MCFs) and, more specifically, our definition of throughput is consistent with maximum concurrent flow problems [22]. A good overview of routing algorithm design using MCFs is given by Bertsekas and Gallager [8], while Ahuja et al. [7] give a more general treatment of MCFs. One of the first MCF formulations of the routing algorithm design problem is given by Fratta et al. [23] where the cost function approximates the queuing delay of a particular routing algorithm. Other early work by Ros Peran [24] uses maximum channel load as a cost function. The work presented in the paper introduces two new cost functions, worst-case and average-case throughput. The convexity of worst-case channel load ensures that globally optimal solutions to worst-case routing problems can be efficiently found. While the average-case must be approximated, a simple expression yields a good estimate and efficient problem formulation. Additionally, a shortcoming of previous cost functions is that an input traffic pattern must be provided. However, this is not practical or even possible in most interconnection network applications. The cost functions used in this work address this problem by optimizing performance over all traffic patterns.

Another set of network problems closely related to throughput-centric routing algorithm design are those considered in adversarial queuing theory [25]. Notably, Andrews et al. [26] show an on-line routing algorithm that finds feasible paths for packets through a network subject to capacity constraints if *any* feasible solution exists. Although similar to the worst-case throughput problem considered in this paper, these results also incorporate delay bounds for the packets through the network. However, the routing algorithm requires solving a shortest-path problem for each packet routed and is well beyond the complexity and latency requirements of almost any interconnection network router. Instead, we focus on oblivious routing algorithms, which have simple hardware implementations.

In virtual-circuit routing, Awerbuch et al. [27] considers throughput competitive algorithms. As our in work, throughput is defined by the maximum channel load. An on-line approximation algorithm is shown to give provably good competitive ratios, but again the algorithm is too complex to consider for implementation in a typical interconnection network. More recently, Räcke [28] addresses a similar problem and develops an oblivious routing algorithm that is throughput competitive with oblivious routing within a polylogarithmic factor. This theoretical observation on the "power" of oblivious routing is consistent with the experimental results presented in this paper.

[6]There is no known method for determining the exact worst-case throughput for a general adaptive routing algorithm.

7. CONCLUSIONS

As shown in this paper, throughput-centric routing algorithm design can be cast in terms of a linear program. Globally optimal solutions to these optimization problems can then be found efficiently. The result is that we can generate optimal oblivious routing algorithms. Also, the tradeoffs considered in their design can be precisely quantified.

Applying these techniques to k-ary 2-cube topologies produced many interesting results. First, existing algorithms sacrifice too much locality to achieve maximum worst-case throughput. This motivated the development of the IVAL and 2TURN routing algorithms, which achieve near-optimal locality while maintaining maximum worst-case throughput. Both algorithms have practical, deadlock-free implementations. Then, interpolated routing was introduced and by interpolating between 2TURN and DOR, a set of routing algorithms that was within 10% of any point of the optimal tradeoff were created. Finally, analysis of average-case throughput showed that existing algorithms had far from optimal average-case throughput. However, the IVAL and 2TURN algorithms developed for the worst-case also performed well in the average-case. This revealed the weak tradeoff between worst-case and average-case throughput in the 2-dimensional torus.

Future work includes application of the methods developed in this paper to larger networks and different topologies. Efficient algorithms for multicommodity flow have received much attention in recent years and adapting either efficient exact methods [29] or approximation algorithms [30][31] could significantly extend the practical application space of our methods.

APPENDIX

As described in Section 3.2, the problem of designing a routing algorithm with optimal worst-case performance can be expressed as a linear program by reformulating the original problem statement using a convex cost function. The first step of this reformulation replaces the maximum channel load $\gamma_{\text{wc}}(R)$ with multiple inequalities, one for each channel and permutation matrix. A new variable w is introduced to store the replaced value of $\gamma_{\text{wc}}(R)$:

$$\begin{aligned} &\text{minimize} && w \\ &\text{subject to} && \textstyle\sum_{p\in P_{s,d}} R(p) = 1, && \forall s,d \in N \\ &&& R(p) \geq 0, && \forall p \in P \\ &&& \gamma_c(R,\pi)/b_c \leq w, && \forall c \in C, \pi \in \Pi. \end{aligned} \tag{16}$$

Since $\gamma_c(R,\pi)/b_c = \gamma_{\text{wc}}(R)$ for some channel c and permutation π, $w \geq \gamma_{\text{wc}}(R)$. This inequality is reduced to an equality by the minimization problem, so this problem is equivalent to the original convex formulation. Although (16) is a linear program, it is not practical because of the exponential number of constraints. To reduce the number of constraints, the dual optimization problem is considered.

The Lagrange dual function corresponding to (16) is

$$g(q,r,t) = \inf_{R,w}\Bigg\{ w + \sum_{s,d\in N}\sum_{p\in P_{s,d}} q_{s,d}\left(R(p)-1\right) - \sum_{p\in P} r_p R(p) + \sum_{c\in C}\sum_{i=1}^{N!} t_{c,i}\left[\gamma_c(R,\pi(i))/b_c - w\right]\Bigg\},$$

where $r_{s,d} \geq 0$, $\pi(i)$ is the i^{th} of the $N!$ permutation matrices.

Focusing on the summation over t, the definition of γ_c is substituted to give:

$$\sum_{c\in C}\frac{1}{b_c}\sum_{s,d\in N}\sum_{\substack{p:c\in p,\\ p\in P_{s,d}}} R(p)\sum_{i=1}^{N!} t_{c,i}\pi(i)_{s,d} - w\sum_{c\in C}\sum_{i=1}^{N!} t_{c,i} \tag{17}$$

The innermost term of the left group of sums represents a weighted combination of permutations matrices and by the result of Birkhoff [32] this can be represented by a scaled doubly-stochastic matrix. Since there are C such sums, let A^c be a doubly-stochastic matrix with row and column sums of ϕ_c. Then,

$$\sum_{i=1}^{N!} t_{c,i}\pi(i)_{s,d} = a^c_{s,d}. \tag{18}$$

Using this simplification, the problem of maximizing the dual function is rewritten as a linear program with a polynomial number of variables and constraints:

$$\begin{aligned} &\text{maximize} && -\textstyle\sum_{s,d\in N} r_{s,d} \\ &\text{subject to} && r_{s,d} + \textstyle\sum_{c\in p} a^c_{s,d}/b_c \geq 0 && \forall s,d \in N, p \in P_{s,d} \\ &&& \textstyle\sum_{s\in N} a^c_{s,d} = \phi_c, && \forall d \in N, c \in C \\ &&& \textstyle\sum_{d\in N} a^c_{s,d} = \phi_c, && \forall s \in N, c \in C \\ &&& \textstyle\sum_{c\in C} \phi_c = 1, && \forall c \in C \\ &&& a^c_{s,d} \geq 0, && \forall s,d \in N, c \in C. \end{aligned} \tag{19}$$

Taking the dual of this linear program yields the primal linear program presented in Section 3.2.

The dual also has an interesting interpretation that could be useful in developing approximation algorithms for the optimal worst-case routing design problem. In the primal problem, designing a good worst-case algorithm corresponds to selecting appropriates paths through the network and assigning them probabilities — any heuristic for picking the paths and their probabilities gives an approximation to the optimal algorithm. The dual has a similar interpretation, but instead of picking paths, a good dual routing selects permutation traffic patterns that would induce the worst-case channel load in the optimal routing algorithm. The weighted sum of these permutations form the A matrices in the dual optimization problem. Again, any heuristic for selecting these permutations gives an approximation algorithm.

A. REFERENCES

[1] W. J. Dally, P. P. Carvey, and L. R. Dennison, "The Avici terabit switch/router," in *Conference Record of Hot Interconnects 6*, August 1998, pp. 41–50.

[2] InfiniBand Trade Association, "InfiniBand architecture specification," `http://www.infinibandta.org`.

[3] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proc. of the ACM Symposium of the Theory of Computing*, Milwaukee, MN, 1981, pp. 263–277.

[4] H. Sullivan and T. R. Bashkow, "A large scale, homogeneous, fully distributed parallel machine, I," in *Proc. of the International Symposium on Computer Architecture*, 1977, pp. 105–117.

[5] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, and J. Kleinberg, "Universal-stability results and performance bounds for greedy contention-resolution protocols," *Journal of the ACM*, vol. 48, no. 1, pp. 39–69, January 2001.

[6] L. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. of the 7th Int.*

Symposium on High-Performance Computer Architecture, January 2001, pp. 255–266.

[7] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: Theory, algorithms, and applications*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1993.

[8] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Inc., Upper Saddle River, NJ, second edition, 1992.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2003.

[10] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, second edition, 1999.

[11] B. Towles and W. J. Dally, "Worst-case traffic for oblivious routing functions," in *Proc. of the Symposium on Parallel Algorithms and Architectures*, Winnipeg, Manitoba, Canada, Aug. 2002, pp. 1–8.

[12] H. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logist. Q.*, vol. 2, pp. 83–97, 1955.

[13] M. Beck and D. Pixton, "The Ehrhart polynomial of the Birkhoff polytope," *to appear in Discrete Comp. Geom.*, `arXiv:math.CO/0202267`.

[14] ILOG, Mountain View, CA, *CPLEX 7.5 User's Manual*, `http://www.ilog.com`.

[15] A. Borodin and J. Hopcroft, "Routing, merging, and sorting on parallel models of computation," *Journal of Computer and System Sciences*, vol. 30, pp. 130–145, 1985.

[16] C. Kaklamanis, D. Krizanc, and A. Tsantilas, "Tight bounds for oblivious routing in the hypercube," in *Proc. of the Symposium on Parallel Algorithms and Architectures*, 1990, pp. 31–36.

[17] W. A. Aiello, F. T. Leighton, B. M. Maggs, and M. Newman, "Fast algorithms for bit-serial routing on a hypercube," *Mathematical Systems Theory*, vol. 24, no. 4, pp. 253–271, 1991.

[18] A. Singh, W. J. Dally, B. Towles, and A. K. Gupta, "Locality-preserving randomized oblivious routing on torus networks," in *Proc. of the Symposium on Parallel Algorithms and Architectures*, Winnipeg, Manitoba, Canada, Aug. 2002, pp. 9–19.

[19] T. Nesson and S. L. Johnsson, "ROMM routing on mesh and torus networks," in *Proc. of the Symposium on Parallel Algorithms and Architectures*, Santa Barbara, CA, 1995, pp. 275–287.

[20] W. J. Dally and C. L. Seitz, "Deadlock free message routing in multiprocessor interconnection networks," *IEEE Trans. on Computers*, vol. 36, no. 5, pp. 547–553, May 1987.

[21] A. Singh, W.J. Dally, A.K. Gupta, and Brian Towles, "GOAL: A load-balanced adaptive routing algorithm for torus networks," in *to appear in Proc. of the International Symposium on Computer Architecture*, San Diego, CA, June 2003.

[22] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM*, vol. 37, no. 2, pp. 318–334, April 1990.

[23] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method — an approach to the store-and-forward communication network design," *Networks*, vol. 3, pp. 97–133, 1973.

[24] F. Ros Peran, *Routing to minimize the maximum congestion in a communication network*, Ph.D. thesis, MIT, 1978.

[25] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson, "Adversarial queueing theory," *Journal of the ACM*, vol. 48, no. 1, pp. 13–38, Jan. 2001.

[26] M. Andrews, A. Fernández, A. Goel, and L. Zhang, "Source routing and scheduling in packet networks," in *Proc. of IEEE Symposium on Foundations of Computer Science*, Las Vegas, NV, Oct. 2001, pp. 168–177.

[27] B. Awerbuch, Y. Azar, and S. A. Plotkin, "Throughput-competitive on-line routing," in *Proc. of IEEE Symposium on Foundations of Computer Science*, Palo Alto, CA, Nov. 1993, pp. 32–40.

[28] H. Räcke, "Minimizing congestion in general networks," in *Proc. of the IEEE Symposium on Foundations of Computer Science*, Vancouver, Canada, 2002, pp. 43–52.

[29] R.D. McBride, "Progress made in solving the multicommodity flow problem," *SIAM J. Optim*, vol. 8, pp. 947–955, Nov. 1998.

[30] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proc. of IEEE Symposium on Foundations of Computer Science*, Palo Alto, CA, Nov. 1998, pp. 300–309.

[31] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 4, pp. 505–520, 2000.

[32] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucumán Rev. Ser. A*, vol. 5, pp. 147–151, 1946.