A Survey of Random Oblivious Routing Algorithms

Andrew Chang, Mattan Erez, Ben Serebrin, Brian Towles

May 16, 2001

1 Introduction

There exist a wide spectrum of algorithms designed to perform the routing of packets through an interconnection network. This report explores research on a specific class of routing algorithms, *randomized oblivious routing*, that have been shown to have several attractive theoretical benefits in addition to relatively low implementation cost. Before the specifics of this class of algorithms are addressed, it is useful to understand the general properties of other classes of routing algorithms, their general properties, and implementation costs.

1.1 Taxonomy of Routing Algorithms

The diversity in routing algorithms is at least partly driven by the tradeoffs in how a routing decision is made:

- *Deterministic*: Given a source and destination node, the single path taken between them is completely defined. Deterministic routing does not contain path diversity and therefore is subject to poor performance on some traffic patterns and lacks fault tolerance.
- *Non-deterministic*: Instead of a single, static path, paths between a source and destination are chosen dynamically. The number possible paths is always greater than one, and therefore path diversity exists. These algorithms can be further classified based on how a these paths are determined:
  - *Oblivious*: The set of paths is only a function of the source and destination of a packet.
  - *Adaptive*: Network state is also used in the selection of a path.

1.2 Deterministic Routing

Perhaps the simplest class of routing algorithms is the deterministic algorithms: routing can be expressed as a function (one-to-one) where the only arguments are the source and destination nodes. This simplicity lends itself to a small and

fast hardware implementation and also reduces the complexity of analysis of network properties such as deadlock conditions. Because of these factors, deterministic routing, or more specifically dimension order routing, has traditionally been a popular choice in many interconnection network implementations. For uniform, random traffic in particular, deterministic routing can realize the maximum throughput of the network and since the routing is generally minimal, low latency is also achieved [8].

However, since deterministic routing algorithms have no path diversity, they are subject to adversarial permutations that are picked to overload particular links. This intuitive idea has been analytically studied first in [3] and [4]. The gist of these results is that using deterministic routing on a general network with nodes of at most degree d, there always exists a permutation that will load a single link $\Omega(\sqrt{N}/d)$. Alternatively, the worst-case throughput of any deterministic routing scheme is $O(d/\sqrt{N})$. So, while deterministic algorithms have simple implementations and acceptable throughput on many traffic patterns, they are also subject to poor worst-case throughput.

1.3 Adaptive Routing

Adaptive routing incorporates some portion of the network state into the routing decision. State used in routing can include information such as link utilization, buffer occupancy, etc. Also, by allowing multiple potential next hops for a packet, path diversity is increased and the poor worst-case throughput seen in deterministic algorithms is at least potentially avoided. This performance benefit is re-enforced in simulation-based studies such as in [2].

The advantages of adaptive algorithms do come at a cost over deterministic algorithms, though. One cost is that adaptive routing relations are more complex since they incorporate network state. As a practical consideration, any state used in the routing relation is local to nodes of the network. This is because any global information must be assembled and often stale by the time it is available to be used.

Also, the nature of adaptive algorithms makes them difficult or impossible to analyze analytically, leaving worst-case performance undetermined. In fact, simple adaptive algorithms may actually perform worse than deterministic routing for some traffic patterns [8].

1.4 Randomized Oblivious Routing

Randomized oblivious routing incorporates the path diversity of adaptive routing, while still allowing some analytical analysis to be performed. The next three sections highlight research in this area. The first section presents Valiant's seminal paper on randomized routing. The second section discusses Nesson's variant which selects randomly between minimal routes. The third section presents Aiello's randomized adaptive bit-serial routing algorithm and his study of the asymptotic bounds for any randomized oblivious bit-serial routing algorithm on an arbitrary network. The final section presents our conclusions.

2 Valiant's Randomized Routing Algorithm

The three papers discussed in this section [10][11][12] present and analyze a set of related non-minimal, oblivious, randomized routing algorithms. The first paper provides analysis of a universal routing algorithm that meets certain criteria. It then presents one instantiation of the algorithm for an *n-dimensional hypercube* (2-ary n-cube). The other two routing schemes in this paper are based on the same ideas as the universal one but are tailored for *butterfly* and *mesh* networks. The second paper presents a more involved derivation of the hypercube algorithm, and the third proves an asymptotic optimality bound on any network that is met by the butterfly and hypercube routing schemes. The analysis is performed by addressing the theoretical problem of *packet permutation routing* - every node has a single packet to be delivered to a destination node, and every node receives only one packet (i.e. the destinations are a permutation of the sources). The analysis is of the *runtime* of the routing algorithm for a random permutation with a bound on the worst case. Runtime is measured as the total number of routing steps, where each step consists of moving a single packet along a particular link. Other important assumptions of the papers include the following:

- The routing algorithm must be distributed, which led the authors to choose an oblivious algorithm - an algorithm that relies only on the source and destination addresses and possibly a vector of random numbers.
- The network is represented by a sparse graph - each node has a limited number of input and output edges, which is relatively small (constant or logarithmically proportional) to the number of nodes. In particular, the analysis is performed for a degree d graph.
- During each cycle a node can transmit at most one packet on each of its out-going edges, and can receive and buffer a packet from each in-coming link.

The main concept employed in this algorithm is to route to the destination node through an arbitrary randomly chosen intermediate node. This has the effect of turning any traffic pattern into a roughly uniform one, thus the analysis of the average case is representative of all permutations.

2.1 General Routing Algorithm (Hypercube)

The algorithm consists of two phases run consecutively and synchronously: phase A completes before the start of phase B. The first phase consists of sending each packet from its source node to a randomly chosen intermediate node. Each packet can be identified by the node number of its origin $s \in V = \{0, \ldots, N-1\}$ where N is the number of nodes in the network. Thus, for each packet s an intermediate node $u(s) \in V$ is independently chosen with probability $\frac{1}{N}$. After phase

A completes and every packet is at its intermediate node, phase B proceeds to route the packets to their destinations[1].

To simplify the analysis a *non-repeating* assumption is made about the routing in each phase. This means that once two routes diverge they will never share any edge again. For example, using dimension-ordered routing in the hypercube meets this assumption. The more involved derivations of [11] do not require dimension-ordered routing for the hypercube. However, since the randomization step turns any traffic pattern to a uniform one, dimensioned-ordered routing is a good match for this algorithm. Another important requirement for the general analysis is that the routing algorithm for each phase be *symmetric* in the sense that after running a phase the distribution of packets retains the same density as in the original permutation, i.e. if a single packet is placed on every node at the beginning of phase A then with high probability each node will contain a single packet at the end of this phase.

The first theorem of the paper applies to a more general routing problem than permutation routing. It is similar to a permutation route, but each node initially holds h packets and is the destination of h packets. It provides an upper bound on the probability that a packet is delayed more that Δ steps during routing (a packet is delayed when it and some other packet try to use the same link simultaneously):

$$Pr\{delay \geq \Delta\} \leq \left(\frac{e\mu\eta h \log N}{\Delta d}\right)^{\Delta} \cdot T$$

Where the network contains N nodes, μ is the maximal distance between any two nodes, η is the expected length of the minimal path between two nodes, d is the degree of the graph representing the network, and $T = hN$ is the total number of packets in the system.

The proof relies on the fact that the routes, and therefore the nodes the packets pass through, are independently chosen for each source→intermediate node pair, and employs the Chernoff Bound on the tail of the resulting binomial distribution of colliding packets. When applied to a permutation route on the hypercube $h = 1$, $N = 2^n$, and $d = \mu = 2\eta = n$. Choosing $\Delta = n + Kn + \mu$ where $K > 2.5$ is some constant gives:

$$4^{-Kn} \cdot 2^n \leq n^{-K}$$

In other words, the probability that the runtime (the delay of the last packet to arrive in phase B) exceeds $(K+1)\log_2 N$ vanishes quickly for suitably large K.

2.2 Routing for Butterfly Networks

The general scheme presented above cannot be directly applied for a butterfly network, since it will violate the non-repeating requirement. Restricting the

[1] [11] inaccurately refers to [10] as containing a "less synchronized" description of the algorithm.

routing on the butterfly to minimal paths solves this problem and allows selecting a random intermediate node to conform to the non-repeating assumption. However the routing is not symmetric, and therefore instead of applying the theorem a direct derivation of the bound is performed. This analysis is relatively involved but follows the same general method as before bounding the number of packets that go through some node. The end result is:

$$Pr\{delay \geq \Delta\} \leq \left(\frac{ehn^2}{(d-1)\Delta}\right)^{\Delta} \cdot T$$

Where d is the radix, $N = d^n$, and $T = hn$ is again the total number of packets in the system.

When choosing $d = n/\log_2 n \propto \log N/\log\log N$ the bound gives the runtime as $O(\log N)$ with very high probability. A conjecture is made that the algorithm retains the logarithmic complexity even for constant d, and not just a relatively small degree.

2.3 Routing on a Mesh

The algorithm presented in this section differs from the previous ones in two ways:

- It is not strictly two-phase, but has $2n-1$ phases for a k-ary n-mesh.
- It does not require that the phases be run serially, and the analysis is performed assuming that a packet proceeds to the next phase as soon as it finishes routing in the current phase.

The algorithm begins by randomly choosing an intermediate node for each source, restricted to the n^{th} dimension, e.g. a node on the same row as the source for a 2-dimensional mesh. The second phase recursively calls the same algorithm but for the sub-cube of dimension $n-1$. This continues until we choose a random node in all but one dimension. These $n-1$ phases constitute phase A of the general algorithm. Phase B simply consists of routing from the intermediate node to the destination along each dimension, where the dimensions are ordered in reverse to the way the intermediate nodes were selected. This takes an additional n phases. Notice that this algorithm is not equivalent to the general one, in that the random intermediate node is restricted to an $n-1$ cube and the last dimension is routed directly. To allow for overlap of different phases a priority scheme is involved for choosing which packets to route from a queue of a particular node: the packet that is in the most advanced state gets priority over all other packets. This ensures that packets are cannot be delayed by packets in a previous phase, thus maintaining the non-repeating property when phases are concurrent.

The details of analyzing the delay bound are left out, but the end result is that for some $C < 1$ and sufficiently large constant K

$$Pr\{delay \geq (2n-1)(k + Km^{\frac{3}{4}})\} \leq C^{K\sqrt{n}}$$

And this is achieved with buffers no bigger than $n \log_2 N$ with very high probability. Alternatively, this can be expressed as $O(\log^2 N)$.

2.4 Optimality of the Algorithms

[12] provides a lower bound on the runtime of a permutation route on any network. This bound is then used to show that although the randomized algorithms presented above may cause some packets to travel twice the expected distance (by routing through an arbitrary random intermediate node) they are still asymptotically optimal. It is shown that in order to achieve a logarithmic runtime ($O(\log N)$) packets must travel at least twice the diameter of the network. Otherwise the routing algorithm will be $\Omega(N^\epsilon)$ for some $\epsilon > 0$.

The hypercube and butterfly algorithms indeed achieve a logarithmic runtime, and the diameter of these networks is also near-optimal ($\log N$) as required by the bound.

2.5 Critique

The most obvious critique of these papers is that the problem they are directly solving is not the one encountered in actual systems. Here, every node starts out with a certain number of packets it must deliver (h), and no new packets are introduced to the system until all hN packets have been delivered. While in practical networks, packets are injected continuously. Moreover, continuing with a theoretic approach, all the analyses are for asymptotically large network and no consideration is given to the constants involved.

Another obvious problem is that although intuitive, no proof is given that the algorithms converge when running both phases concurrently. If the phases must be serialized, some form of global synchronization is required which has not been taken into account, and in fact, contradicts the basic assumption of obliviousness.

Finally, the proof of optimality is only applicable to near-minimal diameter networks, which are less relevant in today's packaging technologies. The analysis for the mesh routing algorithm is not as complete and convincing as for the other two topologies.

3 ROMM: Randomized Oblivious Multi-phase Minimal Routing

Nesson and Johnsson introduce ROMM for n-dimensional binary cubes (2-ary n-cubes) in [6] and extend it to meshes and tori in [7]. The ROMM algorithm operates in n phases and picks $n-1$ intermediate destination nodes along the path from source to destination. They demonstrate that ROMM has advantages over deterministic dimension-order routing and Valiant's 2-phase routing described in the previous section.

3.1 Description of ROMM

A p-phase ROMM algorithm randomly selects $p-1$ intermediate nodes in the minimal sub-cube between the source and the destination nodes. The intermediate nodes are selected separately for each packet such that all routes between successive intermediate nodes are minimal and therefore the overall route is also minimal. Between each phase, the message is delivered using dimension-ordered routing.

ROMM introduces randomness in a degree proportional to the number of phases p. Resource costs increase with p as virtual channel number increases-more buffers and VC resources are required. The papers evaluate 2- and 4-phase ROMM.

3.2 Differences between ROMM and other routing methods

Deterministic, oblivious routing (such as dimension-ordered routing) allows simple routing logic, but always suffers from poor worst-case performance as described in the first section.

Valiant's routing method achieves approximately the same performance for all permutations, but at the cost of making the average number of hops twice that of ROMM. By picking an intermediate node anywhere in the system, Valiant's algorithm changes all traffic patterns to randomized traffic. Because it is minimal, ROMM has lower latency in no-load systems, but less path diversity than Valiant.

ROMM is also simpler to implement than adaptive routing protocols. The authors claim a further advantage: some adaptive algorithms are not compatible with wormhole routing, while ROMM works with store-and-forward, virtual cut-through, and wormhole routing.

3.3 Simulation Results

This section focuses on the results presented in [7], which are calculated for meshes and tori. The authors simulate bit-complement, transpose, random permutation, and arbitrary random traffic on several topologies. The full set of tests is performed on 16x16 mesh and torus networks, a 4x4x4 torus, while only transpose and single-random routing are tested on a 32x32 mesh [2].

The routing algorithms tested are dimension order, 2-phase ROMM, 4-phase ROMM, and Valiant. The torus test uses 3-phase instead of 4-phase. The tests measure the number of cycles required to route a fixed set of packets on the network given different routing algorithms.

In general, 2-phase ROMM routing outperforms the other routing methods. This method bests Valiant and 4-phase ROMM primarily because these two split displacements across dimensions (the restriction that the intermediate nodes be

[2]The authors in [7] refer to random permutation traffic as "single random" and arbitrary random traffic as "full random".

selected along the boundaries of the sub-mesh is not fulfilled). The authors explain that this caused more messages to be driven toward the center of the mesh or torus [3]. They further explain the performance by noting that the increased turn count results in worm "tangling" for wormhole routing.

Dimension ordered routing slightly outperforms 2-phase routing for bit-complement traffic on all topologies and for full-random traffic on the 4x4x4 torus. Dimension ordered routing substantially outperforms (almost half of the latency 2-phase) for bit-complement traffic on the 4x4x4 torus. On the remaining tests, 2-phase ROMM is always best. 2-phase and 4-phase has identical results for both random traffic tests on a 16x16 torus.

The authors predict that 4-phase ROMM would succeed in higher-dimensioned networks, but did not test higher than 3-dimensioned networks in [7].

3.4 Critique

The authors test 256-element 2-dimensional networks, but test only 64-element 3-dimensional networks, so these tests are not fully comparable. They conjecture about, but do not test, the 256-element 4x4x4x4 network. Their assertion about the benefits of having as many phases as dimensions is thus not well tested, since they do not show results of 4 phases in the 3-D torus.

It would be informative if the authors had tested a larger range of network sizes as well. While a 256 (or 64) element processor array is a reasonable size, systems with larger and smaller processor counts exist today with a variety of topologies. As in the above paper, the method of injecting a fixed number of packets per node does not correspond to real traffic patterns.

The authors mention the idea of "tangling" in wormhole routing but fail to explain it. They state that their performance results for 4-phase routing suffer from worm tangling because there were more turns in 4-phase than in 2-phase, but do not justify this.

The authors introduce a confusing and inconsistent variable they call d. d is the largest number of dimensions any path in the network requires; the authors note that for some traffic patterns, all routes require fewer than n dimensions. However, the analytical results in [7] are calculated for routing tasks such as randomized routing, bit complement, and transpose, all of which require all n dimensions. These results still are formulated in terms of d, but no mention is made that d must be equal to n.

The ROMM method itself has merits: it is a good combination of randomness, which allows a high probability of load-balance without the overhead required by adaptive routing. It is also minimal, which guarantees lower bandwidth requirements than non-minimal algorithms can promise.

[3] As a critique, it is unclear what the authors mean by the center of a torus since the topology is symmetric.

4 Fast Algorithms for Bit-Serial Routing on a Hypercube

Ailello et al. [1] present two versions of an asymptotically optimal randomized *adaptive* algorithm for *bit-serial* routing on a hypercube. For a network with N nodes, this algorithm requires $O(\log N)$-steps to route an arbitrary permutation with high probability. For comparison, they derive the lower bound for any randomized oblivious algorithm for bit-serial routing and find it to require $\Omega(\log^2 N/\log\log N)$ steps.

4.1 Problem Context

The authors attempt to address some perceived shortcomings in contemporary work. Several proposed routing algorithms exhibit a logarithmic factor increase in latency ($\Omega(\log^2 N)$) instead of the anticipated optimal $\Omega(\log N)$ in actual implementations. The authors attribute this additional overhead to the bit or byte serialization of packets resulting from the pin limitations in real switches. Also, as this work focuses on bit-serial routing, a packet can occupy up to $\log N$ wires while traveling through the network raising resource contention, deadlock and livelock concerns not found in simpler store-and-forward schemes.

4.2 The Basic Algorithm

The authors' algorithm has three components. First, they apply Valiant's two phase algorithm [10] to initially route to a random intermediate node and then subsequently route to the desired destination node on an $O(\log N)$-dilated butterfly network. A *k-dilated butterfly network* is one in which each edge is replaced by a bundle of k edges. The selection of $O(\log N)$ dilation maintains a high probability that a free output edge exists for each new input packet. Second, they verify the applicability of their algorithm to hypercubes by demonstrating only a constant factor slowdown after mapping the N-node $O(\log N)$-dilated butterfly network into an N-node hypercube. Finally, they implement a bit-serial algorithm which orders packets and bounds the maximum delay experienced by any packet despite the possibility of switch contention. The delay is bounded by $O(M + \log N)$, where M is the length of the message.

4.2.1 Two Versions of the Algorithm on a Dilated Butterfly Network

There are two variations of the basic algorithm for routing permutation traffic. The selected example network is shown in Figure 4.2.1 and is an $O(\log N)$-dilated butterfly employing worm-hole flow control. Nodes in this butterfly network are specified by row and level. Inputs nodes are at *level* 0 and output nodes are at *level* n. Input and output nodes are equivalent so the number of unique nodes is $n2^n$.

The two variations are required due to the differences between *strong switch* and *weak switch* models. A strong switch can examine all input edges from each

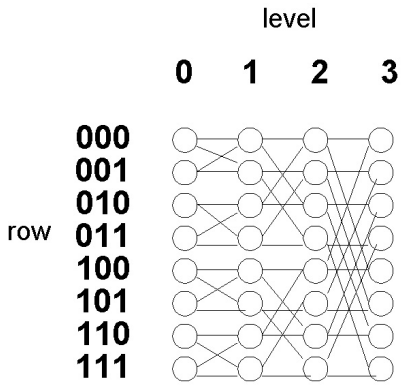

Figure 1: Example Butterfly Network

input bundle within one time step. In contrast, a weak switch can only examine one edge from each input bundle within a time step.

4.2.2 Strong Switch Case

In the strong switch case, for *level* l, each bundle can be reached by 2^l inputs and can select one of $2^{n-(l+1)}$ nodes as its initial random destination. With n packets per input, the probability of more than $c \log N$ packets traversing a bundle is

$$\left(\begin{array}{c} n2^l \\ c \log N \end{array} \right) \left(1/2^{l+1}\right)^{c \log N}$$

This relation results from the need to process n packets from each of the 2^l inputs combined with choosing between $c \log N$ paths and the $1/2^{(l+1)}$ selection probability for each specific path. Applying the relation

$$\left(\begin{array}{c} a \\ b \end{array} \right) \leq (ae/b)^b$$

for $0 < b \leq a$, the authors assert that for any constant k there is a corresponding constant c such that the probability of more than $c \log N$ packets per switch is $1/N^k$ and can be made arbitrarily small. With a message of length M, an additional M steps are required after the arrival of the message head to complete delivery resulting in the final $O(M + \log N)$ bound.

4.2.3 Weak Switch Case

In the weak switch case, packets may need to wait as a switch can only examine one edge from each of the two input bundles per time step. The switch can transfer exactly one packet from the set of input bundles to the set of output bundles. Therefore, packets may be forced to wait due to switch contention and

require buffering. The routing algorithm must guarantee that the maximum delay experienced by any packet remains $O(\log N)$. Ranade [9] proposed a solution to this problem which requires the assignment of a random $O(\log \log N)$-bit *rank* to each packet. Then, a switch may never forward a packet unless it is guaranteed that no packet with smaller rank will arrive in the future. The application of this invariant forces the stream of packets leaving each switch to be ordered. As the packet ranks are randomly distributed through the network, the desired delay bound is maintained. The proposed implementation of this solution required that the packets carry their rank and that ghost packets[4] be produced. The main problem with the original formulation is the additional delay incurred for rank comparison between two packets arriving at a switch.

Aiello et al.'s variation to Ranade eliminates both the requirement of packets to carry rank and the need for ghost packets. Instead, they add a new *token* packet type. Further, they employ the modified invariant that no switch may forward a packet of rank r unless exactly r tokens have already been processed. As before, each packet is assigned a random rank from 0 *to* w where w is $O(\log N)$. Next, the input switches at *level* 0 send out a stream of packets interspersed with tokens so that r tokens precede each packet with rank r. Switches count the number of tokens they have already seen to determine their operation during each time step. Local decisions at each switch are simplified as only one of four cases can occur:

1. Both current input edges have tokens: switch forwards tokens to the output edges and advances all four queue pointers.

2. One current input edge has a token and the other has a packet: switch shifts out the MSB of the packet header to determine which of the two output edges to use and advances the queue pointers for the selected input/output pair.

3. Both current input edges have packets: switch forwards one of the packets and advances the queue pointers for the selected input/output pair.

4. Either of the two input queues is empty: switch does nothing during this time step.

The $O(\log N)$ behavior of this algorithm is demonstrated by first defining and than using the concept of a *k-delay sequence*. This sequence consists of four components: a backwards path from output to input; a set of switches along the path; a sequence of packets; and a sequence of ordered ranks. When a packet is delayed by δ steps, a backtracking algorithm can determine a specific delay sequence. At each switch, there are three possible causes of delay:

1. An alternate packet was forwarded instead - thus the other packet becomes a component in the delay sequence.

[4]Ghost packets are duplicated packets used to broadcast rank state.

2. A token was forwarded instead - trace the token back to its last point of delay and repeat analysis.

3. One of the input queues of the node was empty - trace back to the switch feeding that empty input queue. Either a packet was selected by this switch for an alternate output, in which case add that packet to the delay sequence; or, take one more step back in switch-level and repeat analysis.

The result of this chaining is that a $\delta - w$ packet delay sequence for can be identified. We can now bound the delay for an arbitrary packet by enumerating the possible delay sequences and then calculating the combined probability bound. There are 2^{2n} backward path choices (2^n outputs can serve as beginnings and 2^n inputs can serve as endpoints). For each path, there are

$$\begin{pmatrix} n+\delta-w \\ \delta-w \end{pmatrix}$$

switch combinations on the path. Next, there are $\prod_{i=1}^{\delta-w}(n2^{l_i})$ ways to choose packets. Finally. there are at most

$$\begin{pmatrix} \delta \\ w \end{pmatrix}$$

ways to choose a sequence of ordered ranks. A packet p_i is delayed if it passes through a switch s_i with rank(p_i) equal to r_i. These occur with probabilities $\prod_{i=1}^{\delta-w}(2^{n-l_i}/2^n)$ and $1/w^\delta - w$ respectively. Again, the authors contend that after applying the inequality

$$\begin{pmatrix} a \\ b \end{pmatrix} \leq (ae/b)^b$$

for $0 < b \leq a$, there is a $k1$ such that the product is at most $1/N^k1$, for δ = k2 log N, $w = k3 \log N$, and $k2 > k3$. As δ represents the delay through the network, the delay is bounded to $O(\log N)$.

4.3 Calculating a Lower Bounds for Randomized Oblivious Routing

Aiello et al. present a lower bound on delay for any randomized oblivious bit-serial routing algorithm on an arbitrary network. Assuming message length M, number of nodes N, and maximum degree of the network d, the calculated bound is:

$$\Omega\left(\frac{(M + \log N/\log d)\log N}{\log d + \log\log N}\right)$$

assuming Δ address bits exist when the packet is Δ distance away from its destination. Without this address bit constraint the bound becomes:

$$\Omega\left(\frac{M \log N}{\log d + \log\log N}\right)$$

Taking the probability over the choice of permutations as well as the choice of paths, Aiello et al. demonstrate that for random permutation, the probability that every edge has fewer than $c \log N/(\log d + \log\log N)$ packets traversing it which are $\Omega(\log N/\log d)$ steps away from their destination is at most $e^{-2N^{1/4}}$ for a sufficiently small constant c. Thus, an arbitrary permutation requires $O((M + (\log N/\log d))\text{clog } N/(\log d + \log\log N))$ bit-steps for completion.

The basic proof consists of an iterative partitioning the set of N nodes into two sets, S and T, which each have $N/2$ nodes and then further dividing these sets into arbitrary $\sqrt{N}/8d^{3/2}$ subsets. This process can be repeated $(N/2)/\sqrt{N}/8d^{3/2}$ times. Within each iteration, an edge e, a subset of nodes, and a random destination are selected. Then, the probability that after using the edge e, the $\sqrt{N}/8d^{3/2}$ paths will continue for at least another $\Omega(logN/logd)$ steps is accumulated. After stepping through all combinations and iterations, this effort shows that with high probability $(1 - e^{-2N^{1/4}})$ at least one of the edges will be congested. Therefore, for a random permutation and a random path selection, the lower bound on running time is $\Omega((M + \log N/\log d)/\log N/\log d + \log\log N))$ with probability $1 - e^{-2N^{1/4}}$.

4.4 Contributions and Critiques

The focus of this paper is the analysis of randomized bit-serial routing algorithms. This paper makes two contributions. It introduces of a random adaptive bit-serial routing algorithm with $O(\log N)$ delay bound. It extends the Borodin-Hopcroft lower bound to cover any random oblivious bit-serial routing algorithm on any polylogarithmic degree network. The derived bound is $\Omega(((M + \log N/\log d)\log N)/(\log d + \log\log N))$.

While the majority of the analysis in this study is quite rigorous, there are a few problematic areas. The study relies exclusively on asymptotically optimal bounds as the figures of merit. It makes no attempt to quantify the magnitude of constant factors. While the authors make note that the effects of bit and byte serialization imposed by practical concerns are ignored in other studies, they do not use the same diligence to account for the effects of finite buffering, queuing effects or fairness guarantees in switch allocation in their own study. The authors also neglect to provide any discussion of how the added costs associated with the costs of nxw tokens and the additional $O(N \log N)$ additional edges required to support these tokens compares with the original costs of ghost packets and packet rank identifiers proposed by Rande. Also, as most systems do not employ bit-serial routing, the overall benefit of their analysis is somewhat reduced. Finally, in the latency bound exposition for both strong and weak switches, Aiello et al., are a bit cavalier in declaring that the simplification of their derived probability expressions to a simple $1/N^k$ expression is straightforward.

5 Conclusion

Randomized oblivious routing is an attractive class of routing algorithms because of their low implementation cost, path diversity, and potential to be analyzed. Valiant's early work in this area showed the optimally of randomized oblivious routing from a theoretical standpoint. The later work on ROMM tried to adapt Valiant's results to a more practical and minimal routing algorithm. Finally, implementation effects, such as packet serialization, were studied by Aiello et al.

While a significant amount of theoretical work has been completed in this area, open questions about the efficient implementation of randomized oblivious routing algorithms still exist. For example, the ROMM paper alludes to the potential impact of wormhole routing techniques (flit correlation) on randomized routing algorithms. However, we are unaware of a detailed analysis of this problem. Also, the effects of common hardware limitations, such as finite buffering and pipelining, are still left to be answered. For interconnection network architects to fully realize the potential benefits of randomized oblivious routing, much practical work remains to be done.

References

[1] W. A. Aiello, F. T. Leighton, Bruce M. Maggs, and Mark Newman, "Fast Algorithms for Bit-Serial Routing on A Hypercube", *IEEE Transactions on Computers*, Vol 41. No 5. May 1992. pp. 578-587.

[2] K. Bolding, M. Fulgham, and L. Snyder, "The Case for Chaotic Adaptive Routing," Technical Report UW-CSE-94-02-04, Univerisity of Washington, Feb. 1994.

[3] A. Borodin and J. Hopcroft, "Routing, Merging, and Sorting on Parallel Models of Computation," *Journal of Computer and System Sciences*, Vol. 30. 1985. pp. 130-145.

[4] C. Kaklamanis, D. Krizanc, and A. Tsantilas, "Tight Bounds for Oblivious Routing in the Hypercube," *Symposium on Parallel Algorithms and Architecture*, 1990. pp. 31-36.

[5] F. T. Leighton, Bruce M. Maggs, Abhiram Ranade, and Satish B. Rao, "Randomized Routing and Sorting on Fixed Connection Networks", *Journal of Algorithms*, Vol. 17. No 1. July 1994. pp. 157-205.

[6] T. Nesson and S. L. Johnsson, "ROMM Routing: A Class of Efficient Minimal Routing Algorithms," Technical Report TR-21-94, Center for Research in Computing Technology, Harvard University, Cambridge, MA, August 1994.

[7] T. Nesson and S. Lennart Johnsson, "ROMM Routing on Mesh and Torus Networks," Technical Report TR-08-95, Center for Research in Computing Technology, Harvard University, Cambridge, MA, May 1995.

[8] M. J. Pertel, "A Critique of Adaptive Routing," Technical Report CS-TR92-06, California Institute of Technology, Pasadena, CA, June 1992.

[9] A. G. Ranade, "How to Emulate Shared Memory", *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, October 1987. pp. 185-194.

[10] L.G. Valiant and G.J. Brebner, "Universal Schemes for Parallel Communication," *Proceedings of the* 13^{th} *ACM Symposium on Theory of Computation*, 1981. pp. 263-277.

[11] L.G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM Journal of Computation*, 1981.

[12] L.G. Valiant, "Optimality of a Two-Phase Strategy for Routing in Interconnection Networks," *IEEE Transactions on Computers*, Vol. C-32. No. 9 September 1983.