A Survey of Congestion+Dilation Results for Packet Scheduling

Bruce M. Maggs
Carnegie Mellon University
and
Akamai Technologies
Email: bmm@cs.cmu.edu

***Abstract*— This paper surveys a collection of theoretical results that relate the congestion and dilation of the paths taken by a set of packets in a network to the time required for their delivery, or to the rate at which they can be delivered, where the congestion is the maximum number of paths that pass through a link, over all links, and the dilation is the length of the longest path. Results are stated first for batch routing algorithms and then for continuous routing. Most of the given results are for store-and-forward algorithms (e.g., as implemented in Internet routers), while some are for wormhole or connection-based routing. None of the algorithms described in this survey ever drop packets. The first result in this area was proved by Leighton, Maggs, and Rao in 1988. They showed that any set of packets whose paths have congestion c and dilation d (and are loop free) can be delivered to their destinations in a synchronous store-and-forward network in $O(c + d)$ steps.**

I. Introduction

An algorithm for routing a set of packets to their destinations through a network must perform two tasks. The first, *path selection*, is to determine the path that each packet takes through the network. The second is to resolve any contention for the same resource in the network (e.g., a network link) by specifying the times at which the packets vying for the resource are permitted to use it. This latter task is called *scheduling*. The path selection and scheduling tasks, however, are not necessarily solved in that order. An algorithm may, in fact, solve these tasks simultaneously, and path selection decisions may depend on scheduling decisions, or vice versa.

In this survey, however, we only consider algorithms in which path selection decisions and scheduling decisions are made independently. In particular, we examine different scheduling algorithms under the assumption that the routes taken by the packets are determined solely by their sources and destinations, and not by the presence or behavior of other packets in the network, or by any decisions made by a scheduling algorithm. Such path selection algorithms are called *oblivious*. The path selection algorithms used most heavily by the Internet are in practice either oblivious, or close to oblivious. For example, the decisions made by the BGP and OSPF protocols are typically oblivious, although by changing configuration data, they can be made to modify routes in response to congestion or poor performance in the network. Active load-balancing algorithms are typically not oblivious.

One advantage of viewing packet routing as two independent tasks is that each can then be tackled seperately. In particular, one approach to routing packets to their destinations is to select a set of routes that miminize contention for resources, and, given the routes, schedule the use of the resources to be as efficient as possible. Henceforth in this paper when examining scheduling algorithms, we will assume that the paths taken by the packets have been fixed by the path selection algorithm.

In this paper we describe two types of scheduling algorithms. An *off-line* algorithm is one that has access to a complete description of the network, and all of the paths taken by packets. The algorithm is performed by an external computer that uses this global information to find a schedule. An *on-line* algorithm, on the other hand, is performed by the routers themselves, using only information that is available locally.

A. Congestion and dilation

We begin our discussion of congestion and dilation by focusing on batch routing algorithms in synchronous networks. In the batch scenario, there is a collection of packets whose paths through the network are fixed, and the goal is to schedule the movements of the packets along their paths so as to minimize the total time required to routing all of the packets. In a synchronous network, there is a global clock. In our model, one packet can be transmitted across each network link in a single synchronous time step, and a packet can traverse at most one link in a single step.

Of course, there must be some correlation between the performance of the scheduling algorithm and the selection of the paths. In particular, the maximum distance d traveled by any packet is always a lower bound on the time required to deliver all packets. We call this distance the *dilation* of the set of paths. Similarly, the largest number of packets that must traverse a single edge during the entire course of the routing is a lower bound. We call this number the *congestion* c of the paths. In terms of these parameters, the goal of the path selection algorithm is to select paths for the packets that minimize c and d.

Figure 1 illustrates a graph model for path selection and scheduling. The shaded nodes labeled 1 through 5 represent routers. The edges between the nodes represent network links.

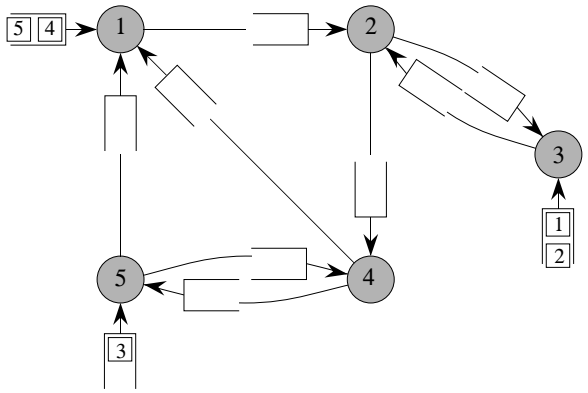

Fig. 1. A graph model for packet routing.

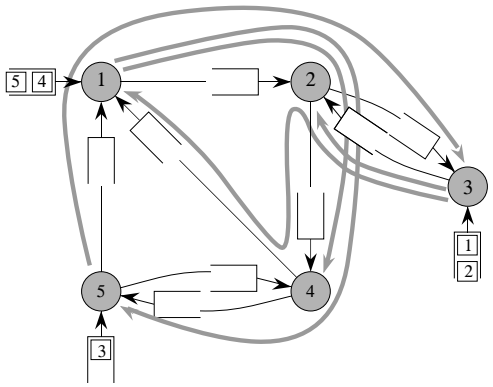

Fig. 2. A set of paths for the packets. Each packet follows a shortest path to its destination. The dilation is $d = 3$ and the congestion is $c = 3$.

At the end of each edge is an *edge queue* that can hold a small number of packets (in this example, two). With this definition, an edge queue might also be called an input queue. The algorithms described in this paper can generally be modified to use output queues instead with no impact on the results.

A packet is depicted by a square box containing the label of its destination. Before the routing process begins, packets are stored at their origins in special *initial queues*. For example, packets 4 and 5 are stored in the initial queue at node 1.

The goal is to deliver each packet from its origin to its destination via a series of synchronized time steps. At each step at most one packet can traverse each edge. Furthermore, a packet can traverse an edge only if at the beginning of the step its edge queue is not full. Upon traversing the last edge on its path, a packet is removed from the edge queue and placed in a special *final queue* at its destination. For simplicity, the final queues are not shown in Figure 1. Independent of the routing algorithm used, the size of the initial and final queues are determined by the particular packet routing instance to solved. Thus, any bound on the maximum queue size required by a scheduling algorithm refers to the edge queues only.

Figure 2 shows one way of choosing the paths for the packets. Here, each packet takes a shortest path from its origin to its destination. For example, packet 1 follows a path from node 3 to 2 to 4 to 1. Since no packet traverses more than three edges, the dilation is $d = 3$. Packets 3, 4, and 5 all traverse the edge from 1 to 2, but no more than three packets traverse any other edge. Thus, the congestion is $c = 3$.

A schedule for the packets is displayed in Figure 3. A schedule simply specifies which packets move and which wait at each time step. An $\times$ in row p and column t indicates that at time t packet p traverses an edge and enters the queue at the end of that edge. A blank indicates that at time t packet p waits in a queue. For example, packet 3 moves at time step 1, waits at steps 2 and 3, and then moves again in steps 4 and 5.

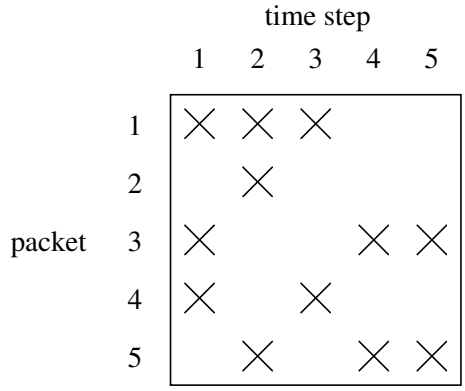

Fig. 3. A schedule for the packets. An $\times$ in row p and column t indicates that at time t packet p moves. A blank indicates that it waits.

B. Outline

In the remainder of this paper we survey the results that have been proved and stated in terms of congestion and dilation. We begin in Section II by reviewing the results for store-and-forward batch routing, both off-line algorithms and on-line. Next, in Section III we cover wormhole routing. Then Section IV reviews bufferless routing. In Section V, we present results for continuous routing. Section VI describes several algorithms for finding paths for the packets that minimize congestion and dilation. We conclude in Section VII with a short discussion of the relevance of these results to routing on the Internet.

II. Store-and-forward batch routing

The idea of decoupling the path selection process from the scheduling process, and then analyzing the scheduling process alone, was first used by Leighton, Maggs, and Rao [1]. They proved that for any set of packets whose paths are edge-simple (i.e., no path uses the same edge more than once) and have congestion c and dilation d, there exists a store-and-forward schedule that routes the packets in $O(c + d)$ packet steps and never stores more than a constant number of packets in the queue at the head of an edge. The $O(c + d)$ bound improves on the naive $O(cd)$ bound, and is optimal. Note that there is no restriction on the structure of the network, and that the size of the network and number of packets do not appear in the number of steps required. The proof uses the Lovász Local

Lemma and is nonconstructive. Leighton et al. also devised a simple randomized on-line algorithm that routes n packets in $O(c + d \log nd)$ packet steps, with high probability (w.h.p.).

It was not obvious prior to [1] that $O(c+d)$-step schedules are guaranteed to exist, as it is easy to imagine that there might be some network for which there is some packet routing instance that requires more than $O(c + d)$ steps to solve. Or one might wonder whether there is some other parameter or characterization of either the network or the paths that figures in the bound on the number of steps. Furthermore, [1] show that many simple scheduling algorithms cannot meet this bound. For example, they showed that for any deterministic non-predictive strategy, there are instances that require $\Omega(cd/\log c)$ steps. A *non-predictive* scheduling algorithm is one that determines the order in which to grant packets access to a link without taking into account their paths beyond the next hop. FIFO is a simple example. Later Cidon, Kutten, Mansour, and Peleg [2] showed that for arbitrary routes, many natural deterministic greedy strategies may require $\Omega(d\sqrt{n} + n)$ steps, where n is the number of packets. A *greedy* scheduling algorithm is one that never leaves a link idle if there is a packet waiting to use it.

Leighton and Maggs and Richa [3] later discovered a sequential off-line algorithm for finding a store-and-forward schedule of length $O(c+d)$ on any network. The algorithm is based on the techniques of Beck [4] and Alon [5] for making the Lovász Local Lemma constructive. It uses information about the entire network and all of the packets, and runs in $O(P \log^{1+\varepsilon} P \log^*(c + d))$ time, w.h.p., for any fixed $\varepsilon > 0$, where ε also (as usual) figures in the constant hidden by the big-O notation, and where P is the sum of the lengths of the paths taken by the packets. Hence [3] makes the proof of [1] constructive.

The $O(c + d)$ bound of Leighton et al. was followed by a number of on-line algorithmic results. For the special case of leveled networks, Leighton, Maggs, Ranade, and Rao [6] presented a simple on-line store-and-forward algorithm for routing any set of n packets in a leveled network with depth D in $O(c+D+\log n)$ steps, w.h.p. In a *leveled* network with depth D, each node is labeled with an integer between 0 and D, and each edge with its tail on level i, $0 \leq i < D$, has its head on level $i + 1$.

For routing instances in which all packets follow shortest paths, there are efficient on-line scheduling algorithms. Mansour and Patt-Shamir [7] first showed that, in any network, if packets are routed greedily on shortest paths, then all of the packets reach their destinations within $d+n-1$ steps, where n is the total number of packets. These schedules may be much longer than optimal, however, because n may be much larger than c. Meyer auf der Heide and Vöcking [8] later devised a simple on-line randomized algorithm that routes all packets to their destinations in $O(c+d+\log nd)$ steps, w.h.p., provided that the packets follow shortest paths.

For arbitrary paths, Tardos and Rabani [9] first designed an on-line scheduling algorithm that is guaranteed to deliver all packets to their destinations within $O(c)+(\log^* n)^{O(\log^* n)}d+\log^k n)$, where $k > 1$ is some fixed constant. We can avoid delving into the meaning of this mathematical expression by observing that Rabani and Ostrovsky [10] then found an algorithm that requires $O(c + d + \log^{1+\varepsilon} n)$ steps, w.h.p., for any $\varepsilon > 0$.

In short, the good news is that for batch routing, the congestion and dilation lower bounds can be met to within a constant factor, and a path selection algorithm, in selecting paths, only needs to minimize these two quantities. Furthermore, when packets follow shortest paths, there is a simple on-line algorithm that delivers the packets in $O(c+d+\log nd)$ steps, w.h.p., where n is the number of packets.

III. Wormhole routing

In a wormhole router, the bits in a message are grouped into a sequence of *flits*, where a flit is the smallest unit of information that can be queued at a node of the network. Typically the number of bits in a flit is a small constant. The message is viewed as a worm that traverses the network in a bit-serial fashion. The header flit or flits, which may specify the message's destination, or the route that it will take, proceed first. As the header flits pass through a switch, they set up a connection between the incoming channel on which they arrived, and an outgoing channel, on which they leave. The remainder of the flits contain no routing information. Instead, it is the responsibility of the switch to forward them from the incoming channel to the outgoing channel. As the header flits work their way through the network, the flits following behind become spread out across the network.

In order to keep the size of the switches small in a wormhole router, the capacity of each switch to queue flits is limited to one per incoming channel. Because a switch cannot identify which message a flit belongs to from its contents, the flits of a message must arrive on the incoming channel in a contiguous fashion. Furthermore, since a switch can only queue one flit of a message, if the header flit of a message cannot advance then the flits following the header must stall.

A natural generalization of wormhole routing is to allow a switch to queue more than one flit per message, perhaps even allowing it to queue an entire message. This approach is called *virtual cut-through* routing [11], and predates wormhole routing.

Wormhole and virtual-cut-through routing have several advantages over store-and-forward routing. In store-and-forward routing, if an l-bit message traverses a path of length d, and is never delayed, it will reach its destination in ld bit steps (assuming each channel can transmit one bit in each bit step). In wormhole (or virtual cut-through) routing, however, the header flit does not wait for the rest of the message. The first bit of the header flit arrives at its destination after d steps, and the last bit of the message arrives after $d + l - 1$ steps. The difference in time is due to the better utilization of network edges by the wormhole router; in a wormhole router, multiple edges of the network can simultaneously transmit flits belonging to a single message, whereas in the store-and-forward algorithm only one edge can transmit a flit of

the messsage at any time. Thus, when the congestion in the network is low, wormhole and virtual-cut-through routing are more efficient than store-and-forward routing. As we shall see, this may not be the case when the network is congested. In addition to reduced latency, wormhole routing also has the advantage that it can be implemented with small, fast switches.

Wormhole routing owes much of its popularity to an influential paper by Dally and Seitz [12], which introduced the method. Much of the paper by Dally and Seitz is devoted to the design of wormhole routing algorithms that avoid *deadlock*. A wormhole routing algorithm can deadlock if the header flit of every worm is prevented from moving because the queue that it wants to enter is full. For example, a pair of worms can deadlock if the head of the first worm wants to enter a queue occupied by the tail of the second worm, and vice versa. The solution to this problem in the paper by Dally and Seitz is to allow each physical channel to emulate several virtual channels, and to construct a virtual network in which the worms cannot form cycles. The virtual channels share the physical wire (or wires) provided by the physical channel, but the switch maintains a separate queue for each virtual channel. This solution has also been implemented in several parallel computers. For example, each physical channel of the iWarp machine [13] supported 4 virtual channels, and each physical channel of the J-Machine [14] supported 2 virtual channels.

In stating the following results, we use q to denote the queue size, i.e., is the maximum number of different messages that can be queued, in whole or in part, at the head of any network edge, i.e., q is the maximum number of virtual channels that can be supported by any physical channel. We assume that after a flit traverses a network edge, it is queued at the head of that edge. We also assume that initially each message is stored in an initial queue at its node of origin, and ultimately is stored in a final queue at its destination.

Greenberg and Oh [15] were the first to state non-trivial network-independent wormhole routing results in terms of l, c, and d. They created a randomized algorithm that produces schedules of $O(mcd + mcl \log n)$ bit steps, w.h.p., where $m = \min\{l, d\}$. Ranade, Schleimer, and Wilkerson [16] then showed that on any network, any set of l-bit messages whose paths have congestion c and dilation d can be routed in $O(lcd)$ bit steps. The $O(lcd)$ bound improves on the naive $O((l + d)cd)$ bound. Neither of these papers considered the case $q > 1$.

Cypher, Meyer auf der Heide, Scheideler, and Vöcking [17] give wormhole routing results that are stated for networks in which each physical channel can simultaneously transmit flits from b messages. They give a simple on-line randomized algorithm for routing n messages in $O((lcd^{1/b} + (l + d) \log n)/b)$ bit steps, w.h.p., for any $b \leq \log n$. If, instead of transmitting b flits simultaneously, a switch can support q virtual channels, where $q = b$, then the number of bit steps can be translated by simply replacing b with q and multiplying by q, i.e., $O(lcd^{1/q} + (l + d) \log n)$

Independently, Cole, Maggs, and Sitaraman showed that if each edge can queue up to q message flits, then it is possible to route any set of l-bit messages whose paths are edge simple and have congestion c and dilation d in $(l + d)c(d \log d)^{1/q} 2^{O(\log^*(c/d))}$ bit steps. The proof is non-constructive, but can be made constructive and turned into an off-line algorithm using the techniques in [3]. They also prove a nearly matching lower bound, i.e., for any values of c, d, q, and l, where $c, d \geq q + 1$ and $l = (1 + \Omega(1))d$, they show how to construct a network and a set of paths with congestion c and dilation d and a set of l-bit messages that require $\Omega(lcd^{1/q})$ bit-steps to route. The proof of the lower bound is a generalization of the $\Omega(lcd)$ lower bound proved by Ranade et al. for the case $q = 1$. These nearly matching upper and lower bounds imply that increasing the queueing capacity of each edge can speed up a wormhole routing algorithm by a superlinear factor, an observation that was first made by Koch [18] for circuit-switching on the butterfly.

IV. Bufferless routing

A *bufferless* scheduling algorithm is one in which no packet waits in a queue (other than its initial or final queue) for more than a single step. Bufferless algorithms are also called "hot-potato" or "deflection" algorithms. The main result in this area is due to Busch, Magdon-Ismail, and Mavronicolas, who designed an algorithm for delivering packets along a fixed set of paths in the batch model that requires $O(T \log^3(n + N))$ steps, w.h.p., where T is the optimal time, n is the number of packets, and N is the size of the network.

There are also several results for specific sizes of buffers. Scheideler [19] shows that a schedule of $O(c + d)$ steps is always possible if buffers of size two are allowed. Scheideler also provides a simpler proof of the $O(c + d)$ result than that of [1], and surveys and simplifies other related results.

For buffers of size one, Meyer auf der Heide and Scheideler [20] designed an algorithm that can deliver the packets within $O((d + c \log(c + d) \log\log(c + d))(\log\log\log(c + d))^{1+\varepsilon})$, w.h.p., for any constant $\varepsilon > 0$.

V. Continuous routing

Scheideler and Vöcking [21] have designed scheduling algorithms for the scenario in which packets arrive in a network according to a stochastic process and put a maximum load $\lambda < 1/e$ on any edge of the network. A load of λ for means for a particular edge means that in any given step, the expected number of new packets that arrive and whose paths pass through that edge is λ. This scenario is called *continuous* routing, in contrast to batch routing.

In the store-and-forward model, assuming that all packets follow shortest paths, and that the dilation of the paths is d, the protocol of Scheideler and Vöcking delivers each packet within $O(d)$ steps of its arrival, with high probability.

In the wormhole model, where each link can simultaneously transmit b flits, they show that for $\lambda \leq b/ld^{1/b}$, the expected time required for a message to be delivered after its arrival is $O(d + l)$ bit steps.

VI. Algorithms for finding paths

Raghavan and Thompson [22] used an algorithmic technique called "randomized rounding" to show that for any set of packets in any network, there is an off-line algorithm that finds a set of paths with congestion $c_{\text{frac\_min}} + O(\sqrt{c_{\text{frac\_min}} \log n})$ when $c_{\text{frac\_min}} = \Omega(\log n)$, and $O(\log n)$ otherwise, where $c_{\text{frac\_min}}$ is the optimum (minimum) possible congestion for the corresponding multicommodity flow problem, which is permitted to split the path between a source and a destination into multiple fractional paths. Thus, $c_{\text{frac\_min}}$ is a lower bound on the congestion, $c_{\min}$, of an optimal integral set of paths. Aumann and Rabani [23] proved that $c_{\min}$ is at most $O(c_{\text{frac\_min}} \log n)$ by proving an approximate max-flow min-cut theorem for multicommodity flow. More recently Andrews and Zhang [24] showed that it is hard to find an algorithm that can find a set of paths whose congestion is within an $\Omega(\log \log^{1-\varepsilon} N)$ factor of optimal, where N is the size of the network. The Andrews-Zhang bound also holds as a lower bound on the integrality gap for the multicommodity flow problem. The integrality gap bound has very recently been tightened by Rao and Zhou to $\Omega(\log \log N / \log \log \log N)$.

Srinivasan and Teo [25] took another approach, not attempting to minimize congestion alone, but instead congestion and dilation taken together. They designed an off-line algorithm for finding a set of paths with congestion c and d for which $c + d$ is within a constant factor of optimal. Combined with the algorithm of Leighton, Maggs, and Richa [3], there is an off-line algorithm for solving any packet routing instance to within a constant factor of the minimum possible number of steps.

On the lower bound side, Borodin and Hopcroft [26] proved that any oblivious path selection algorithm must create congestion at least $\Omega(\sqrt{n/\delta^3})$ for some permutation on any n-node degree-δ network. The Borodin-Hopcroft lower bound was later improved to $\Omega(\sqrt{n}/\delta)$ by Kaklamanis, Krizanc, and Tsantilas [27].

VII. Conclusion

This paper has surveyed a wide variety of algorithms for scheduling the movements of packets in networks of arbitrary topologies. We have seen that for store-and-forward routing, it is possible to find schedules that deliver packets in time proportional to the congestion plus dilation of the paths taken by the packets. For wormhole routing, the length of each message also figures in the time required, and, interestingly, the number of flits that can be simultaneously transmitted across a link or queued at a router appears as an inverse in an exponent! In the continous arrivals model, provided that the load (which is equivalent to the congestion) is sufficiently small, e.g., $\lambda < 1/e$, and the packets follow shortest paths, there is an on-line algorithm that delivers packets in time proportional to the dilation. For wormhole routing with continuous arrivals, the length of the messages and the number of simultaneous flits (as an inverse in an exponent) also figure in.

What are the high-level lessons to be learned for the designers of routing and scheduling algorithms? First, in selecting paths, aim to minimize the congestion and dilation of the paths taken by the packets. These are the only metrics that matter in selecting paths. Second, route along shortest paths, if this can be down without compromising congestion too much. It is easier to design simple but efficient on-line algorithms if the packets follow shortest paths. Third, wormhole routing is more efficient than store-and-forward routing in uncongested networks, but may be much less efficient in congested networks. Fourth, if wormhole routing is to be used, it is important to support either multiple (e.g., at least $\log d$) virtual channels or simultaneous transmissions across each link. Finally, make use of randomness in the scheduling algorithms – all of the simple on-line algorithms do so. Don't expect simple deterministic algorithms to perform well in the worst case – the can't.

References

[1] F. T. Leighton, B. M. Maggs, and S. B. Rao, "Packet routing and job-shop scheduling in O(congestion + dilation) steps," *Combinatorica*, vol. 14, no. 2, pp. 167–180, 1994.

[2] I. Cidon, S. Kutten, Y. Mansour, and D. Peleg, "Greedy packet scheduling," *SIAM Journal on Computing*, vol. 24, no. 1, pp. 148–157, Feb. 1995.

[3] F. T. Leighton, B. M. Maggs, and A. W. Richa, "Fast algorithms for finding O(congestion + dilation) packet routing schedules," *Combinatorica*, vol. 19, no. 3, pp. 89–95, 1999.

[4] J. Beck, "An algorithmic approach to the Lovász Local Lemma I," *Random Structures and Algorithms*, vol. 2, no. 4, pp. 343–365, 1991.

[5] N. Alon, "A parallel algorithmic version of the Local Lemma," *Random Structures and Algorithms*, vol. 2, no. 4, pp. 367–378, 1991.

[6] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao, "Randomized routing and sorting on fixed-connection networks," *Journal of Algorithms*, vol. 17, no. 1, pp. 157–205, July 1994.

[7] Y. Mansour and B. Patt-Shamir, "Greedy packet scheduling on shortest paths," *Journal of Algorithms*, vol. 14, pp. 449–65, 1993.

[8] F. Meyer auf der Heide and B. Vöcking, "A packet routing protocol for arbitrary networks," in *Proceedings of the Twelfth Symposium on Theoretical Aspects of Computer Science.* Volume 349 of *Lecture Notes in Computer Science*. Heidelberg, Germany: Springer–Verlag, Mar. 1995, pp. 291–302.

[9] E. Tardos and Y. Rabani, "Distributed packet switching in arbitrary networks," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, May 1996.

[10] R. Ostrovsky and Y. Rabani, "Universal O(congestion + dilation + $\log^{1+\epsilon} N$) local control packet switching algorithms," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, May 1997, pp. 644–653.

[11] P. Kermani and L. Kleinrock, "Virtual cut-through: a new computer communications switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979.

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

[13] S. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H. T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P. Tseng, J. Sutton, J. Urbanski, and J. Webb, "iWarp, an integrated solution to high-speed parallel computing," in *Proceedings Supercomputing '88*. IEEE Computer Society Press, Nov. 1988, pp. 330–339.

[14] M. D. Noakes, D. A. Wallach, and W. J. Dally, "The J-Machine multicomputer: an architectural evaluation," in *Proceedings of the Twentieth Annual International Symposium on Computer Architecture*, May 1993, pp. 224–235.

[15] R. Greenberg and H. Oh, "Universal wormhole routing," in *Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing*, Dec. 1993, pp. 56–63.

[16] A. Ranade, S. Schleimer, and D. S. Wilkerson, "Nearly tight bounds for wormhole routing," in *Proceedings of the Thirty-Fifth Annual Symposium on Foundations of Computer Science*, 1994.

[17] R. Cypher, F. Meyer auf der Heide, C. Scheideler, and B. Vöcking, "Universal algorithms for store-and-forward and wormhole routing," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, May 1996, pp. 356–365.

[18] R. R. Koch, "Increasing the size of a network by a constant factor can increase performance by more than a constant factor," in *Proceedings of the Twenty-Ninth Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 1988, pp. 221–230.

[19] C. Scheideler, *Universal Strategies for Interconnection Networks*. Berlin, Germany: Springer–Verlag, 1998, vol. 1390 of Lecture Notes in Computer Science.

[20] F. Meyer auf der Heide and C. Scheideler, "Routing with bounded buffers and hot-potato routing in vertex-symmetric networks," in *Proceedings of the Third European Symposium on Algorithms*, 1995, pp. 341–354.

[21] C. Scheideler and B. Vöcking, "Universal continuous routing strategies," in *Proceedings of the Eighth ACM Symposium on Parallel Algorithms and Architectures*, June 1996, pp. 142–151.

[22] P. Raghavan and C. D. Thompson, "Randomized rounding," *Combinatorica*, vol. 7, pp. 365–374, 1987.

[23] Y. Aumann and Y. Rabani, "An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm," *SIAM Journal on Computing*, vol. 27, no. 1, pp. 291–301, 1998.

[24] M. Andrews and L. Zhang, "Hardness of the undirected congestion minimization problem," in *Proceedings of the 37th ACM Symposium on Theory of Computing*, May 2005.

[25] A. Srinivasan and C.-P. Teo, "A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, May 1997, pp. 636–643.

[26] A. Borodin and J. E. Hopcroft, "Routing, merging, and sorting on parallel models of computation," *Journal of Computer and System Sciences*, vol. 30, no. 1, pp. 130–145, Feb. 1985.

[27] C. Kaklamanis, D. Krizanc, and A. Tsantilas, "Tight bounds for oblivous routing in the hypercube," in *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architectures*, July 1990, pp. 31–36.