Packet Routing in Networks with Long Wires*

Ronald I. Greenberg
Department of Electrical Engineering
University of Maryland
College Park, MD 20742
rig@eng.umd.edu

H.-C. Oh
Department of Information Engineering
Korea University
Chochiwon, Korea
hyeong@kusccgx.korea.ac.kr

May 17, 1996

*Work supported in part by NSF grants CCR-9109550 and CCR-9321388.

Corresponding author:

Ronald Greenberg
Electrical Engineering Department
University of Maryland
College Park, MD 20742

(301)405-3649

rig@eng.umd.edu

Abstract

In this paper, we examine the packet routing problem for networks with wires of differing length. We consider this problem in a network independent context, in which routing time is expressed in terms of "congestion" and "dilation" measures for a set of packet paths. We give, for any constant $\epsilon > 0$, a randomized on-line algorithm for routing any set of N packets in $O((C \lg^{\epsilon}(Nd) + D \lg(Nd))/\lg\lg(Nd))$ time, where C is the maximum congestion and D is the length of the longest path, both taking wire delays into account, and d is the longest path in terms of *number* of wires. We also show that for edge-simple paths, there exists a schedule (which could be found off-line) of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$, where d_{max} is the maximum wire delay in the network. These results improve upon previous routing results which assume that unit time suffices to traverse a wire of any length. They also yield improved results for job-shop scheduling as long as we incorporate a technical restriction on the job-shop problem.

1 Introduction

An efficient packet routing algorithm is critical to the design of most large-scale general-purpose parallel computers. One must move data between different locations in an appropriate routing network as quickly as possible and with as little queuing hardware as possible. The packet routing problem has been extensively studied in the past, mostly in the context of specific networks and specific message patterns. Recent works by Leighton, Maggs, Rao, and Ranade have provided very general packet routing results (based on summary measures of the message traffic), which even yield many improvements upon prior analyses of specific networks and message patterns [6, 7, 8]. But these works have made the simplifying assumption that unit time suffices for any transmission of a packet from one network node to another regardless of the actual length of wire connecting the nodes. This assumption becomes less and less tenable as we build larger and larger parallel machines. Hence this paper considers the situation in which an arbitrary delay is associated with each wire.

Except for the introduction of nonunit wire delay, we follow the commonly used *store-and-forward* routing model and the usual graph-based terminology. Packets are atomic objects, which at each time step, either wait in a queue or are in transit on some edge of the network connecting two nodes. Associated with each edge e is an edge delay of $d_e > 0$ time steps required for a packet to traverse that edge, and at any given time, at most one packet can be present on each edge. (There are other interesting routing models, for example allowing the use of transmission lines on which packets can be pipelined, circuit-switching, or wormhole routing, which are not considered in this paper.)

In the model of Leighton, et. al., packets wait in three types of queues. Before routing begins, packets are stored at *initial queues* in the nodes where they are generated. Each time a packet traverses an edge, it enters the *edge queue* at the end of that edge; a packet can begin to traverse an edge only if the queue at the end of that edge is not full. Finally, when a packet reaches its destination, it is placed into a *final queue* at that node. The sizes of the initial and final queues are determined solely by the packet routing problem to be solved, but we seek routing schedules that bound the maximum queue size for edge queues. It is convenient in this paper to also use a second type of edge queue at the *beginning* of each edge. The ability of a node to distribute incoming packets among the queues at the beginnings of the outgoing edges enables simple mechanisms for limiting queue size.

We may view the packet routing problem as being comprised of two tasks, selecting a path through the network for each packet and setting a schedule for when packets move and wait. The second task has traditionally been the more difficult one, and it is the focus of this paper. Of course, the selection of paths affects the time and queue size required by a legitimate schedule. For example, the maximum distance d, in number of edges, traveled by any packet is a lower bound on the routing time; this distance is often referred to as the *dilation* in the literature. In fact, the routing time is lower bounded by the maximum over all packet paths of the sum of edge delays along the path. We refer to this measure as the *generalized dilation* D, which differs from d when the unit wire delay assumption is discarded. Similarly, the routing time is lower bounded by the *congestion* c, the maximum over all edges of the number of packets that must traverse the edge over the entire course of the routing, and by the *generalized congestion* C, the maximum over all edges of the number of packets traversing the edge multiplied by the delay of the edge. We also use the notation d_{max} for the maximum over edges e of the edge delay d_e.

Leighton, Maggs, and Rao have given a randomized on-line algorithm for the unit wire delay case, which (with high probability) produces a schedule of length $O(c + d\lg(Nd))$ with queues of size $O(\lg(Nd))$, where N is the number of packets [7]. This naturally implies that in the problem with general edge delays, we could obtain a schedule of length $O(d_{max}(c + d\lg(Nd)))$ by simply using d_{max} time steps to simulate each step of the unit delay algorithm. In Section 2, we give, for any $\epsilon > 0$, an on-line algorithm that produces a schedule of length $O((C\lg^{\epsilon}(Nd) + D\lg(Nd))/\lg\lg(Nd))$ with queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$. This is a significant improvement, since cd_{max} and dd_{max} may be much larger than C and

D. It should also be noted that the constants hidden in the O-notation are of modest size at least for $\epsilon = 1$, so the algorithm is practical.

Our on-line algorithm is also an improvement upon the result obtained from the (off-line but polynomial time) algorithm of Shmoys, Stein, and Wein [9] for job-shop scheduling. In job-shop scheduling, the problem input consists of a set of jobs and a set of machines. Each job consists of a sequence of operations, each of which has a specified duration and must be processed on a specified machine. The operations of a job must be processed in order, and each machine can handle at most one operation at a time. We can draw a correspondence between job-shop scheduling and packet routing by thinking of jobs as packets and machines as network edges. The schedule length of Shmoys, Stein, and Wein translated into our notation for packet routing is $O\Big((C+D)\frac{\lg^2(Nd)}{\lg\lg(Nd)}\Big)$. Our superior result for packet routing can be applied to job-shop scheduling as long as we impose the restriction that on any given machine, all operations are of the same duration.

Leighton, Maggs, and Rao have also shown, for unit wire delay, that when the paths traversed by the packets are edge-simple, there exists some schedule of length $O(c+d)$ requiring only constant size queues. This immediately implies existence of a schedule of length $O(d_{max}(c+d))$. In Section 3, we show that there exists a schedule of length $O\Big((cd_{max}+D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ with queues of size $O(d_{max})$, a potential improvement when $d > c$. This result also applies to the restricted form of job-shop scheduling with the additional restriction that no job has more than one operation on a single machine.

2 On-Line Algorithm

Our basic approach to produce a schedule on-line is, as in [7, 9], to first produce an "unconstrained" schedule in which several packets may travel on the same edge at the same time and then "flatten" it into a legitimate schedule. We begin by showing how to produce a schedule of length $O\Big((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ with queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ when d_{max} is bounded above by a polynomial in N and d; later we refine the result to obtain, for any constant $\epsilon > 0$, a schedule of length $O((C \lg^{\epsilon}(Nd) + D \lg(Nd))/\lg\lg(Nd))$ with queues of size $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$ and no restriction on d_{max}.

For our initial result, the method of constructing the unconstrained schedule is essentially the same as in the job-shop scheduling approach of Shmoys, Stein, and Wein [9]. (Our advantage is gained through a superior method of flattening to a legitimate schedule that is not applicable to the general form of job-shop scheduling.) Each packet chooses an integral delay randomly and uniformly from the interval $[1, C]$. A packet that is assigned delay x waits in its initial queue for x time steps and then proceeds to its destination without stopping. Though this may cause more than one packet to traverse a single edge at the same time, it is unlikely that too many will do so, as is shown by the following lemma, adapted from [9]. We include the proof here, because we will use similar arguments in proving later results.

Lemma 1 (Shmoys, et. al.) *When d_{max} is bounded above by a polynomial in N and d, the strategy of delaying each packet in its initial queue an integral amount chosen randomly and uniformly from $[1, C]$ yields an unconstrained schedule that is of length at most $C + D$ and, with high probability, has no more than $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$ packets traversing any edge at any time.*

Proof. We begin by considering the probability p that more than τ packets are present on a particular edge e during a particular time step t. Though packets may spend many time steps traversing e, there are at most C total time units of routing on edge e. Thus, there are at most $\binom{C}{\tau}$ ways to choose τ units of packet routing to occur on edge e at time t. The probability that an individual one of these τ units is scheduled on edge e at time t is at most $1/C$ since each packet chose a delay uniformly at random from C possibilities. If these τ units of routing are all from different packets, the probability that they all occur on edge e at time t is at most $\left(\frac{1}{C}\right)^{\tau}$, since packet delays are chosen independently; otherwise the probability is 0. Thus, we have

$$
\begin{aligned}
p &\leq \binom{C}{\tau}\left(\frac{1}{C}\right)^{\tau} \\
&\leq \left(\frac{eC}{\tau}\right)^{\tau}\left(\frac{1}{C}\right)^{\tau} \\
&= \left(\frac{e}{\tau}\right)^{\tau},
\end{aligned}
$$

where the bound on $\binom{C}{\tau}$ can be obtained by using Stirling's approximation to the factorial. For sufficiently large Nd, if $\tau = k\frac{\lg(Nd)}{\lg\lg(Nd)}$, then, $p \leq (Nd)^{-(k-1)}$. To bound the probability that there exists *any* edge and time with more than $k\frac{\lg(Nd)}{\lg\lg(Nd)}$ packets, we multiply p by the Nd bound on the number of edges used by some packet and by the $C + D$ time steps in the unconstrained schedule. The latter factor is also polynomial in N and d, since we have assumed d_{max} is. Thus, choosing k large enough yields the desired result. ■

We must now explain how to flatten the unconstrained schedule into a legitimate schedule. The flattening procedure is trivial when each wire delay is just one unit of time. In that case, an unconstrained schedule S of length L with at most γ packets on an edge at one time can be flattened to a legitimate schedule of length γL by replacing each unit of S's time with γ units of time in which the packets on any given edge are routed in turn. The work of Shmoys, Stein, and Wein [9] on job-shop scheduling shows how to obtain a flattened schedule of length $\gamma L \lg d_{max}$ in the general context of arbitrary durations for each operation (edge traversal) of each job (packet). We show how to obtain a shorter flattened schedule of length γL by taking advantage of a fact that applies to packet routing but not the general form of job-shop scheduling, namely that the time required to traverse a given edge is the same for all packets. We further show that we can flatten schedules produced as in Lemma 1 on-line, whereas Shmoys, Stein, and Wein consider only an off-line context. Also, we bound the queue size, an issue not considered by Shmoys, et. al.

Lemma 2 *Consider any unconstrained schedule of length L with at most γ packets on an edge during any time step. The unconstrained schedule can be simulated by γL steps of* *a legitimate schedule, and the simulation can be performed on-line if all the delays in the unconstrained schedule are in the initial queues. Furthermore, the queue size required by the legitimate schedule is at most γ.*

Proof. The basic idea is that each packet is routed as soon as possible in the legitimate schedule (given the constraint of one packet per edge at any time), except that a packet that begins traversing edge e at time t in the unconstrained schedule does not do so before time γt in the legitimate schedule. (Ties between different packets needing to traverse the same edge are broken arbitrarily.) Figure 1 shows an example of an unconstrained schedule and its flattened version. A more precise specification indicating how the queues are managed is that a packet that traverses edge e and then begins traversing edge e' at time t in the unconstrained schedule is handled as follows: The packet is held in the queue at the end of e until time γt, when it is moved to the queue at the beginning of e'; as soon as it reaches the head of the latter queue and no other packet is traversing e', it begins to traverse e'.

The process just described can be accomplished on-line by having each packet carry a field that holds the time that the packet begins traversing the upcoming edge in the unconstrained schedule. Initially, each packet holds the delay assigned in its source processor; each time a packet is dispatched on an edge, it adds in the delay of that edge.

It remains to be shown that all packets are routed by time γL. Let us refer to the traversal by a particular packet of a particular edge as an *operation*. Also, let t^{ω}_{UB} and t^{ω}_{UE} represent the begin and end times for operation ω in the unconstrained schedule, and let t^{ω}_{LB} and t^{ω}_{LE} represent the times in the legitimate schedule. Our flattening process, clearly enforces $t^{\omega}_{LB} \geq \gamma t^{\omega}_{UB}$, and we now show that $t^{\omega}_{LE} \leq \gamma t^{\omega}_{UE}$.

We proceed by induction on time. Under the assumption that $t^{\omega}_{LE} \leq \gamma t^{\omega}_{UE}$ whenever $t^{\omega}_{UE} \leq t$, we can show that the same is true whenever $t^{\omega}_{UE} \leq t+1$. (The base case with $t=0$ is trivial.) Consider any operation ω with $t^{\omega}_{UE} = t+1$, and denote the packet and edge involved as m and e, respectively. (If there is no such operation, we are done.) We know that

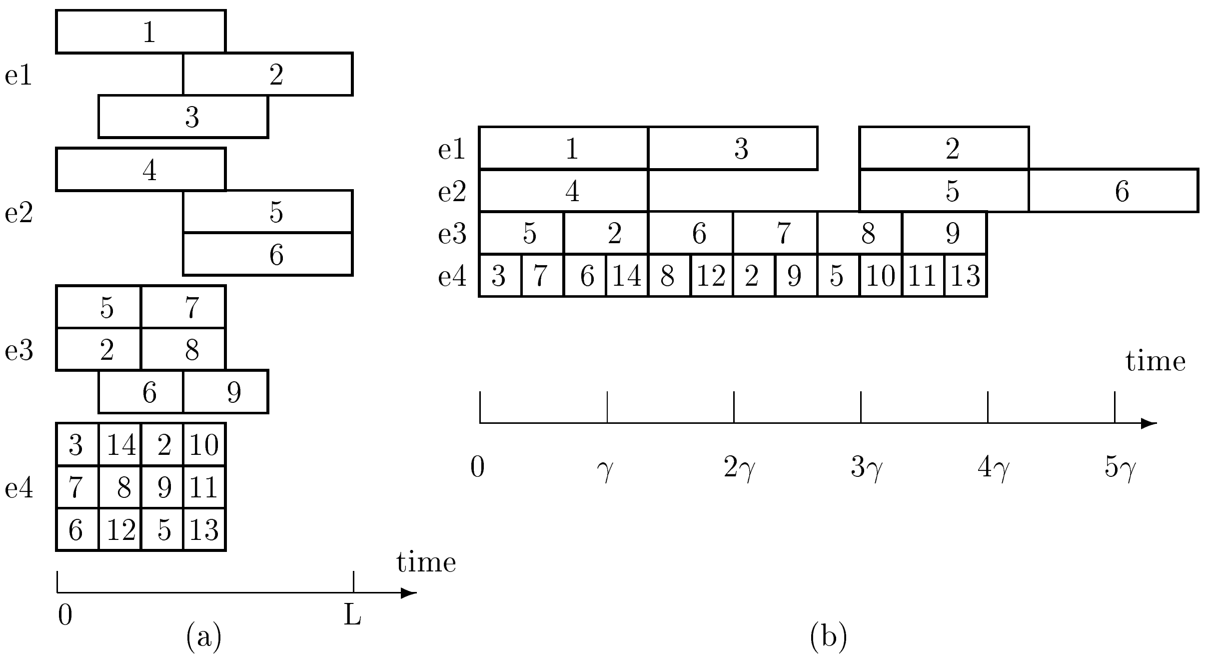

Figure 1: An example of the flattening process for four edges, e_1, e_2, e_3, and e_4 and messages numbered 1 to 14. (a) The initial unconstrained schedule of $L = 7$ and $\gamma = 3$. (b) The flattened version of the schedule in (a).

$t^{\omega}_{UB} = t+1-d_e$ and that there are at most γ packets that begin traversing e at $t+1-d_e$ in the unconstrained schedule. By the induction hypothesis, each of those γ packets is ready to begin traversing e by time $\gamma(t+1-d_e)$ in the legitimate schedule. Even if m is the last of these packets to be sent over e, it completes its traversal by time $\gamma(t+1-d_e)+\gamma d_e = \gamma(t+1)$.

The bound on the queue size can be obtained by using the already established relationship of $\gamma t^{\omega}_{UB} \leq t^{\omega}_{LB} \leq t^{\omega}_{LE} \leq \gamma t^{\omega}_{UE}$ and the fact that packets do not wait at intermediate nodes in the unconstrained schedule. It follows that the set of packets in any edge queue at a given time must have begun traversing the corresponding edge within a fixed period of d_e time steps in the unconstrained schedule; there are at most γ such packets. ■

By putting together Lemmas 1 and 2, we obtain the following result:

Theorem 3 *When d_{max} is bounded above by a polynomial in N and d, any set of packets can be routed on-line in $O\Big((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ steps using queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$, with high* *probability.* ■

We can remove the restriction on d_{max} by using a technique similar to [9], but again we must show how to perform the task on-line:

Theorem 4 *Any set of packets can be routed on-line in* $O\Big((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ *steps using queues of size* $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$*, with high probability.*

Proof. We can begin by thinking of each d_i as being rounded down to the nearest multiple of $\frac{d_{max}}{Nd}$, denoted d_i'. In the resultant network, $\mathcal{N}'$, edges have at most Nd distinct lengths which are multiples of $\frac{d_{max}}{Nd}$. By working with a routing clock period of $\frac{d_{max}}{Nd}$, we can use Lemma 1 to produce the unconstrained schedule we used above, since C and D are polynomial in N and d when expressed in units of $\frac{d_{max}}{Nd}$. The only problem is that in the real network $\mathcal{N}$, each d_i' must be adjusted upward to d_i. But each adjustment is at most $\frac{d_{max}}{Nd}$, which we can handle by simply doubling the clock period to $\frac{2d_{max}}{Nd}$, i.e., giving packets twice as much time at each step to travel or wait on the same edge as before. This adjustment does not change the number of packets using any edge during any time step, so we can proceed with the flattening process just as before. ■

We can also improve the schedule length by tightening the analysis in Lemma 1:

Theorem 5 *For any constant* $\epsilon > 0$*, with high probability, on-line routing of any set of packets can be achieved in* $O\Big(\frac{1}{\epsilon}(C\lg^{\epsilon}(Nd)+D\lg(Nd))/\lg\lg(Nd)\Big)$ *steps using queues of size* $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$.

Proof. We modify Lemma 1 to produce an unconstrained schedule of length $D+\alpha C$ (for α to be determined) by choosing delays from $[1,\alpha C]$. Once τ is determined, the final flattened schedule will be of length $(D+\alpha C)\tau$. In Lemma 1, the upper bound on p becomes $\left(\frac{e}{\alpha\tau}\right)^{\tau}$. Then for $\alpha = (\lg(Nd))^{\epsilon-1}$ (with $\epsilon > 0$) and $\tau = \frac{k}{\epsilon}\frac{\lg(Nd)}{\lg\lg(Nd)}$, we obtain $p \le (Nd)^{-(k-1)}$ for sufficiently large Nd. ■

It should be noted that the constant k in the proof of Theorem 5 is of modest size, so we certainly obtain a practical algorithm for $\epsilon = 1$, the case that leads to Theorem 4. Even Theorem 4 specialized to unit edge delay improves upon the on-line result of Leighton, Maggs, and Rao except when c is somewhat larger than d. But we can also handle this case by obtaining an on-line algorithm with running time more closely parallel to that of Leighton, et. al. We could then interleave different routing algorithms to obtain an algorithm with running time on the order of the minimum of the running times of the individual algorithms.

Theorem 6 *Any set of packets can be routed on-line in $O(C + D\lg(Nd))$ steps using queues of size $O(\lg(Nd))$, with high probability.*

Proof. The proof is the same as for Theorem 5 except that we use $\alpha = 1/\lg(Nd)$ and $\tau = \Omega(\lg(Nd))$. ∎

3 Off-Line Schedule

In this section, we show that for any set of packets with edge-simple paths, there exists a schedule of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ using queues of size $O(d_{max})$. Our proof is nonconstructive, but it may still be helpful to know that such a schedule exists. For a communication pattern that is to be used often enough, it may be worthwhile to spend substantial off-line computation time determining an improved schedule. (In particular, this situation arises in network emulation problems as described in [5, 7].) That is, the basic structure of the proof is to show that there is some way of choosing delays for packets from specified ranges so as to achieve the bounds on schedule length and queue size indicated above. Given enough computation time, one could try every possible combination of delays for the packets to see which set of choices yields the desired schedule. It actually may also be possible to construct the schedule more efficiently by using recent results on algorithmic versions of the Lovász Local Lemma [1, 2]; success in the unit delay case has been reported by Leighton, et. al. [7].

Since part of the off-line proof parallels that of Leighton, et. al., we adopt a few of their definitions. In particular, a set of T consecutive time steps is referred to as a *T-frame* or a *frame of size* T. We also define the congestion in a frame to be the largest number of packets that traverse any edge during the frame. The *relative congestion* in a frame is the ratio of the congestion in the frame to the size of the frame.

The high-level strategy for producing a schedule with the desired bounds is specified in Procedure OFFLINE-ROUTING below.

procedure OFFLINE-ROUTING

1 Produce an unconstrained schedule of length $L = O(cd_{max} + D)$ in which the congestion is $O(d_{max})$ in each every frame of size T or greater, where T is $O(d_{max}^2)$.

2 Convert the schedule to one of length $O(L)$ in which at most $\gamma = O\Big(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ packets use an edge during any unit time step.

3 Flatten the schedule into a legitimate schedule of length $O(L\gamma)$.

The most complicated part of the procedure OFFLINE-ROUTING, and the part that parallels Leighton, et. al., is the first step. Due to the close parallels, we will only sketch the proof that step 1 can be achieved; the interested reader may find details in [4]. We will first prove that steps 2 and 3 can be achieved given a successful completion of step 1. Both steps 1 and 2 depend on the Lovász Local Lemma [10]:

Lemma 7 (Lovász Local Lemma) *Let $A_1, \cdots, A_m$ be events each occurring with dependence at most b, i.e., every one of the events is mutually independent of at least $m - b$ other events. If $\forall i \; \Pr\{A_i\} \leq p$ and $4pb < 1$, then the probability that none of these events occurs is greater than zero.* ∎

We now show that steps 2 and 3 of OFFLINE-ROUTING can be achieved given an unconstrained schedule (allowing several packets on the same edge at the same time as in Section 2) as specified in step 1:

Lemma 8 *Given an unconstrained schedule of length $L = O(cd_{max} + D)$ in which the congestion is $k = O(d_{max})$ in every frame of size T or greater, where T is $O(d_{max}^2)$, we can produce a legitimate schedule of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ using queues of size $O(d_{max})$.*

Proof. We begin by achieving step 2 of the procedure OFFLINE-ROUTING by independently rescheduling frames of size $\Theta(d_{max}^2)$. (We are able to make the frames independent by extending each one by d_{max} time steps so that each operation of routing a packet on an edge occurs entirely within one frame.)

Within each frame, each packet chooses a delay x randomly and uniformly from $[1, \alpha d_{max}^2]$, where α is a constant to be determined. The resultant schedule is of length $O(d_{max}^2)$. We claim that for $\eta = \Omega(\frac{\lg(d_{max})}{\lg\lg(d_{max})})$, there is a way of choosing x's such that at most η packets use an edge during any unit time step. The claim is proved by using the Lovász Local Lemma. For each edge e, we define a bad event as the event that more than η packets use the edge at some time step. For any packet, there are at most $d_e \leq d_{max}$ delays that cause it to be present on e at a particular time step, so the probability that it appears on e at a particular time is at most $\frac{d_{max}}{\alpha d_{max}^2}$. Then, since there are $\Theta(d_{max}^2)$ time steps in the frame, the probability p that a particular bad event occurs is $O\left(d_{max}^2 \binom{k}{\eta} \left(\frac{d_{max}}{\alpha d_{max}^2}\right)^\eta\right)$. Also, since at most k packets pass through any edge, and each of those packets passes through at most $\Theta(d_{max}^2)$ other edges in the frame, the dependence b is at most $kd_{max}^2 = O(d_{max}^3)$. Thus, for a sufficiently large constant α, we have $4pb < 1$ (by bounding the binomial coefficient as in the proof of Lemma 1), from which the claim follows.

Finally we achieve step 3 of the procedure OFFLINE-ROUTING by simply applying Lemma 2. For the queue size bound, notice that queues are of size $O(d_{max})$ at the end of step 2, since the congestion in each frame of size $\Theta(d_{max}^2)$ is $O(d_{max})$, and then the flattening process does not increase queue sizes by more than $\frac{\lg(d_{max})}{\lg\lg(d_{max})}$. ■

All that remains is to sketch the proof that step 1 of the OFFLINE-ROUTING procedure can be achieved:

Lemma 9 *Given any set of packets with edge-simple paths, we can produce an unconstrained schedule of length $O(cd_{max} + D)$ in which the congestion is $O(d_{max})$ in each every frame of size T or greater, where T is $O(d_{max}^2)$.*

Sketch of proof. Note first, that since our goal is to prove a bound of $O(cd_{max} + D)$ on the length of the unconstrained schedule, it suffices to assume that $cd_{max} = D$ and prove a bound of $O(cd_{max})$. Our strategy is to use an approach similar to Leighton, Maggs, and Rao [7, 8] of making a succession of refinements to the "greedy" schedule, in which packets never wait. In this succession of refinements, we bound the congestion in smaller and smaller intervals of time until the number of packets using an edge is at most $O(d_{max})$ during any set of $\Theta(d_{max}^2)$ consecutive time steps.

We begin with a special refinement that transforms the greedy schedule, S_0, into a schedule S_1 in which the relative congestion in each $(d_{max} \lg c)$-frame is $O\left(\frac{1}{d_{max}}\right)$. Each successive refinement transforms a schedule S_i with relative congestion at most r_i in any frame of size (at least) $d_{max}I_i$ (with $I_i = \Omega(d_{max})$) into a schedule S_{i+1} with relative congestion at most r_{i+1} in any frame of size (at least) $d_{max}I_{i+1}$, where $r_{i+1} \approx r_i$ and $I_{i+1} \ll I_i$. This overview is exactly the same as for Leighton, et. al., except that in their analysis d_{max} is replaced by 1.

For the initial refinement that produces S_1 from the greedy schedule of length $|S_0| = cd_{max}$, we assign each packet an integral delay x chosen randomly and uniformly from the interval $[1, \alpha cd_{max}]$ so that $|S_1| = (1+\alpha)cd_{max}$. Then we proceed with the following iterative refinement process to transform S_i into S_{i+1} until the desired unconstrained schedule is obtained.

Begin the ith refinement by breaking S_i into blocks of $(2I_i^3 + 2I_i^2 - I_i)d_{max}$ consecutive time steps and rescheduling each block independently. Within each block, we assign each packet an integral delay x chosen randomly and uniformly from $[1, I_i d_{max}]$. A packet assigned delay x is actually delayed once every I_i steps in the first xI_i steps. In order to make the packets end up in the same positions at the end of the rescheduled block as in the block of S_i (so that the blocks remain independent), we also insert a delay every I_i steps in the last $(I_i d_{max} - x)I_i$ steps, yielding rescheduled blocks of $(2I_i^3 + 2I_i^2)d_{max}$ steps. (Since we are allowing nonunit edge delays, some of the delays we have inserted may occur in the midst of an edge traversal rather than at a queue, but the delays can be moved to the nearest queue after all the refinements have been completed.)

It can be shown that each of the rescheduled blocks of $(2I_i^3 + 2I_i^2)d_{max}$ steps has appropriately reduced frame size except in the first and last $I_i^2 d_{max}$ steps. So we shift the boundaries of the rescheduled blocks so that each one contains a "fuzzy" region of size $2I_i^2 d_{max}$ at its center where the frame size needs to be reduced. Then we reduce the frame size in the fuzzy region by inserting delays in the $I_i^3 d_{max}$ steps before it and the $I_i^3 d_{max}$ steps after it. Each packet is assigned a delay from 1 to $I_i^2 d_{max}$. A packet with delay x waits once every $I_i^3 d_{max}/x$ steps before the fuzzy region and once every $I_i^3 d_{max}/(I_i^2 d_{max} - x)$ steps after the fuzzy region.

To prove that the initial refinement and each of the iterative refinements achieves the goals specified in the second paragraph of this proof sketch, we use the Lovász Local Lemmaas in the proof of Lemma 8. The proofs are similar to those of Leighton, et. al., the main difference being that we must recognize that $T + d_{max}$ delays to a given packet could cause it to be on a given edge during a give T-frame. We also assume that $c = \Omega(d_{max})$ in order to achieve the needed bound on dependence in the construction of the initial refinement; if $c = \Omega(d_{max})$ does not hold, we simply skip step 1 of the Offline-Routing procedure and apply step 2 directly to the greedy schedule.

The result of the analysis of the iterative refinement process may be summarized by the following lemma, which corresponds to the result of Leighton, et. al. with $d_{max} = 1$:

Lemma 10 *As long as $I_i = \Omega(d_{max})$, the ith refinement step described above decreases the frame size from $I_i d_{max}$ to $I_{i+1} d_{max} = (\lg^5 I_i) d_{max}$ for the entire schedule, while the relative congestion becomes $r_{i+1} = r_i(1 + O(1)/\sqrt{\lg I_i})$. Furthermore, we can arrange that every packet waits at most once every I_i steps in S_{i+1}.* ■

We perform refinement steps until we obtain a schedule S_j with $I_j = O(d_{max})$; note that $r_j = O(r_1) = O\left(\frac{1}{d_{max}}\right)$. At this point, we move delays that fall in the midst of an edge traversal to the edge queue for the corresponding edge, which has no effect on the congestion in any frame and does not require queues larger than $O(d_{max})$, since the congestion in frames of size $I_j d_{max}$ is $O(I_j) = O(d_{max})$. Since every packet waits at most once every $I_{j-1} = \Omega(d_{max})$ steps in S_j, we end up with each packet waiting at most once in each edge queue. Thus, we have obtained a schedule S^* of length $O(D) = O(cd_{max})$ and congestion $O(I_j) = O(d_{max})$ in each $I_j d_{max}$-frame. ■

Putting together Lemmas 9 and 8 gives us our final result:

Theorem 11 *For any set of packets with edge-simple paths, there exists a legitimate schedule of length* $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ *using queues of size* $O(d_{max})$. ■

4 Conclusion

We have shown that for any constant $\epsilon > 0$, we can route a set of N packets on-line with high probability in $O((C\lg^{\epsilon}(Nd) + D\lg(Nd))/\lg\lg(Nd))$ time. Also, for edge-simple paths, there exists a schedule of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$. An important open question is whether there exists a schedule of length closer to the lower bound of $C + D$. Also, it would be desirable to obtain improved on-line results and to construct universal networks for parallel computation, as in [3], based on these routing capabilities.

References

[1] N. Alon. A parallel algorithmic version of the local lemma. In *32nd Annual Symposium on Foundations of Computer Science*, pages 586–593. IEEE Computer Society Press, 1991.

[2] J. Beck. An algorithmic approach to the lovász local lemma I. Technical Report 91-21, DIMACS, 1991. To appear in *Random Structures and Algorithms*.

[3] R. I. Greenberg. The fat-pyramid and universal parallel computation independent of wire delay. *IEEE Trans. Computers*, 43(12):1358–1364, Dec. 1994.

[4] R. I. Greenberg and H.-C. Oh. Packet routing in networks with long wires. Technical Report UMIACS-TR-93-22, University of Maryland Institute for Advanced Computer Studies, 1993.

[5] R. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 227–240. ACM Press, 1989.

[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*, 17(1):157–205, July 1994.

[7] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–180, 1994.

[8] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.

[9] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. In *Proceedings of the 2nd Annual SIAM Symposium on Discrete Algorithms*, pages 148–157, 1991.

[10] J. Spencer. *Ten Lectures on the Probabilistic Method*. SIAM, 1987.

Ronald Greenberg received the A.B. degree in Mathematics, the B.S. degree in Computer Science and the B.S. and M.S. degrees in Systems Science and Mathematics all from Washington University, St. Louis, MO in 1983. He received the Ph.D. degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1989.

He is currently an Assistant Professor in the Electrical Engineering Department and the Institute for Advanced Computer Studies at the University of Maryland. His research interests include parallel computation and algorithms for computer-aided design of integrated circuits.

Hyeong-Cheol Oh received the B.S. degree in Electronics Engineering from Seoul National University, Seoul, Korea, and the M.S. degree in Electrical and Electronic Engineering from Korea Advanced Institute of Science and Technology, Seoul, Korea. He received the Ph.D. degree in Electrical Engineering at the University of Maryland, College Park in 1993.

He is currently an Assistant Professor in the Department of Information Engineering, Korea University, Chochiwon, Korea. He also worked for three years at Goldstar Semiconductor Ltd, Korea, where he designed NMOS full-custom and CMOS Gate-Array ICs. His research interests include parallel computation and VLSI design.