10-1992

Packet Routing in Networks with Long Wires

Ronald Greenberg
Rgreen@luc.edu

H.-C. Oh

Packet Routing in Networks with Long Wires

Ronald I. Greenberg and H.-C. Oh
Department of Electrical Engineering
University of Maryland
College Park, MD 20742
rig@eng.umd.edu and ohc@eng.umd.edu

October 22, 1992
Author version for Proceedings of 30th Annual Allerton Conference on Communication, Control, and Computing, pages 664–673, 1992

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 $O\left((C \lg^{\epsilon}(Nd) + D \lg(Nd))/\lg\lg(Nd)\right)$ 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 $O\left((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$, where d_{max} is the maximum wire delay in the network. These results improve upon those of Leighton, Maggs, and Rao, which assume that unit time suffices to traverse a wire of any length. Our results also improve upon those of Shmoys, Stein, and Wein 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 queueing 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 [1, 2, 3]. 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.)

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.

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 $O\left(c + d\lg(Nd)\right)$ with queues of size $O\left(\lg(Nd)\right)$, where N is the number of packets [2]. This naturally implies that in the problem with general edge delays, we could obtain a schedule of length $O\left(d_{max}(c + d\lg(Nd))\right)$ 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 $O\left((C\lg^{\epsilon}(Nd) + D\lg(Nd))/\lg\lg(Nd)\right)$ with queues of size $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$. 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 [4] 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 $O\left((C+D)\frac{\lg^2(Nd)}{\lg\lg(Nd)}\right)$. 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 $O(d_{max}(c+d))$. Though wwe have not removed both the $\lg(Nd)$ factor and all dependence on d_{max} in the off-line case, we show in Section 3 that there exists a schedule of length $O\left((cd_{max}+D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$ with queues of size $O(d_{max})$. 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 [2, 4], 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 $O\left((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$ with queues of size $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$ 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 $O\left((C\lg^{\epsilon}(Nd) + D\lg(Nd))/\lg\lg(Nd)\right)$ 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 for constructing the unconstrained schedule and the analysis of this phase are essentially the same as in the approach of Shmoys, Stein, and Wein. 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:

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

$$p \leq \binom{C}{\tau}\left(\frac{1}{C}\right)^{\tau}$$

$$\leq \left(\frac{eC}{\tau}\right)^{\tau}\left(\frac{1}{C}\right)^{\tau}$$
$$= \left(\frac{e}{\tau}\right)^{\tau},$$

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. In the more general context of nonunit wire delay, Shmoys, Stein, and Wein show how to flatten into a schedule of length $\gamma L \lg d_{max}$, but we show that a flattened schedule of length γL can be obtained by taking advantage of a distinction between packet routing and the general form of job-shop scheduling considered in [4]. In particular, any two packets that traverse the same edge spend the same amount of time in transit on that edge. 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.

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.*

Proof. The flattening process involves routing the packets on each edge e in order of their start times for traversing e in the unconstrained schedule (with ties broken arbitrarily). 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. (Figure 1 shows an example of an unconstrained schedule and its flattened version.)

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. Thus, each network node can decide when to dispatch packets on its outgoing edges by inspecting the packets and associated information queued at its incoming edges.

It remains to be shown that all packets are routed by time γL. Let us refer to the routing of a particular packet on 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 $t^{\omega}_{LE} \leq \gamma t^{\omega}_{UE}$.

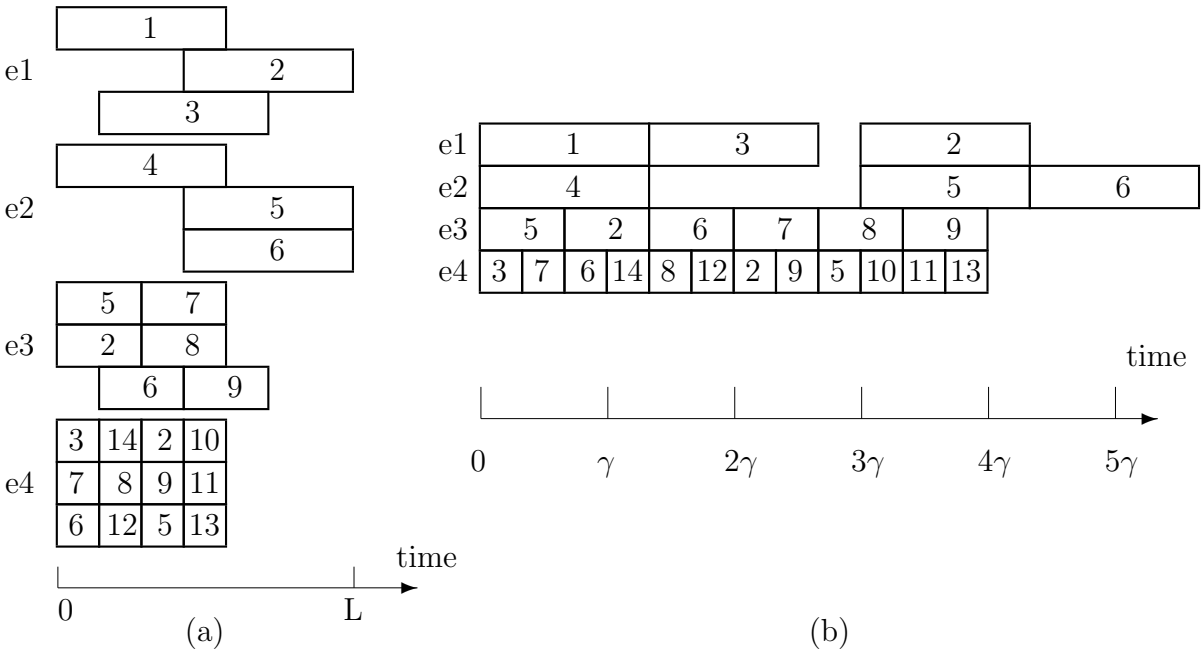

Figure 1: An example of the flattening process for four edges, e_1, e_2, e_3, and e_4. (a) The initial unconstrained schedule of $L = 7$ and $\gamma = 3$. (b) The flattened version of the schedule in (a).

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.) Since $t^{\omega}_{UB} \leq t$, we know from the induction hypothesis that any operation ω' for a packet m' on edge e with $t^{\omega'}_{UB} \leq t^{\omega}_{UB}$ has the property that all other operations for m' on edges that precede e in the path of m' complete by time $\gamma t^{\omega'}_{UB}$ in the legitimate schedule. This implies that there exists such an operation ω' with $t^{\omega'}_{LB} = \gamma t^{\omega'}_{UB}$; we let ω^* be the one maximizing $t^{\omega^*}_{LB}$. By the definition of ω^*, the legitimate schedule places some packet on edge e at every time step from $t^{\omega^*}_{LB}$ to t^{ω}_{LE}. All of these packets begin at times in $[t^{\omega^*}_{UB}, t^{\omega}_{UB}]$ in the unconstrained schedule and therefore end by time t^{ω}_{UE} (since all traversals of edge e take time d_e). Thus the completion time of ω in the legitimate schedule is

$$\begin{aligned} t^{\omega}_{LE} &\leq t^{\omega^*}_{LB} + \gamma(t^{\omega}_{UE} - t^{\omega^*}_{UB}) \\ &= \gamma t^{\omega^*}_{UB} + \gamma(t^{\omega}_{UE} - t^{\omega^*}_{UB}) \\ &= \gamma t^{\omega}_{UE} \ . \end{aligned}$$

■

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 $O\left((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$ steps using queues of size $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$, with high probability.* ■

We can remove the restriction on d_{max} by using a technique similar to [4], but again we must show how to perform the task on-line:

Theorem 4 *Any set of packets can be routed on-line in* $O\left((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$ *steps using queues of size* $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$*, 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 *With high probability, on-line routing of any set of packets can be achieved in* $O\left(\frac{1}{\epsilon}(C\lg^{\epsilon}(Nd)+D\lg(Nd))/\lg\lg(Nd)\right)$ *steps using queues of size* $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$.

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 $p\leq(Nd)^{-(k-1)}$ 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* $O\left(C+D\lg(Nd)\right)$ *steps using queues of size* $O\left(\lg(Nd)\right)$*, 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 $O\left((cd_{max}+D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$ using queues of size $O\left(d_{max}\right)$. Our proof is nonconstructive, but the result may provide useful information. For a communication pattern that is to be used often enough, it may be worthwhile to spend substantial off-line computation time determining a good schedule.

In this section, we make heavy use of the Lovász Local Lemma [5]:

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.* ■

Our strategy to obtain the off-line schedule consists of three stages. In the first stage, we use an approach similar to Leighton, Maggs, and Rao [2, 3] 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. In the second stage, we bound the number of packets using an edge during any unit time step to $O\left(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$. In the final stage, we produce a legitimate schedule by applying the flattening process described in Section 2.

We only sketch the iterative refinement process of the first stage, since it closely parallels that of Leighton, et. al. First, we review a few definitions. 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. 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 $I_{i+1} \ll I_i$.

For the initial refinement that produces S_1 from the greedy schedule of length $|S_0| = D$, we assign each packet an integral delay x chosen randomly and uniformly from the interval $[1, \alpha c d_{max}]$. The resultant schedule is of length $\alpha c d_{max} + D$. Without loss of generality, we assume that $c d_{max} = D$, so $|S_1| = (1 + \alpha) c d_{max}$. We can also assume that $c = \Omega(d_{max})$; otherwise we skip this stage and apply Lemma 9 directly to the greedy schedule.

We claim that there is some way of choosing x's so that for $I_1 = \lg c$, the relative congestion is $O\left(\frac{1}{d_{max}}\right)$ in any frame of size (at least) $I_1 d_{max}$ in S_1. We prove this claim by using the Lovász Local Lemma. For each edge e, we define a bad event as the event that more than $\lg c$ packets use e during any $(d_{max} \lg c)$-frame. 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. Thus, the probability that it appears on e during a particular $(d_{max} \lg c)$-frame is at most $\frac{d_{max} \lg c + d_{max}}{\alpha c d_{max}}$. So, the probability that any bad event occurs is

$$p \leq (1 + \alpha) c d_{max} \binom{c}{\lg c} \left(\frac{2 d_{max} \lg c}{\alpha c d_{max}}\right)^{\lg c} ,$$

since the number of $(d_{max} \lg c)$-frames is certainly at most $(1 + \alpha) c d_{max}$. Also, at most c packets pass through an edge, and each of these packets passes through at most $d \leq D$ other edges. Therefore, since we were able to assume $c d_{max} = D$ and $c = \Omega(d_{max})$, we have a dependence b of at most $cD = O(c^3)$. Thus, for a sufficiently large constant α, we have $4pb < 1$, from which the claim follows for frames of size *exactly* $I_1 d_{max}$. The proof is easily extended to apply to all frames of size *at least* $I_1 d_{max}$, and we henceforth do not make this distinction.

We now sketch the remainder of the refinement process. The first step in the ith refinement is to break S_i into blocks of $(2I_i^3 + 2I_i^2 - I_i)d_{max}$ consecutive time steps and reschedule 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. 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 we will show how to move the delays to queues later. If $2I_i^3 + 2I_i^2 - I_i < I_{i-1}$, which holds as long as I_i is larger than some constant, a packet has been delayed at most once in the entire block before we insert new delays. If any new delay would be within I_i steps of the single old delay, it is not inserted. Omitting these delays has a negligible effect on the probability analysis, but it allows us to maintain the invariant that every packet waits at most once every I_i steps in S_{i+1}.

It can be shown that there is some way of choosing the delays so that in between the first and last $I_i^2 d_{max}$ steps of a block, we can decrease the frame size from $I_i d_{max}$ to $(\lg^2 I_i)d_{max}$ without increasing the relative congestion much. Furthermore, we can keep the relative congestion from increasing much in the first and last $I_i^2 d_{max}$ steps by increasing the frame size in these regions from $I_i d_{max}$ to $I_i^2 d_{max}$. Now, we move the block boundaries so that the regions with the larger frame size form a "fuzzy region" at the center of each new block. Finally, we decrease the frame size in the fuzzy regions with little increase in the frame size elsewhere through another round of assigning random delays to packets. This time packets choose integral delays randomly and uniformly from $[1, I_i^2 d_{max}]$. A packet with delay x waits once every $(I_i^3 d_{max}/x)$ steps in the first $I_i^3 d_{max}$ steps (i.e., before the fuzzy region) and once every $I_i^3 d_{max}/(I_i^2 d_{max} - x)$ steps in the last $I_i^3 d_{max}$ steps (after the fuzzy region).

Lemma 8 summarizes the result of this refinement procedure:

Lemma 8 *As long as $I_i = \Omega(d_{max})$, the ith refinement step described above decreases the frame size from $I_i d_{max}$ to $I_{i+1} d_{max} = (\lg^4 I_i)d_{max}$ for the entire schedule, while the relative congestion becomes $r_{i+1} = r_i(1 + O(1)/\sqrt{\lg I_i})$.*

Proof. This lemma can be proved in a similar fashion to [2, 3]. ■

We perform refinement steps until we obtain a schedule S_j with $I_j = O(d_{max})$. In S_j, the congestion in frames of size $O(I_j d_{max}) = O(d_{max}^2)$ is $O(I_j) = O(d_{max})$. 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 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 $O(cd_{max})$ and congestion $O(d_{max})$ in each $O(d_{max}^2)$-frame.

Now we proceed to the second stage of our off-line construction, which we apply to frames of size $\Theta(d_{max}^2)$ that have congestion $O(d_{max})$. (Recall that the refinement stage actually bounds the relative congestion in all frames of size at least T for some T that is $O(d_{max}^2)$.) The required result for the second stage is embodied in the following lemma:

Lemma 9 *Suppose we have a schedule S_U with congestion $k = O(d_{max})$ and $|S_U| = O(kd_{max})$. Then, there is a schedule, of length $O(kd_{max})$, in which at most $O\left(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$ packets use an edge during any unit time step.*

Proof. Each packet chooses a delay x randomly and uniformly from $[1, \alpha k d_{max}]$, where α is a constant to be determined. The resultant schedule is of length $\alpha k d_{max} + |S_U| = O(kd_{max})$.

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. Again we use the Lovász Local Lemma. For each edge, the bad event of more than η packets using the edge at some time step occurs with probability p at most $O(kd_{max})\binom{k}{\eta}\left(\frac{d_{max}}{\alpha k d_{max}}\right)^{\eta}$. The dependence b is at most $k|S_U| \leq O(d_{max}^3)$. Thus, for a $\alpha \geq e$ and $\eta = \Omega(\frac{\lg(d_{max})}{\lg\lg(d_{max})})$, we have $4pb < 1$, from which the claim follows. ■

To complete the second stage we use Lemma 9 to independently reschedule frames of size $\Theta(d_{max}^2)$ with congestion $O(d_{max})$ in S^*. (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.) We end up with a schedule of length $O(cd_{max})$ with at most $O\left(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$ packets on an edge at any time step and queues that still have size $O(d_{max})$.

Finally, for the third stage of the off-line construction, we simply apply Lemma 2, and we have proved the following theorem:

Theorem 10 *For any set of packets with edge-simple paths, there exists a legitimate schedule of length $O\left((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\right)$ using queues of size $O(d_{max})$.* ■

References

[1] F. T. Leighton, Bruce M. Maggs, Abhiram G. Ranade, and Satish B. Rao. Randomized routing and sorting on fixed-connection networks. Manuscript, 1991.

[2] F. T. Leighton, Bruce M. Maggs, and Satish B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. Manuscript, 1991.

[3] Tom Leighton, Bruce Maggs, and Satish Rao. Universal packet routing algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.

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

[5] Joel Spencer. *Ten Lectures on the Probabilistic Method.* SIAM, 1987.