COMBINATORICA
Bolyai Society – Springer-Verlag

COMBINATORICA **19** (2) (1999) 1–27

FAST ALGORITHMS FOR FINDING O(Congestion + Dilation) PACKET ROUTING SCHEDULES

TOM LEIGHTON, BRUCE MAGGS and ANDRÉA W. RICHA

Received July 8, 1996

In 1988, Leighton, Maggs, and Rao showed that for any network and any set of packets whose paths through the network are fixed and edge-simple, there exists a schedule for routing the packets to their destinations in $O(c+d)$ steps using constant-size queues, where c is the congestion of the paths in the network, and d is the length of the longest path. The proof, however, used the Lovász Local Lemma and was not constructive. In this paper, we show how to find such a schedule in $O(m(c+d)(\log\mathcal{P})^4(\log\log\mathcal{P}))$ time, with probability $1-1/\mathcal{P}^{\beta}$, for any positive constant β, where $\mathcal{P}$ is the sum of the lengths of the paths taken by the packets in the network, and m is the number of edges used by some packet in the network. We also show how to parallelize the algorithm so that it runs in NC. The method that we use to construct the schedules is based on the algorithmic form of the Lovász Local Lemma discovered by Beck.

1. Introduction

In this paper, we consider the problem of scheduling the movements of packets whose paths through a network have already been determined. The problem is formalized as follows. We are given a network with n nodes (switches) and m edges (communication channels). Each node can serve as the source or destination of an arbitrary number of *packets* (or *cells* or *flits*, as they are sometimes referred to). Let N denote the total number of packets to be routed. The goal is to route the N packets from their origins to their destinations via a series of synchronized time steps, where at each step at most one packet can traverse each edge and each packet can traverse at most one edge. Without loss of generality, we assume that all edges in the network are used by the path of some packet, and thus that m gives the number of such edges (all the other edges are irrelevant to our problem).

Mathematics Subject Classification (1991):

Tom Leighton is supported in part by ARPA Contracts N00014-91-J-1698 and N00014-92-J-1799. Bruce Maggs is supported in part by an NSF National Young Investigator Award under Grant No. CCR–94–57766, with matching funds provided by NEC Research Institute, and by ARPA Contract F33615–93–1–1330. Part of this research was conducted while Andréa Richa was at Carnegie Mellon University, supported by NSF National Young Investigator Award under Grant No. CCR–94–57766, with matching funds provided by NEC Research Institute, and ARPA Contract F33615–93–1–1330.

0209–9683/99/$6.00

Figure 1 shows a 5-node network in which one packet is to be routed to each node. The shaded nodes in the figure represent switches, and the edges between the nodes represent channels. A packet is depicted as a square box containing the label of its destination.

During the routing, packets wait in three different kinds of queues. Before the routing begins, packets are stored at their origins in special *initial queues*. When a packet traverses an edge, it enters the *edge queue* at the end of that edge. A packet can traverse an edge only if at the beginning of the step, the edge queue at the end of that edge 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. In Figure 1, all of the packets reside in initial queues. For example, packets 4 and 5 are stored in the initial queue at node 1. In this example, each edge queue is empty, but has the capacity to hold two packets. Final queues are not shown in the figure. Independent of the routing algorithm used, the sizes of the initial and final queues are determined by the particular packet routing problem to be solved. Thus, any bound on the maximum queue size required by a routing algorithm refers only to the edge queues.

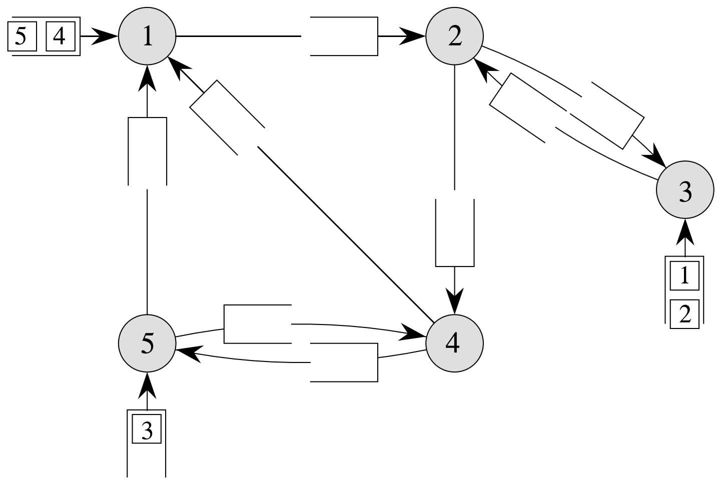

Fig. 1. A graph model for packet routing

We focus on the problem of timing the movements of the packets along their paths. A *schedule* for a set of packets specifies which move and which wait at each time step. The *length* of a schedule is the number of time steps required to route all the packets to their destinations according to the schedule. Given any underlying network, and any selection of paths for the packets, our goal is to produce a schedule for the packets that minimizes the length of the schedule and the maximum queue size needed to route all of the packets to their destinations.

Of course, there is a strong correlation between the time required to route the packets and the selection of the paths. In particular, the maximum distance, d, traveled by any packet is always a lower bound on the time. We call this distance the *dilation* of the 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. Figure 2 shows a set of paths for the packets of Figure 1 with dilation 3 (since the path followed by the packet going from node 5 to node 3 has length 3) and congestion 3 (since three paths use the edge between nodes 1 and 2).

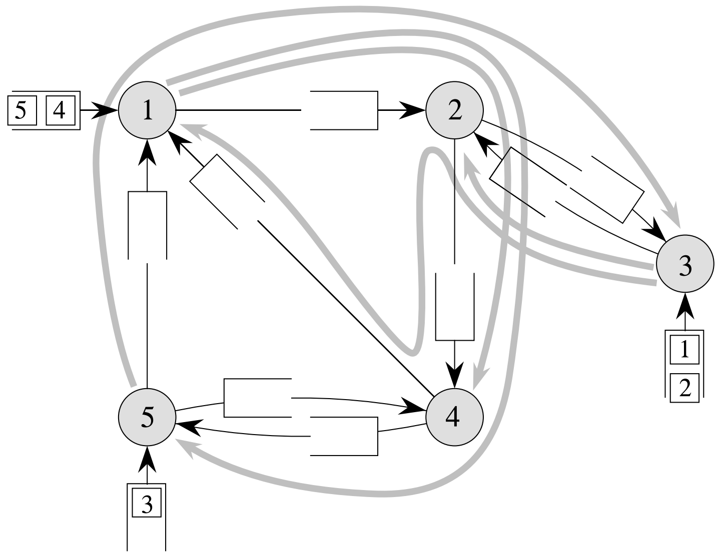

Fig. 2. A set of paths for the packets with dilatation $d=3$ and congestion $c=3$

Given any set of paths with congestion c and dilation d, in any network, it is straightforward to route all of the packets to their destinations in cd steps using queues of size c at each edge. In this case the queues are big enough that a packet can never be delayed by a full queue in front, so each packet can be delayed at most $c-1$ steps at each of at most d edges on the way to its destination.

In [9], Leighton, Maggs, and Rao showed that there are much better schedules. In particular, they established the existence of a schedule using $O(c+d)$ steps and constant-size queues at every edge, thereby achieving the naive lower bounds (up to constant factors) for any routing problem. The result is highly robust in the sense that it works for any set of edge-simple paths and any underlying network. (Ā priori, it would be easy to imagine that there might be some set of paths on some network that requires more than $\Omega(c+d)$ steps or larger than constant-size queues to route all the packets.) The method that they use to show the existence of optimal schedules, however, is not constructive. In other words, prior to this work, the fastest known algorithms for producing schedules of length $O(c+d)$ with constant-size edge queues required time that is exponential in the number of packets.

1.1. Our results

In this paper, we show how to produce schedules of length $O(c+d)$ in $O(m(c+d)(\log \mathcal{P})^4(\log\log \mathcal{P}))$ time steps, with probability at least $1-1/\mathcal{P}^\delta$, for any constant $\delta > 0$, where m is the number of distinct edges traversed by some packet in the network. The schedules can also be found in polylogarithmic time on a parallel computer using $O(m(c+d)(\log \mathcal{P})^4(\log\log \mathcal{P}))$ work, with probability at least $1-1/\mathcal{P}^\delta$.

The algorithm for producing the schedules is based on an algorithmic form of the Lovász Local Lemma (see [6] or [20, pp. 57–58]) discovered by Beck [3]. Showing how to modify Beck's arguments so that they can be applied to scheduling problems is the main contribution of our work. Once this is done, the construction of asymptotically optimal routing schedules is accomplished using the methods of [9].

The result has several applications. For example, if a particular routing problem is to be performed many times over, then it may be feasible to compute an asymptotically optimal schedule once using global control. This situation arises in network emulation problems.

Typically, a guest network G can be emulated by a host network H by embedding G into H. An *embedding* maps nodes of G to nodes of H, and edges of G to paths in H — an edge (x,y) of G is mapped to some path in H between the nodes of H that x and y were mapped to. There are three important measures of an embedding: the load, congestion, and dilation. The *load of an embedding* is the maximum number of nodes of G that are mapped to any one node of H. The *congestion of an embedding* is the maximum number of paths corresponding to edges of G that use any one edge of H. The *dilation of an embedding* is the length of the longest path of H in the embedding. Let l, c, and d denote the load, congestion, and dilation of the embedding, respectively.

Once G has been embedded in H, H can emulate G in a step-by-step fashion as follows. Each node of H first emulates the local computations performed by the l (or fewer) nodes mapped to it. This takes $O(l)$ time. Then for each packet sent along an edge of G, H sends a packet along the corresponding path in the embedding. The algorithm described in this paper can be used to produce a schedule in which the packets are routed to their destinations in $O(c+d)$ steps. Thus, H can emulate each step of G in $O(l+c+d)$ steps, where l is the load of this embedding.

Our packet routing result also has applications to job-shop scheduling. In particular, consider a scheduling problem with jobs $j_1, \ldots, j_r$, and machines $m_1, \ldots, m_s$, for which each job must be performed on a specified sequence of machines. In this application, we assume that each job occupies each machine that works on it for a unit of time, and that no machine has to work on any job more than once. Of course, the jobs correspond to packets, and the machines correspond to edges Hence, we can define the *dilation* of the scheduling problem to be the maximum number of machines that must work on any job, and the *congestion* to be the maximum number of jobs that have to be run on any machine. As a consequence of the packet routing result, we know that any scheduling problem can be solved in $O(c+d)$ steps. In addition, we know that there is a schedule for which each job waits at most $O(c+d)$ steps before it starts running, and that each job waits at most a constant number of steps in between consecutive machines. The queue of jobs waiting for any machine will also always have at most a constant number of jobs.

1.2. Related work

Scheideler recently presented in [18] an alternative simpler proof (than that of [9]) for the existence of $O(c+d)$-step schedules that only require edge queues of size 2. The main idea in his proof is to decompose the problem in a different way by using so-called "secure edges".

In [13], Meyer auf der Heide and Scheideler showed the existence of an off-line protocol that only requires edge queues of size 1. However, the schedule produced by this protocol has length $O([d+c(\log(c+d))(\log\log(c+d))]\log\log\log^{(1+\epsilon)}(c+d))$, for any constant $\epsilon > 0$.

For the class of *leveled* networks, Leighton, Maggs, Ranade, and Rao [8] showed that there is a simple on-line randomized algorithm for routing the packets to their destinations within $O(c+L+\log N)$ steps, with high probability, where L is the number of levels in the network, and N is the total number of packets. (In a leveled network with L levels, each node is labeled with a level number between 0 and $L-1$, and every edge that has its tail on level i has its head on level $i+1$, for $0 \le i < L-1$.)

Mansour and Patt-Shamir [12] showed that if packets are routed greedily on shortest paths, then all of the packets reach their destinations within $d+N$ steps. These schedules may be much longer than optimal, however, because N may be much larger than c. Meyer auf der Heide and Vöcking [14] devised a simple on-line randomized algorithm that routes all packets to their destinations in $O(c+d+\log N)$ steps, with high probability, provided that the paths taken by the packets are shortcut free (e.g., shortest paths).

Recently, Rabani and Tardos [16], and Ostrovsky and Rabani [15] extended the main ideas used in [9], and in the centralized algorithm presented in this work, to obtain on-line local control algorithms for the general packet routing problem that produce near-optimal schedules. More specifically, Rabani and Tardos [16] show a randomized on-line algorithm that with high probability delivers all packets in $O(c+d((\log^* N)^{O(\log^* N)})+\log^6 N)$ steps; Ostrovsky and Rabani [15] improved on this result by presenting a randomized on-line algorithm that delivers all the packets to their destinations in $O(c+d+\log^{(1+\epsilon)} N)$ steps with high probability, for arbitrary $\epsilon > 0$.

It was also in recent work that Srinivasan and Teo [21] answered a long-standing question: Given source and destination nodes for each packet, can we select the routing paths for the N packets, with congestion c and dilation d, in order to approximate the minimum value of $c + d$ (over all possible choices of paths) to within a constant factor? (Finding the minimum value of $c+d$ is NP-hard.) They provided an algorithm that selects such paths in polynomial time; by applying our algorithm on the selected paths, Srinivasan and Teo described the first off-line constant factor approximation algorithm for routing N packets (if we are only given the source and destination nodes of each packet) using constant-size queues. It is interesting to note that there is still no polynomial-time algorithm known for which the congestion c alone is asymptotically optimal: It was clever (and crucial) that Srinivasan and Teo minimized the sum $c+d$ rather than just c.

The problem of scheduling packets through given paths strongly relates to network emulations via embeddings, as we discussed. Koch et al. in [7], and Maggs and Schwabe in [11] address the problem of performing network emulations via embeddings.

Shmoys, Stein, and Wein [19] give randomized and deterministic algorithms that produce schedules of length within a polylogarithmic factor of that of an optimal schedule, for a more general job-shop scheduling problem than the one we considered, when jobs are not assumed to have unit length and a machine may have to work on the same job more than once.

1.3. Outline

The remainder of this paper is divided as follows. In Section 2, we give a very brief overview of the non-constructive proof in [9]. We also introduce some definitions, and present two important propositions and a new lemma that will be of later use. In Section 3, we describe how to make the non-constructive method in [9] constructive. In Section 4, we analyze the running time of the algorithm. The propositions presented in Sections 2 and 3 are meant to replace (and are numbered according to) some of the lemmas in [9].

In Section 5, we show how to parallelize the scheduling algorithm. We conclude with some remarks in Section 6.

2. Preliminaries

In [9], Leighton, Maggs, and Rao proved that for any set of packets whose paths are edge-simple and have congestion c and dilation d, there is a schedule of length $O(c+d)$ in which at most one packet traverses each edge of the network at each step, and at most a constant number of packets wait in each queue at each step. (An *edge-simple* path uses no edge more than once.) Note that there are no restrictions on the size, topology, or degree of the network or on the number of packets.

The strategy for constructing an efficient schedule is to make a succession of refinements to the "greedy" schedule, S_0, in which each packet moves at every step until it reaches its final destination. This initial schedule is as short as possible: Its length is only d. Unfortunately, as many as c packets may traverse an edge at a single time step in S_0, whereas in the final schedule at most one packet is allowed to traverse an edge at each step. Each refinement will bring us closer to meeting this requirement.

The proof uses the Lovász Local Lemma ([6] or [20, pp. 57–58]) at each refinement step. Given a set of "bad" events in a probability space, the lemma provides a simple inequality that, when satisfied, guarantees that with probability greater than zero, no bad event occurs. The inequality relates the probability that each bad event occurs with the dependence among them. A set of bad events $A_1, A_2, \ldots, A_q$ in a probability space has *dependence* at most b if every event is mutually independent of some set of $q-b-1$ other events. The lemma is non-constructive; for a discrete probability space, it shows only that there exists some elementary outcome that is not in any bad event.

Lemma [Lovász]. *Let $A_1, A_2, \ldots, A_q$ be a set of "bad" events, each occurring with probability p, and with dependence at most b. If $4pb < 1$, then with probability greater than zero, no bad event occurs.* ∎

Before proceeding, we need to introduce some notation. A *T-frame* — or a *frame of size T* — is a sequence of T consecutive time steps in a given schedule. The *congestion* of an edge g in a T-frame is the number of packets that traverse g in this frame; the *relative congestion* of edge g in a T-frame is given by the congestion of g in the frame divided by the frame size T. The *frame congestion* in a T-frame is the largest congestion of an edge in the frame; the *relative congestion* in a T-frame is the largest relative congestion of an edge in the frame.

2.1. A pair of tools for later use

In this section we re-state Lemma 3.5 of [9] and we prove Proposition 3.6, which replaces Lemma 3.6 of [9]. Both will be used in the proofs in Section 3.

Lemma 3.5. [9] *In any schedule, if the number of packets that traverse a particular edge g in any y-frame is at most Ry, for all y between T and $2T-1$, then the number of packets that traverse g in any y-frame is at most Ry, for all $y \geq T$.*

Proof. Consider a frame τ of size T', where $T' > 2T-1$. The first $(\lfloor T'/T \rfloor - 1)T$ steps of the frame can be broken into T-frames. In each of these T-frames, at most

RT packets traverse g. The remainder of the T'-frame τ consists of a single y-frame, where $T \le y \le 2T-1$, in which at most Ry packets traverse g. ∎

The following proposition is basically a re-statement of Lemma 3.6 of [9] and will be used here in place of this lemma. Proposition 3.6 applies when the number of distinct edges traversed by the packets in the schedule considered, m', is at most a polynomial in I (I as defined below).

Proposition 3.6. *Suppose that (i) there are positive constants $\alpha_1, \alpha_2, \alpha_3$, where $\alpha_1 \ge \alpha_2$, $\alpha_1 \ge 2\alpha_3$, and $\alpha_3 \ge \alpha_2$; (ii) I is greater than some sufficiently large constant; and (iii) for all edges g, in a schedule of length I^{α_1} (or smaller), the relative congestion of edge g in frames of size I^{α_2} or larger is at most ρ_g, where ρ_g is a constant. Let m' be the number of distinct edges traversed by the packets in this schedule. Furthermore, suppose that each packet is assigned a delay chosen randomly, independently, and uniformly from the range $[0, I^{\alpha_2}]$, and that if a packet is assigned a delay of x, then x delays are inserted in the first I^{α_3} steps and $I^{\alpha_2}-x$ delays are inserted in the last I^{α_3} steps of the schedule.*

1. *Then for any constant $\xi > 0$, there exists a constant $k_1 > 0$ such that with probability at least $1 - m'/I^{\xi}$ the relative congestion of any edge g in any frame of size $\log^2 I$ or larger, in between the first and last I^{α_3} steps of the new schedule is at most $\rho_g(1+\sigma)$, for $\sigma = k_1/\sqrt{\log I}$.*
2. *We can find such a schedule and verify whether it satisfies the conditions in 1. in $O(m' I^{\alpha_1}(\log^2 I))$ time.*

Proof. To bound the relative congestions of each edge in frames of size $\log^2 I$ or larger, we need to consider all m' edges and, by Lemma 3.5, all frames of size between $\log^2 I$ and $2\log^2 I - 1$.

As we shall see, the number of packets that traverse an edge g during a particular T-frame τ has a binomial distribution. In the new schedule, a packet can traverse g during τ only if in the original schedule it traversed g during τ or during one of the I^{α_2} steps before the start of τ. Since the relative congestion of edge g in any frame of size I^{α_2} or greater in the original schedule is at most ρ_g, there are at most $\rho_g(I^{\alpha_2}+T)$ such packets. The probability that an individual packet that could traverse g during τ actually does so is at most T/I^{α_2}. Thus, the probability p that ρ'_g or more packets traverse an edge g during a particular T-frame τ is at most

$$p \le \sum_{k=\rho'_g}^{\rho_g(I^{\alpha_2}+T)} \binom{\rho_g(I^{\alpha_2}+T)}{k} \left(\frac{T}{I^{\alpha_2}}\right)^k \left(1 - \frac{T}{I^{\alpha_2}}\right)^{\rho_g(I^{\alpha_2}+T)-k}.$$

To estimate the area under the tails of this binomial distribution, we use the following Chernoff-type bound [5]. Suppose that there are x independent Bernoulli trials, each of which is successful with probability p'. Let S denote the number of successes in the x trials, and let $\mu = E[S] = xp'$. Following Angluin and Valiant [2], we have

$$\Pr[S \geq (1+\gamma)\mu] \leq e^{-\gamma^2 \mu/3}$$

for $0 \leq \gamma \leq 1$. (Note that e denotes the base of the natural logarithm.)

In our application, $x = \rho_g(I^{\alpha_2} + T)$, $p' = T/I^{\alpha_2}$, and so $\mu = \rho_g(I^{\alpha_2} + T)T/I^{\alpha_2}$. For $\gamma = \sqrt{3k_0/\log I}$ (where k_0 is a positive constant to be specified later), $\rho_g \geq 1$, and $T \geq \log^2 I$, we have $\Pr[S \geq (1+\gamma)\mu] \leq e^{-k_0\rho_g(I^{\alpha_2}+T)T/(I^{\alpha_2}\log I)} \leq e^{-k_0 \log I} \leq e^{-k_0 \ln I} = 1/I^{k_0}$. Set $\rho'_g T$ to be $(1+\gamma)\mu = (1+\sqrt{3k_0/\log I})\rho_g(I^{\alpha_2} + T)T/I^{\alpha_2}$. For I large enough, $2\log^2 I/I^{\alpha_2} \leq 1/\sqrt{\log I}$, and thus $\rho'_g \leq \rho_g(1 + k_1/\sqrt{\log I})$, for some constant $k_1 \geq 2\sqrt{3k_0} + 1$. Let $\sigma = k_1/\sqrt{\log I}$. Then $\rho'_g \leq \rho_g(1+\sigma)$. Thus $p = \Pr[S \geq \rho'_g T] \leq \Pr[S \geq (1+\gamma)\mu] \leq 1/I^{k_0}$.

Since there are at most $(I^{\alpha_1} + I^{\alpha_2}) \leq 2I^{\alpha_1}$ starting points for a frame, and $\log^2 I$ different size frames starting at each point, and there are at most m' distinct edges per frame, the probability that the relative congestion of any edge g is more than ρ'_g in any frame is at most $m' I^{\alpha_1} \log^2 I/I^{k_0} \leq m'/I^{k_0-\alpha_1-2}$ (since $\log^2 I \leq I^2$). For any $\xi > 0$, we set $k_0 = \xi + \alpha_1 + 2$, which in turn sets k_1 and σ, and completes the proof of part 1.

We assign a random delay to each packet, and verify whether the conditions in 1. apply as follows. We construct the schedule by routing all the packets one step at a time. At time t, for $I^{\alpha_3} \leq t \leq (I^{\alpha_1} + I^{\alpha_2} - I^{\alpha_3})$, we compute the congestion of edge g in a T-frame ending at t, for all edges g that are traversed by some packet in the schedule, for all $T \in [\log^2 I, 2\log^2 I - 1]$ using the following rules: (i) if $T = \log^2 I$ then the congestion of g in a T-frame ending at time t can be computed by taking the congestion of g in the T-frame ending at $t-1$, subtracting the number of packets that traverse edge g at time $t-T$ and adding the number of packets that traverse g at time t; otherwise (ii) if $t \geq T$, then the congestion of edge g in a T-frame that ends at t is given by the congestion of g in a $(T-1)$-frame that ends at $t-1$ plus the number of packets that traverse edge g at time t. The relative congestion of an edge in a frame is given by the congestion of the edge in the frame divided by the size of the frame. This can be done in time $O(m'(I^{\alpha_1} + I^{\alpha_2})\log^2 I) = O(m' I^{\alpha_1} \log^2 I)$, m' being the number of distinct edges traversed in this schedule. ∎

In the remainder of this paper, for a schedule of size polynomial in I, we assume that we check for the congestions of all T-frames, $\log^2 I \leq T < 2\log^2 I$, of the schedule as described in the proof of Proposition 3.6.

3. An algorithm for constructing optimal schedules

In this section, we describe the key ideas required to make the non-constructive proof of [9] constructive. There are many details in that proof, but changes are required only where the Lovász Local Lemma is used, in Lemmas 3.2, 3.7, and 3.9 of [9]. The non-constructive proof showed that a schedule can be modified by assigning delays to the packets in such a way that in the new schedule the relative congestion can be bounded in much smaller frames than in the old schedule. In this paper, we show how to find the assignment of delays quickly. We will not regurgitate the entire proof of [9], but only reprove those three lemmas, trying to state the replacement propositions in a way as close as possible to the original lemmas.

In Section 3.1, we provide a proposition, Proposition 3.2, that is a constructive version of Lemma 3.2 of [9]. In Sections 3.2 and 3.3, we provide three propositions that are meant to replace Lemma 3.7 of [9]. Lemma 3.7 is applied $O(\log^*(c+d))$ times in [9]. We will use Propositions 3.7.1 and 3.7.2 to replace the first two applications of Lemma 3.7. The remaining applications will be replaced by Proposition 3.7.3. In Section 3.4, we present the three replacement propositions for Lemma 3.9 of [9]. Our belief is that a reader who understands the structure of the proof in [9] and the propositions in this paper can easily see how to make the original proof constructive. We analyze the running time of our algorithm in Section 4.

3.1. The first reduction in frame size

For a given set of N packets, let c and d denote the congestion and dilation of the paths taken by these packets, and let $\mathcal{P}$ denote the sum of the lengths of these paths. Note that $m \leq \mathcal{P} \leq mc$, and that $c, d < \mathcal{P}$, where m is the number of edges traversed by some packet in the network. The following proposition is meant to replace Lemma 3.2 of [9]. It is used just once in the proof, to reduce the frame size from d to $\log \mathcal{P}$.

Proposition 3.2. *For any constant $\beta > 0$, there is a constant $\alpha > 0$, such that there exists an algorithm that constructs a schedule of length $d + \alpha c$ in which packets never wait in edge queues and in which the relative congestion in any frame of size $\log \mathcal{P}$ or larger is at most 1. The algorithm runs in $O(m(c+d)(\log \mathcal{P}))$ time steps, and succeeds with probability at least $1 - 1/\mathcal{P}^{\beta}$.*

Proof. The algorithm is simple: Assign each packet an initial delay that is chosen randomly, independently, and uniformly from the range $[0, \alpha c]$, where α is a constant that will be specified later; the packet will wait out its initial delay and then travel to its destination without stopping. The length of the new schedule is at most $\alpha c + d$.

To bound the relative congestion in frames of size $\log \mathcal{P}$ or larger, we need to consider all m edges and, by Lemma 3.5, all frames of size between $\log \mathcal{P}$ and $2\log \mathcal{P} - 1$. We assume without loss of generality that $c > 2\log \mathcal{P}$, since any frame of length c or larger has relative congestion at most 1. For any particular edge g, and T-frame τ, where $\log\mathcal{P} \le T \le 2\log\mathcal{P} - 1$, the probability p that more than T packets use g in τ is at most

$$\begin{aligned} p &\le \sum_{i=T+1}^{c} \binom{c}{i} \left(\frac{T}{\alpha c}\right)^{i} \left(1 - \frac{T}{\alpha c}\right)^{c-i} \\ &\le \binom{c}{T+1} \left(\frac{T}{\alpha c}\right)^{(T+1)} \\ &\le \left(\frac{e}{\alpha}\right)^{(T+1)} \end{aligned}$$

since each of the at most c packets that pass through g has probability at most $T/\alpha c$ of using g in τ, and since $\binom{a}{b} \le (ae/b)^b$, for any $0 < b \le a$. The total number of frames to consider is at most $(\alpha c + d)\log\mathcal{P}$, since there are at most $\alpha c + d$ places for a frame to start and $\log\mathcal{P}$ frame sizes. Thus the probability that the relative congestion of any edge is too large in any frame of size $\log\mathcal{P}$ or larger is at most

$$m(\log\mathcal{P})(\alpha c + d)\left(\frac{e}{\alpha}\right)^{\log\mathcal{P}}.$$

We bound the probability above by summing the probabilities that the relative congestion is too large for all $m\log\mathcal{P}$ edge-frame pairs. Using the inequalities $\mathcal{P} \ge c$, $\mathcal{P} \ge m$, and $\mathcal{P} \ge d$, we have that for any constant $\beta > 0$, there exists a constant $\alpha > 0$, such that this probability is at most $1/\mathcal{P}^{\beta}$.

Assigning the delays to the packets takes $O(N)$ time steps. Verifying whether the relative congestion is at most 1 in any T-frame of size $\log\mathcal{P} \le T \le 2\log\mathcal{P} - 1$ can be done in $O(m(c+d)(\log\mathcal{P}))$ time steps (as described in the last paragraph of the proof of Proposition 3.6). ∎

Before applying Proposition 3.7.1, we first apply Proposition 3.2 to produce a schedule S_1 of length $O(c+d)$ in which the relative congestion in any frame of size $\log\mathcal{P}$ or larger is at most 1. For any positive constant β, this step succeeds with probability at least $1 - 1/\mathcal{P}^{\beta}$. If it fails, we simply try again.

3.2. A randomized algorithm to reduce the frame size

In this section, we prove two very similar propositions, Propositions 3.7.1 and 3.7.2, that are meant to replace the first two applications of Lemma 3.7 of [9], which we state below. In proving all these propositions, we use a constructive version of the Lovász Local Lemma that applies to scheduling problems. Let $I \ge 0$. We break a schedule S into *blocks* of $2I^3 + 2I^2 - I$ consecutive time steps. The *size* of a block is the number of time steps it spans.

Lemma 3.7. [9] *In a block of size $2I^3 + 2I^2 - I$, let the relative congestion in any frame of size I or greater be at most r, where $1 \le r \le I$. Then there is a way of* *assigning delays to the packets so that in between the first and the last I^2 steps of this block, the relative congestion in any frame of size $I_1 = \log^2 I$ or greater is at most $r_1 = r(1+\epsilon_1)$, where $\epsilon_1 = O(1)/\sqrt{\log I}$.*

After applying Proposition 3.2 to reduce the frame size from d to $\log \mathcal{P}$, Propositions 3.7.1 and 3.7.2 are used once on each block (for $I = \log \mathcal{P}$ and $I = (\log\log \mathcal{P})^2$ respectively) to further reduce the frame size. Unlike Lemma 3.7 of [9], Propositions 3.7.1 and 3.7.2 may increase the relative congestion by a constant factor. In general, we cannot afford to pay a constant factor at each of the $O(\log^*(c+d))$ applications of Lemma 3.7 of [9], but we can afford to pay it twice.

These two propositions avoid the use of exhaustive search, since they relate to problem sizes that are still large: In these propositions we design separate algorithms that use the fact that the problem sizes are sufficiently large in order to guarantee a "good" solution with high probability. In the remainder of this paper, we assume without loss of generality that $\mathcal{P}$ is not a constant. If $\mathcal{P} = O(1)$, then we can find an optimal schedule in a constant number time steps by using exhaustive search. For the application of Proposition 3.7.1, $I = \log \mathcal{P}$ and $r = 1$. With probability at least $1 - 1/\mathcal{P}^\beta$, for any constant $\beta > 0$, we succeed in producing a schedule S_2 in which the relative congestion is $O(1)$ in frames of size $\log^2 I = (\log\log \mathcal{P})^2$ or greater (if we should fail, we simply try again). In the application of Proposition 3.7.2, $I = (\log\log \mathcal{P})^2$, and $r = O(1)$; in the resulting schedule, S_3, the relative congestion is $O(1)$ in frames of size $\log^2((\log\log \mathcal{P})^2) = (\log\log\log \mathcal{P})^{O(1)}$ or greater, with probability at least $1 - 1/\mathcal{P}^\beta$, for any constant $\beta > 0$. At this point, the problem sizes are small enough for using exhaustive search, and we start using Proposition 3.7.3.

3.7.1. *Let the relative congestion in any frame of size I or greater be at most r in a block of size $2I^3 + 2I^2 - I$, where $1 \le r \le I$ and $I = \log \mathcal{P}$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets in this block. Then, for any constant $\beta > 0$,*

1. *there is an algorithm for assigning initial delays in the range $[0, I]$ to the packets so that in between the first and last I^2 steps of the block, the relative congestion in any frame of size $\log^2 I$ or greater is at most r', where $r' = 2r(1+\sigma)$ and $\sigma = O(1)/\sqrt{\log I}$;*
2. *this algorithm runs in $O(\mathcal{Q}(\log \mathcal{P})^4(\log\log \mathcal{P}))$ time steps, with probability at least $1 - 1/\mathcal{P}^\beta$.*

Proof. We define the *bad event* for each edge g in the network and each T-frame τ, for $\log^2 I \le T \le 2\log^2 I - 1$, as the event that more than $r'T$ packets use g in τ. A particular bad event may or may not *occur* — i.e., may or may not be true — in a given schedule. If no bad event occurs, then by Lemma 3.5, the relative congestion in all frames of size $\log^2 I$ or greater will be at most r'. Since there are

$\log^2 I$ different frame sizes and there are at most $(2I^3 + 2I^2 - I) + I = 2I^3 + 2I^2$ different frames of any particular size, the total number of bad events involving any one edge is at most $(2I^3 + 2I^2)\log^2 I < I^4$, for I greater than some large enough constant.

We now describe the algorithm for finding the assignment. In a first pass of assigning delays to the packets, we process the packets one at a time. To each packet, we assign a delay chosen randomly, independently, and uniformly from 0 to I. We then examine every event in which the packet participates.

A packet can use an edge g in a T-frame τ only if it traversed edge g in τ or in one of the I steps preceding τ in the original schedule. At most $r(T + I)$ packets use edge g in a frame of size $(I + T)$, since the relative congestion in this frame is at most r in the original schedule. Thus at most $r(T + I)$ packets can traverse edge g in the new schedule (after delays are assigned to the packets). We call these at most $r(I+T)$ packets the *candidate* packets to use edge g in τ. Let C_g be the number of candidate packets to use g in τ that have been assigned delays so far. We say that the event for an edge g and a T-frame τ is *critical* if more than $C_g T/I + kr(I+T)T/(I\sqrt{\log I})$ packets actually traverse edge g in τ, where k is a positive constant to be specified later. Intuitively, an event becomes critical if the number of packets assigned delays so far that traverse edge g in τ exceeds the expected number of such packets $(C_g T/I)$ by an excess term $kr(I+T)T/(I\sqrt{\log I})$. Since $C_g \leq r(T + I)$, we allow an excess of at most $k/\sqrt{\log I}$ times the expected number of packets that would use edge g in the frame if *all* of the packets were assigned delays. Hence, the maximum final excess allowed does not depend on C_g. If a packet causes an event to become critical, then we set aside all of the other packets that could also use g during τ, but whose delays have not yet been assigned.

The main difference between our algorithm and Beck's [3] constructive version of the Lovász Local Lemma is that we never allow the number of packets passing through an edge in a T-frame to deviate from the expectation by more than a low order term. In particular, we do not allow this number to deviate by a constant factor times the expectation. In Beck's algorithm, the random variable associated with a bad event (in our case, the number of packets that traverse an edge in a T-frame) may deviate from the expectation by a constant factor times the expectation.

We will deal with the packets that have been set aside later. Let P denote the set of packets that have been assigned delays. As we shall see, after one pass of assigning random delays to the packets, the problem of scheduling the packets that have been set aside is broken into a collection of much smaller subproblems, with probability at least $1 - 1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. Once the size of a subproblem (given by the number of edges involved in the subproblem) gets small compared to the frame length, we can try assigning random delays to the packets that were set aside.

In this initial pass, we assign a random delay to each packet, and check whether the event for an edge g traversed by the packet in this block and T-frame τ becomes critical, for all edges g traversed by the packet in this block and for all frames of size T in $[\log^2 I, 2\log^2 I - 1]$, by following the same procedure as described in the last paragraph of the proof of Proposition 3.6. Here the schedule length after we insert the delays is $2(I^3+I^2)=O(\log^3\mathcal{P})$ (and so there are $O(\log^3\mathcal{P})$ starting points for a T-frame τ) and there are $\log^2 I=(\log\log\mathcal{P})^2$ different frame sizes to consider. The sum of the lengths of the paths traversed by the packets in this block is $\mathcal{Q}$. Thus, a pass takes $O(\mathcal{Q}(\log\mathcal{P})^3(\log\log\mathcal{P})^2)$ time steps. If a pass fails to reduce the component size, we try again.

In order to proceed, we must introduce some notation. The *dependence graph*, G, is the graph in which there is a node for each bad event, and an edge between two nodes if the corresponding events share a packet. Let q denote the number of distinct edges traversed by the packets in this block. Let b denote the degree of G. Whether or not a bad event for an edge g and a time frame τ occurs depends solely on the assignment of delays to the packets that pass through g. Thus, the bad event for an edge g and a time frame τ, and the bad event for an edge g' and a time frame τ' are dependent only if g and g' share a packet. Since at most $r(2I^3+2I^2-I)\leq rI^4$ (for I large enough) packets pass through g and each of these packets passes through at most $2I^3+2I^2-I\leq I^4$ other edges g', and since there are at most $(2I^3+2I^2)(\log^2 I)\leq I^4$ time frames τ', the dependence b is at most rI^{12}. For $r\leq I$, we have $b\leq I^{13}$. Since the packets use at most q different edges in the network, and for each edge there are at most I^4 bad events, the total number of nodes in G is at most qI^4. We say that a node in G is critical if the corresponding event is critical. We say that a node is *endangered* if its event shares a packet with an event that is critical. After each packet has been either assigned a delay or set aside, let G_1 denote the subgraph of G consisting of the critical and endangered nodes and the edges between them. If a node is not in G_1, then all of the packets that use the corresponding edge have already been assigned a delay, and the bad event represented by that node cannot occur, no matter how we assign delays to the packets not in P. Hence, from here on we need only consider the nodes in G_1.

Since different components are not connected by edges in G_1, no two components share a packet. Also, any two events that involve edges traversed by the same packet share an edge in G_1, and so are in the same connected component. Thus there exists a one-to-one correspondence between components of G_1 and disjoint sets in a partition of the packets not in P. Hence, we can assign the delays to the packets in each component separately.

In the following claim, we show that, with high probability, the size of the largest connected component U of G_1 is at most $I^{52}\log\mathcal{P}$, with high probability. Hence we have reduced the maximum possible size of a component from qI^4 in G to $I^{52}\log\mathcal{P}$ in G_1. Recall that the constant k (which we still need to specify) is used to determine whether and event becomes critical.

Claim 1. *For any constant $\beta'>0$, there exists a constant $k>0$ such that the size of the largest connected component of G_1 is at most $I^{52}\log\mathcal{P}$ with probability at least $1-1/\mathcal{P}^{\beta'}$.*

Proof. The trick to bounding the size of a largest connected component U is to observe that the subgraph of critical nodes in U is connected in the cube, G_1^3, of the graph G_1 — i.e., in the graph in which there is an edge between two distinct nodes u and v if in G_1 there is a path of length at most 3 between u and v. In G_1^3, the critical nodes of U form a connected subgraph because any path u, e_1, e_2, e_3, v in G_1 that connects two critical or endangered nodes u and v by passing through three consecutive endangered nodes e_1, e_2, e_3 can be replaced by two paths u, e_1, e_2, w and w, e_2, e_3, v of length 3 that each pass through e_2's critical neighbor w. Let G_2 denote the subgraph of G_1^3 consisting only of the critical nodes and the edges between them. Note that the degree of G_2 is at most b^3, and if two critical nodes lie in the same connected component in G_2, then they also lie in the same connected component in G_1^3, and hence in G_1.

By a similar argument, any maximal independent set of nodes in a connected component of G_2 is connected in G_2^3. Note that if a set of nodes is independent in G_2, then it is also independent in G_1^3 and in G_1. The nodes in an independent set in G_1 do not share any packets, therefore the probabilities that each of these nodes becomes critical are independent. Let G_3 be the subgraph of G_2^3 induced by the nodes in a maximal independent set in G_2 (any such maximal independent set in G_2 will do). The nodes in G_3 form an independent set of critical nodes in G_1. The degree of G_3 is at most b^9.

Our goal now is to show that, for any constant $\beta' > 0$, there exists a constant $k > 0$ such that the number of nodes in any connected component W of G_3 is at most $\log \mathcal{P}$, with probability $1 - 1/\mathcal{P}^{\beta'}$. To begin, with every connected component W of G_3, we associate a spanning tree of W (any such tree will do). Note that, if W and W' are two distinct connected components of G_3, then the spanning trees associated with W and W' are disjoint.

Now let us enumerate the different trees of size ℓ in G_3. To begin, a node is chosen as the root. Since there are at most qI^4 nodes in G_3, there are at most qI^4 possible roots. Next, we construct the tree as we perform a depth-first traversal of it. Nodes of the tree are visited one at a time. At each node u in the tree, either a previously unvisited neighbor of u is chosen as the next node to be visited, or the parent of u is chosen to be visited (at the root, the only option is to visit a previously unvisited neighbor). Thus, at each node there are at most b^9 ways to choose the next node. Since each edge in the tree is traversed once in each direction, and there are $\ell - 1$ edges, the total number of different trees with any one root is at most $(b^9)^{2(\ell-1)} < b^{18\ell}$.

Any tree of size ℓ in G_3 corresponds to an independent set of size ℓ in G_1; moreover, it corresponds to an independent set of ℓ critical nodes in G_1. We can bound the probability that all of the nodes in any particular independent subset U of size ℓ in G_1 are critical as follows. Let p_C be the probability that more than $M = CT/I + kr(I+T)T/(I\sqrt{\log I})$ packets use edge g in τ after C candidate packets to use g in τ have been assigned delays. Then

$$p_C \leq \sum_{j=M+1}^{C} \binom{C}{j} \left(\frac{T}{I}\right)^j \left(1-\frac{T}{I}\right)^{C-j}.$$

For a fixed deviation (in our case, for $kr(I{+}T)T/(I\sqrt{\log I})$) that does not depend on C, the probability p_C of exceeding this deviation is maximized when C is maximized — i.e., when $C = r(I+T)$, which makes $M = r(T+I)T(1+k/\sqrt{\log I})/I$. Thus, $p_C \leq p_{r(I+T)}$. Using the Chernoff-type bound as in the proof of Proposition 3.6, with $\mu = r(I+T)T/I$ and $\gamma = k/\sqrt{\log I}$, and $k_0 = k^2/3$, we have $p_C \leq p_{r(I+T)} = Pr[S \geq (1{+}\gamma)\mu] \leq e^{-\gamma^2\mu/3} = e^{-(k^2 r(I+T)T)/(3I\log I)} \leq e^{-(k_0 \log I)} \leq e^{-(k_0 \ln I)} = 1/I^{k_0}$, since $T \geq \log^2 I$ and $r > 1$. Thus the probability that the event for g and τ becomes critical after C candidate packets to use g in τ have been assigned delays is at most $1/I^{k_0}$. Since the nodes in U are independent in G_1, the corresponding events are also independent. Hence the probability that all of the nodes in the independent set are critical after all packets are assigned delays or put aside is at most $1/I^{k_0\ell}$. Thus the probability that there exists a tree of size ℓ in G_3 is at most $qI^4b^{18\ell}/I^{k_0\ell} \leq qI^{4-(k_0-234)\ell}$ (since there exists at most $qI^4b^{18\ell}$ different trees of size ℓ in G_3 and $b \leq I^{13}$). Since $q \leq \mathcal{P}$, we can make this probability less than $1/\mathcal{P}^{\beta'}$, for $\ell = \log\mathcal{P}$ and any fixed constant $\beta' > 0$, by choosing k to be a sufficiently large constant. Hence, with probability at least $1-1/\mathcal{P}^{\beta'}$, the size of the largest spanning tree in G_3 will be $\log\mathcal{P}$.

We can now bound the size of the largest connected component in G_1. Since (i) the largest connected component in G_3 has at most ℓ nodes with probability at least $1-1/\mathcal{P}^{\beta'}$, (ii) each of these ℓ nodes may have b^3 neighbors in G_2, and (iii) each node in G_2 is either in G_3 or is a neighbor of a node in G_3, the largest connected component in G_2 contains at most $b^3\ell$ nodes with probability at least $1-1/\mathcal{P}^{\beta'}$. As we argued before, the critical nodes in any connected component of G_1 are connected in G_2. Thus, the maximum number of critical nodes in any connected component of G_1 is at most $b^3\ell$. Since each of these nodes may have as many as b endangered neighbors (and each endangered neighbor is adjacent to a critical node), and since $\ell = \log\mathcal{P}$, the size of the largest connected component in G_1 is at most $b^4\ell \leq I^{52}\log\mathcal{P}$, with high probability. ∎

Since $I = \log\mathcal{P}$ in the scope of this lemma, the size of the largest connected component in G_1 is at most $(\log\mathcal{P})^{53}$, for k large enough, with probability at least $1-1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. We still have to find a schedule for the packets not in P. We now have a collection of independent subproblems to solve, one for each component in the dependence graph. We can use Proposition 3.6 to find the initial delays for these packets. Since each node in the dependence graph corresponds to an edge in the routing network, a component with x nodes in the dependence graph corresponds to at most x, and possibly fewer, edges in the routing network.

We apply Proposition 3.6 to each of the independent subproblems. In the original block, let r_g be the relative congestion of edge g with respect to the packets not in P in frames of size I or larger, and let r_g^H be the relative congestion of edge g with respect to the packets in subproblem H in frames of size I or larger ($r_g = \sum_H r_g^H$). Let q^H be the number of distinct edges associated with subproblem H, for all H. Note that $q^H \leq I^{52}\ell \leq I^{52}\log\mathcal{P} = I^{53}$, since $I = \log\mathcal{P}$. After applying Proposition 3.6 to a subproblem H, the relative congestion of any edge g with respect to the packets in H in frames of size $\log^2 I$ or larger, in between the first and last I^2 steps in the final schedule is at most $r_g^H(1+k_1/\sqrt{\log I})$, for some constant $k_1 > 0$, with probability at least $1-q^H/I^\xi \geq 1-1/(\log\mathcal{P})^{\xi-53}$, for any constant $\xi > 0$.

Since the routing subproblems are mutually independent and disjoint, if we apply Proposition 3.6 $\log\mathcal{P}/(\log\log\mathcal{P})$ times to each of the at most $N \leq \mathcal{P}$ subproblems (note that each packet appears in at most one subproblem), then for any constant $\xi > 53$, and $\mathcal{P}$ large enough, with probability at least $1 - N/(\log\mathcal{P})^{(\xi-53)\log\mathcal{P}/(\log\log\mathcal{P})} \geq 1 - 1/\mathcal{P}^{\xi-54}$, the relative congestion of edge g with respect to the packets not in P, in any frame of size $\log^2 I$ or greater is at most $\sum_H r_g^H(1+k_1/\sqrt{\log I}) = r_g(1+k_1/\sqrt{\log I})$.

Applying Proposition 3.6 $\log\mathcal{P}/(\log\log\mathcal{P})$ times for each subproblem takes time $O(\sum_H q^H(I^3+I^2)(\log^2 I)\log\mathcal{P}/\log\log\mathcal{P}) = O(\mathcal{Q}(\log^4\mathcal{P})(\log\log\mathcal{P}))$, since $I = \log\mathcal{P}$ and $\mathcal{Q} \geq \sum_H q^H$.

We now have schedules for the packets in P and for the packets not in P. Fix any edge g traversed by some packet in the block and a T-frame τ, where $T \in [\log^2 I, 2\log^2 I - 1]$. The total number of candidate packets in P to use edge g in τ after the delays have been assigned is given by C_g. The number of packets in P that traverse edge g in τ in the resulting schedule is at most $C_g T/I + kr(I+T)T/(I\sqrt{\log I}) \leq r(I+T)T(1+k/\sqrt{\log I})/I \leq rT(1+(2k+1)/\sqrt{\log I})$, since $(I+T)/I \leq (1+1/\sqrt{\log I})$, for I large enough, and $C_g \leq r(I+T)$. Hence the relative congestion of edge g in τ with respect to the packets in P is at most $r(1+(2k+1)/\sqrt{\log I})$.

Now we consider the relative congestion of the packets not in P. With probability at least $1-1/\mathcal{P}^{\beta'}-1/\mathcal{P}^{(\xi-54)}$, for any positive constants β' and ξ, there exists a constant k_1 such that the number of packets not in P that traverse g in the new schedule is at most $r_g T(1+k_1/\sqrt{\log I}) \leq rT(1+k_1/\sqrt{\log I})$.

Combining the relative congestions for packets in P and not in P, we get that the relative congestion of edge G in τ is at most $2r(1+\max(2k+1,k_1)/\sqrt{\log I})$. Choose $\sigma = \max(2k+1,k_1)/\sqrt{\log I}$. Choose β such that $1/\mathcal{P}^\beta \geq 1/\mathcal{P}^{\beta' q'} + 1/\mathcal{P}^{\xi-54}$. Hence, for any constant $\beta > 0$, there exist positive constants k and k_1 such that the relative congestion of edge g in τ is at most $2r(1+\sigma)$, for any edge g, for any T-frame τ, for any $T \in [\log^2 I, 2\log^2 I - 1]$, with probability at least $1-1/\mathcal{P}^{\beta}$.

The number of time steps taken by the algorithm just described is $O(\mathcal{Q}(\log\log\mathcal{P}+\log\mathcal{P})(\log^3\mathcal{P})(\log\log\mathcal{P})) = O(\mathcal{Q}(\log^4\mathcal{P})(\log\log\mathcal{P}))$. ∎

Proposition 3.7.2. *Let the relative congestion in any frame of size I or greater be at most r in a block of size $2I^3+2I^2-I$, where $1 \le r \le I$ and $I = (\log\log\mathcal{P})^2$. Let $\mathcal{Q}$ be the sum of the lengths of the paths traversed by the packets in this block. Then, for any constant $\beta > 0$,*

1. *there is an algorithm for assigning initial delays in the range $[0, I]$ to the packets so that in between the first and last I^2 steps of the block, the relative congestion in any frame of size $\log^2 I$ or greater is at most r', where $r' = 2r(1+\sigma)$ and $\sigma = O(1)/\sqrt{\log I}$;*
2. *this algorithm runs in $\mathcal{Q}(\log\mathcal{P})(\log\log\mathcal{P})^6(\log\log\log\mathcal{P})^{O(1)}$ time steps, with probability at least $1-1/\mathcal{P}^{\beta}$.*

Proof. The first part of the proof of this proposition is identical to the part where we assign delays to the packets in P in the proof of the Proposition 3.7.1 (we let $I = (\log\log\mathcal{P})^2$ in that proof).

However, since $I = (\log\log\mathcal{P})^2$, we need to make an additional pass assigning delays to the packets in this proof, in order to reduce the component size in the dependence graph to a polynomial in I. From there, we proceed by applying Proposition 3.6 to each component separately, as we did in the proof of Proposition 3.7.1. Let q denote the number of distinct edges used by the packets in this block. In the first pass, we reduce the maximum component size in G_1 from qI^4 to $I^{52}\log\mathcal{P}$ (by taking $\ell = \log\mathcal{P}$, as in Proposition 3.7.1), with probability at least $1-1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. In the second pass, we reduce the component size from $I^{52}\log\mathcal{P}$ down to $I^{52}\log\log\mathcal{P} \le I^{53}$ (by taking $\ell = \log\log\mathcal{P}$ in the part of the proof of Proposition 3.7.1 where we assign delays to the packets in P) and noting that the number of edges in any connected component is now at most $I^{52}\log\mathcal{P}$. For any component, this step will succeed with probability at least $1-1/(I^{52}\log\mathcal{P})^{\beta'}$, for any constant $\beta' > 0$. To make this probability as high as it was in the case $I = \log\mathcal{P}$, if a pass fails for any component, we simply try to reduce the component size again, up to $\log\mathcal{P}/(\log\log\mathcal{P})$ times. Then with probability at least $1-1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$, we have reduced the component size to at most I^{53}. Since (i) for each packet assigned a delay in these two passes, we have to check whether the event for an edge g and a T-frame τ becomes critical, for all edges g traversed by the packet, for all T-frames τ, for $T \in [\log^2 I, 2\log^2 I - 1]$ (using the procedure described in the last paragraph of Proposition 3.6), and since (ii) we repeat the second pass $O(\log \mathcal{P}/\log\log \mathcal{P})$ times, the two passes take

$$O(\mathcal{Q}(I^3+I^2)(\log^2 I)(\log \mathcal{P})/(\log\log \mathcal{P})) \leq$$
$$O(\mathcal{Q}(\log\log \mathcal{P})^6 \log^2((\log\log \mathcal{P})^2)\log \mathcal{P}/(\log\log \mathcal{P})) \leq$$
$$\mathcal{Q}(\log \mathcal{P})(\log\log \mathcal{P})^5(\log\log\log \mathcal{P})^{O(1)}$$

time steps.

The second pass adds some packets to the set P. Let P_1 and P_2 denote the number of packets assigned delays in the first and second pass, respectively. Then the relative congestion due to these packets will be at most

$$[(P_1+P_2)T/I+2kr(I+T)T/(I\sqrt{\log I})]/T \leq r(I+T)/I+2kr(I+T)/(I\sqrt{\log I}) \leq$$
$$r[1+T/I+2k(I+T)/(I\sqrt{\log I})] \leq r(1+(4k+1)/\sqrt{\log I}),$$

since $T < 2\log^2 I$ and $2\log^2 I/I \leq 1/\sqrt{\log I}$. If the two passes fail to achieve the desired relative congestion, we try again.

Now we apply Proposition 3.6 up to $\log \mathcal{P}/(\log\log\log \mathcal{P})$ times, assigning delays to the packets not in P, verifying at the end of each application whether the schedule obtained has relative congestion $r(1+k_1/\sqrt{\log I})$, for some constant k_1 to be specified later. Here we need to apply Proposition 3.6 up to $\log \mathcal{P}/(\log\log\log \mathcal{P})$ times to each resulting component (rather than $\log \mathcal{P}/(\log\log \mathcal{P})$ as in the proof of Proposition 3.7.1) since the component size now is $O(I^{53}) = (\log\log \mathcal{P})^{O(1)}$, and so our bound on the failure probability for each component is only $1/(\log\log \mathcal{P})^{O(1)}$ (since the bound given by Proposition 3.6 is at best polynomially small in I and $I = (\log\log \mathcal{P})^2$), rather than $1/(\log \mathcal{P})^{O(1)}$. The assignment of delays to the packets not in P takes at most $\mathcal{Q}(\log\log \mathcal{P})^6(\log\log\log \mathcal{P})^{O(1)}(\log \mathcal{P})$ time steps, since $\mathcal{Q}$ is an upper bound on the sum of the number of edges in each component. For any constant $\beta' > 0$, there exists a constant $k_1 > 0$ such that we obtain a feasible schedule for these packets with relative congestion $r(1+k_1/\sqrt{\log I})$ with probability at least $1-1/\mathcal{P}^{\beta'}$.

We have schedules for the packets in P and for the packets not in P, with relative congestions $r(1+(4k+1)/\sqrt{\log I})$ and $r(1+k_1/\sqrt{\log I})$, respectively, with probability at least $1-2/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. The two schedules can be found in at most $\mathcal{Q}(\log \mathcal{P})(\log\log \mathcal{P})^6(\log\log\log \mathcal{P})^{O(1)}$ time steps. When we merge the two schedules, the resulting relative congestion may be as large as the sum of the two relative congestions — that is, the resulting relative congestion may be as large as $2r(1+\max(4k+1,k_1)/\sqrt{\log I})$, with probability at least $1-1/\mathcal{P}^{\beta}$, for large enough positive constants k and k_1, for any fixed $\beta > 0$ (choose β' such that $1/\mathcal{P}^{\beta} > 2/\mathcal{P}^{\beta'}$). Let $\sigma = \max(4k+1,k_1)/\sqrt{\log I}$. ∎

3.3. Applying exhaustive search

The remaining $O(\log^*(c+d))$ applications of Lemma 3.7 in [9] are replaced by applications of the following proposition, which uses the same technique as Propositions 3.7.1 and 3.7.2, except that instead of using Proposition 3.6 for each component of the subgraph induced by critical and endangered nodes in the dependence graph, it uses the Lovász Local Lemma and exhaustive search to find the settings of the delays for the packets. Proposition 3.7.3 does not allow a constant factor increase in the relative congestion of the refined schedule, which prevents a $2^{O(\log^*(c+d))}$ blowup in the final relative congestion.

Proposition 3.7.3. *Let the relative congestion in any frame of size I or greater be at most r in a block of size $2I^3+2I^2-I$, where $1 \le r \le I$ and $I \le (\log\log\log\mathcal{P})^{O(1)}$. Let $\mathcal{Q}$ be the sum of the lengths of paths taken by the packets in this block. Then, for any constant $\beta > 0$,*

1. *there is an algorithm for assigning initial delays in the range $[0, I]$ to the packets so that the relative congestion in any frame of size $\log^2 I$ or greater in between the first and last I^2 steps in the resulting schedule is at most r', where $r' = r(1+\sigma)$ and $\sigma = O(1)/\sqrt{\log I}$;*
2. *this algorithm runs in $\mathcal{Q}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$ time steps, with probability at least $1-1/\mathcal{P}^{\beta}$.*

Proof. This proof uses the Lovász Local Lemma to show that an assignment of initial delays satisfying the conditions of the proposition exists, as we will see below.

We first assign delays to some packets by making three passes through the packets using the algorithm of Proposition 3.7.1 (for making the initial pass assigning delays to the packets in P) in each pass. Let $C_g^{(i)}$, $1 \le i \le 3$, be the number of candidate packets to use edge g in τ that were assigned delays in the ith pass. After the first pass, we have that (i) the number of packets assigned delays in this pass that use edge g in the new schedule is at most $C_g^{(1)}T/I + kr(I+T)/(I\sqrt{\log I})$, and (ii) with probability at least $1-1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$, the size of the largest component in the dependency graph is $I^{52}\log\mathcal{P}$.

We need to make two more passes assigning delays to the packets, reducing the size of the largest connected component first to $I^{52}(\log\log\mathcal{P})$, and then to $I^{52}(\log\log\log\mathcal{P}) = (\log\log\log\mathcal{P})^{O(1)}$ (since $I \le (\log\log\log\mathcal{P})^{O(1)}$), by taking $\ell = \log\log\mathcal{P}$ in the second pass and $\ell = \log\log\log\mathcal{P}$ in the third pass. If we fail to reduce the component size as desired, the second pass is repeated up to $\log\mathcal{P}/(\log\log\mathcal{P})$ times and the third pass is repeated up to $\log\mathcal{P}/(\log\log\log\mathcal{P})$ times. The number of packets assigned delays in the second (resp., third) pass that traverse edge g in the new schedule is at most $C_g^{(2)}T/I + kr(I+T)/(I\sqrt{\log I})$ (resp., $C_g^{(3)}T/I + kr(I+$

$T)/(I\sqrt{\log I}))$. As before, k is chosen large enough so that the failure probability in each pass is at most $1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$.

In each pass, we assign a random delay to each packet and check whether the event for any edge g traversed by this packet and any T-frame τ, where $T \in [\log^2 I, 2\log^2 I-1]$, becomes critical, as we did in Propositions 3.7.1-2. Thus each pass takes time $O(\mathcal{Q}(I^3+I^2)(\log^2 I)) = \mathcal{Q}(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$. For any constant $\beta > 0$, choose β' such that $1/\mathcal{P}^{\beta} > 3/\mathcal{P}^{\beta'}$. Hence, since we may repeat the second and third passes up to $\log\mathcal{P}/(\log\log\mathcal{P})$ and $\log\mathcal{P}/(\log\log\log\mathcal{P})$, respectively, we succeed in reducing the component size to $(\log\log\log\mathcal{P})^{O(1)}$ in $\mathcal{Q}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$ time steps, with probability at least $1-1/\mathcal{P}^{\beta}$.

We now use the Lovász Local Lemma to show that there exists a way of completing the assignment of delays (i.e., to assign delays to the packets not in P) so that the relative congestion in frames of size $\log^2 I$ or greater in this block is at most $r(1+O(1)/\sqrt{\log I})$. We associate a bad event with each edge and each time frame of size $\log^2 I$ through $2\log^2 I-1$. The bad event for an edge g and a particular T-frame τ occurs when more than $M_g = (r(I+T)-P_g)T/I + kr(I+T)T/(I\sqrt{\log I})$ packets not in P use edge g in τ, where P_g is the number of packets in P that traverse edge g during τ after the delays have been assigned to the packets in P (note that there are at most $r(T+I)-P_g$ candidate packets not in P to use edge g in τ). As we argued in the proof of Proposition 3.7.1, the total number of bad events involving any one edge is at most I^4. We show that if each packet not in P is assigned a delay chosen randomly, independently, and uniformly from the range $[0,I]$, then with nonzero probability no bad event occurs. In order to apply the lemma, we must bound both the dependence of the bad events, and the probability that any bad event occurs. The dependence b is at most I^{13}, as argued before. For any edge g and T-frame τ that contains g, where $\log^2 I \le T \le (2\log^2 I)-1$, the probability p_g that more than M_g packets not in P use g in τ, can be shown to be at most $1/I^{14}$, for sufficiently large k, using exactly the same Chernoff-bound argument that was used in Proposition 3.7.1. Thus, $4(\max_{g\in G}\{p_g\})b \le 4/I < 1$ (for $I > 4$). Hence, since $\max_{g\in G}\{p_g\}$ is an upper bound on the probability of any bad event occurring, by the Lovász Local Lemma, there is some way of assigning delays to the packets not in P so that no bad event occurs.

Since at most $r(T+I)$ packets pass through the edge associated with any critical node, and there are at most $(I+1)$ choices for the delay assigned to each packet, the number of different possible assignments for any subproblem containing $(\log\log\log\mathcal{P})^{O(1)}$ critical nodes is at most $(I+1)^{r(I+T)(\log\log\log\mathcal{P})^{O(1)}} \le I^{4I^2(\log\log\log\mathcal{P})^{O(1)}}$ (since $r < I$ and $T < 2\log^2 I$). For $I < (\log\log\log\mathcal{P})^{O(1)}$ and $\mathcal{P}$ larger that some constant, this quantity is smaller than $(\log\mathcal{P})^{\gamma}$, for any fixed constant $\gamma > 0$. Hence, we need to try out at most $\log^{\gamma}\mathcal{P}$ possible delay assignments.

After assigning delays to all of the packets, the number of packets that use an edge g in any T-frame τ is at most

$$\sum_{i=1}^{3}\left(\frac{C_g^{(i)}T}{I}+\frac{kr(I+T)T}{I\sqrt{\log I}}\right)+\frac{(r(I+T)-P_g)T}{I}+\frac{kr(I+T)T}{I\sqrt{\log I}}$$
$$\leq \frac{r(I+T)T}{I}+\frac{4kr(I+T)T}{I\sqrt{\log I}}$$

with probability at least $1-1/\mathcal{P}^{\beta}$, since each packet is assigned a delay exactly once, and thus $r(I+T)-P_g+C_g^{(1)}+C_g^{(2)}+C_g^{(3)}\leq r(I+T)$. Thus the relative congestion in any T-frame, for $\log^2 I\leq T<2\log^2 I$, is at most

$$\left(\frac{r(I+T)}{I}\right)(1+\frac{4k}{\sqrt{\log I}}) = r\left(1+\frac{T}{I}\right)(1+\frac{4k}{\sqrt{\log I}})$$
$$= r\left(1+\frac{(8k+1)}{\sqrt{\log I}}\right) = r(1+\sigma),$$

by taking $\sigma=(8k+1)/(I\sqrt{\log I})$, since $2\log^2 I/I\leq 1/\sqrt{\log I}$, for I large enough.

We can bound the total number of time steps taken by the algorithm as follows. The first three passes (including all repeated trials of the second and third passes) take time $\mathcal{Q}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$, with probability at least $1-1/\mathcal{P}^{\beta}$. After the third pass, we solve subproblems containing $(\log\log\log\mathcal{P})^{O(1)}$ critical nodes exhaustively. For each subproblem, for each of the at most $\log^{\gamma}\mathcal{P}$ possible assignment of delays to the packets in the subproblem, for each of the at most $(I^3+I^2)\log^2 I$ T-frames τ in the subproblem, $\log^2 I\leq T<2\log^2 I$, and for every edge g in τ, we check whether more than M_g packets traverse g during τ (using the procedure described in the proof of Proposition 3.6). This takes time $O(\mathcal{Q}(I^3+I^2)(\log^2 I)(\log^{\gamma}\mathcal{P}))$, which is at most $\mathcal{Q}(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}(\log^{\gamma}\mathcal{P})$, for $\mathcal{P}$ large enough, for any fixed $\gamma>0$ (since the sum of the number of distinct edges in each subproblem is at most $\mathcal{Q}$, and since $I=(\log\log\log\mathcal{P})^{O(1)}$). In particular, for $\gamma=1$, this quantity is bounded by $\mathcal{Q}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$. Hence the algorithm runs in $\mathcal{Q}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$ time steps, with probability at least $1-1/\mathcal{P}^{\beta}$, for any constant $\beta>0$. ∎

3.4. Moving the block boundaries

Now we present the three replacement propositions for Lemma 3.9 of [9], which bounds the relative congestion after we move the block boundaries (as in [9]). The three propositions that follow are analogous to the three replacement propositions, Propositions 3.7.1-3, for Lemma 3.7 of [9]. The necessary changes in the proof of Lemma 3.9 of [9], in places where the Lovász Local Lemma is used, are also analogous to the changes made in the proof of Lemma 3.7 of [9], for the cases $I = \log\mathcal{P}$, $I = (\log\log\mathcal{P})^2$, and $I = (\log\log\log\mathcal{P})^{O(1)}$. Therefore, we omit the proofs of Propositions 3.9.1-3.

Suppose we have a block of size $2I^3+3I^2$, obtained after the insertion of delays into the schedule as described in Propositions 3.7.1, 3.7.2, or 3.7.3, according to the current value of I. Then suppose we move the block boundaries as described in [9]. Each Proposition 3.9.1-3 refers to a specific size of I. Note that in [9], the steps between steps I^3 and I^3+2I^2 in the block are called the "fuzzy region" of the block. We assume that the relative congestion in any frame of size I or greater in the block is at most r, where $1 \le r \le I$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets in the block.

Proposition 3.9.1. *For $I = \log\mathcal{P}$, for any constant $\beta > 0$,*

1. *there is an algorithm for assigning delays in the range $[0, I^2]$ to the packets such that in between steps $I\log^3 I$ and I^3 and in between steps I^3+3I^2 and $2I^3+3I^2-I\log^3 I$, the relative congestion in any frame of size $\log^2 I$ or greater is at most $2r(1+\sigma_1)$, where $\sigma_1 = O(1)/\sqrt{\log I}$, and such that in between steps I^3 and I^3+3I^2, the relative congestion in any frame of size $\log^2 I$ or greater is at most $2r(1+\sigma_2)$, where $\sigma_2 = O(1)/\sqrt{\log I}$;*
2. *this algorithm runs in $O(\mathcal{Q}(\log\mathcal{P})^4(\log\log\mathcal{P}))$ time steps, with probability at least $1-1/\mathcal{P}^\beta$.*

Proposition 3.9.2. *For $I = (\log\log\mathcal{P})^2$, for any constant $\beta > 0$,*

1. *there is an algorithm for assigning delays in the range $[0, I^2]$ to the packets such that in between steps $I\log^3 I$ and I^3 and in between steps I^3+3I^2 and $2I^3+3I^2-I\log^3 I$, the relative congestion in any frame of size $\log^2 I$ or greater is at most $2r(1+\sigma_1)$, where $\sigma_1 = O(1)/\sqrt{\log I}$, and such that in between steps I^3 and I^3+3I^2, the relative congestion in any frame of size $\log^2 I$ or greater is at most $2r(1+\sigma_2)$, where $\sigma_2 = O(1)/\sqrt{\log I}$;*
2. *this algorithm runs in $\mathcal{Q}(\log\mathcal{P})(\log\log\mathcal{P})^6(\log\log\log\mathcal{P})^{O(1)}$ time steps, with probability at least $1-1/\mathcal{P}^\beta$.*

Proposition 3.9.3. *For $I = (\log\log\log\mathcal{P})^{O(1)}$, for any constant $\beta > 0$,*

1. *there is an algorithm for assigning delays in the range $[0, I^2]$ to the packets such that in between steps $I\log^3 I$ and I^3 and in between steps I^3+3I^2 and $2I^3+3I^2-I\log^3 I$, the relative congestion in any frame of size $\log^2 I$ or greater is at most $r(1+\sigma_1)$, where $\sigma_1 = O(1)/\sqrt{\log I}$, and such that in between steps*

I^3 and I^3+3I^2, the relative congestion in any frame of size $\log^2 I$ or greater is at most $r(1+\sigma_2)$, where $\sigma_2 = O(1)/\sqrt{\log I}$;

2. *this algorithm runs in $\mathcal{Q}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}$ time steps, with probability at least $1-1/\mathcal{P}^{\beta}$.*

4. Running time

Theorem 4.1. *For any constant $\delta > 0$, the algorithm for finding an $O(c+d)$-steps schedule of the packets takes $O(m(c+d)(\log\mathcal{P})^4(\log\log\mathcal{P}))$ time steps overall, with probability at least $1-1/\mathcal{P}^{\delta}$.*

Proof. For any constant $\beta > 0$, we place an upper bound on the number of time steps taken by the application of Proposition 3.2, followed by the applications of Propositions 3.7.1, 3.9.1, 3.7.2, and 3.9.2, then followed by the applications of Propositions 3.7.3 and 3.9.3. The application of Proposition 3.2 takes $O(m(c+d)\log\mathcal{P})$ time steps, with probability at least $1-1/\mathcal{P}^{\beta}$. Each of the Propositions 3.7.1-3, and each of the Propositions 3.9.1-3 dealt with a single block. For any I, partitioning the schedule into *disjoint* blocks and moving the block boundaries as described in [9] take $O(\mathcal{P})$ time. Let n_I be the number of blocks in the partition of the schedule for any given I.

We place an upper bound on the number of time steps taken by the applications of Propositions 3.7.1-3 and 3.9.1-3 as follows. Assume the n_I blocks in the partition for I are numbered from 1 to n_I. Note that $\sum_{i=1}^{n_I}\mathcal{Q}_i = \mathcal{P}$, where $\mathcal{Q}_i$ is the sum of the lengths of the paths traversed by the packets in block i in this partition, independent of I. Thus the applications of Proposition 3.7.1 and 3.9.1 take $O(\mathcal{P}(\log\mathcal{P})^4(\log\log\mathcal{P}))$ steps; and the applications of Propositions 3.7.2 and 3.9.2 take $\mathcal{P}(\log\mathcal{P})(\log\log\mathcal{P})^6(\log\log\log\mathcal{P})^{O(1)}$ steps. For each partition of the schedule for a given $I \le (\log\log\log\mathcal{P})^{O(1)}$, we apply Propositions 3.7.3 and 3.9.3 to every block i in this partition, $1 \le i \le n_I$, taking overall time $\mathcal{P}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}$ $(\log\log\log\log\mathcal{P})^{O(1)}$. Since we will repartition the schedule $O(\log^*(c+d))$ times after we bring I down to $(\log\log\log\mathcal{P})^{O(1)}$, the overall running time due to applications of Propositions 3.7.3 and 3.9.3 is

$$\mathcal{P}(\log\mathcal{P})(\log\log\log\mathcal{P})^{O(1)}(\log\log\log\log\mathcal{P})^{O(1)}\log^*(c+d).$$

Choose $\delta > 0$ such that $1/\mathcal{P}^{\delta} \ge O(\log^*(c+d))/\mathcal{P}^{\beta}$. Hence, the total number of time steps taken by the algorithm is $O(m(c+d)(\log\mathcal{P})^4(\log\log\mathcal{P}))$, for $\mathcal{P}$ large enough, with probability at least $1-1/\mathcal{P}^{\delta}$, for any constant $\delta > 0$ (Note that we use the inequalities $\mathcal{P} \ge c$, $\mathcal{P} \ge d$, and $\mathcal{P} \le m(c+d)$). ∎

5. A parallel scheduling algorithm

At first glance, it seems as though the algorithm that was described in Section 3 is inherently sequential. This is because the decision concerning whether or not to assign a delay to a packet is made sequentially. In particular, a packet is deferred (i.e., not assigned a delay) if and only if the packet might be involved in an event — i.e., the packet traverses an edge that corresponds to an event — that became critical because of the delays assigned to prior packets.

In [1], Alon describes a parallel version of Beck's algorithm which proceeds by assigning values to all random variables (in this case delays to all packets) in parallel, and then unassigning values to those variables that are involved in bad events. The Alon approach does not work in this application because we cannot afford the constant factor blow-up in relative congestion that would result from this process.

Rather, we develop an alternative method for parallelizing the algorithm. The key idea is to process the packets in a *random* order. At each step, all packets that do not share an edge with an as-yet-unprocessed packet of higher priority are processed in parallel.

To analyze the parallel running time of this algorithm, we first make a dependency graph G' with a node for every packet and an edge between two nodes if the corresponding packets can be involved in the same event. Each edge is directed towards the node corresponding to the packet of lesser priority. By Brent's Theorem [4], the parallel running time of the algorithm is then at most twice the length of the longest directed path in G'.

Let D denote the maximum degree of G'. There are at most ND^L paths of length L in G'. The probability that any particular path of length L has all of its edges directed in the same way is at most $2/L!$ (the factor of 2 appears because there are two possible orientations for the edges). Hence, with probability near 1, the longest directed path length in G' is $O(D+\log N)$. This is because if $L \geq k(D+\log N)$, for some large constant k, then $ND^L \cdot \frac{2}{L!} \ll 1$.

Each packet can be involved in at most $(2I^3+2I^2)(2I^3+I^2)\log^2 I$ events, and at most $r(I+T) \leq O(I)$ packets can be involved in the same event. Hence, the degree D of G' is at most $O(I^7 \log^2 I)$. By using the method of Proposition 3.2 as a preprocessing phase, we can assume that c, d, and thus I, are all polylogarithmic in $\mathcal{P}$. Hence, the parallel algorithm runs in NC, as claimed.

6. Concluding remarks

Our algorithm for packet scheduling can also be used to route messages that are composed of sequences of packets. This is possible since our algorithm can easily maintain the property that any two packets traveling along the same path to the same destination always proceed in order.

The algorithms described in this paper are randomized, but they can be derandomized using the method of conditional probabilities [17, 20].

References

[1] N. Alon: A parallel algorithmic version of the Local Lemma, *Random Structures and Algorithms*, **2**(4) (1991), 367–378.

[2] D. Angluin and L. G. Valiant: Fast probabilistic algorithms for hamiltonian circuits and matchings, *Journal of Computer and System Sciences*, **18**(2) (1979), 155–193.

[3] J. Beck: An algorithmic approach to the Lovász Local Lemma I, *Random Structures and Algorithms*, **2**(4) (1991), 343–365.

[4] R. P. Brent: The parallel evaluation of general arithmetic expressions, *Journal of the ACM*, **21**(2) (1974), 201–208.

[5] H. Chernoff: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, *Annals of Mathematical Statistics*, **23** (1952), 493–507.

[6] P. Erdős and L. Lovász: Problems and results on 3-chromatic hypergraphs and some related questions, In A. Hajnal et al., editor, *Infinite and Finite Sets* Volume 11 of *Colloq. Math. Soc. J. Bolyai*, pages 609–627, North Holland, Amsterdam, The Netherlands, 1975.

[7] R. R. Koch, F. T. Leighton, B. M. Maggs, S. B. Rao, A. L. Rosenberg, and E. J. Schwabe: Work-preserving emulations of fixed-connection networks, *Journal of the ACM*, **44**(1) (1997), 104–147.

[8] 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) (1994), 157–205.

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

[10] F. T. Leighton, B. M. Maggs, and A. W. Richa: Fast algorithms for finding O(congestion + dilation) packet routing schedules, Technical Report CMU–CS–96–152, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, July 1996. To appear in *Combinatorica*.

[11] B. M. Maggs and E. J. Schwabe: Real-time emulations of bounded-degree networks, *Information Processing Letters*, 1998. To appear.

[12] Y. Mansour and B. Patt-Shamir: Greedy packet scheduling on shortest paths, *Journal of Algorithms*, **14** (1993), 449–65.

[13] 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*, pages 341–354, 1995.

[14] 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 439 of *Lecture Notes in Computer Science*, pages 291–302. Springer–Verlag, Heidelberg, Germany, March 1995.

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

[16] Y. Rabani and É. Tardos: Distributed packet switching in arbitrary networks, In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 366–375, May 1996.

[17] P. Raghavan: Probabilistic construction of deterministic algorithms: Approximate packing integer programs, *Journal of Computer and System Sciences*, **37**(4) (1988), 130–143.

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

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

[20] J. Spencer: *Ten Lectures on the Probabilistic Method*, SIAM, Philadelphia, PA, 1987.

[21] 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*, pages 636–643, May 1997.

Tom Leighton

Mathematics Department, and
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
`ftl@math.mit.edu`

Andréa W. Richa

Computer Science and
Engineering Department
Arizona State University
Tempe, AZ 85287
`aricha@asu.edu`

Bruce Maggs

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`bmm@cs.cmu.edu`