Fast Algorithms for Finding O(Congestion+Dilation) Packet Routing Schedules

F. T. Leighton* Bruce M. Maggs Andréa W. Richa

July 1996

CMU-CS-96-152

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

* Mathematics Department, and
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

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 $O(\mathcal{P}(\log\log\mathcal{P})\log\mathcal{P})$ 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. 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.

Tom Leighton is supported in part by ARPA Contracts N00014-91-J-1698 and N00014-92-J-1799. Bruce Maggs and Andréa Richa are supported in part by ARPA Contract F33615-93-1-1330 and in part by an NSF National Young Investigator Award, No. CCR-9457766, with matching funds provided by NEC Research Institute.

Keywords: Store and forward networks, Network problems, Network protocols, Combinatorial algorithms

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 (channels). Each node can serve as the source or destination of an arbitrary number of messages. Each *message* consists 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. (In a dynamic setting, N would denote the rate at which packets enter the network. For simplicity, we will consider a static scenario in which a total of N packets are to be routed through the network.) 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.

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 size 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.

This paper focuses 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. 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 total time and the maximum queue size needed to route all of the packets to their destinations. We would also like to ensure that any two packets traveling along the same path to the same destination always proceed in order.

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 and congestion 3.

1.1 Previous and related work

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 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 required more than $\Omega(c+d)$ steps or larger than constant-size queues to route all the packets.) The method that they used to show the existence of optimal schedules, however, is not constructive. In other words, the fastest known algorithms for producing schedules of length $O(c+d)$ with constant-size edge queues require time that is exponential in the number of packets.

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,

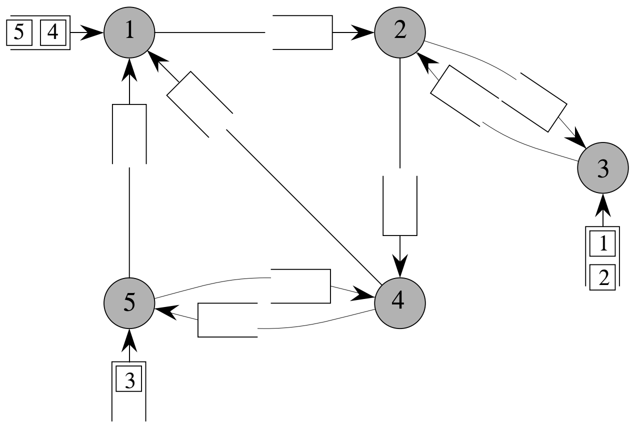

Figure 1: A graph model for packet routing.

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 $0 \leq i < L-1$.)

Mansour and Patt-Shamir [10] then showed that if packets are routed greedily on shortest paths, then all of the packets reach their destinations within $d+N$ steps, where N is the total number of packets. These schedules may be much longer than optimal, however, because N may be much larger than c.

Recently Meyer auf der Heide and Vöcking [11] 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 short-cut free (e.g., shortest paths).

1.2 Our results

In this paper, we show how to produce schedules of length $O(c+d)$ in $O(\mathcal{P}(\log\log\mathcal{P})\log\mathcal{P})$ time, with probability at least $1-1/\mathcal{P}^{\beta}$, for any constant $\beta > 0$, where $\mathcal{P}$ is the *sum of the lengths of the paths taken by the packets*. The schedules can also be found in polylogarithmic time on a parallel computer using $O(\mathcal{P}(\log\log\mathcal{P})\log\mathcal{P})$ work, with probability at least $1-1/\mathcal{P}^{\beta}$.

The algorithm for producing the schedules is based on an algorithmic form of the Lovász Local Lemma (see [6] or [13, 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 the paper. Once this is done, the construction of 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 the optimal schedule once using global control. This situation arises in network emulation problems. Typically, a guest network G is emulated by a host network H by embedding G into H. (For a more complete discussion of emulations and embeddings, see [7].) An *embedding* maps nodes of G to nodes of H, and edges of G to paths in H. 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* is the maximum number of paths corresponding to edges of G that use any one edge of H. The *dilation* is the length of the longest path. Let l, c, and d denote the load, congestion, and dilation of the embedding. Once G has been embedded in H, H can emulate G in a step-by-step fashion. 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.

The result also has applications to job-shop scheduling. In particular, consider a scheduling problem with

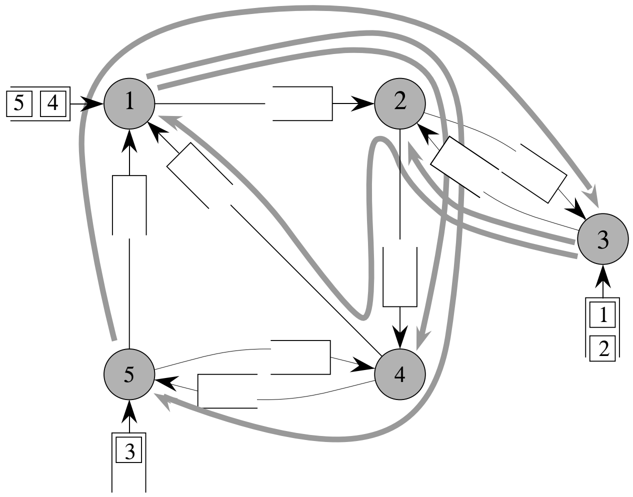

Figure 2: A set of paths for the packets with dilation $d = 3$ and congestion $c = 3$.

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 in the packet routing problem. 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 be at most a constant.

1.3 Outline

The remainder of the paper is divided into sections as follows. In Section 2, we give a very brief overview of the non-constructive proof in [9]. Also we introduce some definitions, and present two important lemmas which will be of later use. In Section 3, we describe how to make the non-constructive method in [9] constructive, and analyze its running time. In Section 4, we show how to parallelize the scheduling algorithm. We conclude with some remarks in Section 5.

2 Preliminaries

In [9], Leighton, Maggs, and Rao proved that for any set of packets whose paths are edge-simple[1] 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. 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 use an edge at a single time step in S_0, whereas in the final schedule at most one packet is allowed to use an edge at each step. Each refinement will bring us closer to meeting this requirement.

[1] An *edge-simple* path uses no edge more than once.

The proof uses the Lovász Local Lemma ([6] or [13, pp. 57–58]) at each refinement step. Given a set of "bad" events in a probability space, the lemma provides a simple inequality which, 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 events $A_1, A_2, \ldots, A_m$ in a probability space has *dependence* at most b if every event is mutually independent of some set of $m-b-1$ other bad 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 $A_1, A_2, \ldots, A_m$ be a set of "bad" events, each occurring with probability p 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* is a sequence of T consecutive time steps. The *frame congestion*, C_f, in a T-frame is the largest number of packets that traverse any edge in the frame. The *relative congestion*, R, in a T-frame is the ratio C_f/T of the congestion in the frame to the size of 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 through Section 3.

Lemma 3.5 *[9] In any schedule, if the number of packets that use 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 use 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 frames, at most RT packets use g. The remainder of the T'-frame τ consists of a single y-frame, where $T \leq y \leq 2T-1$, in which at most Ry packets use g. □

The following proposition will be used in place of Lemma 3.6 of [9].

Proposition 3.6 *Suppose that there are positive constants α_1, α_2, $\alpha_1 \geq \alpha_2$, and ρ, such that in a schedule of size I^{α_1} (or smaller) the relative congestion is at most ρ in frames of size I^{α_2} or larger, and let q be the number of 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 of the schedule and $I^{\alpha_2} - x$ delays are inserted in the last I^{α_3} steps, where α_3 is also a positive constant. Then for any constant $\xi > 0$, with probability at least $1 - q/I^{\xi}$ the relative congestion in any frame of size $\log^2 I$ or larger in-between the first and last I^{α_3} steps in the new schedule is at most $\rho(1+\sigma)$, for some positive $\sigma = O(1)/\sqrt{\log I}$.*

Proof: To bound the relative congestion in frames of size $\log^2 I$ or larger, we need to consider all q 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 use an edge g during a particular T-frame τ has a binomial distribution. In the new schedule, a packet can use g during τ only if in the original schedule it used g during τ or during one of the I^{α_2} steps before the start of τ. Since the relative congestion in any frame of size I^{α_2} or greater in the original schedule is at most ρ, there are at most $\rho(I^{\alpha_2} + T)$ such packets. The probability that an individual packet that could use g during τ actually does so is at most T/I^{α_2}. Thus, the probability p that ρ' or more packets use an edge g during a particular T frame τ is at most

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

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

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

for $0 \leq \gamma \leq 1$.

In our application, $x = \rho(I^{\alpha_2} + T)$, $p' = T/I^{\alpha_2}$, and $\mu = \rho(I^{\alpha_2} + T)T/I^{\alpha_2}$. For $\gamma = \sqrt{3k_0/\log I}$ (where k_0 is any positive constant), $\rho \geq 1$, and $T \geq \log^2 I$, we have $\Pr[S \geq (1+\gamma)\mu] \leq e^{-k_0\rho(I^{\alpha_2}+T)T/(I^{\alpha_2}\log I)} \leq e^{-k_0 \log I} \leq e^{-k_0 \ln I} = 1/I^{k_0}$. Setting $\rho' T = (1+\gamma)\mu = (1+\sqrt{3k_0/\log I})\rho(I^{\alpha_2} + T)T/I^{\alpha_2}$, we have $\rho' \leq \rho(1 + k_1/\sqrt{\log I})$, since $\log^2 I/I^{\alpha_2} \leq 1/\sqrt{\log I}$, for I large enough, for some constant $k_1 > 0$ (that depends on k_0). Let $\sigma = k_1/\sqrt{\log I}$. Then $\rho' \leq \rho(1+\sigma)$. Thus $p = \Pr[S \geq \rho' T] \leq \Pr[S \geq (1+\gamma)\mu] \leq 1/I^{k_0}$.

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 q distinct edges per frame, the probability that the relative congestion is more than ρ' in any frame is at most $q2I^{\alpha_1}\log^2 I/I^{k_0} \leq q/I^{k_0-\alpha_1-2}$ (for $I \geq 1$, $2\log^2 I \leq I^2$). Setting $k_0 = \xi + \alpha_1 + 2$ completes the proof. □

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 in [9], but only reprove those 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]. In this paper 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. Finally, we analyze the running time of our algorithm in Section 3.5.

3.1 The first reduction in frame size

For a given set of N packets, let c and d denote the congestion and the dilation of the paths taken by these packets, and let $\mathcal{P}$ denote the sum of the length of these paths. Let m be the number of edges traversed by the packets (we can ignore the edges not traversed by any packet in the network). Note that $m \leq \mathcal{P} \leq mc$. 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 time at most $O(\mathcal{P})$, 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 $\alpha c + d$. Constructing the new schedule takes time at most $O(\mathcal{P})$.

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 $2\log \mathcal{P} - 1$. 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

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

since each of the at most c packets that pass through g has probability at most $T/\alpha c$ of using g in τ. (Note that e denotes the base of the natural logarithm.) 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 is too large for any frame of size $\log\mathcal{P}$ or larger is at most

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

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 $1/\mathcal{P}^\beta$. $\square$

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. Let $I \ge 0$. We break a schedule S into *blocks* of $2I^3 + 2I^2 - I$ consecutive time steps.

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 $\log\mathcal{P}$, Proposition 3.7.1 is used to reduce the frame size from $\log\mathcal{P}$ to $(\log\log\mathcal{P})^2$, and then Proposition 3.7.2 is used to reduce the frame size from $(\log\log\mathcal{P})^2$ to $\log^2((\log\log\mathcal{P})^2) = (\log\log\log\mathcal{P})^{O(1)}$. 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. 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 at most $O(1)$ in frames of size $\log^2 I = (\log\log\mathcal{P})^2$ (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 at most $O(1)$ in frames of size $\log^2((\log\log\mathcal{P})^2) = (\log\log\log\mathcal{P})^{O(1)}$, with probability at least $1 - 1/\mathcal{P}^\beta$, for any constant $\beta > 0$. At this point, we start using Proposition 3.7.3.

Proposition 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 within this block. Then 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 $2r'$, where $r' = r(1+\sigma)$ and $\sigma = O(1)/\sqrt{\log I}$. For any constant $\beta > 0$, the algorithm runs in time at most $O(\mathcal{Q}(\log\log\mathcal{P})\log\mathcal{P})$, 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, where $\log^2 I \le T \le 2\log^2 I - 1$, 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. 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. We say that the event for an edge g and a T-frame τ is *critical* if delays have been assigned to C packets that could possibly use g in τ, and, of these, more than $CT/I + kr(I+T)T/(I\sqrt{\log I})$ packets actually use g during τ, where k is a positive constant (to be specified later). Intuitively, the 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 (CT/I) by an excess term $kr(I+T)T/(I\sqrt{\log I})$, which is the maximum final excess with respect to the final expected value that we allow. Note that $C \le r(T+I)$. 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. Let P denote the set of packets that are assigned delays. We will deal with the packets that have been set aside later. 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 $\beta' > 0$.

In a pass, we assign a random delay to each packet, and check whether the event for edge g and T-frame τ becomes critical. We can do this checking as we construct the schedule in a per time step fashion. When considering time t, we compute the number of packets that traverse each edge g used during t and then we compute the frame congestion of the T-frames ending at t, for $T \in [\log^2 I, 2\log^2 I - 1]$. Note that the frame congestion of the T-frame ending at time t can be computed by taking the frame congestion of the T-frame ending at $t-1$, subtracting the occurrences of edges in time $t-T$ and adding the occurrences of edges in time t. This can be done in time $O(\mathcal{Q}\log^2 I) \le O(\mathcal{Q}(\log\log\mathcal{P})^2)$, $\mathcal{Q}$ being the sum of the number of edges in the paths taken by the packets within the block. In the remaining of this paper, we assume we check for the congestions of all T-frames of a block, $\log^2 I \le T < 2\log^2 I$, as just described. If a pass fails to reduce the component size, we try again.

Since a total of at most $r(I+T)$ packets that traverse edge g have been assigned delays, the relative congestion for the packets in τ due to the packets in P is at most $[rT(I+T)(1+k/\sqrt{\log I})/I]/T \le r(1 + 1/\sqrt{\log I})(1+k/\sqrt{\log I}) \le r[1 + (2k+1)/\sqrt{\log I}]$, since $T < 2\log^2 I$ and $2\log^2 I/I \le 1/\sqrt{\log I}$, for I large enough. Choose k so that the relative congestion due to the packets in P in τ is at most $r' = r(1+\sigma)$.

In order to proceed, we must introduce some notation. Let m' be the number of edges traversed by some packet within the block. 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 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 $r(2I^3 + 2I^2 - I) \le rI^4$ packets pass through g, and each of these packets passes through at most $2I^3 + 2I^2 - I \le I^4$ other edges g', and there are at most I^4 time frames τ', for I large enough, the dependence b is at most rI^{12}. For $r \le I$, we have $b \le I^{13}$. Since the packets use at most m' 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 $m'I^4$. 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. 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 since this event is not critical, the relative congestion on this edge in the corresponding time frame is at most $r'T$), 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.

We are going to show that, with high probability, the size of the largest connected component U of G_1 is at most $I^{52}\log\mathcal{P}$. 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, and we have reduced the size of a largest component in G_1 from $m'I^4$ to $I^{52}\log\mathcal{P}$.

The trick to bounding the size of the 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., 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 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 three 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 nodes lie in the same connected component in G_2, then they must 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 must also be independent in G_1^3 and in G_1. 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 the number of nodes in any connected component W of G_3 is at most $\log\mathcal{P}$, with probability $1 - 1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. To begin, with every connected component W of G_3, we associate a spanning tree of W, T_W (any such tree will do). Note that, if W and W' are two distinct connected components of G_3, then T_W and $T_{W'}$ are disjoint.

Now let us enumerate the different trees of size t in G_3. To begin, a node is chosen as the root. Since there are at most $m'I^4$ nodes in G_3, there are at most $m'I^4$ 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 (and added to the tree), 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 $t-1$ edges, the total number of different trees with any one root is at most $(b^9)^{2(t-1)} < b^{18t}$.

Any tree of size t in G_3 corresponds to an independent set of size t in G_1, moreover, to an independent set of t critical nodes in G_1. We can bound the probability that all of the nodes in any particular independent subset U of size t 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. Then

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

Since $M = CT/I + kr(I+T)T/I\sqrt{\log I}$ and $C \leq r(I+T)$, using the Chernoff-type bound as in the proof of Proposition 3.6, with $\mu = r(I+T)T/I$, $\gamma = \sqrt{3k_0/\log I}$, $\log^2 I \leq T \leq 2\log^2 I - 1$, for any large enough constants k and k_0, we have $p_C = Pr[S \geq M] \leq Pr[S \geq (1+\gamma)\mu] \leq e^{-\gamma^2\mu/3} = e^{-k_0 r(I+T)T/I\log I} \leq e^{-k_0\log I} \leq e^{-k_0\ln I} = 1/I^{k_0}$. This holds for any $C \leq r(I+T)$ and thus the probability that the event for g and τ becomes critical after C packets 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 is at most $1/I^{k_0 t}$. Thus the probability that there exists a tree of size t in G_3 is at most $m'I^4 b^{18t}/I^{k_0 t} \leq m'I^{4-(k_0-234)t}$ (since there exists at most $m'I^4 b^{18t}$ different trees of size t in G_3 and $b \leq I^{13}$). Since $m' \leq \mathcal{Q} \leq \mathcal{P}$, we can make this probability less than $1/\mathcal{P}^{\beta'}$, for $t = \log\mathcal{P}$, and any constant $\beta' > 0$, by choosing k_0 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 the largest connected component in G_3 has at most t nodes, with probability at least $1 - 1/\mathcal{P}^{\beta'}$, and each of these nodes may have b^3 neighbors in G_2, the largest connected component in G_2 contains at most $b^3 t$ nodes, with the same probability. 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 $b^3 t$. Since each of these nodes may have as many as b endangered neighbors, the size of the largest connected component in G_1 is at most $b^4 t \leq I^{52} t$, with high probability.

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. After applying Proposition 3.6 once for some subproblem with $q \leq I^{52} t = I^{52} \log \mathcal{P} = I^{53}$, since $I = \log \mathcal{P}$, the relative congestion in frames of size $\log^2 I$ or larger in-between the first and last I^2 steps in the new schedule for this subproblem is at most r', with probability at least $1 - q/I^{\xi}$, for any constant $\xi > 0$. Hence, if we apply Proposition 3.6 to each subproblem, then the relative congestion of the packets not in P, in frames of size $\log^2 I$ or larger in-between the first and last I^2 steps in the new schedule, is at most r', with probability at least $1 - q/I^{\xi} \geq 1 - 1/I^{\xi - 53} = 1 - 1/(\log \mathcal{P})^{\xi - 53}$, since the subproblems are mutually independent and disjoint.

If we apply Proposition 3.6 $\log \mathcal{P}/(\log\log \mathcal{P})$ times to each (mutually disjoint and independent) routing subproblem, then for any constant $\xi > 53$ and $\mathcal{P}$ large enough, with probability at least $1 - 1/(\log \mathcal{P})^{(\xi - 53)\log \mathcal{P}/(\log\log \mathcal{P})} \geq 1 - 1/\mathcal{P}^{\xi - 53}$, the relative congestion in any frame of size $\log^2 I$ or greater is at most $r' = r(1 + O(1)/\sqrt{\log I})$. Applying Proposition 3.6 $\log \mathcal{P}/(\log\log \mathcal{P})$ times and checking whether the resulting schedule is feasible takes time at most $O(\mathcal{Q}(\log\log \mathcal{P})^2 \log \mathcal{P}/(\log\log \mathcal{P})) = O(\mathcal{Q}(\log\log \mathcal{P}) \log \mathcal{P})$.

We now have schedules for the packets in P and for the packets not in P. Both have relative congestion $r(1 + O(1)/\sqrt{\log I})$, with probability at least $1 - 1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. When we merge the two schedules, the relative congestion may be as large as the sum of the two relative congestions, that is, $2r(1 + O(1)/\sqrt{\log I})$, with probability at least $1 - 1/\mathcal{P}^{\beta}$, for any fixed $\beta > 0$. The running time of the algorithm is at most $O((\log\log \mathcal{P} + \log \mathcal{P})\mathcal{Q}(\log\log \mathcal{P})) = O(\mathcal{Q}(\log\log \mathcal{P}) \log \mathcal{P})$. $\square$

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 $1 \leq r \leq I$ and $I = (\log\log \mathcal{P})^2$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets within the block. Then 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 $2r'$, where $r' = r(1 + \sigma)$ and $\sigma = O(1)/\sqrt{\log I}$. For any constant $\beta > 0$, the algorithm runs in time at most $\mathcal{Q}(\log\log\log \mathcal{P})^{O(1)} \log \mathcal{P}$, with probability at least $1 - 1/\mathcal{P}^{\beta}$.*

Proof: The proof of this proposition is identical to the one presented for the previous proposition (we let $I = (\log\log \mathcal{P})^2$ in that proof), except for the last part, when we assign delays to the packets that are not in P.

In this case, we need to make another pass through the packets before applying Proposition 3.6 $\log \mathcal{P}/(\log\log\log \mathcal{P})^{O(1)}$ times to each component. In the first pass, we reduce the maximum component size in G_1 from $m' I^4$ to $I^{52} \log \mathcal{P}$, 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 $I^{52} \log(I^{52} \log \mathcal{P}) \leq I^{53}$, for large enough $I = (\log\log \mathcal{P})^2$, by taking $t = \log(I^{52} \log \mathcal{P})$, and noting that we now have $m' \leq I^{52} \log \mathcal{P}$. 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 $O(\log \mathcal{P}/(\log\log \mathcal{P}))$ 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}, and the overall time taken by the two passes was at most $O(\mathcal{Q} \log^2((\log\log \mathcal{P})^2) \log \mathcal{P}/(\log\log \mathcal{P})) \leq \mathcal{Q}(\log\log\log \mathcal{P})^{O(1)} \log \mathcal{P}/(\log\log \mathcal{P})$, with probability at least $1 - 2/\mathcal{P}^{\beta'}$. The second pass adds some packets to the set P. Let P_i denote the number of packets assigned delays in the i-th pass. 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 $2\log^2 I/I \leq 1/\sqrt{\log I}$. Now we apply Proposition 3.6 $\log \mathcal{P}/(\log\log\log \mathcal{P})^{O(1)}$ times, assigning delays to the packets not in $\mathcal{P}$ in time $\mathcal{Q}(\log\log\log \mathcal{P})^{O(1)} \log \mathcal{P}$, obtaining a feasible schedule for these packets with relative congestion $r(1 + O(1)/\sqrt{\log I})$, with probability at least $1 - 1/\mathcal{P}^{\beta'}$, for any fixed $\beta' > 0$.

We have schedules for the packets in P and for the packets not in P. Both have relative congestion $r(1+O(1)/\sqrt{\log I})$, with probability at least $1-2/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$. The total work performed was at most $\mathcal{Q}(\log\log\log\mathcal{P})^{O(1)}\log\mathcal{P}$. When we merge the two schedules, the resulting relative congestion may be as large as the sum of the two relative congestions, that is $2r(1+O(1)/\sqrt{\log I})$, with probability at least $1-1/\mathcal{P}^{\beta}$, for any fixed $\beta > 0$. $\square$

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 *Suppose we have a block of size $2I^3+2I^2-I$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets within this block, and let the relative congestion in any frame of size I or greater in this block be at most r, where $1 \le r \le I$ and $I \le (\log\log\log\mathcal{P})^{O(1)}$. Then there is some way of 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 $\sigma = O(1)/\sqrt{\log I}$. Furthermore, for any constant $\beta > 0$, this assignment can be found in $\mathcal{Q}(\log\log\log\log\mathcal{P})^{O(1)}\log\mathcal{P}$ time, with probability at least $1-1/\mathcal{P}^{\beta}$.*

Proof: The proof uses the Lovász Local Lemma to show that an assignment of initial delays satisfying the conditions of the proposition exists.

We first assign delays to some packets by making three passes through the packets using the algorithm of Proposition 3.7.1. After the first pass, with probability at least $1-1/\mathcal{P}^{\beta'}$, for any constant $\beta' > 0$, the size of the largest remaining subproblem will be $I^{52}\log\mathcal{P}$. We need to make two more passes, reducing the size of the largest subproblem first to $I^{52}\log(I^{52}\log\mathcal{P})$, and then to $I^{52}\log(I^{52}\log(I^{52}\log\mathcal{P})) = (\log\log\log\mathcal{P})^{O(1)}$ (for $I \le (\log\log\log\mathcal{P})^{O(1)}$). As in the second pass in the proof of Proposition 3.7.2, the two additional passes are repeated $O(\log\mathcal{P}/(\log\log\mathcal{P}))$ and $\log\mathcal{P}/(\log\log\log\mathcal{P})^{O(1)}$ times resp., for each component, if we fail to reduce the component size as desired. For any constant $\beta > 0$, we succeed in reducing the component size to $(\log\log\log\mathcal{P})^{O(1)}$ in time at most $\mathcal{Q}(\log\log\log\log\mathcal{P})^{O(1)}\log\mathcal{P}$, 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 $M_g = (r(I+T)-C_g)T/I + \sigma' r(I+T)T/I$ packets not in P use edge g in τ, where $\sigma' = O(1)/\sqrt{\log I}$ and C_g is the number of packets in P that traverse edge g during τ. As we argued before, 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 non-zero 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 σ', using exactly the same Chernoff-bound argument that was used in Proposition 3.7.1. Hence, $4\max\{p_g\}b \le 4/I < 1$ (for $I > 4$), and 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 $(I+1)^{r(I+T)(\log\log\log\mathcal{P})^{O(1)}} \le I^{2I^2(\log\log\log\mathcal{P})^{O(1)}}$ (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 the four passes (consider the exhaustive search we perform to assign delays to the packets not in P as the fourth pass) the number of packets that use an edge g in any T-frame is at most

$$\sum_{i=1}^{4}\left(\frac{C_i T}{I}+\sigma'\frac{r(I+T)T}{I}\right),$$

with probability at least $1-1/\mathcal{P}^{\beta}$, where C_i is the number of packets that could have possibly traversed edge g, and that were assigned delays in the i-th pass. Note that $C_g = C_1+C_2+C_3$. Since $C_1+C_2+C_3+C_4 \leq r(I+T)$, the number of packets that traverse any edge in any T-frame is at most

$$\frac{r(I+T)T}{I}+4\sigma'\frac{r(I+T)T}{I},$$

which means that the relative congestion in any T-frame, where $\log^2 I \leq T < 2\log^2 I$, is at most

$$\begin{aligned}\left(\frac{r(I+T)}{I}\right)(1+4\sigma') &= r\left(1+\frac{T}{I}\right)(1+4\sigma') \\ &= r\left(1+\frac{O(1)}{\sqrt{\log I}}\right) = r(1+\sigma),\end{aligned}$$

as claimed, since $2\log^2 I/I \leq 1/\sqrt{\log I}$, for I large enough.

We can bound the total time taken by the algorithm as follows. The first three passes take time at most $\mathcal{Q}(\log\log\log\log)^{O(1)}\log\mathcal{P}$, 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 T-frame τ in the subproblem, for every edge g in τ, for $\log^2 I \leq T < 2\log^2 I$, we must check whether more than M_g packets traverse g during τ. This takes time at most $O(\mathcal{Q}\log^2 I\log^{\gamma}\mathcal{P}) \leq \mathcal{Q}(\log\log\log\log\mathcal{P})^{O(1)}\log^{\gamma}\mathcal{P}$, for $\mathcal{P}$ large enough, for any fixed $\gamma > 0$. In particular, for $\gamma = 1$ and $\mathcal{P}$ large enough, this quantity is bounded by $\mathcal{Q}(\log\log\log\log\mathcal{P})^{O(1)}\log\mathcal{P}$. Hence the overall time taken is bounded by $\mathcal{Q}(\log\log\log\log\mathcal{P})^{O(1)}\log\mathcal{P}$.

□

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 (see [9]). The three propositions that follow are analogous to the three replacement propositions, Propositions 3.7.1-3, for Lemma 3.7 of [9]. Suppose we have a block of size $2I^3+3I^2$, obtained after we inserted delays into the schedule as described in Propositions 3.6 and 3.7.1-3, and moved the block boundaries as described in [9]. Each proposition refers to a specific size of I. Note that in [9], the steps between steps I^3 and I^3+3I^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 $1 \leq r \leq I$.

Proposition 3.9.1 *Let $I = \log\mathcal{P}$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets within the block. Then 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 $2r_1$, where $r_1 = r(1+\sigma_1)$ and $\sigma_1 = O(1)/\sqrt{\log I}$, 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 $2r_2$, where $r_2 = r(1+\sigma_2)$ and $\sigma_2 = O(1)/\sqrt{\log I}$. For any constant $\beta > 0$, this algorithm runs in time at most $O(\mathcal{Q}(\log\log\mathcal{P})\log\mathcal{P})$, with probability at least $1-1/\mathcal{P}^{\beta}$.*

Proposition 3.9.2 *Let $I = (\log\log\mathcal{P})^2$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets within the block. Then 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 $2r_1$, where $r_1 = r(1+\sigma_1)$ and $\sigma_1 = O(1)/\sqrt{\log I}$, 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 $2r_2$, where $r_2 = r(1+\sigma_2)$ and $\sigma_2 = O(1)/\sqrt{\log I}$. For any constant $\beta > 0$, this algorithm runs in time at most $\mathcal{Q}(\log\log\log \mathcal{P})^{O(1)} \log \mathcal{P}$, with probability at least $1 - 1/\mathcal{P}^{\beta}$.*

Proposition 3.9.3 *Let $I \leq (\log\log\log \mathcal{P})^{O(1)}$. Let $\mathcal{Q}$ be the sum of the lengths of the paths taken by the packets within the block. Then 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 $r_1 = r(1+\sigma_1)$, 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 $r_2 = r(1+\sigma_2)$, where $\sigma_2 = O(1)/\sqrt{\log I}$. For any constant $\beta > 0$, this algorithm runs in time at most $\mathcal{Q}(\log\log\log\log \mathcal{P})^{O(1)} \log \mathcal{P}$, with probability at least $1 - 1/\mathcal{P}^{\beta}$.*

3.5 Running time

Theorem 3.1 *For any constant $\beta > 0$, the overall running time of the algorithm is at most $O(\mathcal{P}(\log\log \mathcal{P}) \log \mathcal{P})$, with probability at least $1 - 1/\mathcal{P}^{\beta}$.*

Proof: Note that each of the propositions in the previous sections dealt only 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 schedule for the given I. Assume the blocks are numbered from 1 to n_I. Note that $\sum_{i=1}^{n_I} m_i$ may be as large as $\mathcal{P}$ and $\sum_{i=1}^{n_I} \mathcal{Q}_i = \mathcal{P}$, where $\mathcal{Q}_i$ and m_i are the sum of the path lengths and the number of edges used within block i, resp.. Let β be any positive constant. For each partition of the schedule for a given $I \leq (\log\log\log \mathcal{P})^{O(1)}$, we apply Propositions 3.7.3 and 3.9.3 for each block i in this partition, $1 \leq i \leq n_I$, taking overall time at most $\mathcal{P}(\log\log\log\log \mathcal{P})^{O(1)} \log \mathcal{P}$. 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 at most $\mathcal{P}(\log\log\log\log \mathcal{P})^{O(1)}(\log \mathcal{P}) \log^*(c+d)$. Thus the total running time of the algorithm is at most $O(\mathcal{P}) + [O(\log\log \mathcal{P}) + (\log\log\log \mathcal{P})^{O(1)} + (\log\log\log\log \mathcal{P})^{O(1)} \log^*(c+d)]\mathcal{P} \log \mathcal{P} \leq O(\mathcal{P}(\log\log \mathcal{P}) \log \mathcal{P})$, for $\mathcal{P}$ large enough, with probability at least $1 - 1/\mathcal{P}^{\beta}$ (since each of the propositions is successful with probability at least $1 - 1/\mathcal{P}^{\beta'}$, for any positive constant β'). Note that we used the inequalities $\mathcal{P} \geq c$ and $\mathcal{P} \geq d$. □

4 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 it might be involved in 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 $ND^L \cdot \frac{2}{L!} << 1$.

Each packet can be involved in at most $r(2I^3 + I^2)\log^2 I$ 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 $O(I^4 \log^2 I)$. 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 $\log N$. Hence, the parallel algorithm runs in NC, as claimed.

5 Remarks

The algorithms described in this paper are randomized, but they can be derandomized using the method of conditional probabilities [12, 13].

References

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

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

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

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

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

[6] P. Erdös and L. Lovás. 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. Koch, T. Leighton, B. Maggs, S. Rao, and A. Rosenberg. Work-preserving emulations of fixed-connection networks. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 227–240, May 1989.

[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):157–205, July 1994.

[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):167–180, 1994.

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

[11] F. Meyer auf der Heide and B. Vöcking. A packet routing protocol for arbitrary networks. In *Proceedings of the 12th Symposium on Theoretical Aspects of Computer Science*, pages 291–302, March 1995.

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

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