General Dynamic Routing with Per-Packet Delay Guarantees of O(distance + 1 / session rate) *

Matthew Andrews[†] Antonio Fernández[‡] Mor Harchol-Balter[§] Tom Leighton[¶]
Lisa Zhang[||]

Laboratory for Computer Science, MIT.
{andrews, anto, harchol, ftl, ylz}@theory.lcs.mit.edu

Abstract

A central issue in the design of modern communication networks is that of providing performance guarantees. This issue is particularly important if the networks support real-time traffic such as voice and video. The most critical performance parameter to bound is the delay experienced by a packet as it travels from its source to its destination.

We study the dynamic routing problem in a connection-oriented packet-switching network. We consider a network with arbitrary topology on which a set of sessions is defined. Each session follows a fixed path from a source to a destination. Packets are injected continually in the sessions with rate r_i for session i. During any time step at most one packet may traverse any edge. Session paths may overlap subject to the constraint that the total rate of sessions using any particular edge is less than 1.

We show the existence of an asymptotically-optimal schedule that achieves a delay bound of $O(1/r_i + d_i)$ with only constant queues at the switches, where d_i is the path length of session i. We also present a simple distributed algorithm that, with high probability, delivers every session-i packet to its destination within $O(1/r_i + d_i \log(m/r_{\min}))$ steps of its injection, where $r_{\min}$ is the minimum session rate, and m is the number of edges in the network. Our results can be generalized to (leaky-bucket constrained) bursty traffic, where session i tolerates a burst size of b_i. In this case, our delay bounds become $O(b_i/r_i + d_i)$ and $O(b_i/r_i + d_i \log(m/r_{\min}))$, respectively.

*Supported by Army grant DAAH04-95-1-0607 and ARPA contract N00014-95-1-1246.

[†]Supported by NSF contract 9302476-CCR.

[‡]Supported by the Spanish Ministry of Education. Current address: Departamento de Arquitectura y Tecnología de Computadores, Universidad Politécnica de Madrid, Spain, anto@eui.upm.es.

[§]Supported by NSF Postdoctoral Fellowship in the Mathematical Sciences.

[¶]Supported by Army grant DAAH04-95-1-0607 and ARPA contract N00014-95-1-1246.

[||]Supported by an NSF graduate fellowship.

1 Introduction

1.1 Motivation

Motivated by the need for quality-of-service guarantees, network designers today offer *connection-oriented* service in many networks, e.g. ATM (Asynchronous Transfer Mode) networks, so that, when setting up a session, a user requests a particular share of the bandwidth, and once the session is established, it injects a stream of packets along the session at the agreed upon rate. An important consequence of the user's predictability is that the network can, in return, guarantee the user an *end-to-end delay bound*, i.e. an upper bound on the time that any packet takes to move from its source to its destination. In order to provide this delay guarantee, the network must determine how to schedule the packets that contend for the same link simultaneously. Apart from delay bounds, it is also important to guarantee small queues at each switch due to limited buffer size. In this paper we show, for the first time, that it is possible to design such a schedule that guarantees asymptotically-optimal per-session delay bounds as well as small queues.

1.2 Model and problem

Consider a network $\mathcal{N}$ of arbitrary topology and a set of sessions defined on this network. A session i is associated with a source node, a destination node, and a simple path from the source to the destination. (A path is *simple* if it uses each edge at most once.) Packets are injected to the network $\mathcal{N}$ in sessions. A packet injected in session i enters the system at the source node of i, traverses the path associated with i, and then is absorbed at its destination. The length d_i is the number of edges on the path from the source to the destination of session i.

Each session i has also defined an injection rate r_i. This rate constrains the injection of new packets from the session so that, during any interval of t consecutive steps, at most $tr_i + 1$ packets can be injected in session i, for any t.

We assume that all packets have the same size. We also assume a synchronized store-and-forward routing, where at each step at most one packet can traverse each edge. When two packets simultaneously contend for the same edge, one packet has to wait in a queue. During the routing, packets wait in two different kinds of queues. After a packet has been injected, but before it leaves its source, the packet is stored in an *initial queue*. Once the packet has left its source, during any time it is waiting to traverse an edge, the packet is stored in an *edge queue*. The *end-to-end delay* (delay for short) for a packet is the total time from the packet injection until it reaches its destination. This includes the total time the packet spends waiting in both types of queues, plus the time it spends traversing edges.

Our goal is to minimize both the end-to-end delay for each packet and the size of all edge queues. In order to achieve delay guarantees and bounded queue sizes, it is necessary to require that, for all edges e, the sum of the rates of the sessions that use edge e is at most 1. Throughout, we shall assume that the sum of the rates of the sessions using any edge e is at most $1 - \varepsilon$, for a constant $\varepsilon \in (0, 1)$.

Our paper focuses on the problem of timing the movements of the packets along their paths. A *schedule* specifies which packets move and which packets wait in queues at each time step. We address the problem of finding a schedule which, for each session, guarantees an asymptotically-optimal delay bound for packets of that session, while maintaining constant-size edge queues.

In this paper all the schedules we obtain are periodic. The schedule defines a fixed *template* for each edge in advance. A template of size M is a wheel with M slots, each of which contains at most one token. Each token is affiliated with some session. The wheel spins at the speed of one slot per time step. A session-i packet can traverse the edge if and only if a session-i token appears. For each session-i token, the session-i packet that uses it will be the one that has been waiting to cross the edge for the longest amount of time, i.e. the session-i packets use the session-i tokens in a First-Come-First-Served manner. The template size and associated tokens do not change over time.

1.3 Lower Bounds

Observe that d_i is always a lower bound on the delay for session i, since every session-i packet has to cross d_i edges. It is also easy to see that $\Omega(1/r_i)$ is an existential lower bound. For instance, consider n sessions, all of which have the same rate $r = (1-\varepsilon)/n$ and the same initial edge e. If a packet is injected in each session simultaneously, one of the packets requires $n = \Omega(1/r)$ steps to cross e.

Furthermore, for *any* given set of sessions, $\Omega(1/r_i)$ is a lower bound for some session i in template-based schedules: It can be proven that there exist two session-i tokens separated by at least $(1-\varepsilon)/r_i$ slots in the template, for some i. Then, an adversary can make sure a session-i packet arrives just after the first token has passed, thereby forcing the packet to wait $\Omega(1/r_i)$ steps.

If the schedule is not restricted to being template-based, the scheduler is more powerful. The scheduler does not have to decide on a fixed schedule in advance, but rather can make a new decision at each step, based on seeing the adversary's injections. In this case it is unknown if for *any* given set of sessions $\Omega(1/r_i)$ is a lower bound.

1.4 Previous Work

The problem of dynamic packet routing in the above setting is well studied. Until recently the best delay bound known was $O(d_i/r_i)$ for packets of session i. It is tempting to believe that this is best possible since a session-i packet may need to wait $O(1/r_i)$ steps to cross each of the d_i edges on its route. However, this leaves a large gap between the upper bound of $O(d_i/r_i)$ and the lower bound of $O(1/r_i + d_i)$. The recent work seeks to close this gap.

In 1990, Demers, Keshav and Shenker [6] proposed a widely-studied routing algorithm called Weighted Fair Queueing (WFQ). WFQ is a packetized approximation of the idealized fluid model algorithm Generalized Processor Sharing (GPS). WFQ is simple and distributed. This same algorithm was proposed independently by Parekh and Gallager [12, 13] in 1992 under the name of Packet-by-Packet Generalized Processor Sharing (PGPS). Parekh and Gallager prove that the algorithm has an end-to-end delay guarantee of $2d_i/r_i$ [13, page 148] in the case when all packets have the same size.

In their 1996 paper, Rabani and Tardos [14] produce an algorithm which routes every packet to its destination with probability $1-p$ in time $O(1/r_{\min}) + (\log^* p^{-1})^{O(\log^* p^{-1})} D + poly(\log p^{-1})$, where $r_{\min} = \min_i r_i$ and $D = \max_i d_i$. Ostrovsky and Rabani improve the bound to $O(1/r_{\min} + D + \log^{1+\varepsilon} p^{-1})$ [11]. The algorithms of Rabani et al. are distributed, where knowledge of the entire network is not assumed, but each packet carries some information. These bounds are not session-based, meaning that if one session has a small rate or a long distance then the delay bounds for all sessions will suffer.

The main technique of Rabani et al. is based on "delay-insertion". The intuition here is that if each packet receives a large random initial delay, then the packets are sufficiently spread out to ensure that they only need to wait $O(1)$ steps at each successive edge rather than $O(1/r_i)$ steps. This delay-insertion technique is introduced by Leighton et al. in [8, 9] in the context of static routing. (In the *static* routing problem, all packets are present in the network initially.) Since our main result employs many techniques from [8], we give a detailed summary in Section 3.1.

A constrasting model is the *connectionless adversarial queueing model* proposed in [2] and later studied in [1]. Here the paths on which packets are injected can change over time giving the adversary more power. In the adversarial queueing model the best delay bound known is polynomial in the maximum path length [1].

1.5 Our Results

We first provide a randomized, distributed scheduler that achieves a delay bound of $O(1/r_i + d_i \log(m/r_{\min}))$ and a bound on the queue size of $O(\log(m/r_{\min}))$, where m is the number of edges in the network and $r_{\min} = \min_i r_i$. While this bound is not optimal, it nevertheless conveys some intuition for our main result.

The main contribution of this paper is an asymptotically-optimal schedule. We prove that a schedule exists for the dynamic routing problem such that the end-to-end delay of each session-i packet is bounded by $O(1/r_i + d_i)$. Our result improves upon previous work in several aspects.

- We provide a session-based delay guarantee. That is, packets from sessions with short paths and high injection rates reach their destinations fast. This is a big improvement over the previous bounds, which are stated in terms of $r_{\min} = \min_i r_i$ and $D = \max_i d_i$. We also guarantee that every packet always reaches its destination within the delay bound, without dropping any packets.

- We guarantee constant-size edge queues. This is interesting because edge queues are much more expensive than initial queues in practice.

- A consequence of our result is a packet-based bound, which improves upon the $O(c + d)$ bound in [8] for the static problem (see Section 3.1 for the problem and parameter definitions). Suppose packet p_i follows a route P_i, then p_i can be routed to its destination within $O(c_i + d_i)$ steps, where c_i is the maximum congestion along P_i and d_i is the number of edges on P_i. The result trivially follows from our result by creating a different session i for each packet p_i, and defining $r_i = (1 - \varepsilon)/c_i$.

The asymptotically-optimal schedule is template-based. Even if the computation of the schedule is time-consuming, it only needs to be done once. Packets can then be scheduled indefinitely as long as the sessions do not change.

Leaky-bucket injection model Our results above can be generalized to bursty traffic streams that are *leaky-bucket regulated*. Here, each session i has a maximum burst size (or bucket size) of $b_i \geq 1$ and an average arrival rate of r_i. During any t consecutive time steps at most $r_i t + b_i$ session-i packets are injected. Leaky-bucket regulated traffic is widely used in the literature, e.g. [4, 5, 7, 12, 13, 16].

Leaky-bucket regulated injections allow traffic shaping. When session-i packets are injected, they first enter the session-i bucket at the source. These packets then leave the bucket one at a time at the rate of r_i. In this way, the end-to-end delay is separated into two components, delay in the bucket and delay in the network. Since delay in the bucket is at most b_i/r_i, the end-to-end delay is increased by at most b_i/r_i steps, and the size of the edge queues is unchanged.

The rest of the paper is divided into sections as follows. We first describe a simple distributed scheduler that has a delay bound of $O(1/r_i + d_i \log(m/r_{\min}))$ in Section 2. In Section 3 we overview the main techniques to achieve a bound of $O(1/r_i + d_i)$ and constant-size edge queues. The proof details are presented in Section 4.

2 A Preliminary Result

Before proving our main result, we first present a simple centralized schedule and a simple distributed schedule that achieve delay bounds of $O(1/r_i + d_i \log(m/r_{\min}))$ with high probability. In addition, the centralized schedule has a maximum queue size of $O(\log(m/r_{\min}))$ with high probability. These preliminary results are substantially simpler to prove because of the relaxed bounds on delay and queue sizes. Nevertheless, they illustrate the basic ideas of the main result.

In this section, for ease of presentation, we omit floors and ceilings where they are necessary. We shall also assume that $1/r_i$ is an integer for all i and there is a constant k such that $k/r_{\min}$ is a multiple of $1/r_i$ for all i.[1]

2.1 Template-Related Definitions

Throughout this paper we are considering template-based schedules. Specifying a template-based schedule reduces to the problem of assigning tokens to the slots on the templates. Our usual strategy will be to assign tokens in *token sequences.* A token sequence for session i will consist of d_i tokens, one for each edge on session i. For each method of assigning tokens we initially provide bounds on delay and queue size for a *token sequence* schedule. In this schedule, packets use tokens for the initial edge in a First-Come-First-Served manner. However, once a packet has used one token from a particular token sequence then for subsequent edges it *only* uses tokens from that token sequence. Below we state a theorem showing that these bounds give us bounds for the corresponding template-based schedule in which session-i packets use session-i tokens in a First-Come-First-Served manner on all edges. The proof is contained in Section 4.5. If no ambiguity arises, we shall often refer to scheduling packets rather than assigning tokens in token sequences.

Let $\kappa_1, \ldots, \kappa_{d_i}$ be a token sequence for session i. If κ_{j+1} appears x_j steps after κ_j, then x_j is the *token lag* for these two tokens and $\sum_{j=1}^{d_i-1} x_j$ is the end-to-end delay for this token sequence. The token sequences for each session i form a partition of all the session-i tokens. In the following we state that, in any template-based schedule, bounding the delay for token sequences is sufficient to bound the packet delays and that bounding the token lag is sufficient to bound the edge queues.

Theorem 1 *Consider any template-based schedule. If the end-to-end delay for each session-i token sequence is bounded by X, then each session-i packet reaches its destination within X steps after it obtains an initial token. If the token lag is bounded by x for* all *token sequences for* all *sessions, then the edge queue size is also bounded by x.*

2.2 A Simple Centralized Schedule

We now describe the centralized schedule which we shall call TEMPLATE. Let $M = k/r_{\min}$ where k is the constant chosen above. Each template will have size M. We first place $r_i M$ *initial tokens* on the template for the first edge of session i, spaced $1/r_i$ slots apart.

[1] In the main text, we choose to assume the existence of k so as to avoid obscuring the main ideas. If there is no such constant k, we can always show the existence of $\hat{r}_i$ for each session i such that the following holds. i) $\hat{r}_i$ is a fraction of s_i/ℓ_i, where s_i, ℓ_i are integers and $\ell_i = \Theta(1/r_i)$ is a power of 2; ii) $r_i \leq \hat{r}_i$; iii) $\sum_i \hat{r}_i \leq 1 - 5\varepsilon/6$ for all edges. Hence, if we choose the template size $M = \max_i \ell_i$, then $M = \Theta(1/r_{\min})$ is a multiple of all ℓ_i's. In the token placement process of Section 2.2, we place s_i tokens in one slot every ℓ_i slots.

Lemma 2 *Each session-i packet will obtain a session-i token at most $2/r_i$ steps after its injection.*

Proof: Suppose that packet p is injected at time t but has not obtained an initial token by time $t + 2/r_i + 1$. Let t' be the last time before $t + 2/r_i + 1$ that there were no session-i packets waiting for initial tokens. (Note that $t' < t$.) Between times t' and t at most $(t - t')r_i + 1$ session i packets are injected. However, at least $(t - t' + 2/r_i)r_i - 1 = (t - t')r_i + 1$ initial tokens for session i appear between times t' and $t + 2/r_i$. By the definition of t', each of these tokens was used by a packet. Hence none of the packets injected between t' and t (and in particular packet p) can still be waiting for an initial token at time $t + 2/r_i$. □

Once the initial session-i tokens are placed we delay each of them by an amount chosen uniformly and independently at random from $[L + 1, L + 1/r_i]$, where $L = \frac{\alpha}{2}\log(mM)$ and α is a constant. The intuition is that the random delays would spread out the tokens. After the tokens have been delayed we can be sure that each packet obtains an initial token within $L + 3/r_i$ steps. We now create the token sequences. (Recall the definition of token sequence from Section 2.1. For every session-i token a placed in the template corresponding to the jth edge, we place a session-i token b on the template corresponding to the $(j + 1)$st edge such that b appears exactly $2L$ steps after a.

We observe that two different session-i packets p and p' have different starting times T and T', and therefore different session-i packets do not cross an edge simultaneously. Unfortunately, tokens from different sessions may be placed in one slot, which would cause packets from different sessions to cross the same edge simultaneously. The following lemma shows that the tokens are not clustered to any great extent.

Lemma 3 *There are at most L tokens in any consecutive L slots on any template with probability $1 - 1/(mM)$, where $L = \frac{\alpha}{2}\log(mM)$ and α is a sufficiently-large constant.*

Proof: Since the initial tokens for session i are spaced $1/r_i$ apart and each is delayed by an amount chosen independently and uniformly at random from $[L + 1, L + 1/r_i]$, the expected number of session-i tokens in a single slot is r_i. For a particular interval of L consecutive slots on a particular template, let the random variable X equal the number of tokens in these slots. By linearity of expectations, $E[X] \le \sum_i r_i L \le (1 - \varepsilon)L$. (Note also that $\sum_i \lfloor r_i L \rfloor \le X \le \sum_i \lceil r_i L \rceil$.) Whether or not a token lands in these L slots is a Bernoulli event. Since the delays to the initial tokens are chosen independently and all session paths are simple, these Bernoulli events are independent. Since $E[X] \le (1 - \varepsilon)L$, we have the following by a Chernoff bound. (See Section 4.1.)

$$\Pr[\ X > L\ ] \le \Pr[\ X > (1 + \varepsilon)(1 - \varepsilon)L\ ] \le e^{-\varepsilon^2(1-\varepsilon)L/3}.$$

In m templates there are at most mM intervals of L consecutive slots. Therefore, by a union bound the probability that more than L tokens appear in *any* L consecutive slots is bounded by,

$$mM \Pr[\ X > L\ ] \le mMe^{-\varepsilon^2(1-\varepsilon)L/3} = mMe^{-\varepsilon^2(1-\varepsilon)\alpha\log(mM)/6}.$$

By choosing a sufficiently large constant α, we can bound the above probability by $1/(mM)$. □

Lemma 3 is not sufficient to guarantee one token per slot. We solve this problem by partitioning each template into intervals of L consecutive slots[2] and "smoothing out" each interval as follows. We take the at most L tokens from these slots and rearrange them arbitrarily so that there is at most one token in each slot. We have,

[2] Here we assume M is a multiple of L. This can be achieved by choosing M and L to be powers of 2. (See the previous footnote.)

Lemma 4 *Consider a packet p. Let $\mathcal{K}^p$ be the token sequence that contains the initial token used by p* before *the smoothing process. Let κ_j^p be the token for the jth edge in this token sequence. Then,* after *the smoothing process, the packet p can use the token κ_j^p to cross its jth edge. Therefore, p reaches its destination within $O(1/r_i + d_i \log(m/r_{\min}))$ steps of its injection.*

Proof: It is sufficient to show that, after the smoothing process, token κ_1^p (the initial token) appears after the injection of p and κ_{j+1}^p appears after κ_j^p. A token is shifted by at most $L-1$ steps by the smoothing process. Before the smoothing, κ_1^p appears at least L steps after the injection of p and κ_{j+1}^p appears exactly $2L$ steps after κ_j^p. The lemma follows. □

Hence, we have presented a schedule TEMPLATE that assigns tokens on the templates with at most one token per slot with high probability. Note that if the first execution of TEMPLATE assigns more than one token per slot, TEMPLATE can be executed again until the condition of one token per slot is satisfied. We have already bounded the delay experienced by packets. We now show that the queue size is small.

Lemma 5 *Every session-i packet waits at most $O(\log(m/r_{\min}))$ steps to cross each edge. Therefore, the queue size is $O(\log(m/r_{\min}))$.*

Proof: Suppose that packet p uses token κ_j^q from token sequence $\mathcal{K}^q$ to cross its jth edge. Then packet p can use κ_{j+1}^q to cross its $j+1$st edge. This token κ_{j+1}^q appears at most $4L = O(\log(m/r_{\min}))$ steps later than κ_j^q. The result follows. □

Therefore,

Theorem 6 *With high probability, the randomized centralized schedule* TEMPLATE *has a delay bound of $O(1/r_i + d_i \log(m/r_{\min}))$ and a queue size of $O(\log(m/r_{\min}))$.*

2.3 A Simple Distributed Schedule

The above scheme TEMPLATE is centralized since the session-i tokens on one template are dependent on the previous template. However, it suggests the following simple *distributed* strategy for scheduling packets so as to achieve small delay. As with the centralized schedule, we place initial tokens on the first edge of session i and then delay each token by an amount chosen independently and uniformly at random from $[L+1, L+1/r_i]$. Suppose that a packet now has its initial token at time T. Then for the kth edge on this packet's path the packet is given a "deadline" of $T + 2L(k-1) + L$, where $L = \frac{\alpha}{2}\log(mM)$. Whenever two or more packets contend for the same edge simultaneously, the packet with the earliest deadline moves. We call this scheme EARLIEST-DEADLINE-FIRST (EDF).

Lemma 7 *For any edge, there are at most L deadlines in any consecutive L time steps with probability at least $1 - 1/(mM)$, where $L = \frac{\alpha}{2}\log(mM)$ and α is a sufficiently large constant.*

Proof: The proof is almost identical to that of Lemma 3. □

Lemma 8 *If for any edge, there are at most L deadlines in any consecutive L time steps, then each packet crosses every edge by its deadline.*

Proof: For the purpose of contradiction, let D be the first deadline that is missed. This implies that all deadlines earlier than D are met. Let p be the packet that misses deadline D for edge e. Since packet p makes its previous deadlines, p must have crossed its previous edge by time $D-L$, or else e must be p's first edge and p must have obtained its initial token by time $D-L$. Hence, at every time step from time $D-L+1$ to D packet p is held up by another packet with deadline no later than D. Furthermore, these deadlines must be later than $D-L$ since all deadlines earlier than D are met. Therefore, at least $L+1$ packets have deadlines for edge e from time $D-L+1$ to D. This contradicts Lemma 7. □

Lemmas 7 and 8 imply,

Theorem 9 *With high probability, the randomized distributed schedule* EARLIEST-DEADLINE-FIRST *achieves a delay bound of* $O(1/r_i + d_i \log(m/r_{\min}))$.

Note that EDF does not generate a template for each edge. Instead, it generates a list of $r_i M$ initial deadlines for the first edge of session i, and gives them in order to the session-i packets injected.

3 Summary of the Main Result

Our main result for the dynamic routing problem parallels an earlier result on static routing. In Section 3.1 we review the method used for solving the static case, and in Section 3.2 we give an overview of the additional complexities that need to be addressed in the dynamic case.

3.1 A Bound of $O(c+d)$ for Static Routing

Leighton, Maggs and Rao consider the static routing problem for arbitrary networks in [8]. For static routing, all packets are present in the network initially. Each packet is associated with a source, a destination, and a route. The *congestion* on each edge is the total number of routes that require this edge, and the *dilation* of a route is the number of edges on the route. Leighton et al. show that for any set of routes with maximum congestion c (over all edges) and maximum dilation d (over all routes), there is a schedule of length $O(c+d)$ and edge queue size $O(1)$. In this schedule, at most one packet traverses each edge at each time step. A packet waits $O(c+d)$ steps initially before leaving its source, and it waits $O(1)$ steps to cross each edge afterwards.

We summarize here the techniques in [8]. The strategy for constructing an efficient schedule is to make a succession of *refinements* to an initial schedule $\mathcal{S}^{(0)}$. In $\mathcal{S}^{(0)}$, each packet moves at every step until it reaches its destination. This schedule has length d, but as many as c packets may traverse the same edge at the same step. Each refinement brings the schedule closer and closer to the requirement that at most one packet uses one edge per time step.

A *T-frame* is a time interval of length T. The *frame congestion*, C, in a T-frame is the largest number of packets that use any edge during the frame. The *relative congestion* in a T-frame is the ratio C/T. The frame congestion (resp. relative congestion) on an edge e during a T-frame is defined to be the frame congestion (resp. relative congestion) associated with edge e.

It is obvious that the initial schedule $\mathcal{S}^{(0)}$ has relative congestion at most 1 for any c-frame. A *refinement* transforms a schedule $\mathcal{S}^{(q)}$ with relative congestion at most $c^{(q)}$ in any frame of size $I^{(q)}$ or larger into a schedule $\mathcal{S}^{(q+1)}$ with relative congestion at most $c^{(q+1)}$ in any frame of size $I^{(q+1)}$ or larger. The resulting frame size $I^{(q+1)}$ is much smaller than $I^{(q)}$, whereas the relative congestion $c^{(q+1)}$ is only slightly bigger than $c^{(q)}$. In particular, $I^{(q+1)} = \log^5 I^{(q)}$ and $c^{(q+1)} = (1+o(1))c^{(q)}$.

After a series of $O(\log^* c)$ refinements, a schedule $\mathcal{S}^{(\zeta)}$ is obtained where the relative congestion is $O(1)$ for any $O(1)$-frame. A final schedule can be constructed by stretching $\mathcal{S}^{(\zeta)}$ by a constant factor. Each refinement is achieved by inserting delays to the packets. It is the central issue in [8] to show that a set of delays always exists satisfying the criteria in the table below.

Schedule	Frame size	Relative congestion
$\mathcal{S}^{(q)}$	$I^{(q)}$	$c^{(q)}$
Refinement	$\log^5 I^{(q)}$	$(1+o(1))c^{(q)}$
$\mathcal{S}^{(q+1)}$	$I^{(q+1)}$	$c^{(q+1)}$

3.2 A Bound of $O(1/r_i + d_i)$ for Dynamic Routing

Our result for the dynamic routing problem is parallel to that in [8]. For an arbitrary network where paths (sessions) are defined, we show that there is a schedule such that every session-i packet reaches its destination within $O(1/r_i + d_1)$ steps of its injection, where r_i and d_i are the injection rate and path length for session i, respectively. A session-i packet waits $O(1/r_i + d_i)$ steps initially before leaving its source, and it waits $O(1)$ steps to cross each edge afterwards.

To achieve a session-based, end-to-end delay bound of $O(1/r_i + d_i)$ for our dynamic routing problem, we adopt the general approach in [8]. However, there are three major problems in transforming the solution for the static problem into a solution for the dynamic problem. In the following we present these three problems and their solutions.

Problem 1: Infinite time

In [8] all the packets to be scheduled are present initially. In the dynamic model, packets are injected over an infinite time line. We would like to partition the infinite time line into finite time intervals which can be scheduled independently of each other. We divide time into intervals of length $\mathcal{T}$, where $\mathcal{T}$ is defined in Section 4.2. We shall then independently schedule the time intervals $[0, \mathcal{T})$, $[\mathcal{T}, 2\mathcal{T})$, $[2\mathcal{T}, 3\mathcal{T})$, etc.

We associate each session i with a quantity $\mathcal{T}_i = O(1/r_i + d_i)$, which is also defined in Section 4.2. For any integer $k \geq 0$ consider all the session-i packets that are injected during interval $[k\mathcal{T} - \mathcal{T}_i, (k+1)\mathcal{T} - \mathcal{T}_i)$. We shall provide a schedule in which all these packets leave their sources no earlier than time $k\mathcal{T}$ and reach their destinations before time $(k+1)\mathcal{T}$. (See Figure 1.) From now on, we shall concentrate on scheduling the arrivals that would be serviced during interval $[\mathcal{T}, 2\mathcal{T})$.

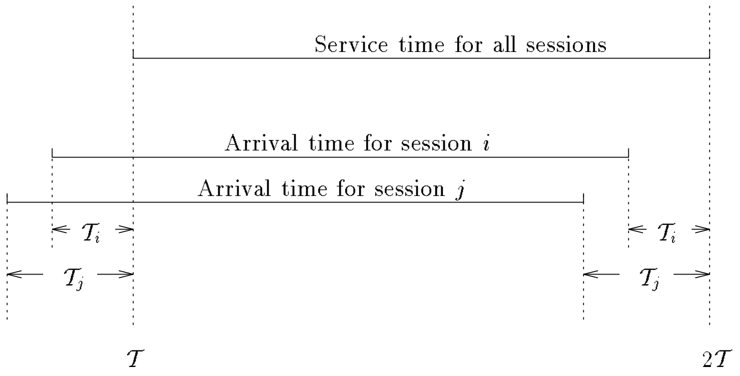

Figure 1: All the session-i packets that arrive during $[k\mathcal{T} - \mathcal{T}_i, (k+1)\mathcal{T} - \mathcal{T}_i)$ are serviced during $[k\mathcal{T}, (k+1)\mathcal{T})$. In this figure, $k = 1$.

Problem 2: Session-based delay guarantees

Once we restrict ourselves to interval $[\mathcal{T}, 2\mathcal{T})$, it seems that the dynamic routing problem is similar to the static problem. However, we cannot simply proceed with the successive refinements as in Section 3.1, since some sessions need tighter delay bounds than others. Session-i packets can only tolerate a delay proportional to $1/r_i + d_i$. We group sessions according to their associated $1/r_i + d_i$ value. We start by inserting delays to sessions having large values of $1/r_i + d_i$, reducing the frame size and bounding the relative congestion. When the frame size becomes small enough, sessions with smaller $1/r_i + d_i$ join in.

More precisely, we introduce the concept of *integral* and *fractional* sessions. When session i is *integral*, packets of size 1 are injected at rate r_i. When session i is *fractional*, a packet of size $\hat{r}_i$ is injected at every time step, where $\hat{r}_i \approx r_i$ is defined in Section 4.2. A packet from a fractional session always crosses one edge at a time, whether or not other packets are crossing the edge at the same time. Therefore, a fractional packet from session i always contributes exactly $\hat{r}_i$ to the congestion. Integral sessions are those to which we can afford to insert delays in order to bound the congestion. Fractional sessions are those to which we cannot insert delays. However, congestion due to a fractional session i is only $\hat{r}_i$, which is small.

As before, $\mathcal{S}^{(q)}$ represents the schedule in the qth iteration. The set of integral sessions for $\mathcal{S}^{(q)}$ is denoted by $A^{(q)}$. For the initial schedule $\mathcal{S}^{(0)}$, all the sessions are fractional and we show that the relative congestion is less than 1. For schedule $\mathcal{S}^{(q)}$ we inductively assume that the relative congestion due to the current integral and fractional sessions is at most $c^{(q)}$ for any frame of size $I^{(q)}$ or larger. To create a schedule $\mathcal{S}^{(q+1)}$ from schedule $\mathcal{S}^{(q)}$ we carry out a frame-refinement step in Section 4.3.2 and a conversion step in Section 4.3.3.

The frame-refinement step reduces the frame size from $I^{(q)}$ to $I^{(q+1)} = \log^5 I^{(q)}$, while slightly increasing the relative congestion from $c^{(q)}$ to $(1 + o(1))c^{(q)}$. This step is achieved by delaying the integral packets by up to $\Theta\left((I^{(q)})^2\right)$ steps. We make sure that if session i is in $A^{(q)}$ then $1/r_i + d_i \geq (I^{(q)})^2$, and therefore the delays inserted can be tolerated. The conversion step converts some sessions from fractional to integral, while maintaining the frame size of $I^{(q+1)}$ and slightly increasing the relative congestion to $c^{(q+1)} = (1 + o(1))^2 c^{(q)}$. These newly-converted sessions form a set $B^{(q+1)}$ and have associated values $1/r_i + d_i \geq (I^{(q+1)})^2$. This bound is chosen so that the sessions in $A^{(q+1)}$, which is $A^{(q)} \cup B^{(q+1)}$, will be able to tolerate the delays inserted during the next iteration of frame-refinement. During the conversion step we delay the packets in $B^{(q+1)}$ by up to $\Theta(1/r_i + d_i)$ steps. We shall show the existence of "good" delays for both frame-refinement and conversion steps. The following table summarizes our approach.

Schedule	Integral sessions	Frame size	Relative congestion
$\mathcal{S}^{(q)}$	$A^{(q)}$	$I^{(q)}$	$c^{(q)}$
Refinement	$A^{(q)}$	$\log^5 I^{(q)}$	$(1+o(1))c^{(q)}$
Conversion	$A^{(q)} \cup B^{(q+1)}$	$\log^5 I^{(q)}$	$(1+o(1))^2 c^{(q)}$
$\mathcal{S}^{(q+1)}$	$A^{(q+1)}$	$I^{(q+1)}$	$c^{(q+1)}$

In Section 4.3.4 we show that at the termination of our algorithm we have a schedule $\mathcal{S}^{(\zeta)}$ in which every session is integral and the relative congestion is at most 1, for all frames of size larger than a certain constant. In $\mathcal{S}^{(\zeta)}$ all session-i arrivals during $[\mathcal{T} - \mathcal{T}_i, 2\mathcal{T} - \mathcal{T}_i)$ are serviced during $[\mathcal{T}, 2\mathcal{T})$. Furthermore, all session-i packets reach their destination within $O(\mathcal{T}_i)$ steps of their injections.

Problem 3: Constant-factor stretching in the final schedule

As discussed above, we repeat the process of refinement and conversion until we have a schedule, $\mathcal{S}^{(\zeta)}$, in which all sessions are integral and in which the relative congestion is 1 for all frames of size larger than a certain constant w, defined at the end of Section 4.2. In the static problem, a final schedule can easily be obtained by stretching $\mathcal{S}^{(\zeta)}$ by a constant factor. However, we cannot afford to have a constant blowup in our final schedule for the dynamic problem. This is because we need to independently schedule all time intervals $[0, \mathcal{T})$, $[\mathcal{T}, 2\mathcal{T})$, etc, and a constant blowup would make these time intervals overlap.

To overcome this problem, we shall first devise a schedule for a new network $\mathcal{M}$ that is constructed from the original network $\mathcal{N}$ as follows. Each edge e of $\mathcal{N}$ is replaced by $2w$ consecutive edges $e_1, \ldots, e_{2w}$, where w is the constant presented above. The rates and routes of the sessions are unaffected. In $\mathcal{M}$, session i has length $D_i = 2wd_i = O(d_i)$.

All the techniques described earlier are applied to the network $\mathcal{M}$. We carry out successive conversion and refinement steps for $\mathcal{M}$ and obtain a schedule $\mathcal{S}^{(\zeta)}$, where the relative congestion is 1 for any frame whose size is larger than w. We then "smooth" $\mathcal{S}^{(\zeta)}$ and convert it to a schedule for $\mathcal{N}$ where only one packet at a time traverses any link.

The idea behind the smoothing process is as follows. In $\mathcal{S}^{(\zeta)}$, there may be more than 1 packet which requires some edge of $\mathcal{M}$ during a given time step. However, at most w packets will require any given edge in $\mathcal{M}$ within w time steps. This fact means we can shuffle in time the packets that require the edge, so that exactly one packet traverses the edge at any time step. Unfortunately, this shuffling in time can lead to an impossible schedule for $\mathcal{M}$, in that a packet is assigned to traverse its edges out of order (timewise). However, it turns out that if one only considers the schedule with respect to the packets traversing edge e_{2w}, for all e, then the schedule is completely legal, i.e. the packets cross *these* edges in order. The idea is then to schedule edge e in $\mathcal{N}$ in the same way that the corresponding edge e_{2w} is being scheduled in $\mathcal{M}$. We shall explain the smoothing process in detail in Section 4.4.

Figure 2 is a schematic picture of our overall approach.

The main theorem of this paper is stated below. The proof details are left to the Appendix.

Theorem 10 *Consider an arbitrary network in which sessions are defined. Each session i is associated with an injection rate r_i and path length d_i. Packets are injected to the network along these sessions subject to the injection rates. If the total rate on each edge is at most $1 - \varepsilon$ for a constant $\varepsilon \in (0, 1)$, then there exists a template-based schedule such that each session-i packet reaches its destination within $O(1/r_i + d_i)$ steps of its injection and at most one packet crosses an edge at each time step. This schedule also maintains constant edge queues.*

Acknowledgments

The authors wish to thank Bruce Maggs, Greg Plaxton and Salil Vadhan for many helpful comments.

References

[1] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proceedings of the 37th Annual*

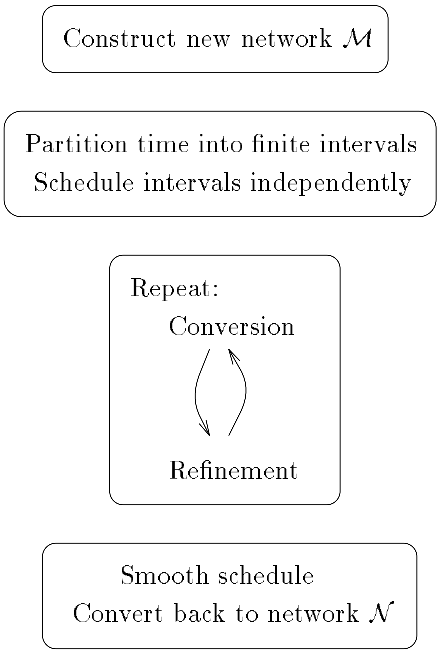

Figure 2: An overview of our approach for the dynamic routing problem.

Symposium on Foundations of Computer Science, pages 380 – 389, Burlington, VT, October 1996.

[2] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queueing theory. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 376 – 385, Philadelphia, PA, May 1996.

[3] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493 – 509, 1952.

[4] R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory*, pages 114 – 31, 1991.

[5] R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, pages 132 – 141, 1991.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Research and Experience*, 1:3 – 26, 1990.

[7] A. Elwalid, D. Mitra, and R. H. Wentworth. A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node. *IEEE Journal on selected areas in communications*, pages 1115 – 1127, 1995.

[8] 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 – 186, 1994.

[9] 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, Carnegie Mellon University, 1996.

[10] F. T. Leighton and G. Plaxton. Hypercubic sorting networks. *SIAM Journal of Computing (to appear)*, 1997.

[11] R. Ostrovsky and Y. Rabani. Local control packet switching algorithm. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (to appear)*, May 1997.

[12] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344 – 357, 1993.

[13] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137 – 150, 1994.

[14] Y. Rabani and E. Tardos. Distributed packet switching in arbitrary networks. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, Philadelphia, PA, May 1996.

[15] J. Spencer. *Ten Lectures on the Probabilistic Methods.* Capital City Press, Philadelphia, Pennsylvania, 1994.

[16] J. S. Turner. New directions in communications, or which way to the information age. *IEEE Communications Magazine*, pages 8 – 15, 1986.

4 Appendix

The analysis culminating in Theorem 10 is provided below in full detail.

4.1 Lemmas for Probabilistic Analysis

Throughout the probabilistic analysis of the proof we use the Lovász Local Lemma [15, pages 57-58] and a Chernoff Bound [3]. We include them here for easy reference.

[Lovász Local Lemma] *Let $E_1, \ldots, E_m$ be a set of "bad events" each occurring with probability p and with dependence at most d (i.e. every bad event is mutually independent of some set of $m-d$ other bad events). If $4pd < 1$, then with probability greater than zero no bad event occurs.*

[Chernoff Bound] *Let X_i be n independent Bernoulli random variables with probability of success p_i. Let $Y = \sum_{i=1}^{n} X_i$ and let $\mu \geq \sum_{i=1}^{n} p_i$. Then for $0 < \delta < 1$, we have*

$$\Pr[\, Y > (1+\delta)\mu \,] \leq e^{-\delta^2 \mu/3}.$$

4.2 Parameter Definitions

Interval length $\mathcal{T}$ and $\mathcal{T}_i$ As discussed in Section 3.2 we independently schedule intervals $[0, \mathcal{T})$, $[\mathcal{T}, 2\mathcal{T})$, etc. The templates will be of size $\mathcal{T}$. Our proof will concentrate on the interval $[\mathcal{T}, 2\mathcal{T})$. All the session-$i$ packets that arrive during $[\mathcal{T} - \mathcal{T}_i, 2\mathcal{T} - \mathcal{T}_i)$ are serviced during $[\mathcal{T}, 2\mathcal{T})$. We define $\mathcal{T}$ and $\mathcal{T}_i$ for session i as follows. Recall $D_i = 2wd_i$, where w is a constant defined at the end of this section.

$$\begin{aligned}
\mathcal{T}_i &= 4D_i + 2 + (24/\varepsilon + 2)/r_i, \\
\mathcal{T} &= \left\lceil \frac{(1 + 12/\varepsilon) \max_i \mathcal{T}_i}{w} \right\rceil w.
\end{aligned}$$

In other words, $\mathcal{T}$ is the smallest multiple of w that is greater than or equal to $(1 + 12/\varepsilon) \max_i \mathcal{T}_i$. Clearly $\mathcal{T}_i = O(1/r_i + D_i) = O(1/r_i + d_i)$.

Packet size for a fractional session In this section we define $\hat{r}_i$, the packet size for a fractional session i. For reasons that will become clear in the conversion step of Section 4.3.3, we need $\hat{r}_i$ to be slightly larger than r_i and we shall need to express $\hat{r}_i$ as the ratio of two integers. Let,

$$\begin{aligned}
\hat{r}_i &= s_i/\ell_i, \\
\ell_i &= \lceil 24/(\varepsilon r_i) \rceil, \\
s_i &= \lfloor \ell_i r_i (1 + \varepsilon/6) \rfloor,
\end{aligned}$$

where ε is such that the total injection rate on any edge is at most $1 - \varepsilon$. By the definitions of ℓ_i and s_i, one can verify that,

$$r_i(1 + \varepsilon/12) + 1/\ell_i \leq \hat{r}_i = s_i/\ell_i \leq r_i(1 + \varepsilon/6). \tag{1}$$

A fractional session-i injects a packet of size $\hat{r}_i = s_i/\ell_i$ at every step. Since $\varepsilon \in (0,1)$ and the total injection rate on any edge is at most $1 - \varepsilon$, one can verify that,

Lemma 11 *For all edges e, $\sum_{i \in E} \hat{r}_i < 1 - 5\varepsilon/6$ where E is the set of sessions that use edge e.*

Parameters for schedule $\mathcal{S}^{(q)}$ We shall show later that, in schedule $\mathcal{S}^{(q)}$, the relative congestion, due to all integral and fractional sessions, is at most $c^{(q)}$ for any frame of size $I^{(q)}$ or larger. For $\mathcal{S}^{(q)}$, the set $A^{(q)}$ consists of all the integral sessions. As we construct schedule $\mathcal{S}^{(q+1)}$ from $\mathcal{S}^{(q)}$, sessions in $B^{(q+1)}$ become integral and join $A^{(q)}$. The schedule at the end of the refinement and the conversion is $\mathcal{S}^{(\zeta)}$. The parameters $I^{(q)}$, $c^{(q)}$, $A^{(q)}$ and $B^{(q+1)}$ are defined by the following recurrences. Let $X_i = D_i + 1/r_i$ for session i, and let $X_{\max} = \max_i X_i$.

$$
\begin{aligned}
I^{(0)} &= e^{\log^{2/5} X_{\max}} \\
I^{(q+1)} &= \log^5 I^{(q)} \\
c^{(0)} &= 1 - 5\varepsilon/6 \\
c^{(q+1)} &= (1+\delta^{(q)})^2 c^{(q)} \\
\delta^{(q)} &= \beta/\sqrt{\log I^{(q)}} \\
A^{(0)} &= \emptyset \\
A^{(q+1)} &= A^{(q)} \cup B^{(q+1)} \\
B^{(q+1)} &= \left\{ i \notin A^{(q)} : \left(I^{(q+1)}\right)^2 \le X_i \le e^{\sqrt{I^{(q+1)}}} \right\} \qquad \text{for } q \ne \zeta - 1 \\
B^{(q+1)} &= \left\{ i \notin A^{(q)} : X_i \le e^{\sqrt{I^{(q+1)}}} \right\} \qquad \text{for } q = \zeta - 1
\end{aligned}
$$

The parameter β is a sufficiently large positive constant. Note that $I^{(q)}$ decreases polylogarithmically and $c^{(q)}$ increases by a factor of $1 + o(1)$. One can verify that $B^{(q)}$ forms a partition of all the sessions and that sessions with large X_i values become integral first. We make use of the bound $X_i \ge \left(I^{(q+1)}\right)^2$ in the frame-refinement step and we use the bound $\log^2 X_i \le I^{(q+1)}$ in the conversion step.

Definition of w We define a constant w that has two purposes. First, the process of refinement and conversion terminates when the frame size becomes smaller than or equal to w. Second, the intermediate network $\mathcal{M}$ is constructed from the original network $\mathcal{N}$ by replacing each edge in $\mathcal{N}$ with $2w$ edges. We define w to be a constant that satisfies the following two bounds:

1. $w \ge x$, where x satisfies $(1 - \frac{\alpha}{\sqrt{\log x}})^2 = 1 - 5\varepsilon/6$, i.e. $x = e^{\alpha^2(1-\sqrt{1-5\varepsilon/6})^{-2}}$;
2. $w \ge 2\log^{15} w + 2\log^{10} w - \log^5 w$.

The first bound ensures that the relative congestion $c^{(\zeta)}$ is at most 1. (See Lemma 23.) The second bound is to maintain an invariant throughout the frame-refinement steps. (See Section 4.3.2.)

4.3 A Schedule for the Intermediate Network $\mathcal{M}$

In this section we start with an initial schedule $\mathcal{S}^{(0)}$, defined in section 4.3.1, and describe the successive iterations needed to transform it to schedule $\mathcal{S}^{(\zeta)}$ (see Section 3.2, Problem 2), where $\mathcal{S}^{(\zeta)}$ satisfies Theorem 12 below. All our schedules are defined for network $\mathcal{M}$, as defined in Section 3.2, Problem 3.

Theorem 12 *Given network $\mathcal{M}$ and a set of sessions as defined in Section 1.2, there is a schedule $\mathcal{S}^{(\zeta)}$ such that the following hold.*

1. *The relative congestion is at most 1 for any frame of size larger than a certain constant;*

2. *After leaving its source, each packet waits at most once every $O(1)$ steps, which implies that all edge queues in $\mathcal{M}$ have size $O(1)$;*

3. *For all sessions i, any session-i packet reaches its destination within $O(1/r_i + D_i)$ steps of its injection;*

4. *All session-i arrivals during $[\mathcal{T} - \mathcal{T}_i, 2\mathcal{T} - \mathcal{T}_i)$ are serviced during $[\mathcal{T}, 2\mathcal{T})$, i.e. all packets leave their source no earlier than $\mathcal{T}$ and reach their destination before $2\mathcal{T}$.*

We first define or recall several basic concepts. Given some schedule $\mathcal{S}$, a *region* R of the schedule is some interval of contiguous time steps in the schedule. A *T-frame* is a region of length T. The *congestion* C in a T-frame is the maximum number of packets that cross any edge in that interval and the *relative congestion* is the ratio C/T. A fractional packet from session i always contributes exactly $\hat{r}_i$ to the relative congestion of any frame.

4.3.1 An Initial Schedule $\mathcal{S}^{(0)}$

In $\mathcal{S}^{(0)}$, all sessions are fractional, i.e. $A^{(0)} = \emptyset$. Each packet (of a fractional size) crosses one edge per time step whether or not other packets are using the same edge at the same time. Since the relative congestion is entirely due to fractional sessions, the relative congestion is at most $\sum \hat{r}_i < 1 - 5\varepsilon/6 = c^{(0)}$ on any edge e (see Lemma 11).

Note that the above relative congestion holds for any frame size. We choose the initial frame size $I^{(0)} = e^{\log^{2/5} X_{\max}}$, so that $I^{(1)} = \log^2 X_{\max}$, which implies $X_{\max} = e^{\sqrt{I^{(1)}}}$. This allows the sessions with the largest X_i value to be converted in the first iteration of the algorithm (see definition of $B^{(1)}$).

4.3.2 Frame Refinement for Schedule $\mathcal{S}^{(q)}$

In this section we describe the frame-refinement process. For each schedule, a frame refinement delays the packets from integral sessions in a way that dramatically reduces the frame size, but does not increase the relative congestion and the length of the schedule by much.

To be more precise, for schedule $\mathcal{S}^{(q)}$, we inductively assume that the relative congestion is at most $c^{(q)}$ for frames of size $I^{(q)}$ or larger and that each integral packet waits at most once every $I^{(q-1)}$ steps after leaving its source. In this frame refinement step we show that there is a way to delay (by an amount related to the frame size) the packets from $A^{(q)}$ so that, in the resulting schedule $\mathcal{S}^{(q+\frac{1}{2})}$, the relative congestion is at most $(1+\delta^{(q)})c^{(q)}$ for any frame of size $I^{(q+1)} = \log^5 I^{(q)}$ or larger, where $\delta^{(q)} = \beta/\sqrt{\log I^{(q)}}$, and each integral packet waits at most once every $I^{(q)}$ steps.

The base case of the initial schedule $\mathcal{S}^{(0)}$ is described in Section 4.3.1. Since there are no integral sessions, no delays are inserted in this step. Trivially, the resulting relative congestion is at most $(1 + \delta^{(0)})c^{(0)}$ for any frame of size $I^{(1)}$ or larger at the end of this step, and no packet ever waits.

Let us now consider refining schedule $\mathcal{S}^{(q)}$, for $q > 0$. The refinement is divided into two steps. In the first refinement step we divide the current schedule into blocks of length $2(I^{(q)})^3 + 2(I^{(q)})^2 - I^{(q)}$, and insert delays into each block so that its length increases to $2(I^{(q)})^3 + 2(I^{(q)})^2$. We show that these delays can be introduced in such a way that in the central $2(I^{(q)})^3$ steps of each block the relative congestion of frames of at least length $I^{(q+1)}$ is only a little larger than $c^{(q)}$. (See Figure 3.) At the beginning and end of each block there are "fuzzy" regions of length $(I^{(q)})^2$ each. In the second step we move the block boundaries so that the fuzzy regions at the end and beginning of adjacent blocks are at the center of the new blocks of $2(I^{(q)})^3 + 2(I^{(q)})^2$ steps. Again, we insert delays into each block increasing the size of the block by $(I^{(q)})^2$ steps. We show that there is a way to insert these delays so that the final conditions for refining $\mathcal{S}^{(q)}$ are indeed satisfied. (See Figure 4.)

In the following we present Lemma 14 that will be used extensively in both steps of the refinement. We continue by presenting both steps in detail.

A Useful Lemma

The following lemma is used to prove Lemma 14.

Lemma 13 *Let X and Y be independent random variables. Let Y be binomially distributed with mean μ_y, and let σ_1, σ_2, and v be values such that $\sigma_2 = \sigma_1 - 1/v$. Then,*

$$\Pr[\ X + \mu_y > (1+\sigma_1)v\ ] \leq 2\Pr[\ X + Y > (1+\sigma_2)v\ ].$$

Proof: Let $z = (1+\sigma_1)v - \mu_y$. We have,

$$\begin{aligned} \Pr[\ X + \mu_y > (1+\sigma_1)v\ ] &= \Pr[\ X > z\ ], & (2) \\ \Pr[\ X + Y > (1+\sigma_2)v\ ] &= \Pr[\ X + Y > \mu_y + z - 1\ ]. & (3) \end{aligned}$$

Note also that,

$$\begin{aligned} \Pr[\ X + Y > \mu_y + z - 1\ ] &\geq \Pr[\ X > \mu_y + z - 1 - \lfloor \mu_y \rfloor \text{ and } Y \geq \lfloor \mu_y \rfloor\ ] \\ &= \Pr[\ X > z - 1 + \mu_y - \lfloor \mu_y \rfloor\ ]\Pr[\ Y \geq \lfloor \mu_y \rfloor\ ]. \end{aligned}$$

This last equality follows from the independence of X and Y. Theorem B.1 in [10] shows that $\Pr[\ Y \geq \lfloor \mu_y \rfloor\ ] \geq 1/2$. Since $\mu_y - \lfloor \mu_y \rfloor < 1$, we have,

$$\Pr[\ X + Y > \mu_y + z - 1\ ] \geq \frac{1}{2}\Pr[\ X > z\ ].$$

Our Lemma follows from equalities (2) and (3) and the above inequality. □

We say that a packet is *active* during some region of a schedule if the packet belongs to some integral session and it traverses at least one edge during the region. Since we maintain the invariant that a packet waits at most once every $I^{(q-1)}$ steps after leaving its source, an *inactive* packet is either at its source or its destination during the entire region. Lemma 14 below is a stepping stone that allows us to reduce the frame size from $I^{(q)}$ to $poly \log I^{(q)}$. We invoke this lemma for various values of s, t, r and R.

Lemma 14 *Consider some region R of a schedule where the relative congestion is at most $r = \Theta(1)$ for frames of length s or more, where $\log^3 I^{(q)} \leq s \leq (I^{(q)})^2$. Consider any edge e and any t-frame, where $\log^2 I^{(q)} \leq t \leq 2\log^2 I^{(q)}$. Assume each active packet in the region is delayed between the beginning of R and the beginning of the t-frame by a number of steps randomly, independently, and uniformly chosen from $[1, s]$. Then, for any constant k there is some value $\gamma = \Theta(1)/\sqrt{\log I^{(q)}}$ such that the probability of having a relative congestion larger than $r(1+\gamma)$ on e during the t-frame is at most $(I^{(q)})^{-k}$.*

Proof: Let the random variable X be the frame congestion on e during the t-frame due to the active packets after they are delayed. If the relative congestion due to fractional sessions is r_f, the frame congestion due to fractional sessions in the t-frame is exactly $r_f t$. Since the active packets are the only integral-session packets that can cross e during the region, the frame congestion on e during the t-frame is $X + r_f t$ after the delay.

Consider now a binomial random variable Y with parameters $(r_f s, t/s)$ and mean $E[Y] = r_f t$. From Lemma 13, the probability p that the congestion in the t-frame is larger than $(1+\gamma)rt$ after the packets have been delayed is,

$$p = \Pr[\, X + r_f t > (1+\gamma)rt \,] \leq 2\Pr[\, X + Y > (1+\sigma)rt \,],$$

where $\sigma = \gamma - 1/rt$. Since $t \geq \log^2 I^{(q)}$ and $r = \Theta(1)$, then $\gamma = \Theta(1)/\sqrt{\log I^{(q)}}$ if and only if $\sigma = \Theta(1)/\sqrt{\log I^{(q)}}$. Let $\sigma = v/\sqrt{\log I^{(q)}}$, where v is a constant. We shall choose an appropriate value v so that the lemma is satisfied.

We first concentrate on X. Since the active packets are delayed up to s steps, an active packet that crosses e in the t-frame after the delay could cross e in an interval of $t+s$ steps before the delay. The relative congestion due to active packets is at most $r - r_f$ in that interval before the delay. Hence, at most $(t+s)(r-r_f)$ active packets can cross e in the t-frame after the delay, and each of them has probability of at most t/s of doing so.

Recall that Y is a binomial random variable with parameters $(r_f s, t/s)$. We define Z to be a binomial random variable with parameters $(n, t/s)$, where $n = r(t+s) > (r-r_f)(t+s) + r_f s$. It is easy to see that,

$$p \leq 2\Pr[\, X + Y > (1+\sigma)rt \,] \leq 2\Pr[\, Z > (1+\sigma)rt \,].$$

Therefore, we bound the probability p,

$$p \leq 2 \sum_{i=(1+\sigma)rt}^{r(t+s)} \binom{r(t+s)}{i} (t/s)^i (1-t/s)^{r(t+s)-i}.$$

We bound the sum by observing that $(1+\sigma)rt$ is larger than $E[Z] = (t+s)rt/s$, since $t/s \leq 2/\log I^{(q)}$. Thus, the first term of the sum is the largest. By applying the inequalities $(1+x) \leq e^x$, $\ln(1+x) \geq x - x^2/2$ for $0 \leq x \leq 1$, and $\binom{a}{b} \leq (ae/b)^b$ for $0 < b < a$, and the fact that there are at most $r(t+s)$ terms in the sum, we have

$$p \leq 2r(t+s)e^{-rt\sigma^2(1/2 - \sigma/2 - t/\sigma^2 s - 2t/\sigma s)}.$$

The bounds on s and t and the definitions of r and σ imply that we can choose a constant v large enough so that $p < (I^{(q)})^{-k}$ for any constant $k > 0$. □

The First Refinement Step for Schedule $\mathcal{S}^{(q)}$

We first divide the interval $[\mathcal{T}, \mathcal{T} + |\mathcal{S}^{(q)}|)$ into blocks of length $2(I^{(q)})^3 + 2(I^{(q)})^2 - I^{(q)}$. We shall reschedule each block B independently. During a block B we only delay active packets.

For each block B, each active packet in B is assigned a delay randomly, uniformly, and independently chosen from $[1, I^{(q)}]$. An active packet p, whose assigned delay is x, is delayed in the first $xI^{(q)}$ steps of B once every $I^{(q)}$ steps. In order to independently reschedule the next block, packet p is also delayed in the last $(I^{(q)} - x)I^{(q)}$ steps of B once every $I^{(q)}$ steps. Therefore, a rescheduled block has length $2(I^{(q)})^3 + 2(I^{(q)})^2$.

Before the delays are inserted to reschedule block B, an active packet p is delayed at most once within the block, provided that $2(I^{(q)})^3 + 2(I^{(q)})^2 - I^{(q)} < I^{(q-1)}$, which holds as long as $I^{(q)}$ is larger than some constant. Prior to inserting any new delay to a block, we check if it is within $I^{(q)}$ steps of the single old delay. If the new delay would be too close to the old delay, then it is simply not inserted. The loss of one delay in a block has a negligible effect on the probability analysis that follows.

Lemma 16 shows that with the insertion of delays we can dramatically reduce the frame size in the center of the block and increase the relative congestion only slightly. In order to prove Lemma 16, we need the following fact.

Lemma 15 *If the relative congestion in every frame of size T to $2T-1$ is at most r, then the relative congestion in any frame of size T or greater is at most r.*

Proof: Consider a frame of size T', where $T' > 2T-1$. The first $\lfloor T'/T \rfloor T - T$ steps of the frame can be broken into T-frames, each with relative congestion r. The remainder of the T'-frame consists of a single frame of size between T and $2T-1$ steps in which the relative congestion is also at most r. $\square$

Lemma 16 *There exists a way of choosing delays so that in between the first and last $(I^{(q)})^2$ steps of the block B, the relative congestion of any frame of size $\log^2 I^{(q)}$ or larger is at most $(1+\gamma_1)c^{(q)}$, for some $\gamma_1 = \Theta(1)/\sqrt{\log I^{(q)}}$.*

Proof: With each edge e, we associate a bad event. A bad event on e happens when the frame congestion on edge e is more than $(1+\gamma_1)c^{(q)}I$ during *any* I-frame of size $\log^2 I^{(q)}$ or larger. Due to Lemma 15, it is sufficient to prove the statement for all frames of size between $\log^2 I^{(q)}$ and $2\log^2 I^{(q)}$. We shall use the Lovász Local Lemma to show that the probability that no bad event occurs is nonzero.

We first bound the dependence, d, of bad events. Two bad events on two edges are dependent only if a packet from a session $i \in A^{(q)}$ can use both edges. At most $c^{(q)}(2(I^{(q)})^3 + 2(I^{(q)})^2 - I^{(q)})$ packets (from sessions in $A^{(q)}$) can cross the same edge in the block B, and each packet crosses at most $2(I^{(q)})^3 + 2(I^{(q)})^2 - I^{(q)}$ edges in B. As we shall show later, $c^{(q)} \leq 1$. Therefore, a bad event can be dependent on at most $O((I^{(q)})^6)$ other bad events.

We now bound the probability, p, that a bad event happens on e. Consider a particular I-frame, where $\log^2 I^{(q)} \leq I \leq 2\log^2 I^{(q)}$, that lies completely between the first and last $(I^{(q)})^2$ steps of B. By setting $R = B$, $r = c^{(q)}$, $s = I^{(q)}$ and $t = I$, we apply Lemma 14 to show that for any constant k_1 there is some value $\gamma_1 = \Theta(1)/\sqrt{\log I^{(q)}}$ such that the probability p_1 of a bad event happening on e in the I-frame is smaller than $(I^{(q)})^{-k_1}$.

Since there are $O((I^{(q)})^3 \log^2 I^{(q)})$ possible I-frames in B, the probability that a bad event happens on e during any I-frame is $p < p_1 O((I^{(q)})^3 \log^2 I^{(q)})$. We can set the value k_1 appropriately so that this probability is upper bounded by $O((I^{(q)})^{-7})$.

Therefore, we have $4pd < 1$ and our lemma follows from the Lovász Local Lemma. $\square$

At the end of the first refinement step, the center of each block has small relative congestion for small frame sizes. However there are regions of $(I^{(q)})^2$ steps at the beginning and end of each block that may have very large relative congestion. We call these "fuzzy" regions, and we deal with them in the second refinement step.

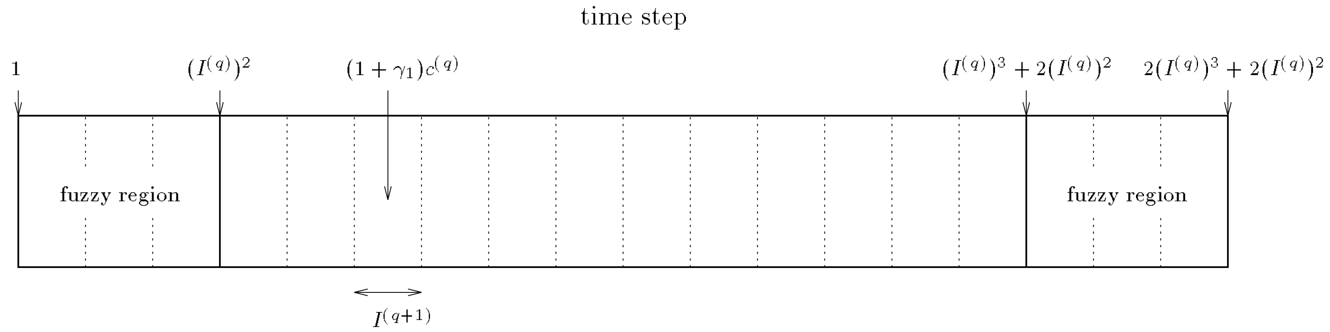

Figure 3: Situation after the first refinement step

The Second Refinement Step for Schedule $\mathcal{S}^{(q)}$

We start the second step of the refinement by relocating the block boundaries so that blocks still have $2(I^{(q)})^3 + 2(I^{(q)})^2$ steps, but now the fuzzy regions that were at the beginning and end of adjacent blocks are in the center of new blocks. Then, a new block has two "clean" regions of $(I^{(q)})^3$ steps each at the beginning and the end, and a fuzzy region of length $2(I^{(q)})^2$ steps in the center.

As in the first step of the refinement we now concentrate on individual blocks. We first show that the relative congestion is not very large for frames of size $(I^{(q)})^2$ or larger (even in the fuzzy region).

Lemma 17 *For any choice of delays in the first step of the refinement, the relative congestion in any frame of size* $(I^{(q)})^2$ *or larger is at most* $(1 + 2/I^{(q)})c^{(q)}$.

Proof: Without loss of generality we shall assume that all the sessions are integral. Consider an I-frame with I_1 steps before the center of the block and I_2 steps after the center ($I = I_1 + I_2$, and either I_1 or I_2 could be zero). A packet crosses some edge e in the I_1-frame only if it did so in some frame of length $I_1 + I^{(q)}$ before the delays where inserted. Therefore, at most $(I_1 + I^{(q)})c^{(q)}$ packets can cross edge e in the I_1-frame. Similarly, at most $(I_2 + I^{(q)})c^{(q)}$ packets can cross edge e in the I_2-frame. Therefore, the congestion in the I-frame can be at most $(I_1 + I_2 + 2I^{(q)})c^{(q)} = (I + 2I^{(q)})c^{(q)}$, and for $I \geq (I^{(q)})^2$ the relative congestion is at most $(1 + 2/I^{(q)})c^{(q)}$. □

Now, in order to reduce the frame size in the fuzzy region, we consider only the active packets in each block B, and assign a delay randomly, independently, and uniformly chosen from $[1, (I^{(q)})^2]$ to each active packet. A packet p with delay x waits once every $(I^{(q)})^3/x$ at the beginning of the block and once every $(I^{(q)})^3/((I^{(q)})^2 - x)$ at the end. As in the first step a delay is not inserted if it is going to be within $I^{(q)}$ steps of an existing delay for a moving packet.

The block length after the delay insertion is $2(I^{(q)})^3 + 3(I^{(q)})^2$, and the fuzzy region can be $(I^{(q)})^2$ steps longer, spanning steps $(I^{(q)})^3$ to $(I^{(q)})^3 + 3(I^{(q)})^2$.

The next lemma shows that there is some way of inserting delays so that the frame size in the fuzzy region is reduced, and the frame size and relative congestion in the rest of the block are increased by only a small amount.

Lemma 18 *In a block B, there exists a way of choosing delays so that in the fuzzy region (i.e. interval $[(I^{(q)})^3, (I^{(q)})^3 + 3(I^{(q)})^2]$) the relative congestion of any frame of size $\log^2 I^{(q)}$ or larger is at most $(1+\gamma_2)c^{(q)}$, for some $\gamma_2 = \Theta(1)/\sqrt{\log I^{(q)}}$, and so that in the intervals $[I^{(q)} \log^3 I^{(q)}, (I^{(q)})^3]$*

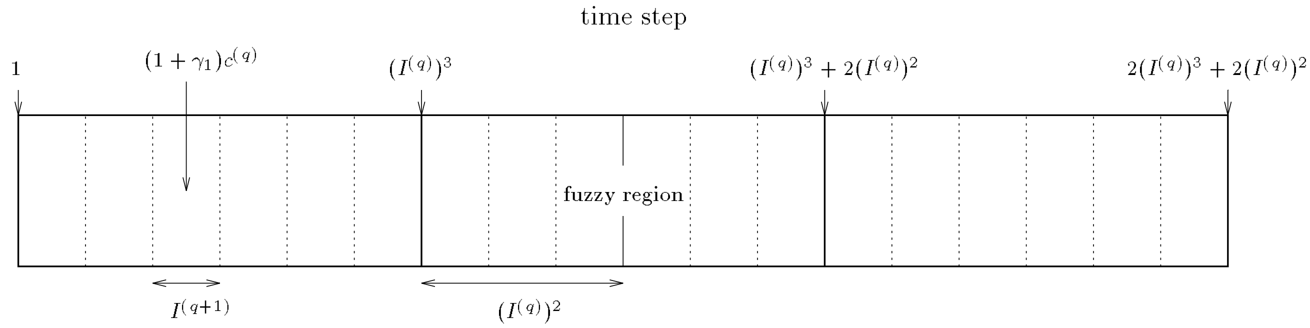

Figure 4: Situation after relocating block boundaries

and $[(I^{(q)})^3+3(I^{(q)})^2, 2(I^{(q)})^3+3(I^{(q)})^2 - I^{(q)}\log^3 I^{(q)}]$ *the congestion of any frame of size* $\log^2 I^{(q)}$ *or larger is at most* $(1+\gamma_3)c^{(q)}$*, for some* $\gamma_3 = \Theta(1)/\sqrt{\log I^{(q)}}$*.*

Proof: As in Lemma 16, we will use the Lovász Local Lemma to prove the claim. We associate a bad event with every edge e, so that a bad event happens on e if, for any $I \geq \log^2 I^{(q)}$,

- more than $(1+\gamma_2)c^{(q)}I$ packets cross e in any I-frame in $[(I^{(q)})^3, (I^{(q)})^3+3(I^{(q)})^2]$ (the fuzzy region), or
- more than $(1+\gamma_3)c^{(q)}I$ packets cross e in any I-frame in $[I^{(q)}\log^3 I^{(q)}, (I^{(q)})^3]$ or $[(I^{(q)})^3 + 3(I^{(q)})^2, 2(I^{(q)})^3+3(I^{(q)})^2 - I^{(q)}\log^3 I^{(q)}]$.

The dependency, d, of the bad events is bounded as in Lemma 16. Two bad events on two edges are dependent if packets from some session $i \in A^{(q)}$ can use both edges. At most $O((I^{(q)})^3)$ packets cross any edge in a block, and each of them can cross at most $O((I^{(q)})^3)$ other edges. Therefore, $d = O((I^{(q)})^6)$.

Now, to bound the probability p of a bad event happening on some edge e, we consider the three intervals separately and sum their respective probabilities. From Lemma 15 we only consider frames of length I such that $\log^2 I^{(q)} \leq I \leq 2\log^2 I^{(q)}$.

Take first some I-frame in $[(I^{(q)})^3, (I^{(q)})^3+3(I^{(q)})^2]$ (the fuzzy region). From Lemma 17 we know that the relative congestion for frames of size $(I^{(q)})^2$ or longer is at most $(1+2/I^{(q)})c^{(q)} = \Theta(1)$. Then, by choosing $R = B$, $r = (1+2/I^{(q)})c^{(q)}$, $s = (I^{(q)})^2$, and $t = I$, we can use Lemma 14 to show that, for any constant k_2, there is some $\sigma_2 = \Theta(1)/\sqrt{\log I^{(q)}}$ such that the probability of having relative congestion on e in the I-frame larger than $c^{(q)}(1+2/I^{(q)})(1+\sigma_2) = c^{(q)}(1+\gamma_2)$ is smaller than $(I^{(q)})^{-k_2}$. Note that $\gamma_2 = \Theta(1)/\sqrt{\log I^{(q)}}$.

Take now some I-frame in $[I^{(q)}\log^3 I^{(q)}, (I^{(q)})^3]$, which starts at step j. Given the way delays are inserted, by the jth step an active packet with delay x has been delayed $jx/(I^{(q)})^3$ steps. Thus, the delay of an active packet at the jth step is essentially a random value uniformly chosen from $[1, j/I^{(q)}]$. For $j \geq I^{(q)}\log^3 I^{(q)}$ the value $j/I^{(q)} \geq \log^3 I^{(q)}$.

Note that before inserting delays, from Lemma 16 the relative congestion in any frame of length $\log^2 I^{(q)}$ or larger in the interval $[1, (I^{(q)})^3]$ was at most $(1+\gamma_1)c^{(q)}$. Then, we can make $R = [1, (I^{(q)})^3]$, $r = (1+\gamma_1)c^{(q)}$, $s = \log^3 I^{(q)}$, and $t = I$, and use Lemma 14 to show, for any constant k_3, the existence of some $\sigma_3 = \Theta(1)/\sqrt{\log I^{(q)}}$ such that the probability of having relative congestion larger than $(1+\sigma_3)(1+\gamma_1)c^{(q)} = (1+\gamma_3)c^{(q)}$ on e in the I-frame is smaller than $(I^{(q)})^{-k_3}$. Again, $\gamma_3 = \Theta(1)/\sqrt{\log I^{(q)}}$.

By symmetry, the same value γ_3 makes the probability of a bad event happening on e in some I-frame in $[(I^{(q)})^3 + 3(I^{(q)})^2, 2(I^{(q)})^3 + 3(I^{(q)})^2 - I^{(q)} \log^3 I^{(q)}]$ smaller than $(I^{(q)})^{-k_3}$.

There are $O((I^{(q)})^3 \log^2 I^{(q)})$ possible I-frames as described in total. Hence, we can choose values for k_2 and k_3 such that the probability of a bad event is bounded as $p \leq (p_1+2p_2)O((I^{(q)})^3 \log I^{(q)}) < O((I^{(q)})^7)$. Therefore, we can guarantee $4pd < 1$ and invoke the Lovász Local Lemma to prove the claim. □

Finally, we bound the frame size and the relative congestion in the remaining intervals of the block in the following lemma.

Lemma 19 *The relative congestion in any frame of size* $\log^4 I^{(q)}$ *or larger in the intervals* $[1, I^{(q)} \log^3 I^{(q)}]$ *and* $[2(I^{(q)})^3 + 3(I^{(q)})^2 - I^{(q)} \log^3 I^{(q)}, 2(I^{(q)})^3 + 3(I^{(q)})^2]$ *is at most* $(1+\gamma_1)(1 + 1/\log I^{(q)})c^{(q)} = (1+\gamma_4)c^{(q)}$.

Proof: Let us first consider some I-frame in $[1, I^{(q)} \log^3 I^{(q)}]$. Recall that, before inserting delays, the relative congestion for frames of size $\log^2 I^{(q)}$ or more was at most $(1+\gamma_1)c^{(q)}$. In the interval no packet is delayed more than $\log^3 I^{(q)}$ steps. Therefore, the packets crossing some edge e in the I-frame could have crossed e in some interval of at most $I + \log^3 I^{(q)}$ steps, and the congestion in the I-frame can be of at most $(I + \log^3 I^{(q)})(1+\gamma_1)c^{(q)}$. For $I \geq \log^4 I^{(q)}$ the claim follows. The proof for interval $[2(I^{(q)})^3 + 3(I^{(q)})^2 - I^{(q)} \log^3 I^{(q)}, 2(I^{(q)})^3 + 3(I^{(q)})^2]$ is similar. □

From the above two lemmas we have that any frame of length at least $\log^4 I^{(q)}$ in each of the different intervals has at most relative congestion $(1+\gamma)c^{(q)}$, where $\gamma = \max(\gamma_2, \gamma_3, \gamma_4)$ and $\gamma = O(1)/\sqrt{\log I^{(q)}}$. We need to be careful now with the relative congestion in frames that overlap several intervals or several blocks. We can safely say that for any frame of size $I^{(q+1)} = \log^5 I^{(q)}$ or larger in the schedule $\mathcal{S}^{(q+\frac{1}{2})}$ obtained after the frame refinement, the relative congestion is at most $(1+\delta^{(q)})c^{(q)}$, for some $\delta^{(q)} = \beta/\sqrt{\log I^{(q)}}$ large enough.

4.3.3 Conversion for Schedule $\mathcal{S}^{(q)}$

In the conversion process we transform the schedule $\mathcal{S}^{(q+\frac{1}{2})}$, obtained from the frame-refinement step, into a new schedule $\mathcal{S}^{(q+1)}$. In this new schedule, all the sessions in $B^{(q+1)}$, which were fractional in $\mathcal{S}'^{(q)}$, have been made integral, and the relative congestion of frames of size $I^{(q+1)}$ or larger is at most $c^{(q+1)} = (1+\delta^{(q)})^2 c^{(q)}$.

At the beginning of this step, we inductively assume that the relative congestion is at most $(1+\delta^{(q)})c^{(q)}$ for any frame of size $I^{(q+1)}$ or larger, where $\delta^{(q)} = \beta/\sqrt{\log I^{(q)}}$. If the set $B^{(q+1)}$ is empty then we skip this conversion step; clearly, the relative congestion is at most $c^{(q+1)}$ for any frame of size $I^{(q+1)}$, and we are done.

On the other hand, if the set $B^{(q+1)}$ is not empty then for each session $i \in B^{(q+1)}$ we apply the following two processes. (a) In the discretization process we convert the schedule for fractional session-i packets into a schedule for integral packets in which no packet has to wait too much before it starts moving. (b) In the delay-insertion process we delay the time at which packets start moving (i.e. we insert *initial delays*) in such a way that the relative-congestion requirements are satisfied.

Discretization

We first show how to transform a fractional session in $B^{(q+1)}$ into an integral session. Consider a session i in $B^{(q+1)}$. When session i is fractional, a packet of size $\hat{r}_i = s_i/\ell_i$ is injected at every time step, where ℓ_i and s_i are integer constants defined in Section 4.2. A fractional packet marches to its destination one edge at a time with no delay.

We want to replace these fractional packets by integral packets. An integral packet waits at its source until it finds an unused *initial token*. Then, it crosses one edge every time step until it reaches its destination. The number of initial tokens and their distribution have to be carefully chosen so that no packet waits at its source for too long.

To transform session i, we consider the two intervals shown in Figure 5, $U = [\mathcal{T} - \mathcal{T}_i, 2\mathcal{T} - \mathcal{T}_i)$ and $V = [\mathcal{T}, 2\mathcal{T} - \mathcal{T}_i)$. When session i is converted, we distribute enough *initial tokens* in the interval V to accommodate all the session-i arrivals during U. Integral packets arrive at a rate r_i during U and initial tokens will appear at a rate roughly equal to $\hat{r}_i$ during V. Recall from Section 4.2, that $\hat{r}_i$ is slightly larger than r_i. By choosing the interval U long enough (i.e. $\mathcal{T}$ large enough), we guarantee that there are more initial tokens than arrivals.

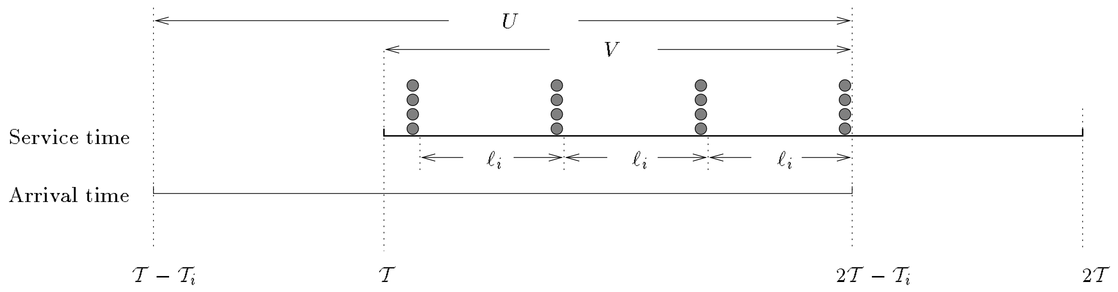

Figure 5: Session-i packets that are injected in interval U are assigned initial tokens in interval V. The interval V is divided into consecutive intervals of length ℓ_i, each of which has s_i initial tokens. The initial tokens are shown in solid dots.

Let $|V| = \mathcal{T} - \mathcal{T}_i$ be the length of interval V. We divide V into consecutive intervals of length ℓ_i (starting from the end), and we put s_i initial tokens in the last slot of each ℓ_i-interval. Note that if $|V|$ is not an integer multiple of ℓ_i, then the first ℓ_i-interval is "incomplete". (See Figure 5.) We show that there are enough initial tokens, and that no packet waits too long for an unused one.

Lemma 20 *For a converted session $i \in B^{(q+1)}$, every session-i packet that is injected during U finds an unused initial token in V within $\mathcal{T}_i + \ell_i = O(1/r_i + D_i)$ steps of its injection.*

Proof: Let $x = \mathcal{T}/(\mathcal{T} - \mathcal{T}_i)$ be the ratio of the length of interval U to the length of interval V. It suffices to show that s_i, the number of initial tokens in an ℓ_i-interval (shown in Figure 5), is as large as the number of session-i arrivals during an interval of length $x\ell_i$. At most $n = x\ell_i r_i + 1$ packets can arrive during $x\ell_i$ steps. Since $\mathcal{T} = (1 + 12/\varepsilon) \max_i \mathcal{T}_i$ by definition, we have $x \leq 1 + \varepsilon/12$ and $n \leq \ell_i r_i(1 + \varepsilon/12) + 1$. By the left-hand side of Inequality 1 in Section 4.2, we have $n \leq s_i$. Therefore, we have enough initial tokens. Since the initial tokens are at the end of an ℓ_i-interval, each packet can use an initial token that appears after the packet arrival time. It is also easy to verify that an unused initial token appears within $\mathcal{T}_i + \ell_i = O(\mathcal{T}_i)$ steps of the packet injection. $\square$

Delay Insertion

Before any delay is inserted to a packet from session $i \in B^{(q+1)}$, the packet leaves its source at the time of its initial token and marches to its destination with no more waiting. Now we insert an initial delay for each session-i packet, which has the effect of deferring the start time of the packet. We choose the delays uniformly from $[1, \ell_i]$. After the initial delay each packet travels to its destination without further delay.

Lemma 21 *Consider a particular edge e and a particular t-frame during interval $[\mathcal{T}, 2\mathcal{T})$. Suppose session i requires edge e, then the expected number of session-i packets that use e in the t-frame is at most $ts_i/\ell_i = t\hat{r}_i$.*

Proof: Let us assume first that delays have not been inserted yet. Due to the way initial tokens are distributed, session-i packets cross edge e in a very synchronous manner: a batch of at most s_i packets crosses every ℓ_i steps. Since we want an upper bound on the expectation, we assume that exactly s_i packets cross e every ℓ_i steps.

Let us now partition time in ℓ_i-intervals, so that each interval ends with a step in which packets cross e (i.e. all packets cross e in the last step of the intervals). Observe that, once delayed, all the packets that crossed e in the last step of some ℓ_i-interval will cross it in the following interval. Then, the total number of packets crossing e in an ℓ_i-interval after the delay insertion is exactly s_i. Also, after the insertion of delays, the expected number of packets crossing e in some subinterval of length ℓ of an ℓ_i-interval is exactly $\ell s_i/\ell_i$.

Take now the t-frame, and consider the incomplete ℓ_i-intervals it contains. There can be at most one at the beginning and one at the end. Assume they have lengths t_1 and t_2, respectively. From the above observations, the expected number of packets crossing e in the t_1 (resp. t_2) subinterval is $t_1 s_i/\ell_i$ (resp. $t_2 s_i/\ell_i$). In the remainder of the t-frame the number of packets crossing is exactly $(t - t_1 - t_2)s_i/\ell_i$. Hence, the expected number of packets crossing e in the t-frame is $(t - t_1 - t_2)s_i/\ell_i + t_1 s_i/\ell_i + t_2 s_i/\ell_i = ts_i/\ell_i$. □

We now use a Chernoff bound and the Lovász Local Lemma to show the following.

Lemma 22 *There exists a way of choosing the initial delays for sessions in $B^{(q+1)}$ such that the relative congestion in any frame of size $I^{(q+1)}$ or bigger is at most $c^{(q+1)}$ after the delays are inserted.*

Proof: Due to Lemma 15, it is sufficient to prove the result for all frames of size $I^{(q+1)}$ to $2I^{(q+1)}$. We associate a bad event with each edge e and each I-frame, where $I^{(q+1)} \le I \le 2I^{(q+1)}$. A bad event $E_{\{e,I\}}$ happens when more than $Ic^{(q+1)}$ packets use e during frame I. We use the Lovász Local Lemma to show that with nonzero probability no bad event occurs. Let $D_{\max} = \max_{i \in B^{(q+1)}} D_i$, $r_{\min} = \min_{i \in B^{(q+1)}} r_i$, $X = \max_{i \in B^{(q+1)}} D_i + 1/r_i$, and $\ell_{\max} = \max_{i \in B^{(q+1)}} l_i$.

We first bound the dependency d of bad events. Note that the probability space is given by the delays assigned to packets from sessions in $B^{(q+1)}$. Hence, a bad event $E_{\{e,I\}}$ is dependent on another bad event $E_{\{e',I'\}}$ only if a packet p from a session $i \in B^{(q+1)}$ such that there is a nonzero probability that p uses e during the I-frame and there is a nonzero probability that p uses e' during the I'-frame.

There are at most $1/r_{\min}$ sessions in $B^{(q+1)}$, each of which is at most $D_{\max}$ long. Therefore, $E_{\{e,I\}}$ depends on $E_{\{e',I'\}}$ for at most $D_{\max}/r_{\min} = O(X^2)$ choices of e'. Furthermore, intervals I and I' cannot be more than $D_{\max} + \ell_{\max}$ steps apart. (Otherwise any session-i packet either has probability 0 of crossing edge e during I or probability 0 of crossing e' during I'.) Therefore, the starting point of I' is limited to $2D_{\max} + 2\ell_{\max} + 4I^{(q+1)}$ locations, and the total possible choices for I' is at most $(2D_{\max} + 2\ell_{\max} + 4I^{(q+1)})I^{(q+1)} = O(X(I^{(q+1)})^2)$. We conclude that the dependency d is of $O(X^3(I^{(q+1)})^2)$

We now bound the probability p that a bad event $E_{\{e,I\}}$ happens. By our inductive assumption, the frame congestion on edge e during the I-frame is at most $(1 + \delta^{(q)})c^{(q)}I$ before the conversion.

Let S be the set of sessions in $B^{(q+1)}$ that use edge e. When sessions in $B^{(q+1)}$ are fractional, they contribute exactly $I\sum_{i\in S}\hat{r}_i$ to the frame congestion. Lemma 21 implies that the expected frame congestion due to the sessions in $B^{(q+1)}$ is at most $I\sum_{i\in S}\hat{r}_i$ after the initial delays are inserted. The congestion due to sessions not in $B^{(q+1)}$ does not change during the conversion. Hence, the expected frame congestion on edge e during the I-frame is at most $(1+\delta^{(q)})c^{(q)}I=\mu$. We bound the probability of $E_{\{e,I\}}$ as follows.

$$\begin{aligned} p &= \Pr\Big[\text{ Frame congestion on } e \text{ in } I > c^{(q+1)}I\Big] \\ &= \Pr\Big[\text{ Frame congestion on } e \text{ in } I > (1+\delta^{(q)})\mu\Big] \\ &\leq e^{-(\delta^{(q)})^2\mu/3} \\ &\leq e^{-(1-\varepsilon)\beta^2 I^{(q+1)}/(3\log I^{(q)})} \\ &\leq e^{-(1-\varepsilon)\frac{\beta^2}{3}(I^{(q+1)})^{1/5}(I^{(q+1)})^{3/5}} \\ &\leq e^{-(1-\varepsilon)\frac{\beta^2}{3}(I^{(q+1)})^{1/5}\log^{6/5}X} \end{aligned}$$

The first inequality follows from a Chernoff bound. The second inequality holds since $\mu > (1-\varepsilon)I \geq (1-\varepsilon)I^{(q+1)}$ and from the definition of $\delta^{(q)}$. The third inequality follows from the recurrence for $I^{(q+1)}$. The last inequality follows from the fact that $\log^2 X \leq I^{(q+1)}$. (This explains the need for $\log^2 X_i \leq I^{(q+1)}$ in the definition of $B^{(q+1)}$.)

When β is a sufficiently large constant, we have $4dp < 1$. Hence, the Lovász Local Lemma implies that with nonzero probability no bad events occur. That is, there exists a way to choose the initial delays for sessions in $B^{(q+1)}$ such that for all frames of size $I^{(q+1)}$ or larger the relative congestion is at most $c^{(q+1)}$.

Note that in the proof of this lemma we associate a bad event with each edge e and each interval I. Why couldn't we associate a bad event with each edge e only and then use a union bound on the number of intervals, as in Lemma 16? This is because we are considering all the session-i packets during an interval of length $\mathcal{T}$, which can be much bigger than $1/r_i + D_i$ for some sessions i. □

4.3.4 The Termination of the Algorithm

The succession of refinement and conversion terminates at schedule $\mathcal{S}^{(\zeta)}$ when the frame size $I^{(\zeta)}$ becomes smaller than or equal to w, a constant defined in Section 4.2. The following lemma shows that the relative congestion of $\mathcal{S}^{(\zeta)}$ is small.

Lemma 23 *In the schedule $\mathcal{S}^{(\zeta)}$ all sessions are integral and the relative congestion is at most $c^{(\zeta)} < 1$ for any frame of size $I^{(\zeta)}$ or larger.*

Proof: One can verify that $B^{(q+1)}$ forms a partition of all the sessions. Therefore, all the sessions are integral in the schedule $\mathcal{S}^{(\zeta)}$. By our induction, the relative congestion is at most $c^{(\zeta)}$ for all frames of size $I^{(\zeta)}$ or larger. Hence, we only need to show that $c^{(\zeta)} < 1$.

Due to the termination conditions, $x \leq I^{(\zeta-1)}$, where x is defined above. Let $\Delta = \beta/\sqrt{\log x}$, and observe that $\delta^{(\zeta-1)} \leq \Delta < 1$. By the recursive definition of $c^{(\zeta)}$, we have,

$$\begin{aligned} c^{(\zeta)} &= (1+\delta^{(\zeta-1)})^2(1+\delta^{(\zeta-2)})^2\cdots(1+\delta^{(0)})^2c^{(0)} \\ &< (1+\Delta)^2(1+\Delta^2)^2(1+\Delta^4)^2(1+\Delta^8)^2\ldots c^{(0)} \\ &\leq (1-\Delta)^{-2}\Big\{(1-\Delta)^2(1+\Delta)^2(1+\Delta^2)^2(1+\Delta^4)^2(1+\Delta^8)^2\ldots\Big\}c^{(0)} \end{aligned}$$

$$\begin{aligned}
&\leq (1-\Delta)^{-2} c^{(0)} \\
&= \left(1-\beta/\sqrt{\log x}\right)^{-2} c^{(0)} \\
&= \frac{1-5\varepsilon/6}{1-5\varepsilon/6} \\
&= 1
\end{aligned}$$

The first inequality holds since $\delta^{(q)} < (\delta^{(q+1)})^2$, for all q, by the recurrence defined in Section 4.2. The third inequality holds since $\Delta < 1$, and therefore the "telescope product" in the braces is less than 1. The last equality holds by the above choice of x and the definition of $c^{(0)}$ in Section 4.2. $\square$

Now, we have to make sure that in the resulting schedule $\mathcal{S}^{(\zeta)}$ no packet waits too much. The conversion step guarantees that when a session i becomes integral, no packet waits more than $O(D_i+1/r_i)$ steps before it starts moving, and it does not wait anymore. The last frame-refinement step also guarantees that a moving packet never waits more than once every $I^{(\zeta-1)}$ steps. However, all the frame-refinement steps that an integral packet has to go through can, in fact, delay the time it starts moving. The following lemma shows that this delay does not add up to a large amount, and therefore that a session-i packet reaches its destination in at most $O(D_i + 1/r_i)$ steps in the schedule $\mathcal{S}^{(\zeta)}$.

Lemma 24 *During frame-refinement a session-i packet is delayed by at most $2(D_i + 1/r_i)$ steps before it starts moving.*

Proof: Suppose session i first becomes integral in schedule $\mathcal{S}^{(q')}$. Consider a session-i packet p. For schedule $\mathcal{S}^{(q)}$, where $q \leq q'-1$, p is never delayed during frame-refinement. For schedule $\mathcal{S}^{(q)}$, where $q \geq q'$, p is delayed by at most $I^{(q)} + (I^{(q)})^2$ steps before it starts moving. Therefore, the total delay inserted during all the frame-refinement steps is at most $\sum_{q\geq q'} I^{(q)} + (I^{(q)})^2$. Since session i becomes integral for schedule $\mathcal{S}^{(q')}$, we must have $i \in B^{(q')}$. By the definition of $B^{(q')}$, $D_i + 1/r_i \geq (I^{(q')})^2$. Since $I^{(q)}$ decreases polylogarithmically, a session-i packet is delayed during frame-refinement by at most $2(D_i + 1/r_i)$ steps before it starts moving. $\square$

We proceed to prove that $\mathcal{S}^{(\zeta)}$ has all the properties stated in Theorem 12.

Proof of Theorem 12:

1. By Lemma 23, the relative congestion is at most 1 for any frame of size $I^{(\zeta)}$ or larger. Due to the termination conditions $I^{(\zeta)}$ is a constant.

2. By the invariant maintained throughout the frame-refinement steps, a packet waits at most once every $I^{(\zeta-1)}$ steps once it leaves its source. In addition, by Property 1 above, at most $I^{(\zeta)}$ packets cross an edge during any time step. Therefore, the edge queues have size at $2I^{(\zeta)}$.

3. We first show that a session-i packet reaches its destination within $\mathcal{T}_i$ steps after it obtains an initial token. After the initial token, a session-i packet is deferred by an initial delay during the conversion step and other delays during the frame-refinement step before it could leave its source. The initial delay is at most $\ell_i < 1+24/(\varepsilon r_i)$, and the delay during the refinement is at most $2(D_i+1/r_i)$ by Lemma 24. Once the packet starts moving, it reaches its destination in at most $2D_i$ steps by Property 2. Therefore, a session-i packet reaches its destination within $4D_i + 1 + (24/\varepsilon + 2)/r_i < \mathcal{T}_i$ steps after obtaining its initial token.

Since any session-i packet obtains an initial token within $\mathcal{T}_i + \ell_i$ steps of its injection by Lemma 20, the packet reaches its destination within $2\mathcal{T}_i + \ell_i = O(1/r_i + D_i)$ steps of its injection.

4. For all session-i arrivals during $[\mathcal{T} - \mathcal{T}_i, 2\mathcal{T} - \mathcal{T}_i)$, the initial tokens are in $[\mathcal{T}, 2\mathcal{T} - \mathcal{T}_i)$. From the discussion of Property 3, a session-i packet reaches its destination within $\mathcal{T}_i$ steps after it obtains an initial token. Therefore, all packets leave their sources no earlier than $\mathcal{T}$ and reach their destinations before $2\mathcal{T}$.

□

4.4 The Final Schedule for the Original Network $\mathcal{N}$

We now describe how to create a schedule $\mathcal{S}_{\mathcal{N}}$ for network $\mathcal{N}$ from $\mathcal{S}^{(\zeta)}$. In $\mathcal{S}_{\mathcal{N}}$ at most one packet at a time crosses each edge in $\mathcal{N}$. Recall that in the construction of $\mathcal{M}$ from $\mathcal{N}$, each edge e in $\mathcal{N}$ is replaced by $2w$ consecutive edges $e_1, \ldots, e_{2w}$, where w is a constant defined in Section 4.2.

We first partition the time interval $[\mathcal{T}, 2\mathcal{T})$ into consecutive intervals of length w (recall that by definition $\mathcal{T}$ is a multiple of w). For each w-interval and each edge f in $\mathcal{M}$, as many as w packets, $p_1, p_2, \ldots, p_w$, can cross f during the w-interval by schedule $\mathcal{S}^{(\zeta)}$. We *smooth out* $\mathcal{S}^{(\zeta)}$ so that p_j is the jth packet to cross f in the w-interval, where $p_1, \ldots, p_w$ represents an arbitrary ordering. After smoothing, a packet may not be scheduled to cross the edges on its route in order. For example, a packet may be scheduled to cross edge f before g, whereas f follows g on the route in $\mathcal{M}$. A packet may also be scheduled to leave its source before its injection time. However, $\mathcal{S}^{(\zeta)}$ after smoothing does have the property that at most one packet at a time crosses each edge. We define $\mathcal{S}_{\mathcal{N}}$ as follows. $\mathcal{S}_{\mathcal{N}}$ schedules a packet p to cross e in $\mathcal{N}$ at time t if and only if $\mathcal{S}^{(\zeta)}$ after smoothing schedules p to cross e_{2w} in $\mathcal{M}$ at time t.

Lemma 25 *In $\mathcal{S}_{\mathcal{N}}$, each packet is scheduled to leave its source after its injection and is scheduled to cross the edges on its route in order.*

Proof: We first show that each packet crosses the edges on its route in order. Consider a packet p. Let e and $\hat{e}$ be two edges on p's route in $\mathcal{N}$, where $\hat{e}$ follows e. Let t and $\hat{t}$ be the times that p crosses e and $\hat{e}$ in schedule $\mathcal{S}_{\mathcal{N}}$. We shall show that $t < \hat{t}$.

Let e_{2w} and $\hat{e}_{2w}$ be the edges in $\mathcal{M}$ that correspond to e and $\hat{e}$. Let τ and $\hat{\tau}$ be the times that p crosses e_{2w} and $\hat{e}_{2w}$ in the schedule $\mathcal{S}^{(\zeta)}$ *before* smoothing. Since p crosses the edges in $\mathcal{M}$ in order before smoothing, we have,

$$\tau + 2w \quad \leq \quad \hat{\tau}. \tag{4}$$

In schedule $\mathcal{S}_{\mathcal{N}}$, packet p crosses e at time t, which is shifted by at most $w-1$ steps from τ. Similarly, $\hat{t}$ is shifted by at most $w-1$ steps from $\hat{\tau}$. Hence we have,

$$\begin{array}{rcccl} \tau - (w-1) & \leq & t & \leq & \tau + (w-1), \\ \hat{\tau} - (w-1) & \leq & \hat{t} & \leq & \hat{\tau} + (w-1). \end{array}$$

From Inequality 4 and the above inequalities, we have $t < \hat{t}$. Therefore, p crosses the edges on its route in order.

The proof that packet p leaves its source after its injection time is similar. Suppose that p is injected to the network at time s. Let edge e be the first edge on the route of p in network $\mathcal{N}$, and let t be the time that p crosses e in $\mathcal{S}_{\mathcal{N}}$. Also let e_{2w} be the corresponding edge in $\mathcal{M}$, and let τ be the time that p crosses e_{2w} in $\mathcal{S}^{(\zeta)}$ before smoothing. Since in $\mathcal{S}^{(\zeta)}$ before smoothing p crosses the edges in order and leaves its source after its injection we have,

$$s + 2w \quad \leq \quad \tau.$$

In schedule $\mathcal{S}_{\mathcal{N}}$, packet p crosses e at time t, which is shifted by at most $w-1$ steps from τ. Hence we have,

$$\tau - (w-1) \quad \leq \quad t \quad \leq \quad \tau + (w-1).$$

Therefore, $s < t$ and packet p leaves its sources in $\mathcal{N}$ after the injection time. $\square$

We summarize the properties of $\mathcal{S}_{\mathcal{N}}$.

Theorem 26 *Schedule $\mathcal{S}_{\mathcal{N}}$ satisfies the following properties.*

1. *At most one packet at a time crosses each edge in $\mathcal{N}$;*
2. *After leaving its source, each packet waits constant number of steps to cross an edge, which implies all the edge queues in $\mathcal{N}$ have constant size;*
3. *For all sessions i, any session-i packet reaches its destination within $O(1/r_i + d_i)$ steps of its injection;*
4. *All session-i arrivals during $[\mathcal{T} - \mathcal{T}_i, 2\mathcal{T} - \mathcal{T}_i)$ are serviced during $[\mathcal{T}, 2\mathcal{T})$, i.e. all packets leave their source no earlier than $\mathcal{T}$ and reach their destination before $2\mathcal{T}$.*

Proof: The smoothing process guarantees Property 1. Properties 2 and 3 come from Properties 2 and 3 of $\mathcal{S}^{(\zeta)}$ given in Theorem 12, the construction of $\mathcal{M}$ from $\mathcal{N}$ and the fact that each packet is scheduled to reach its destination in $\mathcal{S}_{\mathcal{N}}$ at most w steps later than in $\mathcal{S}^{(\zeta)}$.

To see Property 4, recall that the interval $[\mathcal{T}, 2\mathcal{T})$ is partitioned into intervals of size w (with one interval possibly longer than w), and schedule $\mathcal{S}^{(\zeta)}$ is smoothed out within each w-interval. Therefore, if a packet is scheduled to cross an edge e during $[\mathcal{T}, 2\mathcal{T})$ according to $\mathcal{S}^{(\zeta)}$, the packet must also be scheduled to cross e during $[\mathcal{T}, 2\mathcal{T})$ according to $\mathcal{S}_{\mathcal{N}}$. Property 4 follows. Property 4 of the above theorem implies that all intervals of $[0, \mathcal{T})$, $[\mathcal{T}, 2\mathcal{T})$, etc. can be scheduled independently. $\square$

4.5 Making a Template-Based Schedule

In this section we show how to transform $\mathcal{S}_{\mathcal{N}}$ into a template-based schedule. This transformation makes use of Theorem 1, which we now restate and prove.

Theorem 1 *Consider any template-based schedule. If the end-to-end delay for each session-i token sequence is bounded by X, then each session-i packet reaches its destination within X steps after it obtains an initial token. If the token lag is bounded by x for* all *token sequences for* all *sessions, then the edge queue size is also bounded by x.*

We first state and prove a result on vectors and permutations. A vector $\vec{v} = [v_1, v_2, \ldots, v_n]$ is *sorted* if $v_1 \leq v_2 \leq \ldots \leq v_n$. We define $perm(\vec{v})$ to be a sorted vector whose components form a permutation of the components of $\vec{v}$. We also use the notation $\vec{u} < \vec{v}$ to indicate that the jth component of $\vec{u}$ is smaller than the jth component of $\vec{v}$ for each j.

Lemma 27 *Let $\vec{u} = [u_1, u_2, \ldots, u_n]$ and $\vec{v} = [v_1, v_2, \ldots, v_n]$ be two vectors, each of which consists of n distinct numbers. If $\vec{u} < \vec{v}$, then,* *1. $perm(\vec{u}) < perm(\vec{v})$;*

2. If $\vec{v} < \vec{u} + \vec{z}$, then $perm(\vec{v}) < perm(\vec{u}) + \vec{z}$, where $\vec{z} = [z, \ldots, z]$ is a vector of m z's for a scalar z.

3. Let $|\vec{v}|$ represent the maximum component of $\vec{v}$, then $|perm(\vec{v}) - perm(\vec{u})| \leq |\vec{v} - \vec{u}|$.

Proof: Without loss of generality, we assume $\vec{u}$ is sorted, i.e. $\vec{u} = perm(\vec{u})$. We also assume $perm(\vec{v}) = [v_{\sigma(1)}, \ldots, v_{\sigma(n)}]$, where σ represents the sorted permutation of $\vec{v}$.

1. Let us compare u_j and $v_{\sigma(j)}$. There are two cases to consider. If $j \leq \sigma(j)$, then $u_j \leq u_{\sigma(j)} < v_{\sigma(j)}$. These inequalities hold since $\vec{u}$ is sorted by assumption and $\vec{u} < \vec{v}$. If $j > \sigma(j)$, then there exists $j' \geq j$ such that $v_{j'} \leq v_{\sigma(j)}$. (Otherwise, for all $j' \geq j$, $v_{j'} > v_{\sigma(j)}$. However, only $n - j$ components of $\vec{v}$ can be greater than $v_{\sigma(j)}$.) Combining the fact that $\vec{u}$ is sorted and $\vec{u} < \vec{v}$, we have $u_j \leq u_{j'} < v_{j'} \leq v_{\sigma(j)}$. Therefore, $perm(\vec{u}) < perm(\vec{v})$ in both cases.

2. Since $perm(\vec{u} + \vec{z}) = perm(\vec{u}) + \vec{z}$ for $\vec{z} = [z, \ldots, z]$, Property 1 implies $perm(\vec{v}) < perm(\vec{u} + \vec{z}) = perm(\vec{u}) + \vec{z}$.

3. Suppose $|perm(\vec{v}) - perm(\vec{u})| = v_{\sigma(j)} - u_j$. There are two cases to consider. If $v_{\sigma(j)} \leq v_j$, then $v_{\sigma(j)} - u_j \leq v_j - u_j$, which implies $|perm(\vec{v}) - perm(\vec{u})| \leq |\vec{v} - \vec{u}|$. If $v_{\sigma(j)} > v_j$, then there exists $j' < j$ such that $v_{\sigma(j)} \leq v_{j'}$. (Otherwise, for all $j' \leq j$, $v_{\sigma(j)} > v_{j'}$. However, only $j - 1$ components of $\vec{v}$ can be smaller than $v_{\sigma(j)}$.) Since $u_{j'} < u_j$ by the assumption that $\vec{u}$ is sorted, we have $v_{\sigma(j)} - u_j \leq v_{j'} - u_{j'}$, which implies $|perm(\vec{v}) - perm(\vec{u})| \leq |\vec{v} - \vec{u}|$. Property 3 follows.

□

We are ready to transform a token-sequence-based bound into a packet-based bound. Although it might seem straight-forward, the difficulty is that, in a template-based schedule, a packet is unable to identify a token sequence. This means if a session-i token appears, then the session-i packet that has been waiting the longest has to move.

Proof of Theorem 1: It suffices to show the following. For any $y \geq 1$, consider the first y session-i packets injected. After obtaining its initial token, each of these y packets reaches the destination within X steps and it waits at most x steps to advance each edge.

Let T_{k1} be the time that the kth packet catches an initial token κ_k and advances its first edge. Let T_{kj} be the time that the kth packet would cross the jth edge if it followed the token sequence initiated at κ_k. Note that T_{kj} is not necessarily the time that the kth packet crosses the jth edge in a template-based schedule. However, T_{kj} does represent a time that a token would appear on the jth edge. We have $T_{11} < T_{21} < \ldots < T_{y1}$, and $T_{k1} < T_{k2} < \ldots < T_{kd_1}$ for $1 \leq k \leq y$.

We first apply Property 1 of Lemma 27 to show that packets 1 through y are able to cross the jth edge by times $perm(T_{1j}, T_{2j}, \ldots, T_{yj})$, for $1 \leq j \leq d_i$. Take an example of the second edge. Let $perm(T_{12}, T_{22}, \ldots, T_{y2}) = [T_{\sigma(1),2}, T_{\sigma(2),2}, \ldots, T_{\sigma(y),2}]$, where σ represents the sorted permutation. Property 1 of Lemma 27 implies $[T_{11}, T_{21}, \ldots, T_{y1}] < [T_{\sigma(1),2}, T_{\sigma(2),2}, \ldots, T_{\sigma(y),2}]$. Since packet 1 has left its first edge by time T_{11} and an unused token for the second edge appears by $T_{\sigma(1),2}$, packet 1 is able to advance its second edge by $T_{\sigma(1),2}$. Since packet 1 has left by $T_{\sigma(1),2}$, packet 2 is able to obtain an unused token by $T_{\sigma(2),2}$ and advance its second edge. Similar reasoning applies to packets 3 through y for the second edge. Inductively, packets 1 through y are able to advance their last edge by $perm(T_{1d_i}, T_{2d_i}, \ldots, T_{yd_i})$. This quantity is bounded by $[T_{11} + X, T_{21} + X, \ldots, T_{y1} + X]$ by

Property 2 of Lemma 27. Hence, all the session-i packets reach their destination within X steps after they obtain the initial tokens.

Let us bound the queue size now. Consider the jth edge, where $1 \leq j \leq d_i$. Suppose packet k, for $1 \leq k \leq y$, uses token κ_{kj} to cross its $j$the edge at time t_{kj}. Let $\kappa_{k,j+1}$ be the $(j+1)$st token on the same token sequence as κ_{kj}, and let $t_{k,j+1}$ be the time that $\kappa_{k,j+1}$ appears. (Note that κ_{kj} is not necessarily on the same token sequence as the initial token that packet k used to cross its first edge, and that $\kappa_{k,j+1}$ is not necessarily the token that packet k would use to cross the $(j+1)$st edge.) Since $t_{kj} < t_{k,j+1}$, Property 1 of Lemma 27 and our argument for the delay bound above imply that packets 1 through y are able to cross the $(j+1)$st edge by $perm(t_{1,j+1}, t_{2,j+1}, \ldots, t_{y,j+1})$. Property 3 of Lemma 27 shows that $|perm(t_{1,j+1}, t_{2,j+1}, \ldots, t_{y,j+1}) - [t_{1j}, t_{2j}, \ldots, t_{yj}]|$ is bounded by x, the token lag. Hence, a packet waits at most x steps to advance each edge once it obtains an initial token. □

We now describe how to transform $\mathcal{S}_{\mathcal{N}}$ into a template-based schedule. Theorem 1 implies that it suffices to define the token sequences. Let $\mathcal{T}$ be the size of each template. Suppose each session-i initial token (in the conversion step of Section 4.3.3) is owned by a session-i packet. We have shown the existence of an optimal schedule $\mathcal{S}_{\mathcal{N}}$ for these packets. The movement of each packet scheduled by $\mathcal{S}_{\mathcal{N}}$ determines a token sequence, and these token sequences define the locations of all the tokens.

Obviously, the token lag is $O(1)$ for all sequences and the end-to-end delay is $O(1/r_i + d_i)$ for all session-i token sequences. Since each session-i packet is able to obtain an initial token within $O(1/r_i + d_i)$ steps of its injection, Theorem 1 implies that the template-based schedule defined by the token sequences achieves a delay bound of $O(1/r_i + d_i)$ and constant edge queues.

In summary, we restate Theorem 10:

Theorem 10 *Consider an arbitrary network in which sessions are defined. Each session i is associated with an injection rate r_i and path length d_i. Packets are injected to the network along these sessions subject to the injection rates. If the total rate on each edge is at most $1 - \varepsilon$ for a constant $\varepsilon \in (0, 1)$, then there exists a template-based schedule such that each session-i packet reaches its destination within $O(1/r_i + d_i)$ steps of its injection and at most one packet crosses an edge at each time step. This schedule also maintains constant edge queues.*