Packet Routing with Arbitrary End-to-End Delay Requirements

Matthew Andrews
andrews@research.bell-labs.com

Lisa Zhang
ylz@research.bell-labs.com

Bell Laboratories
600-700 Mountain Avenue
Murray Hill, NJ 07974

October 7, 1998

Abstract

We study the problem of scheduling packets to meet an arbitrary set of delay requirements. Consider a connection-oriented network in which a set of sessions is defined. Each session i follows a fixed route and requests an end-to-end delay requirement Δ_i. The packet injections for each session are subject to a specified session rate and burst size; the packet movement is restricted to one packet per link per time step.

If an arbitrary set of delay requirements is schedulable for every link in isolation and for every session in isolation, we present a simple distributed protocol in which every session i achieves a delay bound of $O((\alpha + \log \frac{m}{\rho_{\min}})\Delta_i)$. Here, α is a term that depends logarithmically on the burst sizes; m is the number of links in the network and $\rho_{\min}$ is the smallest session rate. In addition, we show the existence of a schedule that achieves a bound of $O(\alpha\Delta_i)$.

We also construct an example to show that the α factor cannot be removed in general. Hence, per-session schedulability and per-link schedulability is *not* sufficient to guarantee network schedulability for arbitrary delay requirements. This provides a contrast with previous work in that the bounds are not simply composed of a per-link delay bound and a per-session bound.

Finally, we examine the problem of route selection in which the session routes are not given *a priori*. Our aim is to choose the routes so that the delay requirements can be met.

1 Introduction

The use of scheduling to minimize end-to-end delays in modern communication networks remains an important and widely studied problem. Many real-time audio and video applications rely on the ability of the network to provide delay guarantees.

We focus on the situation in which packets are injected into fixed *sessions* in the network. Most of the work on providing end-to-end delay bounds has concentrated on bounds that are dependent on the burst size σ_i, rate ρ_i and length K_i of session i. For example, if the packet movement is restricted to one packet per link per time step, Parekh and Gallager showed in [22, 23] that a special case of the Weighted Fair Queueing scheme [8] guarantees a delay bound of $O(\frac{\sigma_i}{\rho_i} + \frac{K_i}{\rho_i})$ for every session i. In a later paper [2], Andrews, Fernández, Harchol-Balter, Leighton and Zhang showed that a bound of $O(\frac{\sigma_i}{\rho_i} + K_i)$ is achievable.

However, the fact that these bounds are dependent on the session parameters is not always desirable. Some sessions with low rates might have stringent delay requirements, whereas some sessions with high rates might have looser requirements. For example, a typical voice call has a low rate (64kbps) but a delay requirement of a fraction of a second. In contrast, a large file transfer may consume large amounts of bandwidth but a delay of many seconds may be acceptable.

In this paper, we associate an *arbitrary* delay requirement with each session. We say that a set of delay requirements is *schedulable* if the packets can be scheduled so that all the requirements are met. For a *single link*, Liebeherr, Wrege and Ferrari [18] and Georgiadis, Guérin and Parekh [12] characterized a necessary and sufficient condition for schedulability.[1] (We quote this condition in (2).) They also showed that the earliest-deadline-first scheme, which gives priority to the packet that is closest to violating its delay bound, can satisfy any schedulable set of requirements.

In this paper, we are concerned with meeting an arbitrary set of requirements in the *network* setting. There are two obvious necessary conditions for schedulability. First, the schedulability conditions of Liebeherr *et al.* and Georgiadis *et al.* must hold for every link. Second, each delay requirement must be at least the length of the session route. Unfortunately, these two conditions are not sufficient to guarantee network schedulability. Briefly, our results are as follows. Given these two necessary conditions, we show how to schedule the packets so that all delay requirements are satisfied up to a logarithmic factor. Furthermore, we also construct an example in which some packet has to miss its requirement by this logarithmic factor. Before presenting our results in detail we need to describe our model.

The Model We consider a *connection-oriented* network, where a set of n sessions are defined. Each session is specified by a source, a destination and a fixed route from the source to the destination. The session length K_i is the number of links along the route of session i.

Packets are injected into the network in sessions. In particular, we adopt the *leaky-bucket constrained* injection model of Cruz [6, 7]. The session-i injections are specified by parameters (σ_i, ρ_i), where $\sigma_i \geq 1$ is the burst size and $\rho_i < 1$ is the session rate. During any time interval $(t_1, t_2]$, a total of at most,

$$\sigma_i + \rho_i(t_2 - t_1)$$

session-i packets can be injected. Upon injection, a session-i packet arrives at the source of session i. It then traverses the links of the session-i route until it reaches its destination.

We assume all packets have unit size and all links have unit bandwidth. At any time step, at most one packet can traverse each link. If two packets contend for the same link simultaneously, one packet has to wait in a queue. The *end-to-end delay* (delay for short) experienced by a packet is the total time from its injection to its arrival at its destination, including queueing time and

[1]These two papers considered slightly different models. However, for our purposes the schedulability conditions in the two papers are the same.

the time to traverse all session links. In order to achieve any bounded delay, the following *stability condition* is necessary for all links e,

$$\sum_{i \in S^e} \rho_i \leq 1 - \varepsilon, \tag{1}$$

where S^e is the set of sessions going through e and ε is a positive constant. This stability condition states that the total rate on any link has to be smaller than 1.

Each session i also specifies a delay requirement of Δ_i. If the network consists of a single link e, the following *link schedulability condition* is both necessary and sufficient [12, 18].

$$\forall t \geq 0, \qquad \sum_{i \in S^e, \Delta_i \leq t} \sigma_i + \rho_i(t - \Delta_i) \leq t. \tag{2}$$

This schedulability condition states that the maximum number of packet arrivals with deadlines at or before time t cannot exceed the maximum number of packets that can traverse the link by time t. For a set of delay requirements to be schedulable in a network with multiple links, it is certainly necessary to have the *per-session schedulability condition* of $\Delta_i \geq K_i$ for all i and to have the *per-link schedulability condition* of (2) for all e.

Our Results In this paper, we aim to schedule the packets so that the end-to-end delay experienced by every session-i packet is within a small factor of Δ_i. Let $\alpha = \Theta(\log(\max_e \sum_{i \in S^e} \sigma_i))$ and $\rho_{\min} = \min_i \rho_i$. Let m be the number of links in the network.

- In Section 2, we describe a simple distributed protocol in which every session-i packet experiences an end-to-end delay of $O(\Delta_i \alpha + K_i \log \frac{m}{\rho_{\min}})$, as long as the Δ_i's satisfy the per-link and per-session schedulability conditions. Due to the fact that $\Delta_i \geq K_i$, this bound is within a logarithmic factor, $O(\alpha + \log \frac{m}{\rho_{\min}})$, of Δ_i. Our protocol uses an earliest-deadline-first approach.

- In Section 3, we show that there exists a schedule with delay bound $O(\Delta_i \alpha + K_i)$ for session i.

- In Section 4, we show that the factor of α cannot be removed in general. We construct an example in which the Δ_i's satisfy the per-session and per-link schedulability conditions. However, *some* packet of *some* session i has to suffer a delay of $\Omega(\alpha \Delta_i)$, no matter how the packets are scheduled. This result shows that the per-session and per-link conditions are not sufficient to guarantee network schedulability for arbitrary delay requirements.

- In Section 5, we examine a related problem of route selection in which the session routes are not given *a priori*. Our aim is to choose the routes so as to meet the delay requirements. In particular, we assume the existence of a set of routes for which the Δ_i's are schedulable. For this set of routes let $\alpha = \Theta(\log(\max_e \sum_{i \in S^e} \sigma_i))$. Let $U = \max_i\{\Delta_i \rho_i, \sigma_i\}$. If $4 + 2U$ packets are allowed to cross a link in each time step, we show how to choose a set of session routes and schedule the packets so that session i achieves a delay bound of $O((\alpha + \log \frac{m}{\rho_{\min}})\Delta_i)$. Moreover, for these routes there exists a schedule with bounds $O(\alpha \Delta_i)$.

The fact that every link is schedulable in isolation and the fact that every session is schedulable in isolation do not guarantee that the network is schedulable. This conclusion is somewhat surprising and is fundamentally different from previous work. For example, consider the *static* routing problem, where all packets are present in the network initially. If C is the maximum number of packets routed through the same link (the *congestion*) and D is the maximum packet path length (the *dilation*), Leighton, Maggs and Rao showed in [16] that all packets can reach their destinations within time $O(C + D)$. Their result indicates that if a delay requirement Δ satisfies a simple per-link condition, i.e. $C \leq \Delta$, and a simple per-path condition, i.e. $D \leq \Delta$, then Δ is schedulable for the network. In another example, the $O(\frac{\sigma_i}{\rho_i} + K_i)$ bound for *dynamic* routing [2] is the sum of a delay bound, $O(\frac{\sigma_i}{\rho_i})$, that is achievable for a single link in isolation and a delay bound, $O(K_i)$, that is achievable for a session in isolation.

The schemes to achieve our logarithmic upper bounds have an unexpected feature. They produce schedules that are *independent* of the exact packet arrival times. With arbitrary delay requirements this seems suboptimal, since packets with stringent requirements need to be scheduled as soon as they arrive. Indeed, Liebeherr *et al.* and Georgiadis *et al.* took advantage of the arrival times when scheduling a single link. However, in the network environment, our lower bound shows that one cannot do better in general even if packet arrival times are taken into account.

Both of our upper bounds are motivated by the protocols given in [2], although the analysis here is somewhat more involved. The central technique is "delay insertion", a method introduced by Leighton, Maggs and Rao [16]. The essential idea is to delay packets in such a way that not many packets try to traverse a link simultaneously.

Other Related Work Delay insertion is widely used to derive analytical delay bounds. In addition to [2, 16, 17], it is also adopted by Rabani and Tardos [24] and Ostrovsky and Rabani [21]. A simulation study [3] compares the schemes of [2] favorably against Weighted Fair Queueing, demonstrating the practical benefits of such schemes.

As mentioned earlier, earliest-deadline-first (EDF) when applied to a *single link* can separate session rates from delay bounds and produce optimal delays [12, 18, 28]. For *networks*, Georgiadis, Guérin, Peris and Sivarajan [13] showed that by "reshaping" session traffic at each node, EDF can achieve the same bound $O(\frac{\sigma_i}{\rho_i} + \frac{K_i}{\rho_i})$ as Weighted Fair Queueing [22, 23]. Early papers on EDF include Ferrari and Verma [11] and Verma, Zhang and Ferrari [27]. For an overview of different scheduling schemes see [15, 30].

Our algorithm for choosing session routes combines the path filtering technique of Lin and Vitter [19] with a rounding scheme of Karp, Leighton, Rivest, Thompson, Vazirani and Vazirani [14]. This approach was first used by Srinivasan and Teo [26] in the context of static routing.

An alternative approach is to assume that the packet arrivals are generated by stochastic processes. Each packet is allowed to miss its delay requirement with a small probability. In this framework of statistical multiplexing, more stringent delay requirements are schedulable. For examples, see Elwalid, Mitra and Wentworth [10] and Elwalid and Mitra [9].

2 The Distributed Protocol

2.1 Overview

Our protocol uses randomization and an earliest-deadline-first approach. For each session-i packet, the protocol assigns deadlines $D_1, D_2, \ldots, D_{K_i}$ for each link that the packet goes through. In case of contention, a link services the waiting packet with the earliest deadline. Ties are broken arbitrarily. We define $D_1 = rand + G$ and $D_k = D_{k-1} + G$, where G is a logarithmic parameter defined later and $rand$ is a random number chosen from an appropriate range. Roughly speaking, if Δ_i is large then $rand$ if chosen from a range proportional to $1/\rho_i$; if Δ_i is small then $rand$ is chosen from a range dependent on Δ_i. Note that randomness is only added to the first deadline of each packet. This randomness has the effect of spacing out the deadlines so that no time period contains too many deadlines. In this way, every packet can meet its deadline at every link.

2.2 Description

We now concentrate on defining the deadlines.

Tokens We use *tokens* to specify the first deadlines, D_1. All tokens appear periodically with a *period* of M. For session i, let M_i be the range from which random numbers are chosen. In particular, let $\tau_1, \tau_2 \ldots \tau_{M/M_i}$ be numbers chosen uniformly at random from each of the intervals $[0, M_i), [M_i, 2M_i) \ldots [M - M_i, M)$. Session-$i$ tokens appear periodically with period M at the following times.

$$\begin{array}{llll}
\tau_1 & \tau_2 & \ldots & \tau_{M/M_i} \\
\tau_1 + M & \tau_2 + M & \ldots & \tau_{M/M_i} + M \\
\tau_1 + 2M & \tau_2 + 2M & \ldots & \tau_{M/M_i} + 2M \\
 & \vdots & &
\end{array}$$

We now let,[2]

$$\begin{aligned}
M_i &= \min\left\{\alpha \cdot \frac{\Delta_i}{\sigma_i}, \frac{1}{\rho_i}\right\}, \qquad \text{where } \alpha = \frac{2}{\varepsilon} \max_e H(\sum_{i \in S^e} \sigma_i), \qquad (3) \\
M &= \max_i M_i.
\end{aligned}$$

Here, $H(n) = \Theta(\log n)$ is the nth harmonic number.

Deadlines For each session-i packet, we define a sequence of deadlines $D_1, D_2 \ldots D_{K_i}$ for crossing the K_i links on its session route. Suppose a session-i packet p is injected at time t_{inj}. Packet p *obtains* the first session-i token that appears after t_{inj} but is not yet obtained by any earlier session-i packets. Let τ be the time that the token appears. We define the deadlines as follows.

$$\begin{aligned}
D_1 &= \tau + G, \\
D_k &= D_{k-1} + G, \\
\text{where} \quad G &= a \log(m/\rho_{\min}).
\end{aligned}$$

Here, a is a constant chosen later, m is the number of links in the network and $\rho_{\min} = \min_i \rho_i$.

Now that all deadlines are defined, each link gives priority to the packet that has the earliest deadline.

Remarks We emphasize that the only per-session processing is the determination of which token a packet obtains. This can be done at the point where the session *enters* the network. Once the token has been obtained, the deadlines for the packet are independent of its session parameters. This means that we need no per-session state *within* the network. The idea of restricting per-session processing to the network entry points is known as *differential service* [20, 29], and is becoming increasingly popular.

Our protocol can be implemented by stamping each packet with its current deadline, in a manner similar to the protocols described in [4].

2.3 Analysis

We say that our protocol is *successful* if, for any link e and any time interval $I = [t, t + G)$, fewer than G deadlines fall into interval I. We shall see in Lemma 4 that when the protocol is successful every packet meets its deadline at every link. In Lemma 3 we show that our protocol is successful with high probability. The proof of Lemma 3 relies on the following observation about the per-link schedulability condition.

[2]To be precise, we need M to be an integral multiple of every M_i. This can be achieved by defining the M_i's to be powers of 2. For clarity of presentation, we use the definition of M_i given in (3), since it does not affect our results asymptotically.

Lemma 1 *If the delay requirements Δ_i satisfy the per-link schedulability condition (2) for all links, then for all e,*

$$\sum_{i \in S^e} \frac{\sigma_i}{\Delta_i} \leq H(\sum_{i \in S^e} \sigma_i) = \Theta(\log(\sum_{i \in S^e} \sigma_i)).$$

Proof: Without loss of generality, we assume sessions $1, 2, \ldots, n_e$ go through link e and $\Delta_1 \leq \Delta_2 \leq \ldots \leq \Delta_{n_e}$. The link schedulability condition (2) implies,

$$\begin{aligned} & \sigma_1 + \ldots + \sigma_i \leq \Delta_i \\ \Longrightarrow \quad & \frac{\sigma_i}{\Delta_i} \leq \frac{\sigma_i}{\sigma_1 + \ldots + \sigma_i} \\ \Longrightarrow \quad & \frac{\sigma_i}{\Delta_i} \leq H(\sigma_1 + \ldots + \sigma_i) - H(\sigma_1 + \ldots + \sigma_{i-1}) \end{aligned}$$

Summing over all i, we obtain the lemma. □

Lemma 2 *For any link e and any time t, the expected number of packets with deadlines at time t for link e is at most* $1 - \varepsilon/2$.

Proof: It suffices to show that,

$$\sum_{i \in S^e} 1/M_i \leq 1 - \varepsilon/2. \tag{4}$$

This is because one session-i token is placed at random in each of the intervals $[0, M_i)$, $[M_i, 2M_i)$, etc., and the deadlines for each session are a fixed amount of time after the tokens.

If $\alpha \cdot \frac{\Delta_i}{\sigma_i} < \frac{1}{\rho_i}$, then $1/M_i = \frac{\sigma_i}{\alpha \Delta_i}$. If $\alpha \cdot \frac{\Delta_i}{\sigma_i} \geq \frac{1}{\rho_i}$, then $1/M_i = \rho_i$. Summing over all sessions in S^e, we have that the expected number of deadlines at time t is at most,

$$\sum_{i \in S^e} 1/M_i = \sum_{i \in S^e} \left(\frac{\sigma_i}{\alpha \Delta_i} + \rho_i \right) \leq \varepsilon/2 + (1 - \varepsilon) \leq 1 - \varepsilon/2.$$

We bound the first term using Lemma 1 and the definition of α, and we bound the second term using the stability condition (1). □

Lemma 3 *With high probability our protocol is successful, i.e. fewer than G deadlines fall into interval I for any link e and any time interval* $I = [t, t + G)$.

Proof: For each token γ of each session, let the random variable X_γ indicate whether or not γ corresponds to any deadline during interval I at link e. By linearity of expectation and Lemma 2, we have,

$$E[\sum_\gamma X_\gamma] \leq (1 - \varepsilon/2)G.$$

For fixed I and e, the X_γ's are independent Bernoulli random variables for all tokens γ. Thus, a Chernoff bound shows that,

$$\Pr\left[\sum_\gamma X_\gamma \geq G\right] < e^{-\varepsilon^2 G/12(1-\varepsilon/2)} < e^{-\varepsilon^2 G/12}.$$

Since the token placement is periodic with period M, we only need to consider a fixed time period of length M. Over all m links, there are a total of mM such I-intervals. By a union bound argument, the probability that some link e has at least G deadlines during some interval I is at most $mMe^{-\varepsilon^2 G/12}$. Since $G = a \log(m/\rho_{\min})$ and $M = O(1/\rho_{\min})$, this probability is at most $\frac{1}{poly(m/\rho_{\min})}$ for a sufficiently large constant a.

□

Lemma 4 *When our protocol is successful, each packet meets its deadline at each link on its route.*

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. Suppose that packet p misses deadline D at link e. Since packet p meets its previous deadlines, it must be waiting at link e at time $D - G + 1$. Hence, one packet crosses link e during every time step from $D - G + 1$ to D. Let p' be such a packet, then p' must have a deadline $D' \leq D$ by the definition of earliest-deadline-first. Moreover, $D' \geq D - G + 1$ since D is the first deadline missed. Hence, the total number of deadlines in $[D - G + 1, D]$ is at least G. This contradicts the assumption that the protocol is successful. $\square$

It remains to bound τ, the time when a packet catches a token after its injection.

Lemma 5 *If a session-i packet p arrives at its first link at time t_{inj}, then $\tau \leq t_{inj} + 2\alpha\Delta_i$.*

Proof: Let t_0 be the last time before t_{inj} that no session-i packet is waiting at its first link. During (t_0, τ) every session-i token must be obtained by some session-i packet injected during $(t_0, t_{inj}]$. Otherwise, either (t_0, t_{inj}) contains a time when no session-i packet is waiting or p would obtain a token before τ. The total number session-i packets injected during $(t_0, t_{inj}]$ is at most $\sigma_i + (t_{inj} - t_0)\rho_i$. The total number of session-i tokens during $(t_0, \tau]$ is at least $(\tau - t_0 - M_i)/M_i$. Therefore, the total number of session-i packets that obtain tokens during $(t_0, \tau]$ is at least $(\tau - t_0 - M_i)/M_i$. Since p does not obtain a token until time τ and packets with earlier deadlines are given priority, we have,

$$\begin{aligned}
& \frac{\tau - t_0 - M_i}{M_i} \leq \sigma_i + (t_{inj} - t_0)\rho_i \\
\Longrightarrow\quad & \tau < t_0 + M_i(1 + \sigma_i) + M_i\rho_i(t_{inj} - t_0) < t_0 + 2\alpha\Delta_i + (t_{inj} - t_0) = t_{inj} + 2\alpha\Delta_i.
\end{aligned}$$

The second term is bounded due to the fact that $M_i \leq \alpha\frac{\Delta_i}{\sigma_i}$. The third term is bounded due to the fact that $M_i \leq \frac{1}{\rho_i}$. $\square$

To summarize, Lemma 3 shows that our protocol is successful with high probability. Lemmas 4 and 5 show that, for a successful run of the protocol, the end-to-end delay experienced by every session-i packet is at most,

$$2\alpha\Delta_i + \sum_{k=1}^{K_i} G \leq 2\alpha\Delta_i + K_i a \log(m/\rho_{\min}).$$

When the delay requirements satisfy the per-session condition $\Delta_i \geq K_i$, the above bound is within a factor $2\alpha + a\log(m/\rho_{\min})$ of Δ_i.

Theorem 6 *Consider an arbitrary set of delay requirements that satisfy the per-link and per-session schedulability conditions. With high probability, every session-i packet experiences an end-to-end delay of $O(\alpha\Delta_i + K_i \log\frac{m}{\rho_{\min}}) = O(\Delta_i(\alpha + \log\frac{m}{\rho_{\min}}))$, where $\alpha = \Theta(\max_e \log\sum_{i\in S^e}\sigma_i)$.*

Note that the per-link schedulability condition is only used in Lemma 2. We have the following more general result.

Theorem 7 *Consider an arbitrary set of delay requirements. With high probability, every session-i packet experiences an end-to-end delay of $O(\beta\Delta_i + K_i \log\frac{m}{\rho_{\min}})$, where $\beta = \Theta(\max_e \sum_{i\in S^e}\frac{\sigma_i}{\Delta_i})$.*

3 Improved Bound of $O(\Delta_i\alpha + K_i)$

By combining the techniques of [2] with the analysis in Lemma 2, we can improve the bound $O(\Delta_i\alpha + K_i \log \frac{m}{\rho_{\min}})$ to $O(\Delta_i\alpha + K_i)$. Due to space limitations, we offer an overview in the Appendix. The technical details omitted can be reconstructed from [2].

Theorem 8 *For an arbitrary set of delay requirements that satisfy the per-link and per-session schedulability conditions, there exists a schedule in which every session-i packet experiences an end-to-end delay of* $O(\alpha\Delta_i + K_i) = O(\alpha\Delta_i)$, *where* $\alpha = \Theta(\max_e \log \sum_{i\in S^e} \sigma_i)$. *In general, there exists a schedule in which every session-i packet experiences an end-to-end delay of* $O(\beta\Delta_i + K_i)$, *where* $\beta = \Theta(\max_e \sum_{i\in S^e} \frac{\sigma_i}{\Delta_i})$.

4 A Lower Bound

Given the per-session and per-link schedulability conditions, we have shown how to achieve a delay bound of $O(\alpha\Delta_i)$ for all sessions i. (Recall $\alpha = \Theta\left(\max_e(\log \sum_{i\in S^e} \sigma_i)\right)$. In this section, we construct an example in which some session-i packet has to suffer a delay of $\Omega(\alpha\Delta_i)$ even if the per-session and per-link conditions hold.

A Simple Example. In order to convey the idea of our construction we first give a simple example to show that the delay requirements cannot always be met *exactly*. Consider a 2-link network. Packet A is injected at time 0, requires links 0 and 1 and has delay requirement 3. Packet B is injected at time 0, requires link 0 and has delay requirement 1. Packet C is injected at time 2, requires link 1 and has delay requirement 1.

Clearly, the schedulability conditions are met for both links. However, it is impossible to meet all three delay requirements since it would require scheduling packet B at time 0 and packet C at time 2. In addition, if packet A is to meet *its* delay requirement then it must either cross link 0 at time 0 or else it must cross link 1 at time 2. This is impossible since two packets cannot cross a link at the same time step.

Lower Bound. We now generalize the above construction to obtain our logarithmic lower bound. Our network is a linear array with 2^{2x+1} links, numbered $0, 1, \ldots, 2^{2x+1} - 1$. The sessions are defined as follows. Every session injects one packet only, which corresponds to a burst size of $\sigma_i = 1$ and a session rate of $\rho_i = 0$.[3] Let $R[e_1, e_2)$, where $e_1 < e_2$, be the route that consists of links $e_1, e_1 + 1, \ldots, e_2 - 1$. For $1 \le a \le 2x + 1$, we define 2^{a-1} sessions on each of the following routes,

$$R[0, 2^a) \qquad R[2^a, 2\cdot 2^a) \qquad R[2\cdot 2^a, 3\cdot 2^a) \qquad \ldots \qquad R[2^{2x+1} - 2^a, 2^{2x+1})$$

Each session that is defined on the route $R[b, b + 2^a)$, sends a packet at time xb and has a delay requirement 2^a. (Note that b is an integral multiple of 2^a.)

The stability condition holds trivially since each session only injects one packet. The per-session condition follows from the fact that $\Delta_i = K_i$. To verify the per-link schedulability condition, observe that for all e,

$$\sum_{i\in S^e, \Delta_i \le t} \sigma_i + \rho_i(t - \Delta_i) \le \sum_{a:2^a \le t} 2^{a-1} \le t, \qquad \forall t \ge 0.$$

The number of sessions traversing any link e is $\sum_{a=1}^{2x+1} 2^{a-1}$, which implies $\alpha = \Theta(\log 2^{2x+1})$. Hence, $x = \Theta(\alpha)$. Note that x, and hence α, can be arbitrarily large.

[3] We could make $\rho_i = 1/\Gamma$ for a sufficiently large Γ. Every session then injects one packet during the first Γ time steps, and the injection pattern repeats every Γ steps. The stability and the per-link schedulability conditions still hold if Γ is chosen appropriately.

Theorem 9 *Some session-i packet has to suffer an end-to-end delay larger than $x\Delta_i = \Omega(\alpha\Delta_i)$.*

Proof: For the purpose of contradiction, let us assume that for all i every session-i packet experiences a delay of at most $x\Delta_i$. We use this assumption to prove the following lemma. Let $T[t_1, t_2)$, where $t_1 < t_2$, denote the time interval $t_1, t_1 + 1, \ldots, t_2 - 1$. Let $f(a) = (2x + 2 - a)2^{a-1}$.

Lemma 10 *For $1 \le a \le 2x + 1$, there exists some b_a, where b_a is an integral multiple of 2^a, such that at least $f(a)$ packets traverse the route $R[b_a, b_a + 2^a)$ during $T[xb_a, xb_a + x2^a)$.*

Proof: We proceed by a backwards induction on a. We first note that $f(a)$ satisfies the following recurrence,

$$f(a) = \begin{cases} f(a+1)/2 + 2^{a-1} & \text{if} \quad a \le 2x, \\ 2^{2x} & \text{if} \quad a = 2x + 1. \end{cases}$$

Consider the base case of $a = 2x + 1$. At time 0, 2^{2x} packets are injected to follow the route $R[0, 2^{2x+1})$, each of which has a delay requirement of 2^{2x+1}. By assumption, all these packets can be routed during $T[0, x2^{2x+1})$. The base of the induction holds for $b_a = 0$.

Now suppose that the lemma holds for $a + 1$, and we prove it for a. We know that at least $f(a+1)$ packets traverse the route $R[b_{a+1}, b_{a+1} + 2^{a+1})$ during $T[xb_{a+1}, xb_{a+1} + x2^{a+1})$. Let P_{a+1} denote this set of packets.

Case 1. At least $f(a+1)/2$ of the packets in P_{a+1} traverse the route $R[b_{a+1}, b_{a+1} + 2^a)$ during $T[xb_{a+1}, xb_{a+1} + x2^a)$. Let $b_a = b_{a+1}$. Since b_a is an integral multiple of 2^a, 2^{a-1} packets are injected to traverse the route $R[b_a, b_a + 2^a)$ at time xb_a, each of which has a delay requirement of 2^a. By assumption, all these packets can be routed during $T[xb_a, xb_a + x2^a)$. Hence, the total number of packets that traverse the route $R[b_a, b_a + 2^a)$ during $T[xb_a, xb_a + x2^a)$ is at least,

$$f(a+1)/2 + 2^{a-1} = f(a).$$

The inductive step holds.

Case 2. Fewer than $f(a+1)/2$ of the packets in P_{a+1} traverse the route $R[b_{a+1}, b_{a+1} + 2^a)$ during $T[xb_{a+1}, xb_{a+1} + x2^a)$. This implies that at least $f(a+1)/2$ of the packets in P_{a+1} traverse the route $R[b_{a+1} + 2^a, b_{a+1} + 2^{a+1})$ during $T[xb_{a+1} + x2^a, xb_{a+1} + x2^{a+1})$. Let $b_a = b_{a+1} + 2^a$. The inductive step holds using a similar argument to Case 1. □

The proof of Theorem 9 is now simple. For $a = 1$, Lemma 10 implies that $2x + 1$ packets traverse links $2b_1$ and $2b_1 + 1$ during $2x$ time steps. This contradicts the fact that at most one packet at a time can traverse each link. □

5 Determining Session Routes

So far we have focused on packet scheduling when the session routes are given. A natural related problem is the *selection* of the session routes when only the sources and destinations are given.

The analogous problem in the context of static routing was considered by Srinivasan and Teo [26]. (See the Introduction for a discussion on static routing.) They showed how to choose the packet routes so as to minimize the sum of the congestion and the dilation up a constant factor. In conjunction with the scheduling results of [16, 17], this provides a constant-factor approximation for minimizing the routing time. Srinivasan and Teo constructed their routes using the "path-filtering" technique of Lin and Vitter [19] coupled with a rounding technique for "column-sparse" matrices due to Karp, Leighton, Rivest, Thompson, Vazirani and Vazirani [14].

In this section, we study the following problem. We are given a set of sessions, each of which is specified by its burst size σ_i, its rate ρ_i, its source and destination and a delay requirement Δ_i. We assume the *existence* of some session routes and a schedule for which all the delay requirements can be met. Our goal is then to *find* a set of routes such that, by using the scheduling results of Sections 2 and 3, we can closely approximate the delay requirements.

In general, it is NP-hard to even find a set of routes for which the rates at each link add up to less than 1. Hence, we opt to relax the constraint of "one packet per link per time step" by increasing the link bandwidth, thereby allowing more than one packet to cross a link per time step. Thus, a second goal is to keep this bandwidth increase as small as possible. If we allow arbitrarily large delay requirements, then this problem of minimizing bandwidth is all that remains, and we have a standard multicommodity flow problem, e.g. [25]. Here we are more interested in what is achievable when the delay requirements are stringent. Let $U = \max_i\{\Delta_i\rho_i, \sigma_i\}$. We focus on the situation when U is small.[4]

We have assumed the existence of a set of session routes for which the delay requirements are schedulable. For these routes (which we do not know), define S_e to be the set of sessions that go through link e and let $\alpha = \max_e H(\sum_{i\in S^e}\sigma_i)$. We use a similar framework to Srinivasan and Teo. However, we must be more careful since different sessions have different delay requirements.

Theorem 11 *We can choose a set of routes for which every session i achieves a delay bound of $O(\Delta_i(\alpha + \log\frac{m}{\rho_{\min}}))$ if $4+2U$ packets are allowed to cross a link simultaneously, and the distributed protocol of Section 2 is used. Moreover, for these routes there exists a schedule in which the delay bounds $O(\alpha\Delta_i)$ are achieved.*

Proof: Let $M_i = \min\{\alpha\frac{\Delta_i}{\sigma_i}, \frac{1}{\rho_i}\}$. For the routes on which the delay requirements are achieved, the argument that proves Inequality (4) in Lemma 2 implies that $\sum_{i\in S^e} 1/M_i \le 2$ for all e. Hence, the following system of equations is feasible. Note that although we do not know α, we can use a standard doubling technique to estimate α until the system becomes feasible.

$$\begin{aligned} &\text{Flow conditions} & \\ \sum_e x_i^e &\le \Delta_i & \forall i \\ \sum_i \max\left\{\rho_i, \frac{\sigma_i}{\alpha\Delta_i}\right\} x_i^e &\le 2 & \forall e \\ x_i^e &\in \{0,1\} & \forall e, i \end{aligned}$$

The flow conditions ensure that the variables x_i^e define one unit flow from the source to the destination of session i. These conditions are standard and we do not write them explicitly.

We solve the fractional relaxation of the above system, in which $x_i^e \in \{0,1\}$ is relaxed to $0 \le x_i^e \le 1$. Let $\bar{x}_i^e$ be the fractional solution thus obtained.

Path Filtering For each session i, we use standard path decomposition techniques to decompose the flow between the source and the destination of session i into a polynomial number of paths. For session i, let P_{ij} denote its jth flow path, let y_{ij} be the fraction of session-i flow carried on path P_{ij}, and let K_{ij} be the number of links along P_{ij}. Observe that for all sessions i, $\sum_j y_{ij} = 1$ and $\sum_j y_{ij}K_{ij} \le \Delta_i$. Path P_{ij} is *long* if $K_{ij} \ge 2\Delta_i$; P_{ij} is *short* otherwise. We now filter out the long paths and scale up the short paths as follows. Let $f_i = \sum_{P_{ij}\text{ short}} y_{ij}$ be the fraction of session-i flow carried by short paths. By our earlier observation and the definition of short paths, we have $f_i > 1/2$ for all i. We now let each short path P_{ij} carry a flow of $y'_{ij} = y_{ij}/f_i$, and let each long path P_{ij} carry zero flow, i.e. $y'_{ij} = 0$. Let $P_{i1}, \ldots, P_{ij_i}$ be the short paths for session i.

[4]The parameter U corresponds to the bandwidth increase. It must be dependent on $\sigma_{\max}$, otherwise there can be an arbitrary gap between delay requirements that are schedulable using fractional paths and delay requirements that are schedulable using integral paths. We omit the details.

The variables y'_{ij} satisfy the following constraints.

$$\sum_{1 \le j \le j_i} y'_{ij} = 1 \qquad \forall i \tag{5}$$

$$\sum_{ij: e \in P_{ij}} \max\left\{\rho_i, \frac{\sigma_i}{\alpha \Delta_i}\right\} y'_{ij} \le 4 \qquad \forall e \tag{6}$$

Note that we now have the value 4 on the right-hand side of Constraint (6) due to the filtering of the long paths. Our goal is to round y'_{ij} to $\{0, 1\}$ so that the left-hand side of Constraint (5) does not decrease and the left-hand side of Constraint (6) increases as little as possible.

Rounding We now apply a rounding result of Karp *et al.* [14] to the following linear system.

$$\sum_{1 \le j \le j_i} -2U y'_{ij} = -2U \qquad \forall i \tag{7}$$

$$\sum_{ij: e \in P_{ij}} \max\left\{\rho_i, \frac{\sigma_i}{\alpha \Delta_i}\right\} y'_{ij} \le 4 \qquad \forall e \tag{8}$$

Rounding Theorem *Let A be a real-valued $r \times s$ matrix, let y and b be a real-valued vectors such that $Ay = b$, and let t be a positive real number such that in every column of A, the sum of all the positive entries is at most t and the sum of all the negative entries is at least $-t$. Then we can compute an integral vector $\bar{y}$ such that for every i, either $\bar{y}_i = \lfloor y_i \rfloor$ or $\bar{y}_i = \lceil y_i \rceil$, and $A\bar{y} = \bar{b}$, where $\bar{b}_i - b_i < t$ for all i.*

In order to apply this Rounding Theorem to our linear system, observe that by the path filtering, any path used by session i passes through at most $2\Delta_i$ links. Hence, the variable y'_{ij} appears in at most $2\Delta_i$ rows with a positive coefficient. This in turn implies that any column sum of positive entries in our linear system is at most,

$$\max_i\{2\Delta_i \max\{\rho_i, \frac{\sigma_i}{\alpha \Delta_i}\}\} \le \max_i\{2\Delta_i \rho_i, \frac{2\sigma_i}{\alpha}\} \le 2U.$$

Any column sum of negative entries is exactly $-2U$. We apply the Rounding Theorem. From (7) the rounded variables $\bar{y}_{ij}$ satisfy,

$$\sum_{1 \le j \le j_i} -2U \bar{y}_{ij} < 0 \qquad \forall i$$

Since $\bar{y}_{ij}$ are 0-1 integers, $\sum_{1 \le j \le j_i} \bar{y}_{ij} \ge 1$, which implies that we have chosen a route for every session. Furthermore, the right-hand side of (8) increases by at most $2U$. This implies,

$$\sum_{ij: e \in P_{ij}} \max\left\{\rho_i, \frac{\sigma_i}{\alpha \Delta_i}\right\} \bar{y}_{ij} \le 4 + 2U \qquad \forall e \tag{9}$$

Wrapping Up Thus far, we have chosen a route of length at most $2\Delta_i$ for every session i. For these routes, let $\bar{S}^e$ be the set of sessions that pass through e. We have $\sum_{i \in \bar{S}^e} \rho_i \le 4 + 2U$ for all e from (9).

Now suppose that we relax the restriction of one packet per link per time step and allow u packets to cross a link simultaneously. It is straightforward to adapt the analysis of Section 2 to obtain the following analogue of Theorem 7. For any arbitrary set of delay requirements δ_i, we can achieve a delay bound $O(\beta \delta_i + K_i \log \frac{m}{\rho_{\min}})$ for session i, where $\beta = \Theta(\frac{1}{u} \max_e \sum_{i \in S^e} \frac{\sigma_i}{\delta_i})$.

In our case, we set $u = 4+2U$ and $\delta_i = \min\{\alpha\Delta_i, \frac{\sigma_i}{\rho_i}\}$. From (9), we have $\sum_{i\in\bar{S}^e} \frac{\sigma_i}{\delta_i} \le 4+2U$ for all e, which implies $\beta = O(1)$. We also have $K_i < 2\Delta_i$. Hence, if we use the distributed protocol of Section 2, session i achieves a delay bound of $O(\Delta_i(\alpha + \log \frac{m}{\rho_{\min}}))$. A similar generalization can be applied to Theorem 8 to remove the factor $\log \frac{m}{\rho_{\min}}$. Hence, there exists a schedule in which session i achieves a delay bound of $O(\alpha\Delta_i)$. $\square$

6 Open Questions

The following questions remain. Given that the obvious per-session and per-link conditions are insufficient for network schedulability, is there a simple characterization that is both necessary and sufficient for the network environment? If so, is there a distributed protocol that can guarantee end-to-end delays within a constant factor of the requirements?

We have concentrated on a connection-oriented model, in which packets follow a fixed set of session routes. In a contrasting *connectionless adversarial model* [5, 1], no fixed rate is associated with any route and the adversary is given the power to choose the route upon each packet injection. In this model, it only makes sense to associate a delay requirement with each packet. Similar questions arise such as identifying and meeting schedulable sets of delay requirements.

Acknowledgment

We are indebted to Kavita Ramanan for her help.

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

[2] M. Andrews, A. Fernández, M. Harchol-Balter, T. Leighton, and L. Zhang. Dynamic packet routing with per-packet delay guarantees of O(distance + 1/session rate). In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 294 – 302, Miami Beach, FL, October 1997.

[3] M. Andrews and L. Zhang. Minimizing end-to-end delay in high-speed networks with a simple coordinated schedule. Submitted, July 1998.

[4] A. Banerjea, D. Ferrari, B. Mah, M. Moran, D. Verma, and H. Zhang. The Tenet real-time protocol suite: Design, implementation, and experiences. *IEEE/ACM Transactions on Networking*, 4(1):1 – 11, February 1996.

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

[6] R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory*, pages 114 – 131, 1991.

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

[8] 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.

[9] A. Elwalid and D. Mitra. Design of generalized processor sharing schedulers which statistically multiplex heterogeneous QoS classes. Bell Labs. Technical Memorandum, May 1998.

[10] 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*, 13(6):1115 – 1127, August 1995.

[11] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368 – 379, April 1990.

[12] L. Georgiadis, R. Guérin, and A. Parekh. Optimal multiplexing on a single link: delay and buffer requirements. *IEEE Transactions on Information Theory*, 43(5):1518 – 1535, September 1997.

[13] L. Georgiadis, R. Guérin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. In *Proceedings of IEEE INFOCOM '96*, pages 102 – 110, 1996.

[14] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, 2:113 – 129, 1987.

[15] S. Keshav. *An engineering approach to computer networking*. Addison Wesley, Reading, MA, 1997.

[16] 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, 1993.

[17] 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.

[18] J. Liebeherr, D. Wrege, and D. Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking*, 4(6):885 – 901, December 1996.

[19] J. Lin and J. S. Vitter. ε-Approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771 – 782, Victoria, B.C. Canada, May 1992.

[20] K. Nichols and S. Blake. Differentiated services operational model and definitions, 1998. Internet Draft.

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

[22] 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.

[23] 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.

[24] 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.

[25] P. Raghavan and C.D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365 – 374, 1991.

[26] A. Srinivasan and C. Teo. A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 636 – 643, El Paso, TX, May 1997.

[27] D. Verma, H. Zhang, and D. Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of Tricomm '91*, Chapel Hill, NC, April 1991.

[28] D. Wrege and J. Liebeherr. A near-optimal packet scheduler for QoS networks. In *Proceedings of IEEE INFOCOM '97*, 1997.

[29] J. Wroclawski. Differential service for the Internet. http://diffserv.lcs.mit.edu, 1998.

[30] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. In *Proceedings of IEEE*, October 1995.

Appendix

A. Chernoff Bound

The proof of Lemma 3 uses a slightly nonstandard statement of the Chernoff Bound in which we only have an upper bound on the expectation. We include the statement here.

Chernoff Bound *Let Y be a sum of independent Bernoulli random variables, where $E[Y] \leq \mu$. Then,*

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

for $\delta < 1$.

B. Overview of Proof of Theorem 8

In this section, we give an overview of the proof of Theorem 8. The proof uses the ideas of Lemma 2 combined with some results from [2]. The omitted details can be found in [2]. We begin by restating the theorem.

Theorem 8 *For an arbitrary set of delay requirements that satisfy the per-link and per-session schedulability conditions, there exists a schedule in which every session-i packet experiences an end-to-end delay of $O(\alpha\Delta_i + K_i) = O(\alpha\Delta_i)$, where $\alpha = \Theta(\max_e \log \sum_{i \in S^e} \sigma_i)$.*

For this schedule, packet movement at each link is governed by tokens. A session-i packet can cross a link e at time step t if and only if a session-i token is placed at link e in time slot t. The restriction of one packet per link per time step is enforced by having at most one token per time slot.

We concentrate on placing tokens at every link in the time interval $[0, T)$ for some $T = \Theta(\Delta_{\max}\alpha + K_{\max})$. The same token placement is repeated in the intervals $[T, 2T), [2T, 3T)$, etc. In placing the tokens, we make sure that all the session-i packets injected during $[jT - T_i, (j+1)T - T_i)$ move from their source to their destination during $[jT, (j+1)T)$ for some $T_i = \Theta(\Delta_i\alpha + K_i)$.

We borrow the terminology of *relative congestion* from [16]. A schedule has relative congestion c for frames of size I if for any link e and for any time interval $[t, t+I')$, where $I' \geq I$, the total number of tokens placed in $[t, t+I')$ is at most cI'. Enforcing the restriction of one token per link per time slot is equivalent to creating a schedule with relative congestion 1 for frame size 1. In the following we first create a schedule $\mathcal{S}^{(\zeta)}$ that has relative congestion 1 for a *constant* frame size. We then indicate how to reduce the frame size to 1.

The schedule $\mathcal{S}^{(\zeta)}$ is the last in a sequence of schedules $\mathcal{S}^{(0)}, \mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(\zeta)}$. In each schedule, sessions fall into two groups, the *fractional* sessions which have a small Δ_i, and the *integral* sessions which have a large Δ_i. For a fractional session i, a fractional token of size about $1/M_i$ is placed in every time slot at every link on the session route. (Recall the definition of M_i in Equation (3).) For an integral session we carefully place integral packets. For the initial schedule $\mathcal{S}^{(0)}$, all sessions are fractional. During the construction of each successive schedule, some fractional sessions are converted into integral ones. When we terminate with the schedule $\mathcal{S}^{(\zeta)}$, all sessions are integral.

Lemma 12 *The relative congestion in $\mathcal{S}^{(0)}$ is at most $1 - \varepsilon/2$ for frames of any size.*

The proof is similar to that of Lemma 3. This is where the factor α is required. We omit the details. From now on, we inductively assume that the relative congestion in $\mathcal{S}^{(q)}$ is at most $c^{(q)}$ for frames of size $I^{(q)}$. For the base of the induction, we choose $c^{(0)} = 1 - \varepsilon/2$ and $I^{(0)} = poly(\Delta_{\max}\alpha + K_{\max})$. (The exact choice of $I^{(0)}$ is somewhat technical.)

For $\mathcal{S}^{(q)}$ we first carry out a *refinement* step, which significantly reduces the frame size without increasing the relative congestion by much. We then carry out a *conversion* step, which converts some sessions from fractional to integral while maintaining the frame size and slightly increasing the relative congestion. Both steps are achieved by randomly delaying the integral tokens. The existence of a "good" set of delays is guaranteed by an argument that uses the Lovász Local Lemma. The following Lemmas 13 and 14 summarize the refinement and conversion steps respectively.

Lemma 13 *For schedule $\mathcal{S}^{(q)}$, the integral tokens can be delayed by at most $(I^{(q)})^2$ steps so that the resulting relative congestion is at most $(1 + o(1))c^{(q)}$ for frames of size $I^{(q+1)} := \log^5 I^{(q)}$.*

Lemma 14 *After refining schedule $\mathcal{S}^{(q)}$, integral tokens can be placed for fractional sessions i, where $\Delta_i\alpha \geq (I^{(q+1)})^2$, so that the resulting relative congestion is at most $c^{(q+1)} := (1 + o(1))^2 c^{(q)}$ for frames of size $I^{(q+1)}$.*

By applying Lemmas 13 and 14 to $\mathcal{S}^{(q)}$, we obtain the schedule $\mathcal{S}^{(q+1)}$. The proofs can be found in the full version of [2]. The proofs maintain a key invariant that once a packet starts moving it is delayed at most once every constant number of steps. Moreover, an argument similar to that for Lemma 5 can show that every session-i packet catches a token to cross the first link within $O(\Delta_i\alpha)$ steps of its injection.

If we choose the $o(1)$ functions in Lemma 13 and 14 appropriately, then in $O(\log^* I^{(0)})$ iterations of refinement and conversion we can obtain a schedule $\mathcal{S}^{(\zeta)}$ with relative congestion 1 for frames of size w, where w is some constant.

The only remaining problem is that as many as w tokens can be placed in one time slot. The solution is to apply the above process to a modified network in which every link is replaced by w consecutive links. We then convert the resulting schedule for the modified network into a schedule for the original network. We can make sure that at most one token is placed in every time slot and that each session-i packet takes time $O(wK_i) = O(K_i)$ to reach its destination once it starts moving. For a detailed explanation, see [2].