Source Routing and Scheduling in Packet Networks *

Matthew Andrews † Antonio Fernández ‡ Ashish Goel § Lisa Zhang ¶

Abstract

We study routing *and* scheduling *in packet-switched networks. We assume an adversary that controls the injection time, source, and destination for each packet injected. A set of paths for these packets is* admissible *if no link in the network is overloaded. We present the first on-line routing algorithm that finds a set of admissible paths whenever this is feasible. Our algorithm calculates a path for each packet as soon as it is injected at its source using a simple shortest path computation. The length of a link reflects its current congestion. We also show how our algorithm can be implemented under today's Internet routing paradigms.*

When the paths are known (either given by the adversary or computed as above) our goal is to schedule the packets along the given paths so that the packets experience small end-to-end delays. The best previous delay bounds for deterministic and distributed scheduling protocols were exponential in the path length. In this paper we present the first deterministic and distributed scheduling protocol that guarantees a polynomial end-to-end delay for every packet.

Finally, we discuss the effects of combining routing with scheduling. We first show that some unstable scheduling protocols remain unstable no matter how the paths are chosen. However, the freedom to choose paths can make a difference. For example, we show that a ring with parallel links is stable for all greedy scheduling protocols if paths are chosen intelligently, whereas this is not the case if the adversary specifies the paths.

1 Introduction

Two of the most important problems in the control of packet-switched networks are *routing* and *scheduling*. The goal of routing is to assign a path to a packet from its source to its destination. The goal of scheduling is to deal with the *contention* that occurs when two or more packets wish to cross a link simultaneously. Each link must have a *scheduler* that resolves this contention by deciding which packet to advance.

The scheduling problem typically assumes that the paths of the packets are given as part of the input. The goal is then to schedule the packets along their paths in such a way that they all reach their destinations in a short time. Much recent work has focused on the *Adversarial Queueing Model*, e.g. [7, 2, 8]. We follow their convention and assume that all packets are unit size and each link processes one packet per time step. In this Adversarial Queueing Model, the adversary chooses the injection time, source, destination, and route for each packet injected. A sequence of injections is called (w, r)*-admissible* for a window size w and injection rate $r < 1$, if in any time interval of $T \geq w$ the total number of packets injected into the network whose paths pass through any link e is at most Tr. These paths are also called (w, r)*-admissible*. Previous work has examined the performance of a number of simple scheduling protocols in this model. A packet scheduling protocol is said to be *universally stable* if it guarantees bounded buffer sizes and packet transmission delays for any (w, r)-admissible injections. In [2] it was proved that several natural protocols (Longest-In-System, Shortest-In-System, Furthest-To-Go) are universally stable, whereas several others (First-In-First-Out, Last-In-First-Out, Nearest-To-Go) are not.

In this paper we study both routing and scheduling. The adversary no longer specifies the route of each packet; it merely specifies the source and destination. However, we are guaranteed that (w, r)-admissible paths for the injections do exist. The problem is now two-fold. We first need to find some (W, R)-admissible paths, possibly for a different window size W and a different $R < 1$. These admissible paths combined with a universally stable scheduling scheme, such as the ones in [2] or the one presented in Section 3 of this paper, result in a universally stable protocol for

*Partially supported by DIMACS funding.

†Bell Laboratories. andrews@research.bell-labs.com.

‡GSyC, ESCET, Universidad Rey Juan Carlos, Spain. anto@gsyc.escet.urjc.es.

§Department of Computer Science, University of Southern California. agoel@cs.usc.edu.

¶Bell Laboratories. ylz@research.bell-labs.com.

routing and scheduling.

1.1 Source Routing for Stability

Our result. In Section 2 of the paper we present the first online algorithm for assigning admissible routes to packets. If the adversary can assign (w, r)-admissible routes, then our algorithm finds a set of (W, R)-admissible routes where $R \in (r, 1)$ is of our choice and $W \geq w$ is determined by the choice of R. Hence, if the parameter of merit is the window size w, then our algorithm is a W/w-approximation algorithm (modulo a small increase in the rate). Moreover, our algorithm is online in that it assigns routes to packets as soon as they are injected into the network. Hence it can also be regarded as a W/w-competitive algorithm for this problem. This is the first approximation algorithm/competitive algorithm for this problem. Once the routes are chosen, we can use any "good" scheduling protocol in the Adversarial Queueing Model.

Our algorithm is based on the ε-approximation algorithm for fractional maximum multicommodity concurrent flow given by Garg and Könemann [10], which in turn builds upon the work of Plotkin, Shmoys, and Tardos [13] and Young [18]. In the maximum multicommodity concurrent flow problem, the demands for each commodity remain constant as the algorithm progresses. In our setting, the demands between source-destination pairs correspond to the packets injected by the adversary, which can change over time. Even though the algorithm of Garg and Könemann [10] is an *offline* algorithm that assigns *fractional* paths to a fixed set of commodities, in our setting we are able to convert it into an *online* algorithm that assigns an *integral* path to each packet as soon as it is injected.

Implementation under Internet routing paradigms. At a high level, our algorithm works as follows. Each link maintains a measure of *congestion* that represents how many packets have been routed through it in the recent past. Packets are then routed on shortest paths with respect to this congestion measure. Hence we need a mechanism for distributing congestion information from the links to the source nodes. We also need a mechanism by which a source node can inform a link whenever it routes a packet through that link.

The first requirement could be satisfied by something akin to the OSPF (Open Shortest Path First) link state flooding protocol. (See e.g. [11].) This is a protocol that is used for flooding link state information to the nodes in a network so that packets may be routed along shortest paths. The second requirement may be satisfied by the MPLS (Multi-Protocol Label Switching) protocol that is gaining increasing acceptance in the Internet. (See e.g. [15].) With this protocol a source node can compute an explicit route to each destination and then distribute a label for the route to each of the links that comprise the route. In combination with this label distribution the source can also specify how much traffic it is going to send on the route.

In Section 2 we first assume that this control information is transmitted instantaneously and does not contribute to the congestion in the network. We then consider a model in which the control information is transmitted in-band through the network and must contend with the data traffic.

Relation to previous work. Routing and scheduling as a combined problem has been studied in the past. For example, Aiello et al. presented a distributed algorithm [1] motivated by the Awerbuch-Leighton multicommodity flow algorithm [5]. In [9] Gamarnik gave a solution based on an approximation algorithm for static routing. However, both these algorithms require a dependence between how a packet is routed and how it is scheduled. Hence, their routing schemes only work in association with their specific scheduling schemes, but not with generic scheduling algorithms. Neither routing algorithm can be used to provide packets with admissible paths at injection time. Using networking terminology, these routing algorithms correspond to *active routing* [17], where intermediate routers need to actively participate in determining routes for each individual packet. In contrast, our algorithm corresponds to *source routing*where the entire path of a packet is known at the source.

1.2 Deterministic Distributed Scheduling with Polynomial Delays

In Section 3 of the paper we study the scheduling problem in isolation assuming that (w, r)-admissible paths are given. In recent years, a number of scheduling algorithms have been proposed that guarantee *network stability*, i.e. the number of packets in the network remains bounded and the end-to-end delay experienced by packets remains bounded. For example, the Longest-In-System protocol that always gives priority to the packet injected into the system earliest, was shown in [2] to guarantee a delay bound of $O(w/(1-r)^{d_{\max}})$, where $d_{\max}$ is the maximum length of a path assigned to any packet. Note however, that this bound is exponential in $d_{\max}$. It has been an open problem whether or not any deterministic, distributed scheduling protocol has a polynomial delay bound in the Adversarial Queueing Model. Indeed, [2] remarked that "it is of considerable interest to determine whether such a protocol exists".

A *randomized* protocol based on Longest-In-System can guarantee that each packet experiences a delay of $poly(w, 1/(1-r), d_{\max}, \log m)$ with high probability [2], where m is the number of links in the network. In essence, for most of the time the protocol is successful and keeps all delays small. However, even if the failure probability is small, if the algorithm is run for an extended period of time then the algorithm is likely to make some random choices that are bad. This causes packets to violate the delay bound. Moreover, if one packet violates the delay bound then other packets injected along the same path at similar times are also likely to violate the delay bound. Hence, all of the packets that make up a single file transfer could be excessively delayed. Although this randomized protocol can be derandomized in a centralized manner it seems hard to convert it into a deterministic, *distributed* protocol. This is because the "success condition" involves packets injected at multiple source nodes and hence it cannot be verified locally.

Our result. In Section 3 we present the first deterministic, distributed scheduling protocol with a polynomial delay bound. It guarantees that *all* packets reach their destination within $poly(w, 1/(1-r), m)$ steps of their injection. We start by presenting a randomized protocol in which the "success condition" can be verified at the source nodes independently. This allows us to derandomize the protocol in a distributed fashion.

1.3 The Effects of Combining Source Routing with Scheduling

In the final part of the paper we consider the following question: Is it possible for unstable scheduling protocols to become stable if paths can be chosen by a routing algorithm as opposed to being dictated by the adversary? We first present a network and a sequence of packet injections such that regardless of how the routes for these packets are chosen, many greedy protocols (including FIFO) remain unstable. Thus, we cannot hope to achieve stability using FIFO even if we have the freedom to choose routes. However, we also present an example in which the ability to select the routes does make a difference. We show that in a "ring" with multiple parallel links, if we are allowed to choose the routes intelligently then we can ensure that all greedy scheduling protocols are stable. However, if the adversary dictates the routes then many scheduling protocols (including FIFO) are unstable.

1.4 Other Related Work

Much traditional work on routing focuses on the problem of routing *flows* online, e.g. [3, 4]. Each flow requests a bandwidth from a source to a destination and we must choose a path for each accepted flow without violating any link capacity. The goal is to maximize the on-line acceptance rate. However, this work does not consider packet-level behavior.

The problem of choosing routes for a fixed set of packets was studied by Srinivasan and Teo [16] and Bertsimas and Gamarnik [6]. For example, [16] presents an algorithm that minimizes the congestion and dilation of the routes up to a constant factor. This result complemented the paper of Leighton, Maggs and Rao [12] which showed that packets could be scheduled along a set of paths in time O(congestion+dilation).

2 Source Routing for Stability

For convenience we use the following weaker notion of admissibility in this section. We say that a set of packet paths is *weakly* (w, r)*-admissible* if we can partition time into windows of length w such that for each window *in the partition* and each link e, the number of paths that pass through e and correspond to packets injected during the window is at most wr. However, this distinction is not important due to Lemma 1. Moreover, all of the delay bounds that have been derived in the past for the Adversarial Queueing Model apply to weakly (w, r)-admissible paths.

Lemma 1 *If a set of paths is (w, r)-admissible then it is also weakly (w, r)-admissible. Conversely, weak (w, r)-admissibility implies (w', r')-admissibility for some $w' \geq w$ and $r' \in [r, 1)$.*

Proof: Suppose the injections are weakly (w, r)-admissible. We show that they are (w', r')-admissible for $r' = (1+r)/2$ and $w' = 4wr/(1-r)$. For any $T \geq w'$, let T be in the range of $[nw, (n+1)w)$ where n is an integer at least $4r/(1-r)$. Due to weak admissibility and our choices of n, T and r', the number of injections during T steps for any link e is at most,

$$(n+2)rw \leq nw(1+r)/2 \leq Tr'.$$

The other direction is trivial. ■

We assume an adversary that injects weakly (w, r)-admissible packets into the network[1]. Our aim is to choose weakly (W, R)-admissible routes for these packets where $R \in (r, 1)$ is of our choice and $W \geq w$ is determined by the choice of R.

2.1 The Basic Routing Protocol

We first assume that control information is communicated instantaneously. Whenever a source node chooses a route for a packet, this information is instantaneously transmitted to all the links on the route. Whenever the congestion on a link changes, this fact is instantaneously transmitted to all the source nodes. Later on we relax these assumptions. As mentioned in the Introduction, the algorithm is

[1] In fact, as will be seen later, we only need to assume that the adversary can choose *fractional* paths that are weakly (w, r)-admissible.

Find routes.
Initialize $c(e) = \delta, \forall e$
for the ith window, $i = 1, \ldots, t$
    for each packet injected during ith window
        $p \leftarrow$ least congested route under c (i.e. shortest path with respect to c)
        $c(e) \leftarrow c(e)(1 + \mu/w), \forall e \in p$

Figure 1. Procedure to find routes for packets injected during one phase.

based on the Garg-Könemann offline approximation algorithm for fractional maximum concurrent flow. However, in our setting we can convert it into an *online* algorithm that chooses *integral* paths for the packets.

Protocol. We route every packet injected along the path whose total congestion is the smallest under the current congestion function $c(\cdot)$, i.e. we route along shortest paths with respect to $c(\cdot)$. Initially, the congestion along every link is set to δ where δ is defined in (2). For every link e along the chosen route, its congestion $c(e)$ is updated to $c(e)(1+\mu/w)$ where μ is defined in (1). We reset the congestion of every link to its initial value of δ at the beginning of each *phase*. A phase terminates in t windows of w steps, where t is an integer defined in (3). Figure 1 illustrates the procedure for one phase. The values of μ, δ and t are defined as follows. Let m be the number of links in the network. For any $R \in (r, 1)$ of our choice, let

$$\mu = 1 - \left(\frac{r}{R}\right)^{1/3} \tag{1}$$

$$\delta = \left(\frac{1 - r\mu}{m}\right)^{1/r\mu} \tag{2}$$

$$t = \left\lfloor \frac{1 - r\mu}{r\mu} \ln \frac{1 - r\mu}{m\delta} \right\rfloor + 1 \tag{3}$$

Our objective is to show,

Theorem 2 *For all packets injected during one phase, at most twR of their routes chosen by our procedure go through the same link. In other words these routes are weakly (tw, R)-admissible.*

Analysis. To prove Theorem 2 let us examine an integer program formulation for routing the set of packets injected during a window of w time steps. Let P_j be the set of possible routes for the jth packet, and let variable $x_j(p) \in \{0, 1\}$ indicate whether or not route $p \in P_j$ is chosen for packet j. The following linear relaxation of the integer program (LP) has an optimal solution $\lambda \geq 1$ since the injections are (w, r)-admissible. We present both the primal and the dual.

Primal

$$\begin{array}{lll} & \max \lambda & \\ s.t. & & \\ & \sum_{p \in P_j} x_j(p) \geq \lambda & \forall j \\ & \sum_j \sum_{p: e \in p, p \in P_j} x_j(p) \leq rw & \forall e \\ & x_j(p) \geq 0 & \forall j, \forall p \in P_j \end{array}$$

Dual

$$\begin{array}{lll} & \min \sum_e rw \cdot c(e) & \\ s.t. & & \\ & \sum_{e \in p} c(e) \geq z(j) & \forall j, \forall p \in P_j \\ & \sum_j z(j) \geq 1 & \\ & c(e) \geq 0 & \forall e \\ & z(j) \geq 0 & \forall j \end{array}$$

For any non-negative congestion function $c(\cdot)$, let $D = \sum_e c(e)$ be the total congestion of all links. For packet j let q_j be the least congested path in terms of c. We use $\alpha = \sum_j \sum_{e \in q_j} c(e)$ to represent the total congestion of these least congested paths. It can be shown that the dual is equivalent to,

$$\min_c \; rw \cdot D/\alpha.$$

The congestion found at the end of window i by our protocol (see Figure 1) defines a valid solution to this reformulated dual for window i. We exploit this connection to prove Theorem 2. The key here is to bound the total link congestion since the link congestion increases only when a path goes through it. In particular, the following three lemmas show that the total link congestion is no more than 1 at the end of a phase. Let $c_i(e)$, D_i and α_i represent the values of $c(e)$, D and α at the end of the ith window.

Lemma 3 $D_i/\alpha_i \geq 1/rw$ *for* $1 \leq i \leq t$.

Proof: Since the injections are (w, r)-admissible, the primal LP for window i has $\max \lambda \geq 1$. Since the congestion c_i found by our protocol defines a dual solution, our lemma follows from duality. ∎

Lemma 4 $D_i \leq \frac{D_{i-1}}{1 - r\mu}$.

Find routes.
Initialize $c(e) = \delta, \forall e$
for ith window, $i = 1, \ldots, t$
    for each packet injected during ith window
        $p \leftarrow$ least congested route under c
    $c(e) \leftarrow c(e)(1 + N_i(e) \cdot \mu/w)$.

Figure 2. Procedure to find routes for packets injected during one phase with fewer updates.

Proof: It suffices to show $D_i \leq D_{i-1} + \alpha_i \cdot \mu/w$ since $D_i/\alpha_i \geq 1/rw$ by Lemma 3. Let c_{ij} be the congestion function after routing the jth packet injected during the ith window and let D_{ij} be defined in terms of c_{ij}. Suppose path p_j is chosen for the jth packet injected during the ith window. By definition we have,

$$\begin{aligned} D_{ij} &= \sum_e c_{ij}(e) \\ &= \sum_{e \notin p_j} c_{i,j-1}(e) + \sum_{e \in p_j} c_{i,j-1}(e)(1 + \mu/w) \\ &= D_{i,j-1} + \sum_{e \in p_j} c_{i,j-1}(e) \cdot \mu/w. \end{aligned}$$

Now we repeatedly apply the recurrence above. We also observe that the congestion function c only increases. Hence, if q_j is the least congested path for j under c_i then $\sum_{e \in p_j} c_{i,j-1}(e)$ is necessarily no more than $\sum_{e \in q_j} c_i(e)$. (We emphasize that p_j and q_j may be two different paths. The path p_j is least congested with respect to $c_{i,j-1}$ and q_j is least congested with respect to c_i.) We have,

$$\begin{aligned} D_i &= D_{i-1} + \sum_j \sum_{e \in p_j} c_{i,j-1}(e)\mu/w \\ &\leq D_{i-1} + \alpha_i \cdot \mu/w. \end{aligned}$$

■

Lemma 5 $D_t \leq 1$.

Proof: By definition $D_0 = m\delta$ where m is the number of links in the network. By applying Lemma 4, we have,

$$\begin{aligned} D_t &\leq \frac{m\delta}{(1 - r\mu)^t} \\ &= \frac{m\delta}{1 - r\mu}\left(1 + \frac{r\mu}{1 - r\mu}\right)^{t-1} \\ &\leq \frac{m\delta}{1 - r\mu} e^{\frac{r\mu(t-1)}{1-r\mu}} \\ &\leq 1. \end{aligned}$$

The second inequality follows from $1 + x \leq e^x$ for $x \geq 0$. The last inequality follows from the definition of t in (3). ■

We are now ready to prove Theorem 2.

Proof of Theorem 2: Consider any link e. For every w paths routed though e, the congestion of e is increased by a factor at least $1 + \mu$. Initially, $c_0(e) = \delta$. Since $D_t \leq 1$, $c_t(e) \leq 1$. Hence, the total number of paths that are routed through e in a phase is at most $w \log_{1+\mu} 1/\delta$. It suffices to show that this quantity is no more than wtR.

$$\begin{aligned} \frac{w \log_{1+\mu} 1/\delta}{wtR} &\leq \frac{\ln 1/\delta}{\ln(1+\mu)} \cdot \frac{r\mu}{1 - r\mu} \cdot \frac{1}{\ln \frac{1-r\mu}{m\delta}} \cdot \frac{1}{R} \\ &= \frac{r}{R} \cdot \frac{\mu}{\ln(1+\mu)(1 - r\mu)^2} \\ &\leq \frac{r}{R} \cdot (1 - \mu)^{-3} \\ &= 1. \end{aligned}$$

The first inequality and the first equality follow from the definitions of t and δ respectively. The second inequality follows from the fact that $r < 1$ and $\ln(1+\mu) \geq \mu - \mu^2/2$. The last equality follows from the definition of μ. Our proof is complete. ■

2.2 Routing with Less Frequent Updates

In this section we show that Theorem 2 still holds even if the congestion function c is updated less frequently. In particular, we only update the congestion at the end of each window, not for each packet injection. Hence the source nodes only need to communicate with the links at the end of each window. For this new protocol we redefine μ to be

$$\frac{1}{m}\left(1 - \left(\frac{r}{R}\right)^{1/3}\right). \tag{4}$$

Suppose $N_i(e)$ packets are routed through link e during the ith window, then we update $c(e)$ to $c(e)(1 + N_i(e) \cdot \mu/w)$. See Figure 2.

We prove that Theorem 2 remains true. We first show that Lemma 4 still holds. As before, we show $D_i \leq D_{i-1} + \alpha_i \cdot \mu/w$. For any packet j injected during the ith window, let p_j be the path chosen for j.

$$D_i = \sum_e c_i(e)$$

$$
\begin{aligned}
&= \sum_e c_{i-1}(e)(1 + N_i(e) \cdot \mu/w) \\
&= D_{i-1} + \sum_e c_{i-1}(e) N_i(e) \cdot \mu/w \\
&= D_{i-1} + \sum_j \sum_{e \in p_j} c_{i-1}(e) \cdot \mu/w \\
&\leq D_{i-1} + \alpha_i \cdot \mu/w
\end{aligned}
$$

Hence $D_t \leq 1$. Now, for every mw paths routed through e, the congestion on e is increased by a factor at least $1 + m\mu$. Therefore the congestion on any link at the end of a phase is at most,

$$
\begin{aligned}
\frac{mw \log_{1+m\mu} 1/\delta}{wtR} &\leq \frac{\ln 1/\delta}{\ln(1+m\mu)} \cdot \frac{r\mu}{1-r\mu} \cdot \frac{1}{\ln \frac{1-r\mu}{m\delta}} \cdot \frac{1}{R} \\
&= \frac{r}{R} \cdot \frac{m\mu}{\ln(1+m\mu)(1-r\mu)^2} \\
&\leq \frac{r}{R} \cdot (1-m\mu)^{-3} \\
&= 1,
\end{aligned}
$$

with the revised definition of μ in (4).

2.3 Implementation Using In-band Signaling

In the previous sections we assumed that sources can communicate with the links on their chosen routes via instantaneous setup messages. In turn, we also assumed that the links can instantaneously broadcast their congestion to the sources. In this section, we first extend our result in Section 2.2 to the case where each of these communications takes τ time steps. We then give an upper bound on τ for which the communication may be carried out in-band using packets transmitted through the network.

Assume without loss of generality that $w > 2\tau$ (since admissibility for a small window implies admissibility for a large window). Each source only updates the link congestion at the end of every window. Since the congestion does not change during a window, all the packets for a given source-destination pair (s,t) are routed along the *same path* p. At the end of window $[w(i-1), wi)$ a *control packet* is sent along path p that contains the number of (s,t)-packets injected during window $[w(i-1), wi)$. This packet takes time τ to traverse the path. Hence, at time $wi + \tau$, each link can update its congestion due to all the packets injected during $[w(i-1), wi)$. Then by time $wi + 2\tau \leq w(i+1)$ this new congestion can be distributed via control packets to all the sources.

Note that at the end of window $[wi, w(i+1))$, every link has updated its congestion according to the injections in window $[w(i-1), wi)$. The exact form of this update is as follows. Let $N_i(e)$ be the number of packets routed through e that were injected during $[w(i-1), wi)$. Let $c_i(e)$ be the congestion of e at the end of window $[w(i-1), wi)$. We update $c_i(e)$ by,

$$
c_{i+1}(e) = c_i(e) + c_{i-1}(e) N_i(e) \cdot \mu/w,
$$

for

$$
\mu = \frac{1}{2m}\left(1 - \left(\frac{r}{R}\right)^{1/3}\right). \tag{5}
$$

To show that Theorem 2 remains true, we observe,

$$
\begin{aligned}
D_{i+1} &= \sum_e c_{i+1}(e) \\
&= \sum_e c_i(e) + c_{i-1} N_i(e) \cdot \mu/w \\
&= D_i + \sum_e c_{i-1}(e) N_i(e) \cdot \mu/w \\
&= D_i + \sum_j \sum_{e \in p_j} c_{i-1}(e) \cdot \mu/w \\
&\leq D_i + \alpha_{i,i+1} \cdot \mu/w.
\end{aligned}
$$

Here $\alpha_{i,i+1}$ is the sum of the congestion along the paths chosen for packets injected during $[w(i-1), wi)$ with respect to $c_{i+1}(e)$. This is sufficient to imply $D_t \leq 1$. Note also that for every $2mw$ (non-control) packets routed through an link, the congestion function of the link increases by at least a factor $1 + 2m\mu$. The remainder of the analysis follows through for the revised definition of μ in (5).

To ensure that the transmission time of the control packets is upper bounded, the scheduling protocol always gives priority to control packets. Observe that a total of at most $n^2 + mn$ control packets can be sent out during one window, where m is the number of links and n is the number of nodes in the network. If we let $\tau = n^3 + mn^2$, the transmission of a control packet takes at most τ time steps. Without loss of generality we assume that $w \geq 2\tau$ and $w(1-r)/2 \geq n^2 + mn$. The latter condition ensures that together with the control packets the injections are $(w, (1+r)/2)$-admissible.

3 A Scheduling Protocol with Polynomial Delay Bounds

In this section we assume that (w,r)-admissible paths are known (either given by the adversary or computed as in Section 2). Hence, in order to achieve network stability we can use any of the scheduling protocols that are known to be stable for Adversarial Queueing. However, the best previous delay bounds known for distributed, deterministic protocols are exponential in the maximum packet path length. In this section we present a deterministic, distributed scheduling protocol with a polynomial delay bound.

In [2] a randomized protocol was presented for which the delay bound is $O(\frac{d_{\max}}{\varepsilon} \log m)$ with high probability, where

$\varepsilon = 1 - r$. This protocol is hard to derandomize because its success depends on a condition that can only be checked globally. In this section we first present a new randomized protocol and then show how to derandomize it in a distributed manner. The key idea of this protocol is that the conditions that determine the "success" of the protocol only depend on packets that share the same initial link. This allows derandomization in a distributed manner.

Our new randomized protocol is defined in terms of two parameters M and T which are defined below. We partition time into intervals of length M, which we call *M-intervals*. We save up all packets that are injected into the network during each M-interval and then schedule these packets during the next M-interval. We give each packet a deadline for every link on its path. Our goal is to make sure that no more than T packets have a deadline for link e during any time interval of length T. If this condition holds then we are able to bound the end-to-end delay experienced by a packet.

Randomized protocol. For a packet p injected during an M-interval $[(\gamma-1)M, \gamma M)$ for an integral γ, let us suppose its path is $e_0, e_1, \ldots, e_{d_p}$. We define a deadline τ_k^p for p at link e_k as follows. We choose the initial deadline τ_0^p uniformly at random from $[\gamma M + T, (\gamma+1)M - d_{\max}T)$. We then define the remaining deadlines inductively by $\tau_{k+1}^p = \tau_k^p + T$. Our protocol always gives priority to the packet with the smallest deadline at each link. We define M and T such that,

$$T = \frac{36m}{\varepsilon^3}\log(2Mm^2), \qquad (6)$$

$$M \geq \max\left\{\frac{1-\varepsilon/2}{\varepsilon/6}(d_{\max}+1)T, w\right\}. \qquad (7)$$

These properties are satisfied for,

$$M = O\left(\frac{d_{\max}m}{\varepsilon^4}\log\frac{m}{\varepsilon} + w\right).$$

When a packet meets its deadlines, it reaches its destination within $2M$ steps.

Analysis. Our objective is to show that all packets injected during a given M-interval meet all their deadlines with a constant probability. Lemma 6 gives a sufficient condition for all deadlines to be met. For any packet p and link e let $X_{[t,t+T)}^{p,e} = 1$ if e is the kth link on packet p's path and τ_k^p lies in the time interval $[t, t+T)$. Let $X_{[t,t+T)}^{p,e} = 0$ otherwise.

Lemma 6 *If $\sum_p X_{[t,t+T)}^{p,e} \leq T$ for all t and all links e, then all packets meet all their deadlines.*

Proof: Suppose not. Let p be a packet that misses its kth deadline τ_k^p and suppose that no deadline earlier than τ_k^p is missed. Then p has arrived at its kth link e_k by time $\tau_k^p - T$. (This is true regardless of whether e_k is the initial link of p or not.) By our assumption that τ_k^p is the first deadline that is missed, all the packets with deadlines for e_k that are earlier than $\tau_k^p - T + 1$ meet those deadlines. Therefore, the only packets that block packet p in the interval $[\tau_k^p - T + 1, \tau_k^p]$ have deadlines in the interval $[\tau_k^p - T + 1, \tau_k^p]$. By the assumption in the statement of the lemma there are at most $T - 1$ such packets (excluding p). Therefore packet p is served by link e_k at time τ_k^p or earlier. This is a contradiction. ■

Given Lemma 6 we show,

Lemma 7 *Consider packets injected during an M-interval, $[(\gamma-1)M, \gamma M)$. The number of deadlines from these packets on any link e during any interval $[t, t+T)$ is at most T with a constant probability.*

Proof: We use a Chernoff bound to prove the number of deadlines is small. Let $S_{e_0,e}^\gamma$ be the set of packets injected into the network during the interval $[(\gamma-1)M, \gamma M)$ that have e_0 as their initial link and that have link e on their path. The expected number of deadlines is,

$$E\left[\sum_{p \in S_{e_0,e}^\gamma} X_{[t,t+T)}^{p,e}\right] \leq \frac{|S_{e_0,e}^\gamma|}{M - (d_{\max}+1)T}T.$$

When $|S_{e_0,e}^\gamma|$ is large, the expectation is large and the argument is straightforward. However, for small $|S_{e_0,e}^\gamma|$ a direct application of the Chernoff bound may not suffice. To rectify this, let us define a new quantity,

$$\beta_{e_0,e}^\gamma = \frac{M}{M - (d_{\max}+1)T}\max\{|S_{e_0,e}^\gamma|/M, \varepsilon/3m\}.$$

The quantity β has the following properties.

1. $\beta_{e_0,e}^\gamma \geq \varepsilon/3m$;
2. $\sum_{e_0} \beta_{e_0,e}^\gamma \leq \frac{M}{M-(d_{\max}+1)T}((1-\varepsilon) + m\varepsilon/3m) \leq \frac{1-\varepsilon/2}{1-2\varepsilon/3}(1 - 2\varepsilon/3) \leq 1 - \varepsilon/2$.

The second property follows from the requirement of M in (7) and the admissibility of the paths. Our lemma follows if we show that the following holds with constant probability,

$$\sum_{p \in S_{e_0,e}^\gamma} X_{[t,t+T)}^{p,e} \leq (1+\varepsilon/2)\beta_{e_0,e}^\gamma T, \forall e_0, e \text{ and } \forall [t, t+T). \qquad (8)$$

If the above holds, the number of deadlines on link e in the interval $[t, t+T)$ is at most $(1+\varepsilon/2)\sum_{e_0}\beta_{e_0,e}^\gamma T$, which is less than T due to the second property of β. We have,

$$
\begin{aligned}
& Pr\left[\sum_{p\in S^{\gamma}_{e_0,e}} X^{p,e}_{[t,t+T)} > (1+\varepsilon/2)\beta^{\gamma}_{e_0,e}T\right] \\
\leq\ & \frac{\prod_p E[(1+\varepsilon/2)^{X^{p,e}_{[t,t+T)}}]}{(1+\varepsilon/2)^{(1+\varepsilon/2)\beta^{\gamma}_{e_0,e}T}} \qquad (9)\\
\leq\ & \exp(-\varepsilon^2\beta^{\gamma}_{e_0,e}T/12) \\
\leq\ & \frac{1}{2Mm^2}. \qquad (10)
\end{aligned}
$$

The first inequality is due to a Chernoff bound. The second inequality holds since $E[\sum_{p\in S^{\gamma}_{e_0,e}} X^{p,e}_{[t,t+T)}] \leq \beta^{\gamma}_{e_0,e}T$. The third inequality follows from the definition of T in (6) and the fact that $\beta^{\gamma}_{e_0,e} \geq \varepsilon/3m$. By taking a union bound over all links e_0, e and all intervals $[t,t+T) \subseteq [\gamma M,(\gamma+1)M)$, we have that the number of deadlines from all packets on e during $[t,t+T)$ is at most T with probability at least $1/2$. ■

Remarks. To prove Lemma 7 a condition weaker than (8) would be sufficient. It would suffice to show that the number of deadlines on any e during any $[t,t+T)$ is at most $(1+\varepsilon/2)\sum_{e_0}\beta^{\gamma}_{e_0,e}T$. Indeed, this would even allow T and M to be a factor of m smaller, as in [2]. However, such a weaker condition only allows derandomization in a centralized manner.

We emphasize that the condition (8) depends only on sets of packets that are injected into *one particular* initial link. Therefore we can choose the deadlines for a packet simply by considering the other packets that are injected at the same initial link. Hence, we can carry out a derandomization independently at each initial link and obtain a *distributed*, deterministic protocol. This is in contrast to the randomized protocol of [2] in which the success condition depends on packets that are injected across all initial links in the network.

Derandomization. We use the method of conditional expectations to derandomize the protocol for each M-interval. (See e.g. [14].) In summary,

Theorem 8 *Our derandomized protocol is distributed and guarantees a delay bound of $2M = poly(m,w,1/\varepsilon)$ for every packet.*

Proof: Let $S^{\gamma}_{e_0,e} = \{p_0,p_1,\ldots,p_\ell\}$. For $i \leq \ell$, let $g(\delta_0,\delta_1,\ldots,\delta_i)$ be equal to

$$\sum_{e,t} Pr\left[\sum_{p\in S^{\gamma}_{e_0,e}} X^{p,e}_{[t,t+T)} > (1+\varepsilon/2)\beta^{\gamma}_{e_0,e}T | \tau_0^{p_0}=\delta_0,\ldots,\tau_0^{p_i}=\delta_i\right],$$

where t is summed over the range $[\gamma M,(\gamma+1)M-T)$. By a calculation similar to the Chernoff calculation of (9), the value of $g(\cdot,\ldots,\cdot)$ is upper bounded by the following function h,

$$
\begin{aligned}
& h(\delta_0,\delta_1,\ldots,\delta_i) \\
=\ & \sum_{e,t} \frac{\prod_p \exp(\frac{\varepsilon}{2}E[X^{p,e}_{[t,t+T)}|\tau_0^{p_0}=\delta_0,\ldots,\tau_0^{p_i}=\delta_i])}{(1+\varepsilon/2)^{(1+\varepsilon/2)\beta^{\gamma}_{e_0,e}T}}.
\end{aligned}
$$

For fixed $\delta_0,\ldots,\delta_{i-1}$, the definition of conditional expectation implies that there exists an initial deadline δ_i for the packet p_i such that $h(\delta_0,\delta_1,\ldots,\delta_{i-1}) \geq h(\delta_0,\delta_1,\ldots,\delta_{i-1},\delta_i)$. If we always choose the initial deadline so that this inequality is satisfied then,

$$
\begin{aligned}
g(\delta_0,\delta_1,\ldots,\delta_\ell) &\leq h(\delta_0,\delta_1,\ldots,\delta_\ell)) \\
&\leq h(\emptyset) \\
&\leq \exp(-\varepsilon^2\beta^{\gamma}_{e_0,e}T/12),
\end{aligned}
$$

The third inequality follows from (10). We have chosen the parameters M and T so that $\exp(-\varepsilon^2\beta^{\gamma}_{e_0,e}T/12)$ is less than 1. In addition, since $g(\delta_0,\delta_1,\ldots,\delta_\ell)$ involves no randomness every term of g is either 0 or 1. The above inequalities imply that $g(\delta_0,\delta_1,\ldots,\delta_\ell)$ is less than 1 and so condition (8) fails with probability zero. Hence, with probability one all deadlines are met and all packets reach their destinations in time $2M$.

It remains to show that we can calculate $h(\delta_0,\ldots,\delta_i)$. If $j \leq i$ then,

$$E[X^{p_j,e}_{[t,t+T)}|\tau_0^{p_0}=\delta_0,\ldots,\tau_0^{p_i}=\delta_i]$$

is equal to 0 or 1 depending on whether or not the initial deadline δ_j causes packet p_j to have a deadline for link e during $[t,t+T)$. If $j > i$ then,

$$E[X^{p_j,e}_{[t,t+T)}|\tau_0^{p_0}=\delta_0,\ldots,\tau_0^{p_i}=\delta_i] = E[X^{p_j,e}_{[t,t+T)}],$$

which is equal to the probability, over all possible choices of the initial deadline, that packet p_j has a deadline for link e during the interval $[t,t+T)$. (Recall that the initial deadline has at most M choices and all subsequent deadlines are chosen deterministically.) This probability is solely dependent on whether or not the path for packet p_j passes through link e. Hence, for fixed $\delta_0,\ldots,\delta_{i-1}$ we can choose the value of δ_i that minimizes $h(\delta_0,\delta_1,\ldots,\delta_{i-1},\delta_i)$. ■

4 Instability in Combined Routing and Scheduling

In [2] it was shown that if the packet routes are given by the adversary then the FIFO and Nearest-to-Go (NTG) scheduling protocols can be unstable even if the packet paths are admissible. (FIFO always gives priority to the packet

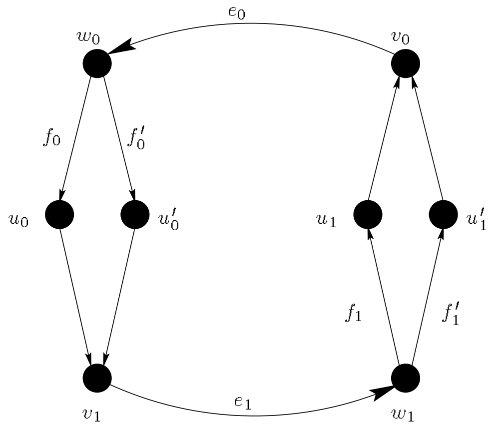

Figure 3. Network G for which FIFO and NTG are unstable even if we are allowed to choose routes.

that arrived at the link earliest. NTG always gives priority to the packet that has the smallest number of hops remaining to its destination.) However, the examples given in [2] do not lead to instability if we are allowed to route packets on paths other than the ones chosen by the adversary.

We therefore have a natural question. If we are allowed to choose the routes, can we guarantee that FIFO and NTG are stable? In this section we show that the answer to this question is negative. We present examples in which regardless of how we choose the routes, the FIFO and NTG scheduling protocols create instability.

Theorem 9 *There exists a network G such that FIFO creates instability under some (w, r)-admissible injections regardless of how packets are routed.*

Proof: Network G is shown in Figure 3. We break the packet injections into phases. We inductively assume that at the beginning of phase j a set S of s packets with destination u_0 is in the queue of e_0. We show that at the beginning of phase $j+1$ more than s packets with destination u_1 are in the queue of e_1. By symmetry this process repeats indefinitely and the number of packets in the network grows without bound. For the basis of the induction, we inject a large burst of packets at source node v_0 with destination node u_0, which is allowed by a large window w. From now on all the injections are at rate r with burst size one. In general the sequence of injections in phase j is as follows.

(1) For the first s steps, we inject a set X of rs packets at node v_0 with destination u_1. These packets are completely held up at e_0 by the packets in S. We also hold up packets in S at f_0 by injecting rs packets at w_0 with destination u_0. These newly injected packets get mixed with those of S into the set S'. At the end of the first s steps, rs packets from S' are at f_0. Note that packets in X will be routed through either f_0 or f'_0.

(2) For the next rs steps, we inject a set Y of r^2s packets at node v_0 with destination u_1. These packets are held up at e_0 by the packets in X. We also inject packets at w_0 with destination u'_0 at rate r. These packets delay the packets from X that are routed through f'_0. Hence, at most $rs/(r+1)$ packets of X cross f'_0. (This only happens if packets in X are routed through f'_0, which is not necessarily the case.) Note that no packet from X crosses f_0 in these steps, since the packets in S' have priority. Hence, at the end of these rs steps, a set $X' \subseteq X$ of at least $r^2s/(r+1)$ packets are still at w_0.

(3) For the next $|X'| + |Y|$ steps the packets in X' and Y move forward, and merge at v_1. Meanwhile, we inject packets at v_1 with destination u_1 at rate r. We end with at least $r(|X'|+|Y|)$ packets at v_1 with destination u_1. This number is at least $r^3s + r^3s/(r+1)$.

This ends phase j. For $r \geq 0.9$ we have $r^3 + r^3/(r+1) > 1$. It is easy to verify that the injections during phase j are admissible. The inductive step is complete. ■

Injections similar to the above can be used to prove the instability of NTG on network G at any rate $r > 1/\sqrt{2}$. The induction hypothesis of phase j now does not require the packets in S to be initially in the queue of e_0, but to cross e_0 in the first s steps of the phase. Hence, subphase (3) is no longer required. Furthermore, after subphase (2) both sets Y and X' contain at least r^2s packets, since single-link injections have higher priority than the packets in X. It follows that the system is unstable since $2r^2s > s$.

5 Stability of a Ring with Parallel Links

In this section we consider source routing on a ring with c parallel links. Consider a decomposition of the network into c disjoint single rings. In the full version of the paper we propose a deterministic on-line source-routing algorithm that routes each packet along one of these rings and guarantees that the routing is admissible. We omit the details here. In [2] it was shown that the single ring is stable under any *greedy* scheduling policy (i.e. one that always schedules a packet whenever packets are waiting). Hence, we conclude that the ring with c parallel links is stable under any greedy scheduling policy if our source-routing algorithm is used.

Note that the 4-ring with 2 parallel links was shown to be unstable under a greedy protocol such as FIFO when the packet paths are given by the adversary [2]. This shows that freedom of routing can make a difference in network stability since we have a network that is unstable under FIFO if the adversary can dictate the routes but is stable under FIFO if we can choose the routes intelligently.

6 Conclusions

In this paper we have presented source routing algorithms for packet-switched networks and we have described the first distributed, deterministic scheduling protocol with a polynomial delay bound. There is much still to be explored in the study of combined routing and scheduling. For example, different packets are often associated with different delay requirements. Some of them may be delay-sensitive whereas others may be delay-tolerant. The problem of scheduling these packets on given routes in order to meet these delay requirements has been studied before. The ability to choose the routes would add an additional dimension to the problem and may even make scheduling easier.

Acknowledgment

The authors wish to thank Adam Meyerson for helpful discussions.

References

[1] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosen. Adaptive packet routing for bursty adversarial traffic. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 359 – 368, Dallas, TX, May 1998.

[2] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results and performance bounds for greedy contention-resolution protocols. *Journal of the ACM*, 48(1):39–69, Jan. 2001.

[3] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 32–40, 1993.

[4] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 321–330, 1994.

[5] B. Awerbuch and T. Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 487–496, 1994.

[6] D. Bertsimas and D. Gamarnik. Asymptotically optimal algorithm for job shop scheduling and packet routing. *Journal of Algorithms*, 33(2):296–318, 1999.

[7] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queueing theory. *Journal of the ACM*, 48(1):13–38, Jan. 2001.

[8] D. Gamarnik. Stability of adversarial queues via fluid models. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 60–70, Palo Alto, CA, November 1998.

[9] D. Gamarnik. Stability of adaptive and non-adaptive packet routing problems in adversarial queueing networks. In *Proceedings of the 31th Annual ACM Symposium on Theory of Computing*, pages 206–214, Atlanta, GA, May 1999.

[10] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 300–309, Palo Alto, CA, November 1998.

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

[12] 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.

[13] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math of Oper. Research*, pages 257–301, 1994.

[14] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130 – 143, 1988.

[15] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, 2001. http://www.ietf.org/rfc/rfc3031.txt.

[16] 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.

[17] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. *IEEE Communications Magazine*, pages 80–86, January 1997.

[18] N. Young. Randomized rounding without solving the linear program. *ACM-SIAM Symposium on Discrete Algorithms*, pages 170–78, 1995.