Universal Stability Results for Greedy Contention-Resolution Protocols*

Matthew Andrews† Baruch Awerbuch‡ Antonio Fernández§ Jon Kleinberg ¶ Tom Leighton‖ Zhiyong Liu**

Abstract

In this paper, we analyze the behavior of communication networks in which packets are generated dynamically at the nodes and routed in discrete time steps across the edges. We focus on a basic adversarial model of packet generation and path determination for which the time–averaged injection rate of packets requiring the use of any edge is limited to be less than 1*. A crucial issue that arises in such a setting is that of* stability *— will the number of packets in the system remain bounded, as the system runs for an arbitrarily long period of time?*

Among other things, we show:

(i) There exist simple greedy protocols that are stable for all networks.

(ii) There exist other commonly–used protocols (such as FIFO) and networks (such as arrays and hypercubes) that are not stable.

(iii) The n–node ring is stable for all greedy routing protocols (with maximum queue-size and packet delay that is linear in n).

(iv) There exists a simple distributed randomized greedy protocol that is stable for all networks and requires only polynomial queue size.

Our results resolve several questions posed by Borodin et al., and provide the first examples of (i) a protocol that is stable for all networks, and (ii) a protocol that is not stable for all networks.

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

†Laboratory for Computer Science, MIT. Supported by NSF contract 9302476-CCR.

‡Department of Computer Science, Johns Hopkins University.

§Laboratory for Computer Science, MIT. On leave from the Dpto. de Arquit. y Tecnol. de Computadores, Univ. Politécnica de Madrid. Supported in part by the Spanish Ministry of Education.

¶Lab Computer Science, MIT. Supported by an ONR Graduate Fellowship. Present address: Department of Computer Science, Cornell University. On leave at IBM Almaden Research Center.

‖Department of Mathematics and Lab for Computer Science, MIT.

**Laboratory for Computer Science, MIT. On leave from Institute of Computing Technology, Academia Sinica, Beijing, China. Supported in part by K. C. Wong Education Foundation, Hong Kong.

1. Introduction

We study the behavior of communication networks in which packets are generated dynamically at the nodes and routed in discrete time steps across the edges. A crucial issue that arises in such a setting is that of *stability* — will the number of packets in the system remain bounded, as the system runs for an arbitrarily long period of time? The answer to this question typically depends on the *rate* at which packets are injected into the system, and on the contention resolution *protocol* that is used when more than one packet wants to cross a given edge in a single time step.

These issues have been investigated in a number of overlapping areas. Within the context of packet routing, there has been recent work focusing on the problem of stability in common interconnection networks; typically, this work assumes that packets are generated according to independent Poisson or Bernoulli processes at the nodes, and they must be routed to random destinations [14, 20, 15, 8, 10, 9, 4, 19, 5]. In related work, Awerbuch and Leighton [1] presented a stable packet-routing algorithm that could be used to provide a local–control approximation for the multicommodity flow problem. The problem of continuous packet injection and routing has also been a major topic of study within the field of queueing theory [12, 11]. Typical assumptions here are that packets are generated according to a Poisson process, and that the time to traverse an edge is an exponentially distributed random variable, rather than a fixed constant. See [2] for a review of previous work on these models.

In this paper, we work within a model of continuous packet injection proposed by Borodin et al. [2], in which probabilistic assumptions are replaced by worst–case inputs. The underlying goal is to determine whether it is feasible to prove stability results even when packets are injected by an *adversary*, rather than an oblivious randomized process. The framework was termed *adversarial queueing theory* in [2], to reflect the fact that the emphasis is on *stability* — the central issue of queueing theory — with respect to an *adversarial* model of packet generation and path determination.

The model of [2] considers the time evolution of a packet–routing network as a game between an *adversary* and a *protocol*. In each time step, the adversary *injects* a set of packets at some of the nodes; for each packet it specifies a sequence of edges that it must traverse, after which the packet will be *absorbed*. If more than one packet wishes to cross an edge e in the current time step, then the *protocol* chooses one of these packets to send across e; the remainder of these packets wait in a *queue* at the tail of e. This game then advances to the next time step. The protocol is said to be *stable* against the adversary if there is a constant C (possibly depending on the underlying network) so that there are never more than C unabsorbed packets in the system, regardless of how long the game is played. In this paper, as in [2], we will only consider *greedy* protocols — those that advance a packet across an edge e whenever there is at least one packet waiting to use e.

A crucial parameter of the adversary is its *rate*. Borodin et al. defined a single *request* by the adversary to be a set of packets requesting edge–disjoint paths; in their terminology, an adversary injects at *rate* r if for all t, no more than $\lceil rt \rceil$ requests are made in any interval of t steps. Among other results, Borodin et al. showed that against any adversary with rate at most 1, (i) any greedy protocol is stable on any DAG, and (ii) the Farthest-to-Go protocol is stable on the ring.

A different but related model of worst-case packet injection was proposed in earlier work of Cruz [6, 7]. In this model, one assumes that packets are injected by k *sessions*, each with a fixed path and a fixed rate (with some *burstiness* allowed), subject to the requirement that the total rate of all sessions using a given edge is strictly less than 1. Cruz proves the stability of every greedy protocol on every layered DAG [7]; Tassiulas and Georgiadis [21] also work within this model, and show that every greedy protocol on the ring is stable. Any set of k sessions in Cruz's model corresponds to an adversary of rate strictly less than 1 in the model of Borodin et al.; hence a stability result in the latter model implies an analogous result in the former. However, the converse direction does not hold: there exist adversaries in the model of Borodin et al. that cannot be captured by the framework of Cruz. (For example, one needs the more general model of [2] to represent connections of limited duration.)

In recent related work, Rabani and Tardos [16] developed a randomized algorithm for static routing problems, and applied it to dynamic problems. They obtain an algorithm with the following performance guarantee in the model of Borodin et al.: against any adversary of rate strictly less than 1 (with some burstiness allowed), every packet is absorbed in a polynomial number of steps with high probability. Their algorithm is different from those we consider here in that it is not greedy and it allows packets to be *discarded*: the algorithm is allowed to pre-emptively remove a packet from the system with inverse polynomial probability.

A number of fundamental open questions were raised in [2] concerning the relationship between rate and stability. In particular they asked,

(i) Is *any* greedy protocol stable against every adversary of rate less than 1, for every network?
(ii) Is *any* greedy protocol stable with small queue size against every adversary of rate less than 1, for every network?
(iii) Does the n–node unidirectional ring have the property that every greedy protocol is stable against every adversary of rate less than 1?
(iv) Does *every* network have this property (namely that every greedy protocol is stable against every adversary of rate less than 1?)

These questions highlight a very basic algorithmic question: when is a given contention resolution protocol stable in a given network, against a given adversary? More specifically, the questions are based on the following definitions, which will be central to the work we do here.

Definition 1.1 *We say that a* graph G *is* universally stable *if every greedy protocol is stable against every adversary of rate less than* 1 *on* G.

Definition 1.2 *We say that* protocol $\mathcal{P}$ *is* universally stable *if it is stable against every adversary of rate less than* 1*, on every network.*

We noted above that [2] showed directed acyclic graphs to be universally stable; but for graphs with directed cycles, the following two extremes were both left open as possibilities: (a) every graph is universally stable; or (b) no graph containing a directed cycle is universally stable. It was also left open whether or not any or all greedy protocols are universally stable.

1.1. Our results

In this paper, we resolve the open questions of [2] described above, and provide additional results on stability. In particular, we show that

(i) There exist commonly-used simple greedy protocols that are universally stable.
(ii) There exists a simple distributed randomized greedy protocol that is universally stable with a bound on queue size that is polynomial in $d \log m$, where d is the maximum path length and m is the number of edges in the network. For many common networks, this bound is therefore polylogarithmic.
(iii) The n–node ring is universally stable, with maximum queue-size and maximum delay that is linear (in n).
(iv) There exist commonly–used graphs and protocols that are not universally stable.

Our universal stability results (i), (ii), and (iii) hold for a broader class of *bounded adversaries*, which we define as follows. The rate of an adversary in our work will be specified by a pair (w, r), where w is a natural number and $0 < r < 1$. The requirement on the adversary is the following: of the packets that the adversary injects in any interval of w steps, at most

rw can have paths that contain any one edge. Such a model allows for adversarial injection patterns that are "bursty," since our rate restriction holds only in an amortized sense. In one time step, an adversary can inject a large number of packets that all request the same edge, provided simply that this does not result in more than rw packets requesting this edge over an interval of w steps.

We first show that several natural protocols are universally stable; we will refer to them as *Farthest-to-Go* (FTG), *Longest-in-System* (LIS), and *Shortest-in-System* (SIS). FTG gives precedence to a packet whose distance to its destination is maximal; LIS gives precedence to the packet injected the earliest; and SIS gives precedence to the packet most recently injected.

Although these protocols are stable, we show that two of them (FTG and SIS) can require queues of exponential size in the worst case. For the third protocol, LIS, the best upper bound on queue size that we can show is exponential, though we do not know of a matching lower bound. Thus it is natural to ask whether there exists a protocol with queues of polynomial size. We show that there is a simple distributed randomized protocol with a polynomial bound on queue size; as is standard, we say that a randomized algorithm in this setting is polynomially bounded if the probability of its having a large queue at any point in time is exponentially small. Our algorithm is based on the Longest-in-System priority rule, with random perturbations, and it has a very simple local-control implementation.

Our examples of instability (result (iv) above) hold even for an adversary that injects at most one set of disjoint paths in each time step. The greedy protocols that turn out to be unstable are FIFO, LIFO, and *Nearest–to-Go* (NTG). FIFO and LIFO maintain the edge queues in First-in-First-out and Last-in-First-out order respectively; the NTG protocol always advances a packet whose distance to its destination is minimal. The FIFO protocol is widely used and NTG has at times been proposed for routing in array–based parallel machines. Curiously, we show that these protocols can be unstable on many common networks, including arrays and hypercubes.

We mentioned above that universal stability is a very basic algorithmic problem for graphs; hence we have the following natural question: Given a graph G, is it universally stable? Our results (iii) and (iv) above show the non–triviality of universal stability as a property; and it is not initially clear that it should even be a *decidable* property, since it is asking whether *every* greedy protocol is stable against *every* bounded adversary on G. For undirected graphs with bi–directional edges, however, we show that universal stability is a decidable property; and in fact it can be decided in polynomial time. To prove this, we show that the set of universally stable graphs is closed under the taking of minors; polynomial–time decidability then follows from results of Robertson and Seymour [17, 18].

1.2. Preliminaries

It is straightforward to formalize the model we have been discussing above. In every time step t, the current *configuration* $\mathcal{C}^t$ of the system is a collection of sets $\{S_e^t : e \in G\}$, such that S_e^t is the set of packets waiting in the queue for e at the end of step t. From the configuration $\mathcal{C}^t$, we obtain the configuration $\mathcal{C}^{t+1}$ for the next time step as follows. (1) We add new packets to some of the sets S_e^t, each of which has an assigned path in G; and (2) for each non-empty set S_e^t, we delete a single packet $p \in S_e^t$ (as specified by a contention–resolution protocol) and insert it into the set S_f^{t+1}, where f is the edge following e on its assigned path. (If e is the last edge on the path of p, then p is not inserted into any set.) A *time–evolution* of G, of rate (w, r), is simply a sequence of such configurations $\mathcal{C}^1, \mathcal{C}^2, \ldots$, such that for all edges e and all intervals I of w consecutive steps, no more than rw packets are introduced during I with an assigned path containing e.

In this version of the paper, however, we prefer to keep the definitions slightly informal for the sake of readability. We therefore will phrase results in terms of an *adversary* that adds packets to the system, and a *protocol* that moves packets across edges. By the *system* $(G, \mathcal{A}, \mathcal{P})$ we simply mean the time–evolution of G induced by adversary $\mathcal{A}$ and protocol $\mathcal{P}$. We view each time step t of this system as consisting of three phases.

(i) Packets are injected by $\mathcal{A}$.

(ii) Packets are moved by $\mathcal{P}$.

(iii) Packets that reach their destinations in phase (ii) are absorbed.

Finally, we give some additional definitions.

Definition 1.3 *A packet is said to* require *an edge e at time t if e lies on the path from its position at time t to its destination.*

For simplicity in this version, we will assume that when a packet is injected, its assigned path is *simple*; namely, it does not contain any edge more than once. It is not difficult, however, to remove this assumption.

Definition 1.4 *We say that $\mathcal{A}$ is a* bounded adversary, *of rate (w, r), if for all e and all intervals I of w consecutive steps, it injects no more than rw packets during I that require e at their time of injection.*

In the sequel we shall speak in terms of both the maximum number of packets in the system and the maximum delay (i.e. time until absorption). The following result shows that there is a close relation between the two quantities; we omit the proof, which is not difficult.

Theorem 1.5 *Let $(G, \mathcal{A}, \mathcal{P})$ be a system, where G is a graph with m edges, $\mathcal{P}$ is any greedy protocol, and $\mathcal{A}$ is an adversary of rate $(w, 1-\varepsilon)$, $0 < \varepsilon < 1$. Suppose there are never more than k packets in the system at the same time, where $k \geq w$.*

Then any packet which is injected with a path of length d will be absorbed in at most $2kd\varepsilon^{-1}$ steps. Conversely, if the maximum delay experienced by any packet is at most r, then there are never more than mr packets in the system at the same time.

2. Universal stability of protocols

In this section we focus on the issue of universal stability for *protocols*: given a contention resolution protocol $\mathcal{P}$, is it stable on every network G, against every bounded adversary $\mathcal{A}$? We first present three simple protocols for which the answer is affirmative. Our upper bounds for all these protocols are exponential in the maximum path length d; thus, while the bounds are large in general, they are fairly good when all packets require only short paths. (In Section 4.2, we will present a randomized protocol with a bound on queue size that is polynomial in $d \log m$.) After our upper bounds, we show in Section 2.2 that several simple and very common protocols are not universally stable.

2.1. Universally stable protocols

SIS is universally stable

Theorem 2.1 *Let G be a directed network, and $\mathcal{A}$ a bounded adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. Then the system $(G, \mathcal{A}, SIS)$ is stable.*

Proof. Let p be a packet in the system $(G, \mathcal{A}, SIS)$ at time step t, waiting in a queue at the tail of edge e, and suppose that there are currently k other packets in the system requiring e that have priority over p. Let $k' > k$ be a multiple of w. We claim that p will cross e within the next $\frac{k'}{\varepsilon}$ steps. For if not, then a distinct packet crosses e in each of the next $\frac{k'}{\varepsilon}$ steps. But any packet in the system during this time that has priority over p, and requires edge e, must either be one of the k packets existing at time t, or one of the (at most) $\frac{k'}{\varepsilon}(1-\varepsilon)$ packets requiring e that were injected during this time. Thus at most $k + \frac{k'}{\varepsilon}(1-\varepsilon) < \frac{k'}{\varepsilon}$ have priority over p during this time, a contradiction.

We define $\beta = w(1-\varepsilon)$, and numbers $k_1, k_2, \ldots$ by the recurrence $k_1 = \beta$, $k_{j+1} = \varepsilon^{-1}k_j + \beta\varepsilon^{-1}$. Now, by induction, we claim that when p arrives at the queue of the j^{th} edge e_j on its path, the following holds: for every edge e on the path of p, there are at most k_j packets requiring e with priority over p. This holds for $j = 1$, since for any edge e, the only packets requiring e that initially could have priority over p are the (at most) $\beta = w(1-\varepsilon)$ packets injected in the same time step as p. Now, suppose that the claim holds for some j. Then by the above argument, p will arrive at the tail of e_{j+1} in another $\frac{w+k_j}{\varepsilon}$ steps, during which time at most another $(1-\varepsilon)\frac{w+k_j}{\varepsilon}$ packets requiring any edge e arrive with priority over p. Thus, when p arrives at the tail of e_{j+1}, at most

$$k_j + (1-\varepsilon)\frac{w+k_j}{\varepsilon} = k_{j+1}$$

packets requiring an edge e have priority over p, and hence the claim holds.

Finally, let m be the number of edges and d the length of the longest simple directed path in G, we claim that there are at most $m(k_d+1)$ packets ever in the system. For if there were ever $m(k_d+1)+1$ packets, then there would be a set of k_d+2 packets all requiring the same edge; the one of these with the lowest priority would contradict the claim of the previous paragraph. ∎

LIS is universally stable

Let us denote as *class* l the set of packets injected in step l. A class l is said to be *active* at the end of step t if and only if at that time there is some packet in the system of class $l' \leq l$. Consider now some packet p, injected at time T_0, and whose path crosses edges $e_1, e_2, \ldots, e_d$, in this order. We use T_i to denote the step in which p crosses edge e_i, and t to denote some step in $[T_0, T_d)$. Let c_t denote the number of active classes at the end of step t, and define $c = \max_{t \in [T_0, T_d)} c_t$.

Lemma 2.2 $T_d - T_0 < (c+w)(1-\varepsilon^d) + \frac{1-\varepsilon^d}{1-\varepsilon}$.

Proof. The packet p reaches the tail of edge e_i at time T_{i-1}. Since p is still in the system, all classes in $[T_0, T_{i-1}]$ are active at the end of that step. Thus, from the definition of c, there are at most $c - (T_{i-1} - T_0)$ active classes of packets that can block p in the queue of e_i. All the packets in these classes were injected in consecutive steps, and hence at most $\lceil (c + T_0 - T_{i-1})/w \rceil (1-\varepsilon)w < (1-\varepsilon)(c + T_0 - T_{i-1} + w)$ packets can block p. Therefore,

$$T_i < \varepsilon T_{i-1} + (1-\varepsilon)(c + w + T_0) + 1.$$

Thus, solving the recurrence, we obtain

$$\begin{aligned} T_d &< ((1-\varepsilon)(c+w+T_0)+1)\sum_{i=0}^{d-1}\varepsilon^i + \varepsilon^d T_0 \\ &= (c+w)(1-\varepsilon^d) + \frac{1-\varepsilon^d}{1-\varepsilon} + T_0 \end{aligned}$$

and the claim follows. ∎

Theorem 2.3 *There are never more than $\frac{w+1}{\varepsilon^d(1-\varepsilon)}$ active classes in the system $(G, \mathcal{A}, LIS)$, where d is the length of the longest simple directed path in G.*

Proof. Let $c = \frac{w+1}{\varepsilon^d(1-\varepsilon)}$ and assume that the end of step t is the first time there are exactly $c+1$ active classes. Hence, at the end of step t there are packets that have been in the system for $c+1$ steps, and during the first c of these steps no more than c classes were active.

However, from the above lemma, any packet that has at most c active classes while in the system (except, maybe, the last step), is absorbed in at most

$$\begin{aligned}&(c+w)(1-\varepsilon^d)+\frac{1-\varepsilon^d}{1-\varepsilon}+1\\&\quad= c+1-\frac{w+1-(1-\varepsilon^d)(w(1-\varepsilon)+1)}{1-\varepsilon}\end{aligned}$$

steps. Since $1-\varepsilon^d < 1$ and $1-\varepsilon < 1$, the last term is negative, the number of steps is smaller than $c+1$, and we reach a contradiction. ■

Corollary 2.4 *Let G be a directed network, and $\mathcal{A}$ an adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. Then, the system $(G, \mathcal{A}, LIS)$ is stable, there are never more than $O(\frac{wm}{\varepsilon^d})$ packets in the system and the maximum number of steps any packet spends in the system is $O(\frac{w}{\varepsilon^d})$, where d is the length of the longest simple directed path in G.*

FTG is universally stable

Theorem 2.5 *Let G be a directed network, and $\mathcal{A}$ a bounded adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. Then the system $(G, \mathcal{A}, FTG)$ is stable.*

Proof. We prove by a backwards induction that this protocol is stable. Let m be the number of edges and d be the length of the longest simple directed path in the graph G. Let us define $k_i = 0$ for $i > d$ and $k_i = m\sum_{j>i} k_j + mw(1-\varepsilon)$ for $1 \le i \le d$. We claim that for all $j > i$ the number of packets in the system that still have to cross *exactly* j edges is at most k_j.

This is trivial for $j > d$ since each packet has to cross at most d edges. Now consider a particular edge e and let $X_i(t)$ be the set of packets in the queue of e that still have to cross *at least* i edges at time t. Let t be the current time, and let t' be the most recent time step preceding t in which $X_i(t')$ was empty. Any packet in $X_i(t)$ must either have had at least $i+1$ edges to cross at time t' or else it must have been injected after time t'. But, from the definition of the protocol, at every step t'' between times t' and t a packet from $X_i(t'')$ must have been chosen to cross edge e. Hence, by the inductive hypothesis,

$$\begin{aligned}|X_i(t)| &\le \sum_{j>i} k_j + \left\lceil \frac{t-t'}{w} \right\rceil w(1-\varepsilon) - (t-t')\\&\le \sum_{j>i} k_j + w(1-\varepsilon) - \varepsilon(t-t').\end{aligned}$$

The above inequalities have two consequences. First, the number of packets in the system that still have to cross i edges is always at most $m\sum_{j>i} k_j + mw(1-\varepsilon) = k_i$ and so the inductive step holds. Second, $t-t'$ cannot be greater than $\frac{1}{\varepsilon}(\sum_{j>i} k_j + w(1-\varepsilon))$. Hence this expression gives the maximum amount of time that a packet with i edges still to cross can remain in a queue. Therefore under FTG the maximum number of packets in the system is bounded by $\sum_{j\ge 1} k_j$ and the maximum amount of time that any packet spends in the system is bounded by k_1/ε. ■

2.2. Protocols that are not universally stable

In this section we show the instability of commonly-used protocols (namely, FIFO, NTG, and LIFO) on simple networks, thus proving that these protocols are not universally stable.

For lower bounds of the type we are interested in obtaining in this section, it is advantageous to have an adversary that is as weak as possible. Thus, for the purposes of this section, we say that an adversary $\mathcal{A}$ has rate r if for every $t \ge 1$, every interval I of t steps, and every edge e, it injects no more than $\lceil rt \rceil$ packets during I that require e at the time of their injection.

We will present our lower bounds for systems that start from a non-empty initial configuration. This implies instability results for systems with an empty initial configuration, by the following simple lemma.

Lemma 2.6 *Let G be a graph, $\mathcal{P}$ be a greedy protocol, and $\mathcal{A}$ an adversary of rate r, and suppose the system $(G, \mathcal{A}, \mathcal{P})$ is unstable starting with some non–empty initial configuration. Then, there exists a system $(G', \mathcal{A}', \mathcal{P})$ that is unstable starting with an empty initial configuration, where $\mathcal{A}'$ is an adversary of rate r.*

Now, we define the graph G to be a four–node directed cycle, with vertices v_0, w_0, v_1, w_1, and two parallel edges between w_i and v_{1-i}. (G has edges e_i from v_i to w_i, and edges f_i, f_i' from w_i to v_{1-i}.)

Theorem 2.7 *Let $r \ge 0.85$. There is an adversary $\mathcal{A}$ of rate r such that $(G, \mathcal{A}, FIFO)$ is unstable, starting from a non–empty initial configuration.*

Proof. We break the construction of $\mathcal{A}$ into phases. Our induction hypothesis will be as follows: at the beginning of phase j, there will be j packets in the queue of e_i requiring to cross edges e_i and f_i, for $i = 0$ or 1 (depending on whether j is even or odd).

To start out, however, we need s_0 packets queued at node v_0, for a large enough constant s_0. Thus the induction hypothesis for phase 0 is certainly met. For a general phase j (suppose j is even), we will show that if at the beginning of j the queue of e_0 contains a set S of s packets requiring edges $e_0 f_0$, then at the start of phase $j+1$, there will be at least $s+1$ packets in the queue of e_1 requiring edges $e_1 f_1$.

The sequence of injections in phase j is as follows. For simplicity, we will omit floors and ceilings; by carrying these through the computations one loses some additive constants, which are offset by the fact that s_0 was a large constant.

(1) For the first s steps, we inject a set X of rs packets that want to traverse edges $e_0 f'_0 e_1 f_1$. These are blocked by the packets in S.

We also hold up the sequence of packets in S at the tail of f_0, using single–edge injections. The newly injected packets get mixed with those of S into the set S'. However, these single–edge injections can only be made at rate r, and so the number of packets of S' that are queued at the tail of f_0 at the end of the first s steps is only rs.

(2) For the next rs steps, we inject a set Y of $r^2 s$ packets that want to traverse edges $f_0 e_1 f_1$. These are blocked by the packets in S'.

We also delay the flow of packets in X though f'_0, using single–edge injections. The new packets get mixed with the packets in X. In the process, $rs/(r+1)$ packets of X cross f'_0 and the size of X shrinks to $r^2 s/(r+1)$.

(3) For the next $r^2 s$ steps the packets in X and Y move forward, and merge at v_1. At the same time, $r^3 s$ new packets that want to traverse edges $e_1 f_1$ are injected in v_1. Since $r^2 s$ packets cross e_1, after these $r^2 s$ steps the queue of e_1 contains $r^3 s + r^2 s/(r+1)$ packets. This ends phase j. Since $r^3 + r^2/(r+1) > 1$, we meet the induction hypothesis for phase $j+1$. ■

An adversary similar to the one described above can be used to prove the instability of the NTG protocol on the network G at any rate $r > 1/\sqrt{2}$. By slightly modifying the network, one can also show the instability of the LIFO protocol. Moreover, all these constructions can be generalized to larger ring-based networks; the resulting systems show that FIFO can be made unstable at any rate $r \geq 0.798$, and that NTG and LIFO can be made unstable at any rate $r > 0.62$.

We will show in Section 3.2 that these instability results also hold for any network that topologically contains any of the graphs used here. This includes k–dimensional arrays, hypercubes, and most other common networks except trees and cycles. As a consequence, we can conclude that FIFO, NTG, and LIFO are unstable for all these networks.

3. Universal stability of networks

We now consider the universal stability of networks. We begin our study with the case of the n-node ring, since it is situated between the class of directed acyclic graphs — which are known to be universally stable by a result of [2] — and the simple cyclic graphs of Section 2.2, which are not universally stable. Thus, it is natural to ask whether there is any universally stable network that contains a directed cycle. In what follows, we establish that cyclicity itself is not the obstacle, by showing that the ring is universally stable.

A natural next question is whether one can characterize the set of universally stable graphs. We show that for undirected graphs, there is a polynomial-time algorithm that decides universal stability.

3.1. The ring is universally stable

Let G denote the n-node ring. We use the numbers $1, \ldots, n$ to denote the edges, and v_i to denote the queue at the tail of edge i. Fix arbitrary $w > 0$ and $\varepsilon > 0$. Our goal is to show that any greedy queueing discipline $\mathcal{P}$ is stable against any adversary of rate $(w, 1-\varepsilon)$.

We begin by developing some general facts about the behavior of $\mathcal{P}$. Let us consider some packet p in the system $(G, \mathcal{A}, \mathcal{P})$. We suppose it was injected in step T_0, at node v_{i_0}, with destination n. Let T' be some time at which it has not yet been absorbed, with $T' - T_0 > w$. (If there is no packet for which such times T_0, T' exist, then all packets are absorbed within w steps, and the system is clearly stable.) Let $v_{i_0}, \ldots, v_{i_0+r}$ be the nodes through which p passes in the interval $[T_0, T']$. We write $i_k = i_0 + k$. For $k = 0, \ldots, r$, let T_k denote the time at which p first reaches v_{i_k} (i.e. when it crosses edge $i_0 + k - 1$, if $k > 0$); by abuse of notation, we will also write $T_{r+1} = T'$.

By the definition of the edges $i_0, \ldots, i_r$, we have

Lemma 3.1 *For each $k = 0, \ldots, r$, and each $t \in [T_k, T_{k+1}]$, some packet crosses edge i_k in step t.*

For j an edge of G, and $t \in [T_0, T']$, we define $P_{j,t}$ to be the number of packets in G at the end of step t that require edge j. Note the following basic property of $P_{j,t}$.

Lemma 3.2 *Let t and t' be such that $t' \leq t \leq t' + w$. Then $P_{j,t} \leq P_{j,t'} + w(1-\varepsilon) - z$, where z is the number of packets that cross edge j in the interval $(t', t]$.*

We define $Q = \max_{j \in G,\, t \in [T_0, T')} P_{j,t}$. For j and t as before, we now define the function $f(j,t) = Q - \varepsilon(t - T_0) + w(1 + j - i_0)$. We note the following properties of this function f.

Lemma 3.3 *(i)* $f(j, t+1) = f(j,t) - \varepsilon$.
(ii) $f(j+1, t) = f(j,t) + w$.

Definition 3.4 *If j is an edge of G and t is a time step, we say that the pair (j,t) is* applicable *if either*
- *$j = i_k$ for some k, and $t \in [T_0, T_{k+1}]$, or*
- *$j > i_r$ and $t \in [T_0, T']$.*

Note the following basic property of applicability.

Lemma 3.5 *If (j,t) is applicable and $(j-1, t)$ is not, then $j = i_k$ for some k, and $t \in (T_k, T_{k+1}]$.*

The crux of our analysis is the following lemma.

Lemma 3.6 *For all applicable pairs (j,t), we have $P_{j,t} \leq f(j,t)$.*

Proof. We prove the lemma by induction on $j \geq i_0$, and for fixed j by induction on t. First, the basis of the induction for any fixed j is easily proved as follows: if (j,t) is applicable, and $t \leq T_0 + w$, then $f(j,t) \geq Q - \varepsilon w + w \geq Q$, and by assumption we have $P_{j,t} \leq Q$ for all $t \in [T_0, T_0 + w] \subseteq [T_0, T')$.

Now, consider any applicable pair (j,t), with $t > T_0 + w$. If for the past w consecutive steps, a packet has crossed edge j in each step, then by Lemmas 3.2 and 3.3, and the induction hypothesis, we have

$$\begin{aligned} P_{j,t} &\leq P_{j,t-w} + w(1-\varepsilon) - w \\ &\leq P_{j,t-w} - \varepsilon w \leq f(j,t-w) - \varepsilon w = f(j,t). \end{aligned}$$

Otherwise, there is a step $t' \in (t-w, t]$ in which no packet crosses edge j. Note that the pair (j,t') is applicable. We claim that in this case the pair $(j-1,t')$ is also applicable. For suppose not; then by Lemma 3.5, $j = i_k$ for some k, and $t' \in (T_k, T_{k+1}]$ — but this contradicts Lemma 3.1. Thus, $(j-1,t')$ is applicable, and so is $(j-1,t'-1)$. Since $\mathcal{P}$ is greedy, the node v_j is empty at the end of step $t'-1$, and hence $P_{j-1,t'-1} \geq P_{j,t'-1}$. Again applying Lemmas 3.2 and 3.3, and the induction hypothesis, we have

$$\begin{aligned} P_{j,t} &\leq P_{j,t'-1} + w(1-\varepsilon) \leq P_{j-1,t'-1} + w(1-\varepsilon) \\ &\leq f(j-1,t'-1) + w(1-\varepsilon) = f(j,t'-1) - \varepsilon w \\ &\leq f(j,t'-1) - \varepsilon(t-t'+1) = f(j,t). \quad \blacksquare \end{aligned}$$

Using this lemma, we now prove the main two results of this section.

Theorem 3.7 *$(G, \mathcal{A}, \mathcal{P})$ is stable, and there are never more than $\frac{wn}{\varepsilon}$ packets in the system that require any given edge.*

Proof. The second statement implies the first, so we will concentrate on proving the second statement. Set $Q = \frac{wn}{\varepsilon}$, and suppose that the theorem is not true. Let T' be the first time at which $Q+1$ packets in the system require a given edge, say edge n; let $T_0 < T'$ denote the time at which the first of these was injected. Note that, from Definition 1.4, in $T' - T_0$ steps at most $\lceil (T'-T_0)/w \rceil (1-\varepsilon)w$ packets can be injected requiring any edge. Therefore, $Q \leq \lceil (T'-T_0)/w \rceil (1-\varepsilon)w \leq ((T'-T_0)/w + 1)(1-\varepsilon)w$, and hence

$$T' - T_0 \geq Q/(1-\varepsilon) - w \geq Q + \varepsilon Q - w = Q + nw - w \geq Q.$$

By our assumption we have $Q < P_{n,T'}$, and by Lemma 3.6, we have $P_{n,T'} \leq f(n,T')$. But since $T' - T_0 \geq Q$, we have

$$f(n,T') = Q - \varepsilon(T'-T_0) + w(1+n-i_0) \leq Q - \varepsilon Q + wn = Q,$$

a contradiction. $\blacksquare$

The proof of the following theorem is analogous, and we omit it.

Theorem 3.8 *The maximum number of steps a packet spends in the system is $O(wn\varepsilon^{-2})$.*

3.2. Deciding universal stability of networks

In Sections 3.1 and 2.2, we have seen that the property of universal stability holds for some graphs, but not for others. We therefore turn to the problem of characterizing those graphs which are universally stable. Initially, it is not at all clear that universal stability should be a decidable property, since we are implicitly quantifying over all adversaries and all protocols. However, we can show that the property of universal stability is closed with respect to the *minor inclusion* relation on graphs; the proof is somewhat lengthy, and we omit it in this version. By results of Robertson and Seymour, this fact implies that the set of graphs that are not universally stable has only finitely many minor-minimal elements. Moreover, the results of Section 2.2 imply that not all planar graphs are universally stable; by [17, 18], we have the following.

Theorem 3.9 *There is an algorithm with running time $O(n^2)$ that decides if a graph is universally stable.*

4. Bounds on queue size for universally stable protocols

The maximum *queue size* required by a queueing protocol is one of the main parameters determining its performance. The issue of *stability* asks whether this parameter can become unbounded; but among universally stable protocols, it is important to identify those that maintain the smallest possible queues. Ideally we would like to have a protocol that never holds more than a constant number of packets in any queue. However, since we are dealing with adversarial packet injection, it is easy to construct examples of networks and adversaries for which any greedy protocol will require queues of super-constant size.

Now, it is interesting to observe that for all three of the universally stable protocols presented in Section 2.1, we have only been able to show exponential upper bounds on the maximum queue size. In this section we show that two of the protocols presented there actually *require* exponential queue size, for some network G and some adversary $\mathcal{A}$.

In Section 4.2, we then present a simple dsitributed randomized greedy protocol that requires only polynomially bounded queues, with high probability.

4.1. SIS and FTG require exponential queue size

We now show that under the protocols SIS and FTG the queue sizes can become exponential. We present a proof of this result for SIS; the proof for FTG is very similar.

In order to make the result more general, we use the type of adversary considered in Section 2.2. We say that an adversary $\mathcal{A}$ has rate $1-\varepsilon$, if for every $t \geq 1$, every interval I of t steps, and every edge e, $\mathcal{A}$ injects no more than $\lceil (1-\varepsilon)t \rceil$ packets during I that require e at the time of injection.

Consider first the linear array L with $m+2$ nodes $0, 1, \ldots, m+1$, with two parallel edges e_i^0 and e_i^1 from node i to node $i+1$, for $0 \leq i \leq m-1$, and with an edge e_m from node m to node $m+1$. Choose an $\varepsilon \leq 1/(m+2)$ and an $s \geq 2m+1$, and construct a tree T such that an adversary $\mathcal{A}$ with rate $1-\varepsilon$ can inject $(1-\varepsilon)s$ packets at the leaves of T during an interval of s steps and they all reach the root of T in the last step of the interval. T can have $O(m^2)$ edges. The graph G is obtained by connecting L and T, making the node 0 of L the root of T.

We now construct an adversary $\mathcal{A}$ with rate $1-\varepsilon$ that injects packets in phases of s steps each. We number the 2^m first phases from 0 to 2^m-1. For some fixed $i \in \{0, \ldots, 2^m-1\}$, let $b_{m-1} \ldots b_0$ be the m-bit binary representation of i. Then, in phase i the adversary injects $(1-\varepsilon)s$ packets at the leaves of the subgraph T of G, all requiring edges $e_0^{b_0} e_1^{b_1} \ldots e_{m-1}^{b_{m-1}} e_m$, so that all of them reach node 0 in the last step of phase i. It also injects $(1-\varepsilon)s$ packets requiring only edge $e_j^{\bar{b}_j}$, for all $0 \leq j \leq m-1$.

Let us define $k_0 = (1-\varepsilon)s$, and $k_j = 2k_{j-1} - \varepsilon s 2^{j-1}$ for $1 \leq j \leq m$. The crucial fact is the following; the proof is by induction on j, and we omit it.

Lemma 4.1 *For all $j \in \{0, \ldots, m\}$, let $i_j \in \{0, 1, \ldots, 2^{m-j}-1\}$ and $b_{m-j-1} \ldots b_0$ be the $(m-j)$-bit binary representation of i_j. Then, at the end of phase $2^j(i_j+1)-1$ there are at least k_j packets in the system $(G, \mathcal{A}, SIS)$ still requiring edges $e_j^{b_0} e_{j+1}^{b_1} \ldots e_{m-1}^{b_{m-j-1}} e_m$. All these packets are in nodes of the subgraph L of G.*

Theorem 4.2 *At the end of phase 2^m-1 there are at least $(2m+1)2^{m-1}$ packets in the system $(G, \mathcal{A}, SIS)$ requiring edge e_m, and there are at least 2^{m-1} packets in some queue of the system.*

Proof. From Lemma 4.1 with $j = m$ and $i_j = 0$, at the end of phase 2^m-1 there are at least k_m packets in the nodes of L requiring edge e_m. Then, the theorem follows, since $k_m = 2^m k_0 - m\varepsilon s 2^{m-1} = s2^{m-1}(2-\varepsilon(m+2)) \geq (2m+1)2^{m-1}$. There are only $2m+1$ queues where these packets can be held, hence some queue contains at least 2^{m-1} packets. ∎

Note that this construction for SIS uses an adversary of rate $1-\Theta(1/m)$. For FTG, it is possible to show an exponential lower bound using an adversary whose rate is independent of the size of the network.

4.2. A randomized greedy protocol with polynomial queue size

In this section we present a randomized greedy protocol with polynomially-bounded queues. We say that a randomized protocol $\mathcal{P}$ has polynomially-bounded queues if there is a polynomial $p(\cdot)$ such that for any network G with m edges, any adversary $\mathcal{A}$, any $t > 0$, and any $k > 1$, the probability that at time t there are more than $kp(m)$ packets in any queue of the system $(G, \mathcal{A}, \mathcal{P})$ is exponential in $-k$.

The bound we obtain for our protocol is polynomial in $d \log m$; thus, for systems in which only short paths are used, this bound is polylogarithmic in the network size.

The definition of the protocol. The protocol is actually simple to state, and so we do this at the outset. Let $\mathcal{A}$ be an adversary of rate (w, r). Let d denote the length of the longest simple directed path and m the number of edges in G. Below, we will define some parameters T, T', and μ in terms of m, r, and d. When a packet p is injected at time t, it is assigned a *label* of value $T' \lceil \frac{t}{T} \rceil + \lambda(p)$, where $\lambda(p)$ is an integer chosen uniformly at random from the interval $[1, \mu]$. At any edge queue, the packet with the smallest label is advanced; this packet's label is then incremented by 1.

The remainder of this section is devoted to defining the parameters T, T', and μ appropriately, and then analyzing the resulting protocol.

Schedules and suffixes. The following lemma will be useful. If X is a set of packets in a graph G, each with a fixed path to traverse, a *schedule* for X is a function σ giving, for each packet $p \in X$ and each edge e in the path of p, the time at which p crosses e. (We assume throughout that time values are non-negative integers.) We require that no two packets cross the same edge at the same time. The *makespan* of σ is the largest absorption time of any packet in X, under the schedule σ.

We say that a greedy schedule σ' is a *suffix* of σ if σ' can be obtained from σ as follows. First, position each packet $p \in X$ at some vertex on its path. Now inductively construct σ' as follows: at a given edge e and time $t = 0, 1, \ldots$, advance the packet in the queue of e that crosses e first under σ.

Lemma 4.3 *If σ' is a suffix of σ, then the makespan of σ' does not exceed that of σ.*

A static protocol. We start by presenting a randomized (non-greedy) protocol for the static routing problem, i.e. the problem of routing a set of N packets, all initially in the system, in the network G. This protocol is derived from a distributed randomized algorithm presented for this problem by Leighton, Maggs, and Rao [13]. We will base our dynamic algorithm on this static one.

Let first assume that we have a set of N packets to be routed in a network G such that no packet has to traverse a path of more that d edges (dilation) and no edge is in more than c

packet paths (congestion). Leighton, Maggs, and Rao [13] presented a distributed randomized algorithm that routes all the packets in $O(c + d\log(Nd))$ steps, with high probability. Here we slightly modify the parameters of the algorithm so the routing takes $(1+\epsilon)c + O(d\log(mcd))$ steps, for any $\epsilon > 1$.

The algorithm for the static problem works as follows. First, each packet p is assigned an integer value $\lambda(p)$ chosen randomly, independently, and uniformly from $[1, \frac{\alpha c}{\log(mcd)}]$, where α is a (small) constant. Let us define $\beta = 1+\epsilon$ and divide the routing time into *intervals* of $\frac{\beta}{\alpha}\log(mcd)$ consecutive steps. A packet p waits in its initial queue for $\lambda(p)$ intervals, and then traverses its path one edge per interval. We say that the algorithm *fails* if more than $\frac{\beta}{\alpha}\log(mcd)$ packets try to cross some edge e in some interval i.

Using Chernoff bounds, one can show that if α is chosen small enough, the probability that the algorithm fails is at most $(mcd)^{-1}$. Therefore, with probability at least $1-(mcd)^{-1}$, the number of steps taken to route with the static protocol is at most

$$(d + \frac{\alpha c}{\log(mcd)})\frac{\beta}{\alpha}\log(mcd) = \beta c + \frac{d\beta}{\alpha}\log(mcd).$$

Back to the dynamic protocol. Let us now go back to the dynamic problem and assume an adversary $\mathcal{A}$ of rate (w, r). Let us choose a positive number β so that $1 < \beta^2 < 1/r$, and then α small enough as in the preceding subsection. We now choose T to be a multiple of w and large enough so that $T > \beta(\beta r T + \frac{d\beta}{\alpha}\log(mrTd))$. Note that we can find such a T that is $\Theta(d\log m/(1-r))$.

We picture time as being divided into *blocks* of length T, and consider the set of packets X_i injected in the i^{th} block of time, $i = 1, 2, \ldots$. We would essentially like to run the static algorithm defined above on each set X_i in turn; note that the congestion of the packets in X_i is at most rT. Thus in the definition of the protocol at the beginning of this section, we set $\mu = \frac{\alpha r T}{\log(mrTd)}$, and $T' > \mu + d$. The effect of this definition of T' is that packets in X_i will always have priority over those in X_j for $j > i$. Let us say that X_i is *successful* if the set of random labels chosen for the packets in X_i, when used in the static algorithm above, causes it to terminate within time

$$\beta^{-1}T > \beta r T + \frac{d\beta}{\alpha}\log(mrTd).$$

By the above analysis, each X_i is successful with probability at least $1-(mrTd)^{-1}$.

A best-case scenario would be the following: the packets in X_1 are successful and hence absorbed by time $2T$ (note that the last packet in X_1 only arrives at time T). This gives priority to the packets in X_2. The packets in X_2 are also successful and hence absorbed by time $3T$. This gives priority to the packets in X_3, and so on.

Unfortunately, the analysis of the static algorithm shows that there is a positive probability of any given set X_i being unsuccessful, and this is what we consider below. Let τ_i denote the time at which the last packet in $X_1 \cup X_2 \cup \cdots X_i$ is absorbed under the dynamic protocol, and $\delta_i = \tau_i - (i+1)T$. Thus δ_i tells how much "behind schedule" the absorption of the sets preceding X_{i+1} was. We now claim the following:

Lemma 4.4 *If $\delta_i \geq T(1-1/\beta)$, and X_{i+1} is successful, then $\delta_{i+1} \leq \delta_i - T(1-1/\beta)$.*

Proof. Consider the set of packets in X_{i+1} at time τ_i; from this time until they are absorbed, these packets have higher priority than any other packet in the system. Consider the schedule σ' on this set of packets defined by their current positions and current labels. Also, consider the schedule σ defined by the initial labels of the packets in X_{i+1}, assuming that they were all released from their sources at the same time. The schedule σ' is a suffix of the schedule σ; thus by Lemma 4.3 and our assumption that X_{i+1} is successful, the packets in X_{i+1} will be absorbed by time $\tau_i + T/\beta$. Hence $\tau_{i+1} - \tau_i \leq T/\beta$, and the lemma follows. ■

Finally, we can show that the protocol has polynomially bounded queues. When the random variable δ_i exceeds $T(1 - 1/\beta)$, it goes down by $T(1-1/\beta)$ with probability at least $1-(mrTd)^{-1}$; and otherwise it goes up by at most $cd \leq rTd$. Thus the expected change in δ_i is less than or equal to

$$-(1-(mrTd)^{-1})T(1-1/\beta) + 1/m.$$

It follows that the probability of δ_i exceeding $krTd$ is exponential in $-k$.

From this it follows that, at any time t, the probability of there being more than krd non-empty sets of packets X_i is exponential in $-k$, and hence the protocol has polynomially bounded queues.

5. Remarks and open questions

We have classified many of the standard simple greedy protocols known to us in terms of their universal stability. However, it would be interesting to study other simple protocols from the point of view of stability, as well as to study the behavior of the protocols of this paper in more detail. We suggest the following three sets of open questions.

First, we do not know of a *deterministic*, *distributed* queueing protocol with polynomially bounded queues. We feel it is of considerable interest to determine whether such a protocol exists. Given the similarities between LIS and our randomized protocol, and the fact that we do not know of an exponential lower bound for LIS, a specific open question is to determine whether LIS itself is polynomially bounded. (We also note that the randomized protocol of Section 4.2 can be converted into a deterministic, centralized protocol with polynomially bounded queues; thus, the emphasis is on finding a protocol that is both deterministic and distributed.)

Throughout most of this paper, our focus has been on adversaries with rates arbitrarily close to 1. But it is interesting to study the behavior of protocols against adversaries of rates bounded away from 1. In Section 2.2, we showed that LIFO and NTG can be unstable at any injection rate greater than 0.62; a recent result of Borodin, Kleinberg, Sudan, and Williamson [3] shows that there exist adversaries of arbitrarily small positive rates that cause NTG to be unstable. However, an analogous result is not known for FIFO, and so we can ask: does there exist a rate $r_0 > 0$ such that FIFO is stable against every adversary of rate (w, r_0), for every w and every network? Similarly, does one of FTG or SIS become polynomially bounded when the injection rate is made small enough?

Finally — the assumption that a packet is injected with a pre–specified path through the network is fairly standard within the context of queueing theory; however, for packet–routing problems, it would be interesting to consider an adversarial model of *adaptive routing*. Here, an adversary injects packets with only their destinations specified, subject to a rate restriction that, say, requires there to be a feasible integral multicommodity flow from the newly injected sources to their destinations. The contention resolution protocol is then free to route each packet on an arbitrary path to its destination. This model is closer to the setting of the Awerbuch–Leighton multicommodity flow algorithm [1]; it would be interesting to investigate these connections further.

Acknowledgments

We thank Allan Borodin for helpful discussions.

References

[1] B. Awerbuch and F. T. Leighton. Improved approximations for the multi-commodity flow problem and local competitive routing in networks. In *Proc. 26th ACM STOC*, 1994.

[2] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queueing theory. In *Proc. 28th ACM STOC*, 1996.

[3] A. Borodin, J. Kleinberg, M. Sudan, and D. P. Williamson. Notes, June 1996.

[4] A. Broder and E. Upfal. Dynamic deflection routing on arrays. In *Proc. 28th ACM STOC*, 1996.

[5] A. Z. Broder, A. M. Frieze, and E. Upfal. A general approach to dynamic packet routing with bounded buffers. In *Proc. 37th IEEE FOCS*, 1996.

[6] R. L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Trans. on Information Theory*, 37(1):114–131, Jan. 1991.

[7] R. L. Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Trans. on Information Theory*, 37(1):132–141, Jan. 1991.

[8] M. Harchol-Balter and P. E. Black. Queueing analysis of oblivious packet-routing algorithms. In *Proc. 5th ACM-SIAM SODA*, 1994.

[9] M. Harchol-Balter and D. Wolfe. Bounding delays in packet-routing networks. In *Proc. 27th ACM STOC*, 1995.

[10] N. Kahale and T. Leighton. Greedy dynamic routing on arrays. In *Proc. 6th ACM-SIAM SODA*, 1995.

[11] F. P. Kelly. *Reversibility and Stochastic Networks*. Wiley, New York, 1979.

[12] L. Kleinrock. *Queueing Systems*. Wiley, New York, 1975.

[13] 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.

[14] T. Leighton. Average case analysis of greedy routing algorithms on arrays. In *Proc. of the 2nd Annual ACM Symp. on Parallel Algorithms and Architectures*, 1990.

[15] M. Mitzenmacher. Bounds on the greedy routing algorithm for array networks. In *Proc. of the 6th Annual ACM Symp. on Parallel Algorithms and Architectures*, 1994.

[16] Y. Rabani and É. Tardos. Distributed packet switching in arbitrary networks. In *Proc. 28th ACM STOC*, 1996.

[17] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. of Combinational Theory, Ser. B*, 41:92–114, 1986.

[18] N. Robertson and P. D. Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. of Combinational Theory, Ser. B*, 48:227–254, 1990.

[19] C. Scheideler and B. Vöcking. Universal continuous routing strategies. In *Proc. of the 8th Annual ACM Symp. on Parallel Algorithms and Architectures*, 1996.

[20] G. D. Stamoulis and J. N. Tsitsiklis. The efficiency of greedy routing in hypercubes and butterflies. *IEEE Trans. on Communications*, 42(11):3051–3061, Nov. 1994.

[21] L. Tassiulas and L. Georgiadis. Any work-conserving policy stabilizes the ring with spatial re-use. *IEEE/ACM Trans. on Networking*, 4(2):205–208, Apr. 1996.