Universal-Stability Results and Performance Bounds for Greedy Contention-Resolution Protocols

MATTHEW ANDREWS

Bell Labs, Lucent Technologies

BARUCH AWERBUCH

Johns Hopkins University, Baltimore, Maryland

ANTONIO FERNÁNDEZ, TOM LEIGHTON, AND ZHIYONG LIU

Massachusetts Institute of Technology, Cambridge, Massachusetts

JON KLEINBERG

Cornell University, Ithaca, New York

Abstract. In this paper, we analyze the behavior of packet-switched communication networks in which packets arrive dynamically at the nodes and are routed in discrete time steps across the edges. We

---

This work was supported by Army grant DAAH 04-95-1-0607 and ARPA contract N00014-95-1-1246.

A preliminary version of this work appeared in *Proceedings of the 1996 IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif.

The work of M. Andrews was performed while a student at MIT, supported by National Science Foundation (NSF) contract 9302476-CCR.

The work of A. Fernández was supported in part by the Spanish Ministry of Education.

The work of J. Kleinberg was supported in part by a David and Lucile Packard Foundation Fellowship, an Alfred P. Sloan Research Fellowship, an ONR Young Investigator Award, and NSF Faculty Early Career Development Award CCR 97-01399. This work was initiated while J. Kleinberg was a student in the MIT Laboratory for Computer Science, supported by an ONR Graduate Fellowship.

Z. Liu was on leave from the Institute of Computing Technology, Academia Sinica, Beijing, China. Z. Liu was supported by the K. C. Wong Education Foundation, Hong Kong.

Authors' present addresses: M. Andrews, Bell Laboratories, 600–700 Mountain Avenue, Murray Hill, NJ 07974, e-mail: andrews@research.bell-labs.com; B. Awerbuch, Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, e-mail: baruch@cs.jhu.edu; A. Fernández, Escuela Superior de Ciencias Experimentales y Technologia, Universidad Rey Juan Carlos, 28933 Mostoles, Madrid, Spain, e-mail: anto@eui.upm.es; J. Kleinberg, Department of Computer Science, Cornell University, Ithaca, NY 14853, e-mail: kleinber@cs.cornell.edu; T. Leighton, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 01219, e-mail: ftl@math.mit.edu; Z. Liu, Institute of Computing Technology, Academia Sinica, Beijing, China.

focus on a basic adversarial model of packet arrival and path determination for which the time-averaged arrival rate of packets requiring the use of any edge is limited to be less than 1. This model can reflect the behavior of connection-oriented networks with transient connections (such as ATM networks) as well as connectionless networks (such as the Internet).

We concentrate on greedy (also known as work-conserving) contention-resolution protocols. 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? We study the universal stability of networks (i.e., stability under all greedy protocols) and universal stability of protocols (i.e., stability in all networks). Once the stability of a system is granted, we focus on the two main parameters that characterize its performance: maximum queue size required and maximum end-to-end delay experienced by any packet.

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 and polynomial delay.

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.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*packet-switching networks*; *store and forward networks*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*sequencing and scheduling*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Adversarial queueing theory, end-to-end delay, network stability, packet scheduling

1. *Introduction*

We study the behavior of packet-switched communication networks in which packets arrive dynamically at the nodes and are 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 arrive 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.[1] In related work, Awerbuch and Leighton [1994] 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 arrivals

[1] See, for example, Leighton [1990], Stamoulis and Tsitsiklis [1994], Mitzenmacher [1994], Harchol-Balter and Black [1994], Kahale and Leighton [1995], Harchol-Balter and Wolfe [1995], Broder and Upfal [1996], Scheideler and Vöcking [1996], and Broder et al. [1996].

and routing has also been a major topic of study within the field of queueing theory [Kleinrock 1975; Kelly 1979]. 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 Borodin et al. [2001] for a review of previous work on these models.

In this paper, we work within a model of continuous packet arrivals proposed by Borodin et al. [2001], 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 Borodin et al. [2001], 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 Borodin et al. [2001] 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 and the load) 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 Borodin et al. [2001], we will only consider *greedy* (also known as *work-conserving*) 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. [2001] 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. [2001] 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 [1991a; 1991b]. 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 [1991b] proves the stability of every greedy protocol on every layered DAG. Tassiulas and Georgiadis [1996] also work within this model, and show that every greedy protocol on the ring is stable. In related work, Rabani and Tardos [1996] developed a randomized algorithm for static routing problems, and applied it to dynamic problems. They obtain an algorithm for which 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. The bounds of Rabani and Tardos were later tightened by Ostrovsky and Rabani [1997].

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. [2001]; 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. [2001] that cannot be captured by the framework of Cruz. (For example, one needs the more general model of Borodin et al. [2001] to represent connections of limited duration.)

A number of fundamental open questions were raised in Borodin et al. [2001] 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 *protocol* $\mathcal{P}$ is *stable* on a network G against an adversary $\mathcal{A}$ if there is a constant C (which may depend on G and $\mathcal{A}$) such that, starting from an empty configuration, the number of packets in the system at all times is bounded by C.

Definition 1.2. 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 1.3. 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 Borodin et al. [2001] 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.

However, from a practical standpoint, stability is not enough. In order to evaluate the performance of a stable system we use two parameters: maximum queue size required and maximum end-to-end delay (delay for short) any packet can experience. The size of a queue is the number of packets waiting to cross the edge associated with the queue. The delay experienced by a packet is the time the packet is in the system (i.e., time from injection until absorption).

1.1. Bounded Adversaries. All the stability results shown in this work 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 (b, r), where $b \geq 1$ 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 I, at most

$r|I| + b$ can have paths that contain any one edge. Such a model allows for adversarial injection patterns that are "bursty." 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 $r|I| + b$ packets requesting this edge over any interval I. Observe that the maximum number of packets injected in the same time step requesting a given edge is b, since for an interval I with $|I| = 1$, $r|I| + b < b + 1$. This implies, for instance, that $b \geq 1$.

The restrictions imposed in our bounded adversaries are realistic in the sense that they are derived from the bucket-based techniques used for the shaping of traffic in packet-switched networks. In order to shape the traffic injected in session-oriented networks, for each session i there is a "bucket" of some size b_i, initially full of "tokens." New tokens are generated and placed into the bucket every $1/r_i$ steps, where r_i is the rate of the session; if the bucket is already full the new tokens are lost. In order to inject a new session-i packet in the network, a token has to be removed from the bucket. If the bucket is empty, the packet has to wait for a new token to be placed there before being injected.

Since we do not want to restrict ourselves to session oriented systems, we have defined the adversary requirements as if each edge in the network had an associated bucket of size b. New tokens appear in each bucket every $1/r$ steps. The adversary, in order to inject a new packet p in the system, has to remove one token from each bucket associated with the edges traversed by p. If any of these are empty, the injection has to wait.

Clearly, the adversaries defined in this way are more powerful than the single-request based adversaries defined in Borodin et al. [2001], since they allow for some burstiness in the packet injections. Furthermore, they are also more powerful than the window-based adversaries defined in a previous version of this work [Andrews et al. 1996]. There, the rate was specified by a pair (w, r), where w (the window size) was a natural number and $0 < r < 1$. The requirement on the adversary was 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. Figure 1 compares the power of the three classes in terms of the number of packets the adversary could inject in a period of time that have to cross a given edge e. In this figure, we take the same value of r for the three classes, and we assume that $b = rw$. Clearly, our class of adversaries can inject at least as many packets as the other two at all times.

1.2. Our Results. In this paper, we resolve the open questions of Borodin et al. [2001] described above, and provide additional results on stability and performance bounds. 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 bounds on queue size and delay that are 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, these bounds are therefore polylogarithmic.

(iii) The n-node ring is universally stable, with maximum queue-size and delay that are linear (in n).

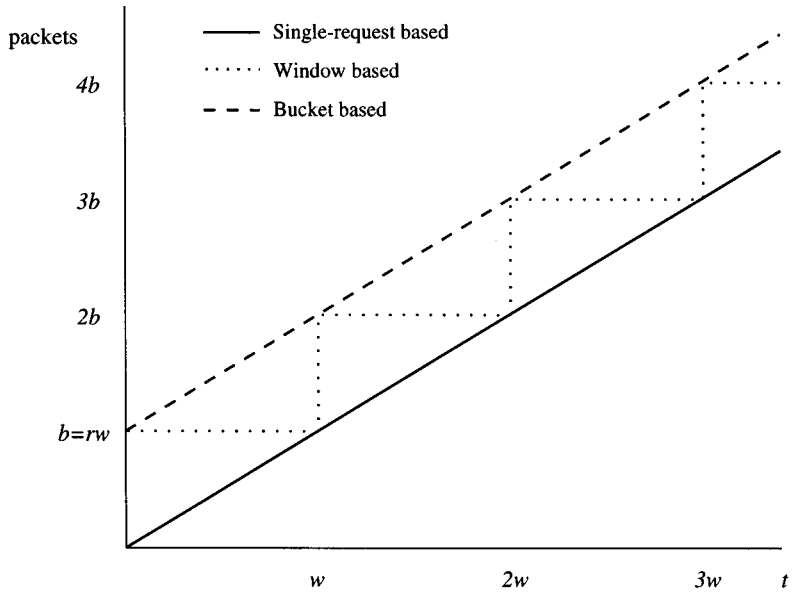

FIG. 1. Number of packets injected over time requiring a given edge for each class of adversaries.

(iv) There exist commonly used graphs and protocols that are not universally stable.

We first show that several natural protocols are universally stable; we will refer to them as *Farthest-to-Go* (FTG), *Nearest-to-Source* (NTS), *Longest-in-System* (LIS), and *Shortest-in-System* (SIS). FTG gives precedence to the packet whose distance to its destination is maximal; NTS gives precedence to the packet whose distance (in number of edges) traversed is minimal; 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 three of them (FTG, NTS, and SIS) can require queues and delays of exponential size in the worst case. For the fourth protocol, LIS, the best upper bounds on queue size and delay that we can show are exponential, though we do not know of matching lower bounds. Thus, it is natural to ask whether there exists a protocol with queues and delays of polynomial size. We show that there is a simple distributed randomized protocol with polynomial bounds on queue size and delay; as is standard, we say that the queue size (respectively, delay) of a randomized algorithm in this setting is polynomially bounded if the probability of its having a large queue (respectively, delay) 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, *Nearest-to-Go* (NTG), and *Farthest-from-Source* (FFS). 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 FFS

protocol always advances a packet whose distance to its source is maximal. 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 nontriviality 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 [1986; 1990].

1.3. 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 nonempty 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 (b, 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, no more than $r|I| + b$ packets are introduced during I with an assigned path containing e.

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 1.4. 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, we will assume that when a packet is injected, its assigned path is *simple*; namely, it does not contain any edge more than once.

Definition 1.5. We say that $\mathcal{A}$ is a *bounded adversary*, of rate (b, r), if for all edges e and all intervals I, it injects no more than $r|I| + b$ packets during I that require edge e at their time of injection.

Note that a system is stable if and only if both the maximum number of packets in any queue and the maximum end-to-end delay of any packet are bounded. The following easy result shows that there is a close relation between these two quantities.

THEOREM 1.6. *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* $(b, 1 - \varepsilon)$, $0 < \varepsilon < 1$. *Suppose there are never more than* k *packets in any queue, where* $mk > b$. *Then any packet that is injected with a path of length* d *will be absorbed in at most* $2mkd/\varepsilon$ *steps. Conversely, if the maximum delay experienced by any packet is at most* s, *then there are never more than* s *packets in any queue.*

PROOF. Suppose there are never more than k packets in any queue at any time. Then, there are never more than mk packets in the system. Consider a queue q at time t. We claim that q becomes empty sometime in the next $2mk/\varepsilon$ steps. For if not, then a packet must leave q on each of the next $2mk/\varepsilon$ time steps. But there are only mk packets in the system at time t, and no more than

$$(1 - \varepsilon)\frac{2mk}{\varepsilon} + b = \frac{2mk}{\varepsilon} - 2mk + b$$

$$< \frac{2mk}{\varepsilon} - mk$$

packets arrive into q over the next $2mk/\varepsilon$ time steps—this contradicts the assumption that a packet leaves q in every one of these time steps.

From this it follows that a packet crosses at least one edge in $2mk/\varepsilon$ time steps. Hence, if a packet must cross d edges before being absorbed, it will be absorbed in at most $2mkd/\varepsilon$ time steps.

Now suppose that the maximum packet delay is at most s. If there were ever a set of more than s packets in some queue q, the last packet in this set to leave q would experience a delay of more than s, a contradiction.

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 four simple protocols for which the answer is affirmative. Our upper bounds on queue size and end-to-end delay 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 bounds on queue size and delay that are 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. Table I summarizes the universal-stability properties of the protocols studied in this section.

2.1. UNIVERSALLY STABLE PROTOCOLS.

In this section, we show the universal stability of four simple protocols. To do that, we upper bound the number of packets waiting in any queue at any time (and therefore the number of packets in

TABLE I. UNIVERSAL STABILITY OF THE SIMPLE PROTOCOLS CONSIDERED

Protocol	Universally stable?
FIFO	NO
LIFO	NO
Nearest-to-Go (NTG)	NO
Farthest-from-Source (FFS)	NO
Farthest-to-Go (FTG)	YES
Nearest-to-source (NTS)	YES
Shortest-in-System (SIS)	YES
Longest-in-System (LIS)	YES

the system at any time) and the end-to-end delay that any packet can experience. These bounds are summarized in Table II.

2.1.1. *SIS Is Universally Stable*. Here we show that the *Shortest-in-System* (SIS) protocol, which at every queue gives priority to the packet that was injected most recently, is universally stable. Let G be a directed network, and $\mathcal{A}$ some bounded adversary of rate $(b, 1 - \varepsilon)$, with $\varepsilon > 0$. We will show the stability of the system $(G, \mathcal{A}, SIS)$. We first show the following lemma:

LEMMA 2.1. *Let p be a packet waiting in the queue of edge e at time t and suppose there are currently $k - 1$ other packets in the system requiring e that have priority over p. Then p will cross e within the next $(k + b)/\varepsilon$ steps.*

PROOF. Assume p does not cross e in the next $(k + b)/\varepsilon$ steps. Then, a distinct packet crosses e in each of the $(k + b)/\varepsilon$ 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 $k - 1$ packets existing at time t, or one of the (at most) $(1 - \varepsilon)(k + b)/\varepsilon + b$ packets requiring e that were injected during this time. Thus, at most $k - 1 + (1 - \varepsilon)(k + b)/\varepsilon + b < (k + b)/\varepsilon$ packets have priority over p during this time, a contradiction.

We now define the numbers $k_1, k_2, \ldots$ by the recurrence $k_1 = b$, $k_{j+1} = (k_j + b)/\varepsilon$.

LEMMA 2.2. *When a packet p arrives at the queue of the jth edge e_j on its path there are at most $k_j - 1$ packets requiring any edge e in the path of p with priority over p.*

PROOF. We use induction to prove the claim. It 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) $b - 1$ packets injected in the same time step as p. Now, suppose that the claim holds for some j. Then by the above lemma, p will arrive at the tail of e_{j+1} in at most another $(k_j + b)/\varepsilon$ steps, during which time at most another $(1 - \varepsilon)(k_j + b)/\varepsilon + b$ packets requiring any edge e arrive with priority over p. Thus, when p arrives at the tail of e_{j+1}, at most

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

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

TABLE II. UPPER BOUNDS ON QUEUE SIZE AND END-TO-END DELAY FOR THE SIMPLE UNIVERSALLY STABLE PROTOCOLS CONSIDERED

Protocol	Queue size	Delay
Farthest-to-Go (FTG)	$O(bm^{d-1}/\varepsilon)$	$O(bm^{d-1}/\varepsilon)$
Nearest-to-Source (NTS)	$O(bm^{d-1}/\varepsilon)$	$O(bm^{d-1}/\varepsilon)$
Shortest-in-System (SIS)	$O(b/\varepsilon^d)$	$O(db/\varepsilon^d)$
Longest-in-System (LIS)	$O(b/\varepsilon^d)$	$O(b/\varepsilon^d)$

THEOREM 2.3. *The system* $(G, \mathcal{A}, SIS)$ *is stable, no queue ever contains more than* k_d *packets, and no packet spends more than* $(db + \Sigma_{i=1}^{d} k_i)/\varepsilon$ *steps in the system, where* d *is the length of the longest simple directed path in* G.

PROOF. First, assume there are $k_d + 1$ packets at some point all requiring the same edge; the one of these with the lowest priority contradicts the claim of the previous lemma.

Combining both lemmas above, a packet p takes at most $(k_j + b)/\varepsilon$ steps to cross the jth edge in its path, once it is in the queue for this edge. The delay bound follows. □

2.1.2. *LIS Is Universally Stable*. The *Longest-in-System* (LIS) protocol gives priority to the packet that has been in the system the longest. Let G be a directed network, and $\mathcal{A}$ an adversary of rate $(b, 1 - \varepsilon)$, with $\varepsilon > 0$. We show that the system $(G, \mathcal{A}, LIS)$ is stable.

Let us denote by *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 $\ell' \leq \ell$. 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 2.4. $T_d - T_0 \leq (1 - \varepsilon^d)c + (1 - \varepsilon^d)/(1 - \varepsilon)b$.

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. Note that, by definition, all the active classes are consecutive. Hence, there are at most $(1 - \varepsilon)(c + T_0 - T_{i-1}) + b$ packets in these classes. Since p is one of these packets, at most $(1 - \varepsilon)(c + T_0 - T_{i-1}) + b - 1$ packets can block p. Therefore,

$$\begin{aligned} T_i &\leq T_{i-1} + (1 - \varepsilon)(c + T_0 - T_{i-1}) + b \\ &= \varepsilon T_{i-1} + (1 - \varepsilon)(c + T_0) + b. \end{aligned}$$

Thus, solving the recurrence, we obtain

$$T_d \leq ((1-\varepsilon)(c+T_0)+b)\sum_{i=0}^{d-1}\varepsilon^i + \varepsilon^d T_0$$

$$= ((1-\varepsilon)(c+T_0)+b)\frac{1-\varepsilon^d}{1-\varepsilon} + \varepsilon^d T_0$$

$$= (1-\varepsilon^d)c + \frac{1-\varepsilon^d}{1-\varepsilon}b + T_0$$

and the claim follows. □

THEOREM 2.5. *There are never more than* $b/(1-\varepsilon)\varepsilon^d$ *active classes in the system* $(G, \mathcal{A}, LIS)$, *where* d *is the length of the longest simple directed path in* G.

PROOF. Let $c = b/(1-\varepsilon)\varepsilon^d$ 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

$$(1-\varepsilon^d)c + \frac{1-\varepsilon^d}{1-\varepsilon}b + 1 = c + 1 - \frac{b-(1-\varepsilon^d)b}{1-\varepsilon} < c+1$$

steps, and we reach a contradiction. The inequality follows from the facts that $b \geq 1$ and $\varepsilon > 0$.

COROLLARY 2.6. *The system* $(G, \mathcal{A}, LIS)$ *is stable, there are never more than* $b/\varepsilon^d + b$ *packets in any queue and no packet spends more than* $b/(1-\varepsilon)\varepsilon^d$ *steps in the system, where* d *is the length of the longest simple directed path in* G.

2.1.3. *FTG Is Universally Stable*. The *Farthest-to-Go* (FTG) protocol gives priority to the packet that still has to cross the largest number of edges. Let G be a directed network, and $\mathcal{A}$ a bounded adversary of rate $(b, 1-\varepsilon)$, with $\varepsilon > 0$. 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 $k_i = mk_{i+1} + mb$ for $1 \leq i \leq d$.

THEOREM 2.7. *The system* $(G, \mathcal{A}, FTG)$ *is stable, there are never more than* k_1 *packets in the system, no queue ever contains more than* $k_2 + b$ *packets, and no packet spends more than* $1/\varepsilon(db + \Sigma_{i=2}^{d} k_i)$ *steps in the system*.

PROOF. We prove by a backwards induction that, for all i, the number of packets in the system that still have to cross at least i edges is at most k_i.

This is trivial for $i > d$ 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,

$$\begin{aligned} |X_i(t)| &\le k_{i+1} + (t - t')(1 - \varepsilon) + b - (t - t') \\ &= k_{i+1} - \varepsilon(t - t') + b. \end{aligned}$$

The above inequalities have three consequences. First, the number of packets in the system that still have to cross i or more edges is always at most $mk_{i+1} + mb = k_i$ and so the inductive step holds. Second, by making $i = 1$, they give a bound of $k_2 + b$ on the maximum queue size. And third, $t - t'$ cannot be greater than $(k_{i+1} + b)/\varepsilon$. Hence, this expression gives the maximum amount of time that a packet with i edges still to cross takes to cross the next edge. Therefore, under FTG, the maximum number of packets in the system is bounded by k_1 and the maximum amount of time that any packet spends in the system is bounded by $(db + \Sigma_{i=2}^{d} k_i)/\varepsilon$.

2.1.4. *NTS Is Universally Stable.* The *Nearest-to-Source* (NTS) protocol gives priority to the packet that has crossed the smallest number of edges. Let G be a directed network, and $\mathcal{A}$ a bounded adversary of rate $(b, 1 - \varepsilon)$, with $\varepsilon > 0$. 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 $\ell_0 = 0$ and $\ell_i = m\ell_{i-1} + mb$ for $i > 0$. Note that $\ell_{d-i+1} = k_i$, for $i \le d + 1$ and k_i defined as in the previous section. We can use a proof similar to the one of Theorem 2.7 to show that the number of packets in the system that have crossed less than i edges is never larger than ℓ_i, and that all of them will cross at least one edge in the next $(\ell_{i-1} + b)/\varepsilon$ steps. Hence, we have the following result:

THEOREM 2.8. *The system* $(G, \mathcal{A}, NTS)$ *is stable, there are never more than* ℓ_d *packets in the system, no queue ever contains more than* $\ell_{d-1} + b$ *packets, and no packet spends more than* $(db + \Sigma_{i=1}^{d-1} \ell_i)/\varepsilon$ *steps in the system.*

2.2. PROTOCOLS THAT ARE NOT UNIVERSALLY STABLE. In this section, we show the instability of commonly-used protocols (namely, FIFO, LIFO, NTG, and FFS) 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 nonempty initial configuration. This implies instability results for systems with an empty initial configuration, by the following simple lemma:

LEMMA 2.9. *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 nonempty 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*.*

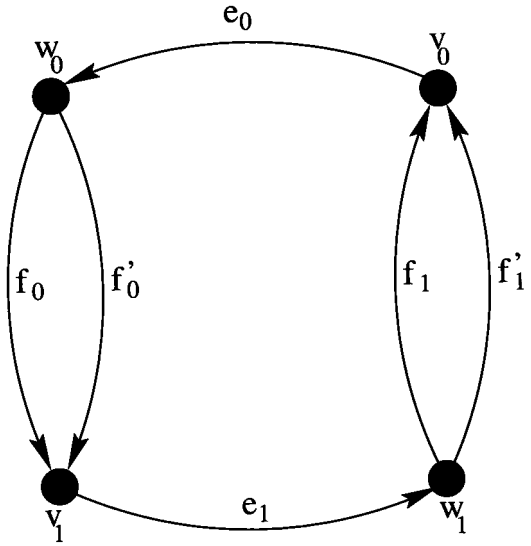

FIG. 2. Graph G used to show instability for FIFO and NTG.

PROOF. For each node of $v \in G$, which in the system $(G, \mathcal{A}, \mathcal{P})$ begins with k_v packets, we define a tree T_v rooted at v and otherwise disjoint from G. We choose all T_v sufficiently large that there is an adversary $\mathcal{A}'$ of rate r that can inject k_v packets into T_v such that

(i) Each packet is injected into $\bigcup_v T_v$ in a different time step.
(ii) No packets in $\bigcup_v T_v$ meet until they reach their roots in v.
(iii) Under any greedy protocol, all packets arrive at their roots in G in the same time step t^*. (This is possible by property (ii).)

Starting from time step t^*, $\mathcal{A}'$ behaves like $\mathcal{A}$.

2.2.1. *Instability of FIFO.* 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}, see Figure 2.

THEOREM 2.10. *Let $r \geq 0.85$. There is a nonempty initial configuration and an adversary $\mathcal{A}$ of rate r such that $(G, \mathcal{A}, FIFO)$ is unstable.*

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 at least $s_0 + j$ packets in the queue of e_i, for $i = 0$ or 1 (depending on whether j is even or odd) and a large enough constant s_0.

To start out, we have s_0 packets queued at node v_0. Then, 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 $s \leq s_0$ packets, then at the start of Phase $j + 1$, there will be more than s packets in the queue of e_1.

The sequence of injections in Phase j is as follows: For simplicity, we will omit floors and ceilings, and sometimes will count steps and packets roughly; by

carrying these through the computations one loses some additive constants, which are offset by the fact that s_0 is a large enough constant.

(1) For the first s steps, we inject a set X of rs packets that want to traverse edges $e_0 f'_0 e_1$. These are blocked by the packets in S.
(2) For the next rs steps, we inject a set Y of $r^2 s$ packets that want to traverse edges $e_0 f_0 e_1$. These are blocked by the packets in X.

    We also delay the flow of packets in X through 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 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 $r^3 s + r^2 s/(r+1) > s$, we meet the induction hypothesis for Phase $j+1$.

2.2.2. *Instability of NTG.* An adversary similar to the one described above can be used to prove the instability of the Nearest-to-Go (NTG) protocol on the same network G at any rate $r > 1/\sqrt{2}$. To do that, we change the claim in the proof above so that we no longer force packets in S be in the queue of e_i at the beginning of Phase j; they can be anywhere in the network, but still require edge e_i. Note that single-edge injections have the highest priority under the NTG protocol. Now, at the end of the second subphase the size of the set X has only shrunk to $r^2 s$, and the claim follows, since $|X| + |Y| = 2r^2 s > s$. Note that we no longer need the third subphase above.

2.2.3. *Instability of FFS.* The proof of instability for the Farthest-from-Source (FFS) protocol is very similar to that of NTG. Note that in any phase j, under FFS, the packets in the initial set S have priority over the packets in set X. Therefore, the first subphase works exactly the same.

However, to block the packets of X during the second subphase we use packets that traverse a two-edge linear array (incident to the node v_i) before the edge f'_i. Hence, at the end of Phase j the set X will contain $r(rs-2)$ packets, and the resulting initial set for Phase $j+1$ contains $2r^2 s - 2r$ packets. This is larger than s for $r > 1/\sqrt{2}$ when s is large.

2.2.4. *Instability of LIFO.* We can also show the instability of the LIFO protocol by slightly modifying the proof for NTG. Let us define the network G' to be an eight-node cycle with some parallel edges. (See Figure 3.) The vertices of G' are denoted v_0 to v_7, and there is an edge leaving v_i denoted e_i. Additionally, v_3 and v_7 have outgoing edges e'_3 and e'_7, respectively. G' also has four extra edges incoming to nodes v_1, v_2, v_5, and v_6, respectively. The edge incident to node v_i will be denoted f_i. These f-edges are considered "faster" than the e-edges, i.e., given an f-edge and an e-edge incoming to the same vertex v, if one packet crosses each of them in the same step, the packet that crossed the e-edge arrives later to v than the packet that crossed the f-edge.

THEOREM 2.11. *Let $r > 1/\sqrt{2}$. There is a nonempty initial configuration and an adversary $\mathcal{A}$ of rate r such that $(G', \mathcal{A}, LIFO)$ is unstable.*

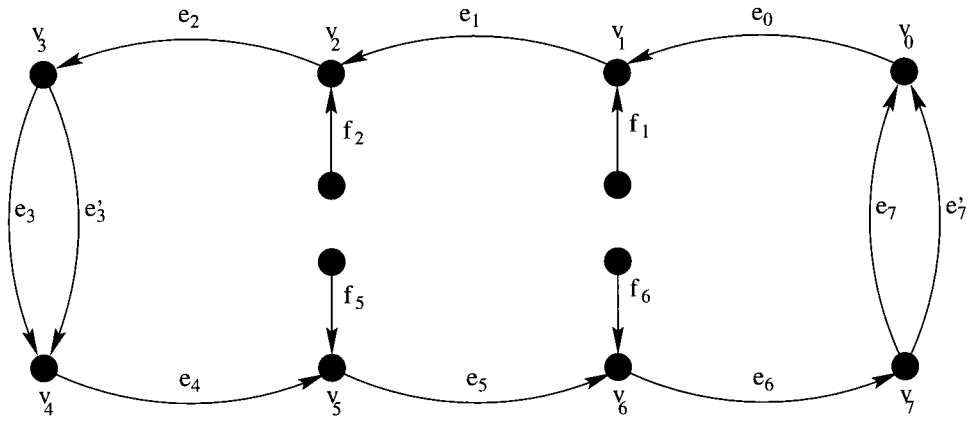

FIG. 3. Graph G' used to show instability for LIFO.

PROOF. The proof is similar to the proof of Theorem 2.10. We use induction and show that, if there is a large enough set S of s packets requiring edges e_0e_1 at the beginning of some phase j (for j even), at the end of the phase there will be more than s packets requiring edges e_4e_5. For j odd, the argument is symmetric.

The subphases of Phase j are now as follows:

(1) For the first s steps, we inject a set X of rs packets that want to traverse edges $f_1e_1e_2e'_3e_4e_5$. Since f_1 is a fast edge, at any step the packet of S crossing e_0 always arrives to v_1 after the packet from X crossing f_1 (if any), and blocks it.

(2) For the next rs steps, we inject a set Y of r^2s packets that want to traverse edges $f_2e_2e_3e_4e_5$. Since f_2 is fast, the rs packets of X initially at the tail of e_1 block the packets in Y at the queue of edge e_2.

Also, we block the packets of X at the queue of e'_3 with single-edge injections. (New injections always arrive later in the step, and block other packets.) At the end of this subphase, there will be r^2s of them there.

This is the end of Phase j, there are $2r^2s > s$ packets at the tail of e_2 and e'_3, and the induction hypothesis for Phase $j + 1$ holds.

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, FFS, 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 Borodin et al. [2001]—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 $b \geq 0$ and $\varepsilon > 0$. Our goal is to show that any greedy queueing discipline $\mathcal{P}$ is stable against any adversary of rate $(b, 1 - \varepsilon)$. Roughly, the proof will proceed as follows: We fix a large number Q', and suppose by way of contradiction that there comes a (first) time at which there is a set S of $Q' + 1$ packets in the system, all requiring a common edge e. For the interval of time between the injection of the first and last of the packets in S, there are no more than Q' packets requiring any common edge; from this we obtain the contradiction that for Q' sufficiently large, one of the packets in S must have crossed e before the last packet in S was injected.

We now give the proof in detail; 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 $T' > T_0$ be some time at which it has not yet been absorbed. Let $v_{i_0}, \ldots, v_{i_0+s}$ be the nodes through which p passes in the interval $[T_0, T']$. We write $i_k = i_0 + k$. For $k = 0, \ldots, s$, 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 $T_{s+1} = T'$.

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

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

Proof. For $0 \leq k \leq s - 1$, the packet p crosses i_k in step T_{k+1}; and at step $T_{s+1} = T'$, either p or some other packet crosses i_s. Since p waits at the tail of i_k for every step $t \in (T_k, T_{k+1}]$, and $\mathcal{P}$ is a greedy protocol, it follows that some packet crosses edge i_k in every step $t \in (T_k, T_{k+1}]$.

For j, an edge of G, and $t \in [T_0, T']$, we define $P_{j,t}$ to be the number of packets in the system 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 $t' \leq t$. Then*

$$P_{j,t} \leq P_{j,t'} + (1 - \varepsilon)(t - t') + b - z,$$

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 as $f(j, T_0) = Q + (b + 1)(j - i_0)$, and $f(j, t) = Q - \varepsilon(t - T_0) + (b + 1)(1 + j - i_0)$, for $t > T_0$. We note the following properties of this function f:

LEMMA 3.3

(*i*) $f(j, t) = f(j, T_0) - \varepsilon(t - T_0) + b + 1$, *for* $t > T_0$.
(*ii*) $f(j, t) = f(j, t') - \varepsilon(t - t')$, *for* $t > t' > T_0$.
(*iii*) $f(j + 1, t) = f(j, t) + b + 1$.

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
—$j > i_s$ 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* $P_{j,t} \le f(j, t)$.

PROOF. We prove the lemma by induction on $j \ge i_0$, 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_0) is applicable, then $f(j, T_0) \ge Q$, and by assumption we have $P_{j,t} \le Q$ for all $t \in [T_0, T')$.

Now, consider any applicable pair (j, t), with $t > T_0$. If, for the past $t - T_0$ 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

$$\begin{aligned} P_{j,t} &\le P_{j,T_0} + (1 - \varepsilon)(t - T_0) + b - (t - T_0) \\ &= P_{j,T_0} - \varepsilon(t - T_0) + b \\ &\le f(j, T_0) - \varepsilon(t - T_0) + b \\ &< f(j, t). \end{aligned}$$

Otherwise, there is some step $t' \in (T_0, t]$ in which no packet crosses edge j. Let us take the most recent such step preceding t. 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 queue v_j is empty at the end of step $t' - 1$, and hence $P_{j-1,t'-1} \ge P_{j,t'-1}$. Again applying Lemmas 3.2 and 3.3, and the induction hypothesis, we have

$$\begin{aligned} P_{j,t} &\le P_{j,t'-1} + (1 - \varepsilon)(t - t' + 1) + b - (t - t') \\ &\le P_{j-1,t'-1} + b + 1 - \varepsilon(t - t' + 1) \\ &\le f(j - 1, t' - 1) + b + 1 - \varepsilon(t - t' + 1) \\ &= f(j, t' - 1) - \varepsilon(t - t' + 1) \\ &= f(j, t). \end{aligned}$$

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 $(b + 1)n/\varepsilon$ 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 $Q' = (b + 1)n/\varepsilon$, 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 without loss of generality; let $T_0 < T'$ denote the time at which the first of these was injected. Note that $Q \leq Q'$. In interval $[T_0, T']$ at most

$$(T' - T_0 + 1)(1 - \varepsilon) + b$$

packets can be injected requiring any edge. Therefore,

$$Q' \leq (T' - T_0 + 1)(1 - \varepsilon) + b,$$

and hence

$$T' - T_0 \geq \frac{Q' - b}{1 - \varepsilon} - 1 \geq Q'.$$

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

$$\begin{aligned} Q' &< f(n, T') \\ &= Q - \varepsilon(T' - T_0) + (b + 1)(1 + n - i_0) \leq Q' - \varepsilon Q' + (b + 1)n = Q', \end{aligned}$$

a contradiction.

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

PROOF. Suppose that a packet p is injected at time T_0, with origin i_0 and destination n, and suppose it is still not absorbed at time T', where $T' = T_0 + (b + 1)n(\varepsilon^{-1} + \varepsilon^{-2})$. Then, by Theorem 3.7, we can apply Lemma 3.6 with $Q = (b + 1)n/\varepsilon$, and we have

$$\begin{aligned} P_{n,T'} &\leq f(n, T') \\ &= Q - \varepsilon(T' - T_0) + (b + 1)(1 + n - i_0) \\ &= \varepsilon^{-1}(b + 1)n - \varepsilon(b + 1)n(\varepsilon^{-1} + \varepsilon^{-2}) + (b + 1)(1 + n - i_0) \\ &\leq 0 \end{aligned}$$

which is a contradiction.

One can show that packet delays of $\Omega(bn)$ and $\Omega(n\varepsilon^{-2})$ can be realized for most common protocols, which means that our bounds are nearly tight in the general case.

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 that 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. Our main result here is that universal stability is decidable, and in fact by an algorithm running in time $O(n^2)$. We note, however, that this result requires tools from the Graph-Minors work of Robertson and Seymour, and hence we do not exhibit an explicit characterization of the set of graphs that are universally stable.

We remark that, subsequent to the initial appearance of these results, an explicit algorithm to decide universal stability of a graph was obtained by Goel [1997]. Goel presents three simple graphs H_1, H_2, H_3 and shows that a directed graph G is universally stable if and only if none of H_1, H_2, H_3 is a minor of G. In work following this, Gamarnik [1998] has demonstrated how universal stability results for networks can also be derived using *fluid models*, an approach that has proved useful in the analysis of stochastic queueing networks.

In this section, we assume that the input is an undirected graph G, which may have self-loops and parallel edges. The network on which packets will be routed is a copy of G, which we call G_d, in which each undirected edge $e = \{u, v\}$ is replaced by two directed edges $e_1 = (u, v)$ and $e_2 = (v, u)$. We require the adversary to inject only packets that traverse *simple* paths in G—namely, paths that do not cross the same edge e of G twice. We then say that G is *universally stable*, as before, if every greedy protocol is stable against every such bounded adversary. Given these definitions, universal stability holds for the (undirected) cycle, by the result of Section 3.1, and for every tree, by a slight variant of results of Borodin et al. [2001]. (Note that for the ring, if the injected paths are simple then the packets can be divided into two classes with no interaction, each class running on a unidirectional ring.) However, the undirected version of the graph in Section 2.2 is not universally stable. We required the injected paths to be simple in order to conclude that cycles and trees are universally stable as *undirected* (rather than directed) graphs.

The crucial fact we show here is that the family of universally stable graphs is a *minor-closed* set; this is what allows us to apply the results of Robertson and Seymour [1995]. We first give some preliminary definitions.

Definition 3.9. Let G be an undirected graph, e an edge of G, and v_1, v_2 the vertices incident to e. We define $T(G, e)$ to be the graph obtained from G by removing the edge e and merging both vertices v_1, v_2 into a single vertex v. The edges incident to v in the resulting graph will be the union of the edges incident to v_1 and v_2, except for the deleted edge e.

We say that a graph H is a *minor* of a graph G if it can be obtained from a subgraph of G by zero or more applications of transformation T. A set of graphs $\mathcal{G}$ is said to be *minor-closed* if whenever $G \in \mathcal{G}$, every minor of G is also in $\mathcal{G}$. Our objective is to show that the set of universally stable graphs is minor-closed. We will do this essentially as follows: If H is not universally stable, then there is a bounded adversary $\mathcal{A}$ and a protocol $\mathcal{P}$ such that the system $(H_d, \mathcal{A}, \mathcal{P})$ is not stable; then, given a graph G such that H is a minor of G, we use $\mathcal{A}$ and $\mathcal{P}$ to construct a bounded adversary $\mathcal{A}'$ and protocol $\mathcal{P}'$ such that the system $(G_d, \mathcal{A}', \mathcal{P}')$ is unstable.

First, the following fact is immediate, since one can always define an adversary that only makes injections on a subgraph of the given network.

LEMMA 3.10. *If H is a subgraph of G and H is unstable, then G is also unstable.*

We now want to show that stability is propagated through applications of the transformation T; to prove this, we argue by contraposition that if $H = T(G, e)$ is not universally stable, then G is not universally stable. Thus, with G and H defined in this way, let $\mathcal{A}$ be an adversary of rate (b, r), $r < 1$, and $\mathcal{P}$ a greedy protocol such that $(H_d, \mathcal{A}, \mathcal{P})$ is unstable. We need to present a bounded adversary $\mathcal{A}'$ of rate (b', r'), $r' < 1$, and a protocol $\mathcal{P}'$ such that $(G_d, \mathcal{A}', \mathcal{P}')$ is unstable.

The transformation T contracts edge e of G, collapsing vertices v_1, v_2 into a single vertex v_0 in H. Let Δ_i denote the number of edges incident to v_i, and $\Delta = \Delta_1 + \Delta_2$. Then, in G_d there are Δ_i incoming edges to v_i and Δ_i outgoing edges from v_i. Let e_1 be the directed edge from v_1 to v_2 and e_2 the directed edge from v_2 to v_1 in G_d. For the sake of readability, we will speak of edges $e \neq e_1, e_2$ and vertices $v \neq v_1, v_2$ as belonging to both G and H (rather than talking about the implicit isomorphism underlying this).

Now, for a path P in H_d, represented as a sequence of edges, we define the *transformed path* $\gamma(P)$ in G_d as follows: Whenever an edge $e \neq e_1, e_2$ incident to v_1 appears directly before (respectively, after) an edge $e' \neq e_1, e_2$ incident to v_2, we insert the edge e_1 (respectively, e_2) between them.

We first derive a new adversary $\mathcal{A}_G$ for the graph G_d from the original adversary $\mathcal{A}$ for H as follows: When $\mathcal{A}$ injects a packet in H with path P, $\mathcal{A}_G$ injects a packet in G with path $\gamma(P)$. Note that for any edge e' common to H_d and G_d, the number of packets injected by $\mathcal{A}_G$ at any given step requiring e' is exactly the same as the number injected by $\mathcal{A}$. For edges e_1 and e_2, we have the following:

LEMMA 3.11. *In the system $(G_d, \mathcal{A}_G, \mathcal{P}_0)$, with $\mathcal{P}_0$ an arbitrary protocol, the number of packets in the queue for e_i $(i = 1, 2)$ at the end of step t is at most Δ more than the number at the end of step $t - 1$.*

PROOF. Without loss of generality, we consider the case $e_i = e_1$. The lemma follows from the fact that at most $\Delta_1 \leq \Delta$ packets requiring e_1 cross incoming edges to v_1 during each step, and the fact that $\mathcal{A}_G$ never injects a packet in v_1 that requires e_1.

However, the number of packets injected by $\mathcal{A}_G$ requiring edges e_i has the following property:

LEMMA 3.12. *The number of packets injected by $\mathcal{A}_G$ requiring edge e_i in t consecutive steps is at most $(rt + b)\Delta/2$.*

PROOF. Again, we consider e_1. A packet injected by $\mathcal{A}_G$ that requires e_1 has to require also one incoming edge to v_1 and one outgoing edge from v_2. We have Δ_1 such incoming edges to v_1 and Δ_2 such outgoing edges from v_2. Since in t steps $\mathcal{A}_G$ injects at most $rt + b$ packets requiring any of these edges, and a packet requiring e_1 requires one edge from each group, the total number of packets injected requiring e_1 is at most $\min\{(rt + b)\Delta_1, (rt + b)\Delta_2\} \leq (rt + b)\Delta/2$.

We now present another adversary $\mathcal{A}'$ and a new protocol $\mathcal{P}'$, and we will show that $(G_d, \mathcal{A}', \mathcal{P}')$ is unstable. The behavior of $\mathcal{A}'$ and $\mathcal{P}'$ in the interval of steps $[(\Delta + 1)(t - 1) + 1, (\Delta + 1)t]$, for any $t > 0$, is described in the two following phases:

(1) In step $(\Delta + 1)(t - 1) + 1$, the adversary $\mathcal{A}'$ behaves exactly like $\mathcal{A}_G$ in step t, and $\mathcal{P}'$ behaves exactly like $\mathcal{P}$ in step t (with arbitrary policy for the edges e_i).
(2) In each of the other Δ steps, $\mathcal{A}'$ injects a packet p_e for each edge $e \neq e_1, e_2$ of G_d; the packet p_e requires only edge e. In these steps, $\mathcal{P}'$ gives maximum priority to the packets $\{p_e\}$, and uses an arbitrary policy for the edges e_i.

We claim that the system $(G_d, \mathcal{A}', \mathcal{P}')$ evolves in essentially the same way as the system $(H_d, \mathcal{A}, \mathcal{P})$, with $\Delta + 1$ steps in the former substituting for one step in the latter. Phase 1 represents the original step of $(H_d, \mathcal{A}, \mathcal{P})$, while Phase 2 is used to move packets across the gap v_1, v_2 if necessary, while the rest of the system is blocked. If we assume both systems are initially empty, we can show the following lemma. Let $\{S^t_{G,e} : e \in G\}$ and $\{S^t_{H,e} : e \in H\}$ denote the configurations of $(G_d, \mathcal{A}', \mathcal{P}')$ and $(H_d, \mathcal{A}, \mathcal{P})$ respectively, at the end of time step t.

LEMMA 3.13. *Let $e \neq e_1, e_2$ be an edge common to H and G, and let t be any time step. Then $S^t_{H,e} = S^{(\Delta+1)t}_{G,e}$. Also, $S^{(\Delta+1)t}_{e_i} = \phi$.*

PROOF. We use induction on t. Since initially both systems are empty, the claim is true for $t = 0$. Now suppose the lemma holds for t, and consider time $t + 1$. In step $(\Delta + 1)t + 1$, the adversary $\mathcal{A}'$ performs injections according to $\mathcal{A}_G$ (Phase (1) above), and hence at the end of this time step, the only difference between the configurations of $(G_d, \mathcal{A}', \mathcal{P}')$ and $(H_d, \mathcal{A}, \mathcal{P})$ is that certain packets which should be in the queue for an edge leaving v_1 (respectively, v_2) are instead in the queue of edge e_2 (respectively, edge e_1). However, by the induction hypothesis, the queue for e_i is empty at the end of step $(\Delta + 1)t$, and so by Lemma 3.11 there are at most Δ packets in the queue for e_i. Thus, over the next Δ steps (Phase (2) above), all these packets will cross edge e_i. No other packet that appears in both G_d and H_d will cross any edge during these Δ steps, since all are blocked by the single-edge injections. Hence, at the end of step $(\Delta + 1)(t + 1)$, the queues for e_1 and e_2 are empty, and all other edge queues are the same in G_d and H_d.

The above lemma shows that the system $(G_d, \mathcal{A}', \mathcal{P}')$ is unstable. However, we still have to argue that $\mathcal{A}'$ is a *bounded* adversary.

LEMMA 3.14. *If $\mathcal{A}$ has rate (b, r), then $\mathcal{A}'$ has rate $((r + b)((\Delta/2) + 1) + 1, (\Delta + r)/(\Delta + 1))$; since $(\Delta + r)/(\Delta + 1) < 1$, $\mathcal{A}'$ is therefore bounded.*

PROOF. Since $\mathcal{A}$ is an adversary of rate (b, r), in t consecutive steps no more that $rt + b$ packets are injected in $(H_d, \mathcal{A}, \mathcal{P})$ requiring any edge. We study the behavior of $\mathcal{A}'$ over an interval I of length t. Such an interval I contains at most $t/(\Delta + 1) + 1$ steps corresponding to Phase 1 of our construction, and at most $\Delta t/(\Delta + 1) + 1$ steps corresponding to Phase 2.

Thus, by Lemma 3.12, the number of packets injected requiring edges e_1 or e_2 in these steps is at most $(r(t/(\Delta + 1) + 1) + b)\Delta/2$. For any other edge e, at most $r(t/(\Delta + 1) + 1) + b$ packets are injected in steps corresponding to Phase

1, and exactly one packet is injected in every one of the at most $\Delta t/(\Delta + 1) + 1$ steps corresponding to Phase 2. Thus, the total number of packets injected during interval I that require e is at most $t(\Delta + r)/(\Delta + 1) + r + b + 1$.

Therefore, in any t consecutive steps, no more than $t(\Delta + r)/(\Delta + 1) + (r + b)((\Delta/2) + 1) + 1$ packets are injected requiring any one edge, and hence $\mathcal{A}'$ is an adversary of rate $((r + b)((\Delta/2) + 1) + 1, (\Delta + r)/(\Delta + 1))$.

By the previous two lemmas, we have

LEMMA 3.15. *If $H = T(G, e)$ is not universally stable, then G is not universally stable.*

Now by Lemmas 3.10 and 3.15, we find that universal stability is indeed a *minor-closed* property.

THEOREM 3.16. *If G is universally stable, and H is a minor of G, then H is universally stable.*

Finally, we discuss the algorithmic consequences of Theorem 3.16. Robertson and Seymour, via their proof of Wagner's Conjecture, have shown that if $\mathcal{G}$ is an arbitrary minor-closed set of graphs, then there is a finite list of graphs $H_1, \ldots, H_k$ such that $G \in \mathcal{G}$ if and only if none of $H_1, \ldots, H_k$ is a minor of G. That is, any minor-closed set of graphs is defined by the exclusion of a finite set of graphs as minors. They also provide an $O(n^3)$ time algorithm to test whether any fixed graph H is a minor of an arbitrary n-node graph [Robertson and Seymour 1995]. It follows from these two facts that there is an $O(n^3)$ time algorithm to test for membership in any minor-closed set of graphs. Finally, when the minor-closed family $\mathcal{G}$ does not contain all planar graphs, results of Robertson and Seymour [1986; 1990] (see also Robertson and Seymour [1995]) imply that one can test for membership in $\mathcal{G}$ in time $O(n^2)$.

Thus, in our case, the minor-closed family of graphs we are dealing with is the set of universally stable graphs, and by Theorem 2.10, this family does not contain all the planar graphs. Hence, we have

THEOREM 3.17. *There is an algorithm with running time $O(n^2)$ that decides if a graph is universally stable.*

Gamarnik [1999] has subsequently considered a related network model in which each edge is undirected and can carry a single packet in one step. (This is in contrast to the bi-directed edges considered above, each of which can carry a single packet *in each direction*.) He obtained a simple characterization theorem for universal stability of networks in this model.

4. *Bounds on Queue Size and Delay for Universally-Stable Protocols*

The maximum *queue size* and *end-to-end delay* required by a protocol are the main parameters determining its performance. The issue of *stability* asks whether these parameters can become unbounded; but among universally stable protocols, it is important to identify those that maintain the smallest possible queues and delays.

Ideally, we would like to have a protocol that never holds more than a constant number of packets in any queue, and therefore no packet is delayed more than a constant times its path length. 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 four 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 and maximum end-to-end delay. In Section 4.1, we show that three of the protocols presented there actually *require* exponential queue size and delay, for some network G and some adversary $\mathcal{A}$.

In Section 4.2, we then present a simple distributed randomized greedy protocol that requires only polynomially-bounded queues, with high probability. Therefore, the end-to-end delay under this protocol is also polynomially bounded.

4.1. SIS, FTG, and NTS Require Exponential Queue Size and Delay. We now show that under the protocols SIS, FTG, and NTS the queue sizes can become exponential. This trivially implies that some packet has to wait an exponential number of steps to reach its destination.

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 $r = 1 - \varepsilon$, if for every $t \geq 1$, every interval I of t steps, and every edge e, $\mathcal{A}$ injects no more than $\lceil rt \rceil$ packets during I that require e at the time of injection.

4.1.1. *The Bound for FTG.* We note that it is easy to get an exponential lower bound on queue sizes under FTG for any rate $r > 1/\sqrt{2}$. The proof is based on the instability result for NTG presented in Section 2.2. The network used is simply the network G used there (see Figure 2) with a directed (away from G) linear array of length n attached to each of its four nodes. We use these "tails" appended to G to force under FTG the priorities in the proof of instability for NTG during $\Theta(n)$ phases.

To do so, in some Phase j, we want the packets in the initial set S to block the packets injected during the phase (which will form the initial set for Phase $j + 1$). To do so, the packets in S instead of being absorbed at node w_i, go down the tail that starts at w_i for an appropriate number of edges. This number of edges is $n - 1$ for the first phase, and decreases by 2 every phase. Similarly, a single-edge injection that was supposed to be absorbed at node v_{1-i} under NTG now goes all the way down the tail that starts at v_{1-i}. That gives it the highest priority (as with NTG).

Therefore, we can apply roughly $n/2$ phases of a process similar to that presented in Section 2.2.2. In each phase the number of packets in G increases by a factor $\ell > 1$, and therefore after the $\Theta(n)$ phases some queue in G will contain $\ell^{\Omega(n)}$ packets.

4.1.2. *The Bound for NTS.* We can show an exponential lower bound for NTS using a similar construction. The graph G from Section 2.2 is expanded with a linear array of length n to each of its four nodes. At the beginning of Phase j (assume j even) we assume that the initial set S is already in the system, and will start crossing edge e_0 in exactly k steps (i.e., the packets in S will cross e_0 in steps k to $k + s - 1$ after the beginning of the phase). We also assume that e_0 is at most the $k + 1$st edge in their path.

Then, in the first subphase (of s steps) we inject the set X of rs packets in the linear array incident to v_0, at a distance of $k + 1$ from v_0. These packets will

reach the queue of edge e_0 roughly at the same time the packets in S do, but under NTS these new packets have lower priority to cross e_0, since this is their $k + 2$nd edge, and will be blocked there.

The second subphase (of rs steps) starts immediately after the first (the packets injected in the first may still be traveling down the linear array). The set Y of r^2s packets is injected in the same node as X. We also inject r^2s packets requiring edge f'_0 in the linear array incident to w_0, at distance k from w_0, which will eventually meet those of X and block them.

This ends Phase j. After another $k + 3$ steps a train of about $2r^2s$ packets (the initial set for Phase $j + 1$) will arrive to v_1, therefore meeting the initial conditions for the next phase. We can repeat this process for about $\Theta(n/3)$ phases. If each phase increases the size of the initial set by a factor of $\ell > 1$, at the end we have a set of at least $\ell^{\Omega(n)}$ packets queued at the nodes of G.

4.1.3. *The Bound for SIS*. The proof for SIS is more involved and is presented now. First, we define the graph G on which the proof works. 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 $0 \le i \le m - 1$, and with an edge e_m from node m to node $m + 1$. Choose an $\varepsilon \le 1/(m + 2)$ and an $s \ge 2m + 1$, and construct a tree T such that an adversary $\mathcal{A}$ with rate $1 - \varepsilon$ can inject $(1 - \varepsilon)s$ packets during an interval of s steps with the following property. They are injected at the leaves of T and they all reach the root of T in the last step of the interval. By a similar argument to the proof of Lemma 2.9, if $s = O(m)$, then T can be constructed with $O(m^2)$ 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 first 2^m 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 $e_0^{b_0}\, e_1^{b_1} \cdots e_{m-1}^{b_{m-1}}\, e_m$, 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 $0 \le j \le m - 1$.

Let us define $k_0 = (1 - \varepsilon)s$, and $k_j = 2k_{j-1} - \varepsilon s 2^{j-1}$ for $1 \le j \le m$. The crucial fact is the following:

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 $e_j^{b_0} e_{j+1}^{b_1} \cdots e_{m-1}^{b_{m-j-1}}\, e_m$. All these packets are in nodes of the subgraph L of G.*

PROOF. We shall use induction on j. The claim is trivially true for $j = 0$ since, by the definition of $\mathcal{A}$, at the end of Phase i there are $(1 - \varepsilon)s = k_0$ packets in node 0 all requiring edges $e_0^{b_0} e_1^{b_1} \cdots e_{m-1}^{b_{m-1}}\, e_m$, where $b_{m-1} \cdots b_0$ is the m-bit binary representation of i.

Let us now assume the result holds for some j and consider some i_{j+1} whose $(m - j - 1)$-bit binary representation is $b_{m-j-2} \cdots b_0$. Let $i_j^0 = 2i_{j+1}$ and let $i_j^1 = 2i_{j+1} + 1$. Then, the $(m - j)$-bit binary representation of i_j^0 is $b_{m-j-2} \cdots b_0 0$ and the $(m - j)$-bit binary representation of i_j^1 is $b_{m-j-2} \cdots b_0 1$.

From the induction hypothesis, at the end of Phase $2^j(i_j^0 + 1) - 1$, there are k_j packets in the nodes of L requiring $e_j^0 e_{j+1}^{b_0} \cdots e_{m-1}^{b_{m-j-2}}\, e_m$. Since $i_j^0 + 1$ is an

TABLE III. ASYMPTOTIC BOUNDS FOR THE SIMPLE UNIVERSALLY STABLE PROTOCOLS CONSIDERED

Protocol	Bound on queue size and delay
Farthest-to-Go (FTG)	$\Theta(exp(d))$
Nearest-to-Source (NTS)	$\Theta(exp(d))$
Shortest-in-System (SIS)	$\Theta(exp(d))$
Longest-in-System (LIS)	$O(exp(d))$

odd number, the m-bit binary representation $b_{m-1} \ldots b_0$ of any $i \in \{2^j(i_j^0 + 1), \ldots, 2^j(i_j^0 + 1) + 2^j - 1\}$ has the bit $b_j = 1$. Hence, during these 2^j phases all the packets injected requiring e_j^0 are single-edge injections. Therefore, during these 2^j phases, there are $(1 - \varepsilon)s2^j$ packets injected that require the single edge e_j^0. Under SIS new injections have higher priority; hence at the end of Phase $2^j(i_j^0 + 1) + 2^j - 1 = 2^j(i_j^1 + 1) - 1$ there are at least $k_j - s2^j + (1 - \varepsilon)s2^j = k_j - \varepsilon s2^j$ packets in the nodes of L still requiring edges $e_j^0 e_{j+1}^{b_0} \cdots e_{m-1}^{b_{m-j-2}} e_m$.

Also by the induction hypothesis, at the end of Phase $2^j(i_j^1 + 1) - 1$ there are at least k_j packets in nodes of L requiring edges $e_j^1 e_{j+1}^{b_0} \cdots e_{m-1}^{b_{m-j-2}} e_m$. Therefore, there are at least $2k_j - \varepsilon s2^j = k_{j+1}$ packets in the nodes of L requiring edges $e_{j+1}^{b_0} \cdots e_{m-1}^{b_{m-j-2}} e_m$ at the end of Phase $2^j(i_j^1 + 1) - 1 = 2^{j+1}(i_{j+1} + 1) - 1$.

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 $k_m = 2^m k_0 - m\varepsilon s2^{m-1} = s2^{m-1}(2 - \varepsilon(m + 2)) \geq (2m + 1)2^{m-1}$. There are only $2m + 1$ queues where these packets can be held, hence some queue contains at least 2^{m-1} packets.

Note that the construction for SIS uses an adversary of rate $1 - \Theta(1/m)$.

4.2. A RANDOMIZED GREEDY PROTOCOL WITH POLYNOMIAL QUEUE SIZE AND DELAY. In Table III, we present the asymptotic bounds we have found for the simple protocols studied. Note that FTG, NTS and SIS require exponential queue size and delay, while we have not been able to prove otherwise for LIS.

In this section, we present a randomized greedy protocol with polynomially bounded queues, and hence polynomially bounded delays. 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 $\gamma > 1$, the probability that at time t there are more than $\gamma p(m)$ packets in any queue of the system $(G, \mathcal{A}, \mathcal{P})$ is exponential in $-\gamma$.

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.

4.2.1. *The Definition of the Protocol.* Let $\mathcal{A}$ be an adversary of rate (b, 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 $T'\lceil t/T \rceil + \lambda(p)$, 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.

4.2.2. *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 nonnegative 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.*

PROOF. We prove by induction on t that for any packet $p \in X$, p is at least as far along its path at time t under σ' as it is under σ. For suppose this is false, and consider the smallest t for which there is a packet p that is farther along under σ than under σ'. Then by the minimality of t, we know that p did not move at time t under σ'—suppose it waited in the queue for edge e—and that p crossed e at time t under σ.

Since σ' is a greedy schedule, it must be that some packet $q \neq p$ traversed the edge e at time t. By the inductive definition of σ', q must have crossed edge e strictly *before* time t under σ. Thus, at time $t - 1$, q has not yet crossed e under σ', but has crossed e under σ; this contradicts the minimality of t.

4.2.3. *A Static Protocol*. In setting up the analysis of our dynamic protocol, it is helpful to consider first a randomized (nongreedy) protocol for the static routing problem—that is, 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 et al. [1994].

Let us 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 et al. [1994] 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 $[1, \alpha c/\log(mcd)]$, where α is a (small) constant. Let us define $\beta = 1 + \epsilon$ and divide the routing time into *intervals* of $\beta/\alpha \log(mcd)$ 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 $\beta/\alpha \log(mcd)$ packets try to cross some edge e in some interval i.

The probability that the algorithm fails can be bounded using Chernoff bounds. Let $N_{e,i}$ be the random variable denoting the number of packets trying to cross edge e in interval i. The expected value of $N_{e,i}$ is at most $\log(mcd)/\alpha$. Thus

$$\Pr\left[N_{e,i} > \frac{\beta}{\alpha}\log(mcd)\right] \leq \exp\left(1 - \frac{1}{\beta} - \ln\beta\right)\frac{\beta}{\alpha}\log(mcd)$$
$$= (mcd)\beta(1 - \ln\beta) - 1/\alpha \ln 2.$$

The probability that the algorithm fails can be crudely upper-bounded by multiplying the above expression by the number of possible choices of e (at most m) and by the number of possible choices of i (at most $d + \alpha c/\log(mcd)$). Thus, assuming c, $d \geq 2$ and $\alpha \leq 1$, the failure probability is at most

$$(mcd)(\beta(1 - \ln\beta) - 1)/\alpha \ln 2 + 1;$$

having chosen β, we can choose α small enough that

$$\frac{\beta(1 - \ln\beta) - 1}{\alpha \ln 2} + 1 < -1, \tag{1}$$

and hence force the probability of failure to be 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

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

4.2.4. *The Analysis of the Dynamic Protocol.* Let us now go back to the dynamic problem and consider an adversary $\mathcal{A}$ of rate (b, r). We may assume without loss of generality that $r \geq 1/2$, since an adversary of rate (b, r') with $r' < 1/2$ is also an adversary of rate $(b, 1/2)$. We will also assume that $m \geq 2$ and $d \geq 2$.

We define $\beta = 1 + (1 - r)/8r$ and then choose α small enough relative to β as in the preceding subsection. Note that

$$\beta^2 < 1 + \frac{1 - r}{2r} < \frac{1}{r}.$$

Finally, we choose a value of T large enough so that

$$T > \beta\left(\beta(rT + b) + \frac{d\beta}{\alpha}\log(md(rT + b))\right).$$

It is easy to check that we may choose such a T that is $\Theta(d \log m/(1 - r))$, and we will assume the constant of proportionality in the $\Theta(\cdot)$ is chosen large enough that $T(1 - (1/\beta)) \geq 4$.

We picture time as being divided into *blocks* of length T, and consider the set of packets X_i injected in the ith 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 $rT + b$. Thus, in the definition of the protocol at the beginning of this section, we set $\mu = \alpha(rT + b)/\log(md(rT + b))$, and $T' > \mu + d$. The effect of this definition of T' is the 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

$$\beta^{-1}T > \beta(rT + b) + \frac{d\beta}{\alpha}\log(md(rT + b)).$$

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

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 positions and labels at time τ_i. 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 $1 - (md(rT + b))^{-1}$; and otherwise it goes up by at most $cd \leq d(rT + b)$. Thus, the expected change in δ_i is less than or equal to

$$\begin{aligned}
&-\left(1 - \frac{1}{md(rT + b)}\right)T\left(1 - \frac{1}{\beta}\right) + \frac{1}{m} \\
&\leq -\frac{3}{4} \cdot 4 + \frac{1}{2} \\
&< -2.
\end{aligned}$$

Thus, the sequence of random variables $\delta_1, \delta_2, \delta_3, \ldots$ exhibits the following properties. The quantity $\delta_{i+1} - \delta_i$ is never greater than $d(rT + b)$; and when δ_i exceeds the constant $T(1 - 1/\beta)$, the expected value of $\delta_{i+1} - \delta_i$ is less than -2. Now a standard result in stochastic processes (see Hajek [1982] and Kahale and Leighton [1995]) implies that the probability of δ_i exceeding $\gamma d(rT + b)$ is exponential in $-\gamma$, independent of i.

Since $T \geq rT + b$, it follows that at any time t, the probability of there being more than γd nonempty sets of packets X_i is exponential in $-\gamma$, 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 and end-to-end delays. We feel it is of considerable interest to determine whether such a protocol exists. (We 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.) In recent work, Andrews and Zhang [2000] show that LIS can produce delays of $\Omega(e^d)$. More generally they show that for any deterministic protocol that selects the packet to advance independently of the packet routes, the delay bound cannot be better than $O(e^{\sqrt{d}})$. Hence, a deterministic, distributed protocol with polynomially bounded end-to-end delays must take the packet routes into account.

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, NTG, and FFS can be unstable at any injection rate greater than $1/\sqrt{2}$; a recent result of Borodin et al. [2001] shows that there exist adversaries of arbitrarily small positive rates that cause these protocols 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 (b, r_0), for every b and every network? Similarly, does one of FTG, NTS, or SIS become polynomially bounded when the injection rate is made small enough?

Finally—the assumption that a packet is injected with a prespecified path through the network is fairly standard within the context of queueing theory; however, for packet-routing problems, an alternative approach is 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 and Leighton [1994] multicommodity-flow algorithm. In recent work, following the initial appearance of these results, Aiello et al. [1998] and Gamarnik [1999] have examined this problem; they present

adaptive routing algorithms that achieve stability in all networks, against all bounded adversaries.

ACKNOWLEDGMENTS. We thank Allan Borodin for helpful discussions.

REFERENCES

AIELLO, W., KUSHILEVITZ, E., OSTROVSKY, R., AND ROSÉN, A. 1998. Adaptive packet routing for bursty adversarial traffic. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (Dallas, Tex., May 23–26). ACM, New York, pp. 359–368.

ANDREWS, M., AWERBUCH, B., FERNANDEZ, A., KLEINBERG, J., LEIGHTON, T., AND LIU, Z. 1996. Universal stability results for greedy contention-resolution protocols. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 380–389.

ANDREWS, M., AND ZHANG, L. 2000. The effects of temporary sessions on network performance. In *Proceedings of the 11th Annual ACM–SIAM Symposium on Discrete Algorithms* (San Francisco, Calif., Jan. 9–11). ACM, New York, pp. 448–457.

AWERBUCH, B., AND LEIGHTON, F. T. 1994. Improved approximations for the multi-commodity flow problem and local competitive routing in networks. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing* (Montreal, Que., Canada, May 23–25). ACM, New York, pp. 487–498.

BORODIN, A., KLEINBERG, J., RAGHAVAN, P., SUDAN, M., AND WILLIAMSON, D. P. 2001. Adversarial queueing theory. *J. ACM 48*, 1 (Jan.), 13–38.

BRODER, A., AND UPFAL, E. 1996. Dynamic deflection routing on arrays. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (Philadelphia, Pa., May 22–24). ACM, New York, pp. 348–355.

BRODER, A. Z., FRIEZE, A. M., AND UPFAL, E. 1996. A general approach to dynamic packet routing with bounded buffers. In *Proceedings of the 37th Annual IEEE Foundations of Computer Science*. IEEE Computer Science Press, Los Alamitos, Calif.

CRUZ, R. L. 1991a. A calculus for network delay. Part I: Network elements in isolation. *IEEE Trans. Inf. Theory 37*, 1 (Jan.) 114–131.

CRUZ, R. L. 1991b. A calculus for network delay. Part II: Network analysis. *IEEE Trans. Inf. Theory 37*, 1 (Jan.) 132–141.

GAMARNIK, D. 1998. Stability of adversarial queues via fluid models. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif. pp. 60–70.

GAMARNIK, D. 1999. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing* (Atlanta, Ga., May 1–4). ACM, New York, pp. 206–214.

GOEL, A. 1997. Stability of networks and protocols in the adversarial queueing model for packet routing. Tech. Rep. STAN-CS-97-59, Stanford Univ. Stanford, Calif.

HAJEK, B. 1982. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Adv. Appl. Prob. 14*, 502–525.

HARCHOL-BALTER, M., AND BLACK, P. E. 1994. Queueing analysis of oblivious packet-routing algorithms. In *Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms* (San Francisco, Calif., Jan. 9–11). ACM, New York, pp. 448–457.

HARCHOL-BALTER, M., AND WOLFE, D. 1995. Bounding delays in packet-routing networks. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing* (Las Vegas, Nev., May 29–June 1). ACM, New York, pp. 248–257.

KAHALE, N., AND LEIGHTON, T. 1995. Greedy dynamic routing on arrays. In *Proceedings of the 6th Annual ACM–SIAM Symposium on Discrete Algorithms* (San Francisco, Calif., Jan. 22–24). ACM, New York, pp. 558–566.

KELLY, F. P. 1979. *Reversibility and Stochastic Networks*. Wiley, New York.

KLEINROCK, L. 1975. *Queueing Systems*. Wiley, New York.

LEIGHTON, T. 1990. Average case analysis of greedy routing algorithms on arrays. In *Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures* (Island of Crete, Greece, July 2–6). ACM, New York, pp. 2–10.

LEIGHTON, F. T., MAGGS, B. M., AND RAO, S. B. 1994. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica 14*, 2, 167–186.

MITZENMACHER, M. 1994. Bounds on the greedy routing algorithm for array networks. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures* (Cape May, N.J., June 27–29). ACM, New York, pp. 346–353.

OSTROVSKY, R., AND RABANI, Y. 1997. Universal O(congestion + dilation + $\log^{1+\epsilon} N$) Local Control Packet Switching Algorithm. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (El Paso, Tex., May 4–6). ACM, New York, pp. 644–653.

RABANI, Y., AND TARDOS, É. 1996. Distributed packet switching in arbitrary networks. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (Philadelphia, Pa., May 22–24). ACM, New York, pp. 366–375.

ROBERTSON, N., AND SEYMOUR, P. D. 1986. Graph minors. V. Excluding a planar graph. *J. Combinat. Theory, Ser. B 41*, 92–114.

ROBERTSON, N., AND SEYMOUR, P. D. 1990. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Combinat. Theory, Ser. B 48*, 227–254.

ROBERTSON, N., AND SEYMOUR, P. D. 1995. Graph minors. XIII. The disjoint paths problem. *J. Combinat. Theory, Ser. B 63*, 65–110.

SCHEIDELER, C., AND VÖCKING, B. 1996. Universal continuous routing strategies. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures* (Padua, Italy, June 24–26). ACM, New York, pp. 142–151.

STAMOULIS, G. D., AND TSITSIKLIS, J. N. 1994. The efficiency of greedy routing in hypercubes and butterflies. *IEEE Trans. Commun. 42*, 11 (Nov.), 3051–3061.

TASSIULAS, L., AND GEORGIADIS, L. 1996. Any work-conserving policy stabilizes the ring with spatial re-use. *IEEE/ACM Trans. Netw. 4*, 2 (Apr.), 205–208.

RECEIVED JANUARY 1999; REVISED AUGUST 2000; ACCEPTED AUGUST 2000