Dynamic Routing on Networks with Fixed-Size Buffers

Extended Abstract

William Aiello* Rafail Ostrovsky† Eyal Kushilevitz ‡ Adi Rosén§

Abstract

The combination of the buffer size of routers deployed in the Internet and the Internet traffic itself leads routinely to routers dropping packets. Motivated by this, we initiate the rigorous study of dynamic store-and-forward routing on arbitrary networks in a model in which dropped packets must explicitly be taken into account. To avoid the uncertainties of traffic modeling, we consider arbitrary traffic on the network. We analyze and compare the effectiveness of several greedy, on-line, local-control protocols using a *competitive analysis of the throughput*. One goal of our approach is for the competitive results to continue to hold as a network grows without requiring the memory in the nodes to increase with the size of the network. Thus, in our model, we have *link buffers of fixed size*, B, which is independent of the size of the network, and B becomes a parameter of the model.

Our results are in contrast to another adversarial traffic model known as Adversarial Queuing Theory (AQT), which studies the stability and growth rate of queues as a function of the network and traffic parameters. For example, in AQT the Furthest-To-Go (FTG) protocol is stable for all networks whereas Nearest-To-Go (NTG) can be unstable for some networks. Unlike AQT, in our setting NTG is preferable to FTG: we show that the NTG protocol is throughput-competitive on all networks whereas the FTG protocol has unbounded competitiveness whenever a network contains even small cycles.

1 Introduction

Packet data networks, such as the Internet, are becoming the dominant technology over which traffic of various types is being carried, and are thus a central topic of study in networking research. In this paper, we are interested in the problem of store-and-forward routing and scheduling on arbitrary networks. In particular, we are interested in *dynamic* routing in which packets arrive continuously and are routed in a continuous fashion.

In the Internet, the combination of the buffer size of deployed routers and the Internet traffic itself leads routinely to the dropping of packets. Motivated by this, we initiate the rigorous study of dynamic store-and-forward routing on arbitrary networks in a setting in which dropped packets must explicitly be taken into account. To avoid the uncertainties of traffic modeling or the assumption of traffic shaping on ingress, we consider arbitrary traffic on the network. We analyze and compare the effectiveness of several greedy, on-line, local-control protocols using the competitive ratio of the throughput.[1] One goal of our approach is for the competitive results to

*AT&T Labs-Research. email: aiello@research.att.com.

†Telcordia Technologies. e-mail: rafail@research.telcordia.com

‡Dept. of Computer Science, Technion, Haifa 32000, Israel. email: eyalk@cs.technion.ac.il. Part of this research was done while the author was a visiting scientist at IBM T.J. Watson Research Lab.

§Dept. of Computer Science, Technion, Haifa 32000, Israel. email: adiro@cs.technion.ac.il

[1]The measure for the number of packets dropped is similarly defined, but as we show in the sequel no online protocol can have a finite competitive ratio for this measure.

continue to hold as a network grows without requiring the memory in the nodes to increase with the size of the network. Thus, in our model the link buffers have a fixed size, B, independent of the size of the network, where B becomes a parameter of the model. Using this model, we analyze several simple and popular contention-resolution and scheduling mechanisms on several classes of networks.

Although our model can be easily generalized to the case of adaptive routing, for this paper we concentrate on the case of non-adaptive routing. In this case the routing algorithm can be divided into two separate steps: (1) selection of paths for packets; and (2) selection of packets contending for an output buffer and scheduling of the packets selected for the output buffers. Such a separation has been used in a wide range of investigations. See, for example, [28, 3, 15, 13] for path selection and [2, 6, 17, 18, 25, 26, 27, 23] for packet-scheduling. Our on-line protocols will have responsibility for the latter part. In our model, as in the AQT model, the path selection step is effectively done by an adversary.

Before describing our model in more detail, we first discuss several issues with previous approaches that our model attempts to resolve.

1.1 Previous Approaches The vast majority of dynamic routing modeling and analysis in the literature can be classified as *stability* analysis. The general framework for such analysis is as follows. First a parameterized model, either probabilistic or adversarial, of packet injections into the network is specified. The former is referred to as queuing theory and the latter as adversarial queuing theory (cf. [7, 2]). Second, it is assumed that no packets are dropped. A routing and/or contention-resolution algorithm is analyzed to bound the maximum number of packets ever waiting to use a link, as a function of the injection model parameters and network parameters. This number is referred to as the buffer size. If, for a certain range of injection parameters, the upper bound on the size of the buffers depends only on the injection and network parameters (and not on the elapsed time), then the protocol is said to be stable (for that range of injection parameters). Ideally, bounds on the latency (i.e., the time between injection and delivery) of the packets are also derived. [2]

Stability analysis can be used to provision memory in the routers or switches of a packet network as follows. Assume that the maximum size of the network is known a priori. Further, assume that the selected traffic model and parameters approximate or bound the behavior of the real traffic. Given these assumptions, stability analysis yields an upper bound $\mathcal{B}$ on the maximum size of the buffers. When the network is first provisioned or when nodes are added to the network (up to the maximum), the routers are equipped with buffers at least as large as $\mathcal{B}$. In this case, no packets will be dropped and the latency bounds given by the analysis will hold. But, when trying to apply stability analysis to a packet network like the Internet, the two assumptions above likely do not hold, giving rise to two major issues: *scalability* and *traffic modeling*, respectively. The resolution of each of these problems requires one inevitably to deal with packet drops. We discuss these two issues below.

To illustrate the scalability problem for stability analysis, consider the situation in which the traffic obeys the assumptions of the traffic model but in which the network continues to grow over time without an a priori upper bound. All stability analyses of which we are aware for reasonably general networks, yield bounds on buffer sizes that are increasing functions of the network parameters.[3] This is the heart of the scalability problem. For large networks or those that span several administrative domains, it is not realistic to add enough memory to every

[2] Since no packets are dropped, the asymptotic fractional throughput of a stable algorithm is 1 even if the latency is unbounded for some packets.

[3] Of course, traffic could be constructed so that the buffers remain constant, e.g., traffic for which all paths are of constant length or for which the arrival rate is a decreasing function of the network size. But we do not assume such restricted traffic patterns.

router in the network in order to continue to obey the stability bounds whenever the network grows, and in practice this is simply not done. Thus, as a network continues to grow, eventually some buffers will be smaller than the bounds given by stability analysis for the new network size. Once this occurs, some packets will be dropped and the assumptions of stability analysis used, say, to derive bounds on latency, will no longer hold.

To illustrate the traffic modeling problem for stability analysis, suppose that the network remains fixed and that all the nodes in the network are provisioned with sufficient memory according to some stability analysis for a certain traffic model. In this case, if the traffic does not follow the restriction of the traffic injection model, stability analysis will give very little insight into the behavior of the network. In particular, nodes may need buffers of sizes larger than the stability analysis postulates, and will not have enough buffer space to store all needed packets. Moreover, it appears to be difficult to formulate models of Internet traffic that are both accurate and tractable for further analysis (see e.g. [31]). In addition, the nature of Internet traffic continues to evolve and change so that even if there were an accurate model now, it may be inaccurate in the future. As an example, the last two years have seen a tremendous increase in peer-to-peer traffic. Therefore, it seems desirable to control the network using protocols that are known to have good performance not only for some specific well defined traffic pattern but for more arbitrary traffic as well.

Due to both the scalability issue and the traffic modeling issue, it is essentially guaranteed that, irrespective of the contention-resolution protocol in the routers, there will be times when the traffic is such that the buffer at the output port of a link becomes full and the router will have to drop packets. Moreover, empirically, the size of router buffers and Internet traffic are such that routers routinely drop packets. Neither probabilistic queuing theory nor adversarial queuing theory are currently equipped to handle dropped packets.

1.2 Our Approach In this paper we develop a scalable model for the analysis of routing protocols on packet networks, that explicitly addresses dropped packets. To do so, we start with the assumption that the sizes of a router's buffers are fixed when the router is provisioned in the network and are not increased when other routers are added to the network. That is, the size of the buffers do not depend on the size of the network. For simplicity, assume that all buffers are of the same size, denoted B. We treat B as a basic parameter of our model. We place no restrictions on the traffic injected into the network. In particular, we certainly do not assume that traffic will be so well-behaved that the routing algorithm can always avoid dropping packets, or that traffic never overloads links or nodes. Thus, one must fundamentally deal with dropped packets in the analysis. The goal is to develop algorithms that have high throughput or low drop rate. The performance in terms of throughput or drop rate will be a function of the network parameters, and the buffer size. As the network grows and routers are added to the network, analysis under this approach will continue to give guarantees on throughput or drop rate.

We are interested in simple local-control contention-resolution algorithms. As stated earlier, we consider dynamic routing where traffic is continuously injected into the network. Given these elements, combined with the fact that the buffers of the nodes have fixed size, the algorithms in each node have to decide in an online manner which packets to drop, and what forwarding-priorities to assign the remaining packets.

To measure the quality of a given algorithm, we use *competitive analysis* [5]. Competitive analysis, applied to this setting, compares the throughput of the local-control, online routing algorithm to that of a centralized, optimal, off-line policy, that uses the same buffers and is given the entire packet arrival sequence in advance. Our aim is to find online queue policies with a small competitive ratio. For a full definition of the model, see Section 2. The competitive ratio for the number of packets dropped can also naturally be defined. However we will show that, in

	NTG	FFO	LIS	FTG	NTO	SIS	FIFO
Compet.	Yes	Yes	Yes	No	No	No	No
Stable	No	No	Yes	Yes	Yes	Yes	No

Table 1: A comparison of several deterministic greedy protocols. The rows denote whether the protocols are throughput-competitive or AQT stable for *all* networks.

terms of drops, the competitive ratio of any online policy is unbounded, and hence this is not a useful measure.

Several recent papers on queuing protocols, this paper included, have used models that share similar properties: the queue size is parametric, the traffic input is arbitrary, and the metric used is the competitive ratio [1, 10, 14, 20]. The papers thus far have studied the throughput (or weighted throughput in the DiffServe model) of a single router or switch. To the best of our knowledge, our paper is the first to apply competitive analysis in this context to the question of throughput for a *network* of routers.

1.3 Our Contributions Our contributions are twofold. The first is the formulation of the present approach and the initiation of a rigorous study of protocol performance in a network setting that explicitly deals with dropped packets. We call our approach the Competitive Network Throughput (CNT) Model. The second are the following main results that we derive within our model:

1. Protocols competitive on all networks. We show that there exist greedy local-control protocols that are throughput-competitive on all networks. In particular, we show that the protocol Nearest-To-Go (NTG) is throughput competitive on all networks (See Theorem 4.1.) Similar proofs hold for the protocols Furthest-From-Origin (FFO) and Longest-In-System (LIS) (and are omitted from this abstract). See Table 1. However, not all greedy protocols are throughput competitive on all networks as we discuss below.

2. Universally competitive networks. We give a characterization of the networks on which *all* greedy protocols are throughput-competitive. We show that on DAGs all greedy protocols are competitive (See Theorem 4.2). We then show that when a network has a cycle, some greedy protocols are not competitive. In particular, the competitive ratio of the Furthest-To-Go (FTG) protocol on a cycle is unbounded (See Theorem 4.4.). Similar proofs hold for the protocols Nearest-To-Origin (NTO), Shortest-In-System (SIS) and First-In-First-Out (FIFO) (and are omitted from this abstract). See Table 1.

3. The topology of the line. We analyze the competitive ratios attainable on the topology of the line. We show that NTG achieves a competitive ratio of $O(n^{\frac{2}{3}})$ (where n in the number of nodes in the line), while other protocols such as FTG and Longest-In-System (LIS) have a lower bound of $\Omega(n)$. Furthermore, Nearest-To-Go is not far from optimal since no greedy protocol can have a competitive ratio better than $\Omega(\sqrt{n})$. This lower bound is complemented by a lower bound of $\Omega(n)$ for the case of $B = 1$. (See Section 5.)

Consider the side-by-side comparison of some of our results to those of adversarial queuing theory (AQT) [7, 2, 9], given in Table 1. Interestingly, this small sample of basic protocols displays all four possible combinations of competitive/not-competitive with stable/not-stable. In particular, there are protocols that are stable on all networks but have unbounded competitive ratio on some networks, e.g., FTG, NTO, and SIS. Conversely, there are protocols that are unstable on some networks but are competitive on all networks, e.g., NTG and FFO.

Our results thus suggest, perhaps surprisingly, that some protocols that are unstable i.e., are deemed undesirable from the perspective of stability analysis, may provide good throughput,

and may be preferable to some stable protocols, when the environment is such that packets must inevitably be dropped. Our results thus provide evidence that for routing on scalable networks with fixed-size buffers and ill-behaved traffic, traditional stability analysis may not be the right means for comparing alternative routing protocols.

1.4 Related Work We briefly review related work on dynamic store-and-forward routing on arbitrary networks. As explained above, the analysis of dynamic packet arrivals has concentrated on the questions of stability and queue sizes. For such analysis, some type of limit must be placed on the traffic injected into the network. Without a restriction, the injection rate can exceed the maximum bandwidth bisection of the network and, hence, the size of the buffers will grow as a function of time. One standard method for limiting the injections is to model the packet injections by a probabilistic process, such as a Poisson arrival process (with sufficiently small rate) at each node with destinations chosen independently and uniformly at random. In this case the performance is often measured in terms of the *expected* latency and buffer size. See, for example, [8, 6, 11, 12, 30, 29, 21, 22].

Another approach is the adversarial queuing theory (AQT), proposed by Borodin et al. [7]. AQT limits the packet traffic in terms of two parameters w and $r \leq 1$. For all time windows of size w, and for all edges e, the number of packets injected during that window of time by the adversary and whose paths include e cannot exceed rw. Borodin et al. and Andrews et al. [7, 2] show that several well-known and simple deterministic greedy queuing protocols yield buffers (and latencies) which are bounded (however, these bounds are exponentially large). On the other hand they show that certain other greedy protocols do not guarantee stability. A subset of their stability results, of particular relevance for comparing our results, are given in Table 1 along with our corresponding results for throughput-competitiveness.

Recently, Awerbuch et al. [4] reported results in a model somewhat similar to ours. As in our model, they allow arbitrary traffic to be injected into the network and use competitive analysis to measure the throughput of on-line algorithms. However, contrary to our work, [4] compares the performance of on-line algorithms to the performance of an adversary restricted to a limited class of routing strategies, rather than to the performance of an unrestricted optimal adversary as in the present paper. In addition, they allow the size of each queue used by the on-line algorithm to be a multiple of the queue size used by the adversary, and for the reported results, this multiple is an increasing function of network parameters.

Organization The rest of the paper is organized as follows. In Section 2, we formally define the model. In Section 3, we show that no algorithm can have a finite competitive ratio for the measure of packet loss. We therefore adopt the measure of throughput for our study. In Section 4, we examine the competitiveness of the NTG protocol. We then show that on DAGs any greedy protocol is competitive, while the cycle does not exhibit this phenomenon. In Section 5 we analyze the competitive ratio achievable on the topology of the line. Due to space limitations, substantial parts of the proofs are omitted from this abstract.

2 The Competitive Network Throughput Model

We model a packet network as a directed graph $G = (V, E)$, $|V| = n$, $|E| = m$, where nodes represent routers and edges represent unidirectional communication links. The system is synchronous. Time proceeds in discrete time steps. All links have unit capacity: in each time step, each link can transmit one unit-sized packet in the direction it is oriented. Each directed link is identified with an output port in the router at the tail of that link and with an input port in the router at the head of that link. At the output port, there is a buffer of size $B > 0$ which can store up to B unit sized packets. At the input port, there is a buffer of size one.

In addition to the input ports and output ports associated with network links, each node

may also have several *traffic input* ports. When packets are injected into the network, they are injected into the traffic input ports. For the sake of uniformity, the buffers of the traffic input ports, like the link input buffers, are of size one, i.e., one packet may be injected into a traffic input port in every time step. Denote by I the maximum number of traffic input ports in any node in the network.

Each packet is injected into a traffic input buffer of a source node and is labeled with a given destination. Packets travel in the network in a store-and-forward manner. During each time step a single packet from each output buffer is forwarded to the input buffer at the head of the link. Then, packets that have not arrived yet to their destination are moved from input buffers to output buffers within each node. If the number of packets destined for a buffer is greater than the free space in the buffer then some packets must be dropped. At the end of each time step, the link and traffic input buffers are empty, and each packet present in the network resides in some link output buffer.

While our model can be adapted to the case of adaptive routing, in this paper we consider the case of non-adaptive routing. In this case the selection of the path for each packet is separate from the scheduling and contention-resolution algorithms. Whenever a packet is injected into the network, its path is determined (either implicitly by routing tables or by explicitly encoding it in the packet itself). We assume that all paths are simple.

Each time step is divided into two substeps: the forwarding substep and the switching substep. In the forwarding substep, for each link, a packet may be selected from the output buffer at the tail of the link. Then the selected packet is forwarded to the input buffer at the head of the link. This substep also includes the injection of at most one packet into each injection buffer. The method for deciding which packet to forward is called the *output scheduling policy*.

In the switching substep, within each router, all the link and traffic input buffers are emptied of their packets. The packets from the input buffers are either transferred to the output buffer of the next link on their path or dropped (or simply output if the node is the packet's destination). When more than B packets are to be placed in a certain buffer then some packets must be dropped. We distinguish between preemptive and non-preemptive policies. For preemptive policies we allow packets from the input buffers to preempt packets already in the output buffer. The preempted packets are dropped. For non-preemptive policies, a packet already in the buffer cannot be dropped from the buffer, and the packets to be dropped must be selected from the packets in the input buffers. We concentrate in this paper on preemptive policies. The method for deciding which packets should be dropped is called the *contention-resolution policy*

We are interested in algorithms that are online and local-control. These algorithms, henceforth called protocols, operate as a set of separate algorithms, one for each node. They make their decision in each time step based only on the information available in that node at that time step (without full view of the network, or knowledge of future arrival of packets).

Of particular interest are *greedy* protocols. A scheduling protocol (and the forwarding substep) is greedy if it always forwards a packet over an edge if there is at least one packet in the output buffer at the tail of that edge. A contention resolution protocol (and the switching substep) is greedy if it keeps each buffer as full as possible in each time step. A protocol is greedy if both the scheduling protocol and the contention-resolution protocol are greedy. We concentrate primarily on greedy protocols and mostly on protocols where the scheduling and contention-resolution policies are the same. That is, in each time step all the packets contending for a given link are ordered by their priority according to the specific policy. The first B packets are placed in the corresponding output buffer and the rest are dropped. The packet with highest priority will be the one that is forwarded over the link in the next time step.

We use competitive analysis to analyze our algorithms. For a minimization problem we say that algorithm A is c-competitive if for every sequence of packets σ injected into the network it

holds that $A(\sigma) \leq c \cdot OPT(\sigma) + \alpha$, where $A(\sigma)$ and $OPT(\sigma)$ are the costs of algorithm A on σ, and the cost of the optimal (centralized, clairvoyant) algorithm on σ, respectively, and α is a constant independent of σ. For a maximization problem the requirement is that $A(\sigma) \geq (1/c)OPT(\sigma) - \alpha$, where $A(\sigma)$ and $OPT(\sigma)$ are the benefits obtained by algorithms A and OPT as above. As we show below, there is no protocol with finite competitive ratio for the measure of the number of dropped packets. Hence, in our model, we use the competitive ratio of throughput as our measure.

We view I and B as characteristics of the routers and hence as network parameters. For a given set of routers with a given set of injection buffers, there is no restriction on the traffic injected into those buffers. The competitive ratio is a function of network parameters such as n, m, B, I, and perhaps others (e.g., the maximum degree of the nodes).

3 Loss of Packets

In this section, we consider the measure of the number of packets that are dropped by the protocol. We show that there is no competitive protocol for this measure. In fact, we show that there is no online algorithm, even centralized, that can guarantee a number of packet-drops that is bounded compared to what the optimal algorithm can do. This holds even for the case where the optimal algorithm is required to be greedy. We prove the above on a network which is a line of at least 4 nodes; it follows that the same impossibility result applies for every network that contains such a sub-network.

THEOREM 3.1. *Assume $I \geq 2$.* [4] *For the topology of the line there is no protocol with finite competitive ratio for the measure of the number of packets dropped.*

Proof Sketch. The proof takes the following approach. For every on-line protocol P, the adversary can inject a finite sequence such that P will drop at least one packet, while the adversary can schedule its own forwarding of the packets so that no packet is dropped and all packets are delivered. To do so the adversary creates a situation where for some buffer, protocol P has more packets in this buffer than the adversary. The adversary then injects to that buffer more packets than P can accept. The above process takes a finite amount of time and thus can be repeated indefinitely. Hence, the number of dropped packets for the on-line protocol is unbounded whereas the number of packets dropped by the optimal off-line algorithm is zero. Details omitted.

4 Throughput Competitiveness

From now on, we consider the measure of throughput; that is, the number of packets that are delivered by any given time t. We first show that there exist greedy protocols that are competitive for all networks. In particular, we show that the protocol Nearest-To-Go (NTG) has a competitive ratio of $O(md)$, where m is the number of edges in the network and d is the maximal length of a path traversed by any packet (Section 4.1). Similar proofs hold for FFO and LIS (details omitted).

We then show that there exist networks on which all greedy protocols are competitive. In particular, we show that on DAGs every greedy protocol has bounded competitive ratio. The universal upper bound on the throughput competitiveness is a function of the specific DAG. We further show that there is a family of DAGs for which this universal upper bound is nearly tight by showing a lower bound on the throughput competitiveness of all greedy algorithms for a complete binary tree. On the other hand, we show that networks that contain cycles render some greedy protocols non-competitive. In particular, the Furthest-to-Go (FTG) heuristic has

[4] If $I = 1$ then, for "most" networks, we can mimic the impossibility proof below.

unbounded throughput-competitiveness on the cycle (Section 4.4). The same result holds for NTO, SIS, and FIFO using a similar proof (details omitted). These results thus also yield a characterization of the network topologies on which all greedy protocols are competitive.

4.1 Nearest-to-Go is Competitive on all Networks In this section we prove the existence of protocols which are competitive for all networks. Specifically, we show an upper bound on the competitive ratio of the protocol NTG (that is, a protocol that uses the NTG priority decision both in the switching substep and in the forwarding substep). Let d be the length of the longest path followed by any packet. We show that NTG is $O(md)$ competitive on any network.

THEOREM 4.1. *The protocol NTG is $O(md)$-competitive for any network G.*

Proof. For the purpose of the analysis assign to each packet p, injected into the network by the adversary, a weight $w(p)$. The weight of p is 1 if this packet is eventually delivered by the adversary, and 0 otherwise. Let ADV^t be the total weight of packets injected by the adversary until time t. Note that ADV^t is an upper bound on the number of packets delivered by the adversary until time t. We start with a claim that gives a lower bound on the number of packets delivered by NTG in a given amount of time (proof omitted).

CLAIM 4.1. *For any $k \leq B$, if at time t NTG has at least k packets in the network, then NTG delivers at least k packets by time $t + dk$.*

We now divide the time axis into *time frames* of dB time steps each, where frame j is $F_j \stackrel{\triangle}{=} [(j-1)dB+1, jdB)$. Let a_j be the total weight of packets *injected* by the adversary during F_j. Let b_j be the number of packets *delivered* by NTG during F_j.

We claim that $b_j + b_{j+1} \geq \min\{a_j, B\}$, for any $j \geq 1$. To see that, distinguish between two cases: either (1) NTG holds in its buffers at some time during F_j at least B packets; or (2) at each time step during F_j NTG holds fewer than B packets. For the first case, let t be a time in F_j where NTG holds at least B packets. It follows from Claim 4.1 that NTG delivers by time $t + dB$ at least B packets. Since time $t + dB$ is in F_{j+1}, this shows that $b_j + b_{j+1} \geq B$. In the second case, NTG never drops a packet during F_j. It follows that if at least a_j packets are injected during F_j, then at the end of F_j NTG has in the network at least $a_j - b_j$ packets. Since there are fewer than B packets in the network at every time step in F_j, $a_j - b_j < B$. By Claim 4.1 at least $a_j - b_j$ packets are delivered during F_{j+1}. We get $b_j + b_{j+1} \geq b_j + (a_j - b_j) = a_j$.

For any time t, we now bound the number of packets injected by the adversary with weight 1 by time t, in terms of the number of packets delivered by NTG by time t. Let s be such that $(s-1)dB \leq t < sdB$. We have

$$2\sum_{j=1}^{s} b_j \geq \sum_{j=1}^{s-1}(b_j + b_{j+1}) \geq \sum_{j=1}^{s-1} \min\{a_j, B\}$$

Now observe that for any j, $a_j \leq mB + mdB \leq 2mdB$. This is because any packet injected with weight 1 cannot be dropped by the adversary. The number of packets injected but not dropped in time interval of length T time units is at most $mB + mT$ (since at most mT packets may be delivered and at most mB packets may be stored in the buffers). Therefore, $\sum_{j=1}^{s-1} \min\{a_j, B\} \geq \sum_{j=1}^{s-1} \frac{a_j}{2md} = \frac{1}{2md}\sum_{j=1}^{s-1} a_j$, and so $\sum_{j=1}^{s} b_j \geq \frac{1}{4md}\sum_{j=1}^{s-1} a_j$. Since $ADV^t \leq \sum_{j=1}^{s-1} a_j + 2mdB$, and $NTG^t = \sum_{j=1}^{s} b_j$, we have,

$$NTG^t = \sum_{j=1}^{s} b_j \geq \frac{1}{4md}\sum_{j=1}^{s-1} a_j \geq \frac{1}{4md} ADV^t - \frac{B}{2} .$$

□

4.2 A Universal Upper Bound for DAGs In this section we show that on DAGs any greedy protocol is competitive. The competitive ratio is a function of the specific DAG $G = (V, E)$. To define this ratio, we define for any $v \in V$: [5]

$$f(v) \stackrel{\triangle}{=} 1 + \sum_{e=(u,v)\in E} f(u).$$

and $f(G) \stackrel{\triangle}{=} \max_{v\in V} f(v)$.

THEOREM 4.2. *For any DAG G and any greedy protocol, the competitive ratio is at most $O(f(G))$.*

Proof. Let P^t be the number of packets delivered by the protocol by time t, and ADV^t be the number of packets delivered by the adversary by time t. We prove that for any t, $P^t \geq \frac{ADV^t}{O(f(G))} - O(mB)$.

For the purpose of the proof we define, in addition to the real packets that the protocol delivers, *virtual packets* and a mechanism that can "virtually deliver" them. These packets cross edges piggybacked on real packets that the protocol transmits and are stored in two types of special buffers, maintained at the tail of every edge: the *virtual holding buffer* and the *virtual transit buffer* . In the following we call a packet that the adversary eventually delivers a D-packet.

We now describe how to handle virtual packets already in the system, and then how virtual packets are created. When virtual packets arrive at node v piggybacked on a packet p, then if v is the destination of p the virtual packets are "virtually delivered". Otherwise let e be the next edge on the path of p. Then the virtual packets are placed in the virtual transit buffer of e. When a (real) packet p leaves some node v on edge e, then all the packets in the virtual transit buffer of e are piggybacked on p and the virtual transit buffer is emptied. In addition, two packets are extracted from the virtual holding buffer of e and are piggybacked on p as well (if there are less than two packets in the buffer, then less than two packets are taken).

Now we describe how virtual packets are created. We create new virtual packets to compensate for each packet that the adversary eventually delivers (i.e., a D-packet) but the protocol drops. Let t be a time step, and let e be an edge emanating from node v. We consider the switching substep of time step t at the tail of e. If no packet is dropped at the tail of e at t, then no new virtual packet is placed in the buffers of e. Assume that at least one packet is dropped. Let x be the number of packets that arrive at v on link input ports in the forwarding substep of t, and for which e is the next edge on their path. Let Δ be the number of D-packets injected into v in the forwarding substep of t and e is the first edge on their path. Let b be the number of packets in the buffer at the tail of e at the end of the forwarding substep. During the switching substep we create x new virtual packets and place them in the virtual transit buffer of e, and we create $\max\{0, \Delta - (B - b)\}$ new virtual packets and place them in the virtual holding buffer of e.

We now analyze the above mechanism. We first claim that no D-packet is ever "lost". That is, when we drop such packets, they are compensated by new virtual packets. We give the following claim.

CLAIM 4.2. *For any time t, the number of D-packets injected by the adversary by time t is at most the total number of real packets and virtual packets delivered by the protocol by time t, plus the total number of real packets and virtual packets stored in all output buffers (of all types) at time t.*

[5] The combinatorial interpretation of $f(v)$ is the number of directed path ending at v, including the null path.

Proof. The proof is by induction on time. For time $t = 0$, the claim is trivial. For any time step $t > 0$, assume the claim holds at the end of time step $t - 1$. We consider each node v and each edge e emanating from v separately. Let x, Δ and b as defined above and examine first the forwarding substep. At this substep, some new D-packets are injected for which e is the next edge on their path – these are included in the quantity Δ; all other packets that are forwarded during this substep have no influence, at this point, on the correctness of the claim as they just change their location (including those, real or virtual, packets that are delivered to their destinations). Next, consider the switching substep of time step t. If no packet is dropped at the tail of e at time step t, then all the $\Delta + x$ (real) packets are added to the buffers of e. Otherwise the buffer at the tail of e is full at the end of t. That is, $B - b$ packets are added to it. In addition we add x new virtual packets to the virtual transit buffer and $\max\{0, \Delta - (B - b)\}$ new virtual packets to the virtual holding buffer . The total number of packets added to the buffers of v in the switching substep is then $(B - b) + x + \max\{0, \Delta - (B - b)\} \geq x + \Delta$. This compensates for the vacated link input buffers, and the D-packets which were in the traffic input buffers. □

LEMMA 4.1. *Let e be an edge emanating from node v. Then, the maximum number of virtual packets piggybacked on a packet that crosses e is at most $3f(v) - 1$.*

Proof. Let the level of a node v be the length of the longest path leading to node v. We prove the claim by induction on the level ℓ of node v. For $\ell = 0$, observe that the virtual transit buffer of node v is always empty since there is no edge leading into v. Therefore, the number of virtual packets piggybacked on a packet that crosses e is at most 2 which proves the claim since $f(v) = 1$.

For $\ell > 0$, we first bound the number of virtual packets in the virtual transit buffer of e at any time step t. The virtual packets in this buffer are either virtual packets that arrived to v piggybacked on some packet that arrived on an edge $e' = (u, v)$ at time step $t-1$, or new virtual packets created at v at time $t - 1$. Since the level of u is less than ℓ then, by the induction hypothesis, the number of packets piggybacked on a packet that crosses e' is at most $3f(u) - 1$. The number of new virtual packets added to the virtual transit buffer in a single time step is at most the in-degree of v. It follows that the total number of packets in the virtual transit buffer of v at t is at most $\sum_{e=(u,v)\in E} 3f(u)$. The number of packets piggybacked on a packet that crosses e is therefore at most $\sum_{e=(u,v)\in E} 3f(u) + 2 \leq 3f(v) - 1$. □

Now, consider the virtual holding buffer at the tail of an edge e emanating from node v. Observe that in any consecutive B time steps the adversary can inject into v at most $2B - 1$ D-packets requiring e. This follows since at most one packet can leave the buffer at the tail of e in each time step, and more packets will overload the buffer, forcing the adversary to drop packets. Therefore, in any B consecutive time steps, the number of new virtual packets placed in the virtual holding buffer of e is at most $2B - 1$. At the same time, a new virtual packet is put in the virtual holding buffer of e at time t only if the buffer of e is full at t, and hence in the next B time steps a packet crosses edge e. It follows that the number of virtual packets in the virtual transit buffer is bounded by $O(B)$ at any time.

Next, consider the number of packets delivered by the greedy protocol and by the adversary by some time t. Clearly the number of packets delivered by the adversary by time t is bounded from above by the number of D-packets it injects by time t. Using Claim 4.2, Lemma 4.1 and the above bound on the size of the virtual holding buffer we have,

$$\begin{aligned} ADV^t &\leq P^t + 3f(G)P^t + O(m(f(G) + B)) \\ &\leq 4f(G)P^t + O(m(f(G) + B)) \ . \end{aligned}$$

We get $P^t \geq \frac{1}{O(f(G))} \cdot ADV^t - O(mB)$. □

4.3 A Universal Lower Bound for Trees We now show that there is a class of DAGs for which the above universal upper bound is nearly tight.

THEOREM 4.3. *On a tree of n nodes with edges directed towards the root and maximum path length h, the competitive ratio of every greedy algorithm is $\Omega(n/h)$.*

Proof omitted. Note that for complete binary trees, the upper and lower bound are within a factor of $O(\log n)$.

4.4 Furthest-to-Go is Not Competitive on the Cycle We have seen that on DAGs, all greedy algorithms are competitive. We have also seen that NTG is competitive for all networks. However, it is not the case that all greedy algorithms are competitive on all networks. We show below that if the rule for discarding packets is that the packet with the furthest destination has the highest priority (to be forwarded and not dropped), i.e. Furthest-To-Go (FTG), then the protocol is not competitive on the cycle.

THEOREM 4.4. *FTG is not competitive on the cycle.*

Proof. Consider a cycle with nodes numbered 0 to $n-1$. All packets injected into node i are destined to node $(i+2)$ `mod` n. The adversary operates by injecting in each time step a single packet into each of the nodes. Now observe that FTG never delivers any packet. To see that, observe that for a packet with destination node i to be delivered, the packet has to cross node $i-1$. But the adversary injects into node $i-1$ a constant stream of packets with destination $i+1$, which have priority over all packets with destination i. Thus, FTG never delivers any packet. The adversary however can deliver at least (roughly) half of the packets: it will accept and deliver all the packets injected into nodes i for (say) even i. □

We remark that it is possible to construct $O(n)$-competitive protocols for the cycle of n nodes. In fact any greedy protocol that drops packets only from traffic input ports and not from link input ports will be $O(n)$ competitive (note that since the in-degrees and out-degrees of all nodes on the cycle are 1 this is possible). Details omitted.

5 The Competitive Ratio on the Line

In this section we analyze the competitive ratio attainable on the topology of the line, and compare the performance of several protocols. Applying the result for general DAGs from Section 4.2 to the line of n nodes, we get that any greedy protocol is $O(n)$ competitive. For the case $B = 1$, this is best possible (up to constant factors), for any protocol. But when $B > 1$, better bounds can be achieved. We show that the protocol Nearest-To-Go (NTG) achieves a competitive ratio of $O(n^{\frac{2}{3}})$ on the line. On the other hand the protocols Longest-In-System (LIS) and Furthest-To-Go (FTG) have a lower bound of $\Omega(n)$. We further show that NTG is not far from optimum by providing a lower bound of $\Omega(\sqrt{n})$ for any greedy protocol.

THEOREM 5.1. *For every greedy protocol on the n-node line, the competitive ratio is $\Omega(n)$ if $B = 1$ and $\Omega(\sqrt{n})$ if $B > 1$.*

The proof is omitted. We now proceed to prove our upper bound for NTG.

THEOREM 5.2. *Consider a line network with $n+1$ nodes, and assume that $B > 1$. Then the protocol NTG is $O(n^{\frac{2}{3}})$ competitive.*

We number the nodes and edges from left to right, starting with 1 in both cases. Let $L = \lceil n^{\frac{2}{3}} \rceil$. Partition the line into intervals. We mark each edge iL, for $i \geq 1$, as a *border edge*. All the edges j for $(i-1)L < j < iL$ belong to *interval number* i. A packet is called *long* if its path contains at least one border edge; otherwise, it is called *short*. A packet for which the last edge on its path is in interval i is called an *interval i packet*. Let A^t denote the number of packets delivered by the adversary until time t. Let N^t denote the number of packets delivered by NTG until time t. Denote by A_s^t the number of short packets delivered by the adversary by time t. Denote by A_l^t the number of long packets delivered by the adversary until time t. Observe that $A^t = A_l^t + A_s^t$. Theorem 5.2 follows from the following two lemmas that we prove below: (1) $A_s^t \leq O(n^{\frac{2}{3}})N^t + O(nB + n^{\frac{5}{3}})$ (Lemma 5.1); and (2) $A_l^t \leq O(n^{\frac{2}{3}})N^t + O(nB + n^{\frac{5}{3}})$ (Lemma 5.5).

We call a packet that the adversary eventually delivers a D-packet. For the purpose of analyzing NTG we define, in addition to the real packets, *virtual packets* and a mechanism that virtually delivers some of these packets (this is a variant of the mechanism used in Section 4.2). Virtual packets cross edges piggybacked on real packets that cross edges according to the schedule of NTG . For the usage of this mechanism, we maintain at the tail of each edge i a *virtual transit buffer* and a *virtual holding buffer*. We first describe how we handle virtual packets already in the system, and then how new virtual packets are created. When a (real) packet p arrives at node i with piggybacked virtual packets on it, then if i is the destination of p, the virtual packets on it are *virtually delivered*. Otherwise, they are placed in the virtual transit buffer. When a packet leaves node i and crosses edge i, all the packets in the virtual transit buffer of i are piggybacked on it, and are extracted from this buffer. In addition, if the virtual holding buffer at the tail of i is not empty, 3 packets are taken from this buffer (or less, if there are fewer packets in this buffer), and piggybacked on the packet that crosses edge i. The virtual holding buffer is maintained using the FIFO policy. Next, we describe how new virtual packets are created. Let t be a time step and consider each node i separately. Let x be the number of packets arriving to i from $i-1$ at time t (x is either 0 or 1) and Δ be the number of D-packets injected into i at time t. Let Δ' be the number of packets injected into i at time t and never delivered by the adversary, and let b be the number of packets in the buffer of i at the beginning of the switching substep of t. If no packet is dropped at i then no new virtual packet is created. Otherwise, NTG orders all the packets in the output and input buffers according to their distance-to-destination, and puts the B packets with highest priority into the buffer of i. We convert the next $\max\{0, x+\Delta-(B-b)\}$ packets (which are some packets that NTG drops) into virtual packets and place them in the *virtual holding buffer* . We now give a claim pertaining to the number of packets and virtual packets that NTG maintains.

CLAIM 5.1. *For any time t, the number of D-packets injected by the adversary by t is at most the number of real packets and virtual packets delivered by NTG by time t, plus the number of real packets and virtual packets stored in the buffers of NTG at time t.*

Proof. The proof is by induction on time. It obviously holds at time 0. Consider time step t. We make the argument separately for each node i. Let x, Δ, Δ' and b be as defined above. During the forwarding substep NTG transfers packets (both real and virtual) which therefore might change their status from "undelivered" to "delivered" but this has no impact on the correctness of the claim; also in this substep some packets might be injected; these are accounted for below in the quantities Δ, Δ'. Now, consider the switching substep. The buffer at node i has $z = (B-b) - x$ "free" slots. If $\Delta + \Delta' \leq z$ then NTG does not drop any injected packet; in this case the number

of real packets grows by Δ or more. If $\Delta + \Delta' > z$ then the number of real packets grows by $\max\{0, z\}$ (note that NTG may accept out of the Δ' packets more than $\max\{0, z\}$ at the expense of real packets that were already in its buffers) and the number of virtual packets by at least $\Delta - z$. Hence the total number of (real and virtual) packets grows also by Δ and the claim still holds. □

We now state two claims about the virtual packets mechanism. Their proofs are omitted.

CLAIM 5.2. *There are at most $3B - 1$ packets in the virtual holding buffer in any node i at any time. Every packet that is added to the virtual holding buffer at time t, leaves this buffer by time $t + B$.*

CLAIM 5.3. *A virtual packet with destination node k, never crosses edge k' for any $k' \geq k$.*

We continue our proof by considering first the short packets and then the long packets.

Short Packets We first show that the number of packets delivered by NTG is $O(n^{\frac{2}{3}})$ competitive compared to the short packets delivered by the adversary. We start with the following claim, which is a variant of Claim 5.1. Proof omitted.

CLAIM 5.4. *For any time t, and for any j, the number of D-packets which are short interval j packets and are injected by the adversary by t, is at most the number of real (short and long) interval j packets delivered by NTG by time t, plus the number of real packets in buffers at the tail of interval j edges at time t, plus the number interval j virtual packets that were converted into virtual packets at the tail of an interval j edge (and are either delivered or undelivered at time t).*

We now claim the following lemma which shows that NTG is $O(n^{\frac{2}{3}})$ competitive compared to the short packets of the adversary. Proof omitted.

LEMMA 5.1. $A_s^t \leq O(n^{\frac{2}{3}})N^t + O(nB + n^{\frac{5}{3}})$.

Long packets We now show that the the total number of packets delivered by NTG is $O(n^{\frac{2}{3}})$ competitive compared to the *long packets* delivered by the adversary. To this end we again consider the piggybacking procedure described above. Let $T_1 = n + B$, let $T_2 = n + n^{\frac{2}{3}}(B - 1)$, and let $T = T_1 + T_2$. For a given final time t, let $0 < t_1 < t_2 < \cdots < t_k \leq t$, be the sequence of time steps in which some packet crosses some edge with more than $3n^{\frac{2}{3}}$ packets piggybacked on it. We now take a subsequence of this sequence so that any two adjacent times are at least T time steps apart. That is, we take the first time step in the original sequence as the first one of the subsequence. The second one in the subsequence is the earliest time which is at least T time steps later than the first one, and so on. Let this subsequence be $t_{j_1}, t_{j_2}, \ldots,$. Based on these times, we define time intervals of the time axis as follows:

1. For each t_{j_i}, we define *red interval* i to be $[t_{j_i} - T_1, t_{j_i} + T_2]$. Let R be the number of red intervals. Observe that any time step can belong to at most two such intervals.

2. The remaining time steps are included in *green intervals* as follows. For $i = 1$, green interval i is $[1, t_{j_1} - T_1)$. For any $1 < i \leq R$, green interval i is $(t_{j_{i-1}} + T_2, t_{j_i} - T_1)$. For $i = R + 1$ green interval i is $(t_{i_R} + T_2, t]$. Observe that some (or all) of these intervals may be empty, but any time step either belongs to one green interval or to one or two red intervals.

Let AG_i be the number of *long* packets that the adversary delivers during green interval i. Let NG_i be the number of packets that NTG delivers during green interval i. Analogously, let AR_i be the number of *long* packets that the adversary delivers during red interval i and NR_i the number of packets that NTG delivers during red interval i.

To prove our bound for long packets, we use the following three lemmas (proofs omitted).

LEMMA 5.2. *For any $i \leq R$, $NR_i \geq n^{\frac{2}{3}}(B-1)$.*

LEMMA 5.3. *For any $i \leq R$, $AR_i \leq nB + Tn^{\frac{1}{3}} = O(nB + n^{\frac{4}{3}})$.*

LEMMA 5.4. *For any $i \leq R$, $AG_i \leq O(n^{\frac{2}{3}})(NG_i + NR_i)$. For $i = R+1$, $AG_i \leq O(n^{\frac{2}{3}})NG_i + O(nB + n^{\frac{5}{3}})$.*

Based on the above three lemmas we can prove the main lemma of this part:

LEMMA 5.5. *For any t, $A_l^t \leq O(n^{\frac{2}{3}})N^t + O(nB + n^{\frac{5}{3}})$.*

Proof. Let R be the number of red intervals until time t, and let $R+1$ be the number of green intervals. Clearly, $A_l^t \leq \sum_{i=1}^{R} \mathrm{AR}_i + \sum_{i=1}^{R} \mathrm{AG}_i + \mathrm{AG}_{R+1}$. By Lemmas 5.2 and 5.3 we get that for any $i \leq R$, $\mathrm{AR}_i \leq O(n^{\frac{2}{3}})\mathrm{NR}_i$. Therefore we have, $A_l^t \leq O(n^{\frac{2}{3}}) \sum_{i=1}^{R} \mathrm{NR}_i + \sum_{i=1}^{R} \mathrm{AG}_i + \mathrm{AG}_{R+1}$. Using Lemma 5.4 we get,

$$\begin{aligned} A_l^t \quad \leq \quad & O(n^{\frac{2}{3}}) \sum_{i=1}^{R} \mathrm{NR}_i + O(n^{\frac{2}{3}}) \sum_{i=1}^{R} (\mathrm{NG}_i + \mathrm{NR}_i) + \\ & [O(n^{\frac{2}{3}})\mathrm{NG}_{R+1} + O(nB + n^{\frac{5}{3}})] \ . \end{aligned}$$

Since each time step appears either in one green interval or in one or two red intervals the last expression is at most $O(n^{\frac{2}{3}})N^t + O(nB + n^{\frac{5}{3}})$, which concludes the proof. □

This concludes the proof that NTG is $O(n^{\frac{2}{3}})$ competitive on the line. As opposed to the protocol NTG , the protocols FTG and LIS have a lower bound of $\Omega(n)$ on the line of n-nodes. (proof omitted).

THEOREM 5.3. *For $I \geq 2$, the competitive ratio of both LIS and FTG on the line of n nodes is $\Omega(n)$.*

Acknowledgments We thank Allan Borodin for useful discussions.

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan, A. Rosén, "Competitive Queue Policies for Differentiated Services," *Proc. of 19th INFOCOM*, pp. 431–440, 2000.
[2] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu, "Universal Stability Results for Greedy Contention-Resolution Protocols", *Proc. of 37th FOCS*, pp. 380–389, 1996.
[3] B. Awerbuch, Y. Azar, and S. Plotkin, "Throughput Competitive On-Line Routing," *34th FOCS*, pp. 32–40, 1993.
[4] B. Awerbuch, A. Brinkman, C. Scheideler, "Anycasting and Multicasting in Adversarial Systems: Routing and Admission Control," manuscript, http://www.cs.jhu.edu/∼scheideler.
[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
[6] A.Z. Broder, A.M. Frieze, and E. Upfal, "A General Approach to Dynamic Packet Routing with Bounded Buffers", *Proc. of 37th FOCS*, pp. 390-399, 1996.

[7] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson, "Adversarial Queuing Theory", *Proc. of 28th STOC*, pp. 376–385, 1996.
[8] A.Z. Broder, and E. Upfal, "Dynamic Deflection Routing on Arrays," *Proc. of 28th STOC*, pp. 348-355, 1996.
[9] D. Gamarnik, Stability of Adaptive and non-Adaptive Packet Routing in Adversarial Queuing Networks. In *Proc. of the 31st STOC*, pp. 206-214, 1999.
[10] E. Hahne, A. Kesselman, Y. Mansour, "Competitve buffer management for shared-memory switches," *Proc. of 13th SPAA*, pp. 53–58, 2001.
[11] M. Harchol-Balter and P. Black, "Queuing Analysis of Oblivious Packet-Routing Algorithms," *Proc. of 5th SODA*, pp. 583-592, 1994.
[12] M. Harchol-Balter and D. Wolfe "Bounding Delays in Packet Routing Networks," *27th STOC*, pp. 248-257, 1995.
[13] A. Kamath, O. Palmon, and S. Plotkin, "Routing and Admission Control in General Topology Networks with Poisson Arrivals," *Proc. of 7th SODA*, pp. 269-278, 1996.
[14] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, M. Sviridenko, "Buffer overflow management in QoS switches," *Proc. of 33rd STOC*, pp. 520–529, 2001.
[15] J. Kleinberg and E. Tardos. "Disjoint Paths in Densely Embedded Graphs." *Proc. of 36th FOCS*, pp. 52-61, 1995.
[16] F.T. Leighton. "Methods for message routing in parallel machines". Invited paper in *24th STOC*, pp. 77-96, 1992.
[17] T. Leighton, B. Maggs, S. Rao, "Packet Routing and Job-Shop Scheduling in O(congestion+dilation) Steps," *Combinatorica*, Vol. 14, No. 2, pp. 167–180, 1994.
[18] T. Leighton, B. Maggs and A. Richa, "Fast Algorithms for Finding O(Congestion+Dilation) Packet Routing Schedules," *Combinatorica*, to appear.
[19] Y. Mansour, and B. Patt-Shamir, "Greedy Packet Scheduling on Shortest Paths", *Journal of Algorithms*, Vol. 14, No. 3, pp. 99–129, 1993.
[20] Y. Mansour, B. Patt-Shamir, O. Lapid, "Optimal smoothing schedules for real-time streams," *Proc. of 19th PODC* , pp. 21-29, 2000.
[21] M. Mihail, "Conductance and Convergence of Markov Chains—A Combinatorial Treatment of Expanders," *Proc. of 30th FOCS*, pp. 526–531, 1989.
[22] M. Mitzenmacher, "Bounds on the Greedy Routing Algorithm for Array Networks", J. Comput. System Sci. 53 (1996), No. 3, pp. 317–327.
[23] R. Ostrovsky and Y. Rabani, "Local Control Packet Switching Algorithm," *Proc. of 29th STOC*, pp. 644–653, 1997.
[24] C. Partridge, The end of simple traffic models. *IEEE Network*, Vol. 7 No. 5, Editor note. 1993.
[25] A. Parekh, and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Transactions on Networking*, 1 (3) pp. 344–357, 1993.
[26] A. Parekh, and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case," *IEEE/ACM Transactions on Networking*, 2 (2) pp. 137–150, 1994.
[27] Y. Rabani and É. Tardos. "Distributed packet switching in arbitrary networks". *28th STOC*, pp. 366-375, 1996.
[28] A. Srinivasan and C.-P. Teo. "A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria". *Proc. of 29th STOC*, pp. 636-643, 1997.
[29] G. Stamoulis and J. Tsitsiklis, "The Efficiency of Greedy Routing in Hypercubes and Butterflies," *IEEE Transactions on Communications*, 42 (11), pp. 3051–208, 1994.
[30] C. Scheideler and B. Vöcking, "Universal Continuous Routing Strategies," *Proc. of 8th SPAA*, 1996.
[31] A. Veres, and M. Boda, The chaotic nature of TCP congestion control. In *Proc. of INFOCOM 2000*, 2000.