Stochastic Stability in Internet Router Congestion Games

Christine Chung[1] and Evangelia Pyrga[2]

[1] Department of Computer Science, University of Pittsburgh, PA, chung@cs.pitt.edu
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany, pyrga@mpi-inf.mpg.de

Abstract. Congestion control at bottleneck routers on the internet is a long standing problem. Many policies have been proposed for effective ways to drop packets from the queues of these routers so that network endpoints will be inclined to share router capacity fairly and minimize the overflow of packets trying to enter the queues. We study just how effective some of these queuing policies are when each network endpoint is a self-interested player with no information about the other players' actions or preferences. By employing the adaptive learning model of evolutionary game theory, we study policies such as Droptail, RED, and the greedy-flow-punishing policy proposed by Gao et al. [10] to find the stochastically stable states: the states of the system that will be reached in the long run.

1 Introduction

Ever since the first congestion control algorithms for TCP endpoints were introduced in [12], the important problem of congestion control at bottleneck routers on the Internet has garnered wide-spread attention. Several algorithms have been proposed for queue management and scheduling of packets in routers. Initially, such algorithms were designed under the assumption that all packets arriving at the routers come from TCP complying sources that produce packet flows with certain characteristics: all flows that become aware of congestion at the router (by seeing some of their packets dropped) will respond by reducing their transmission rates. However, TCP flows are not the only ones competing for available bandwidth or space in router queues. UDP flows behave in a completely different manner, tending to be more aggressive without sharing the same congestion control profile as TCP. Moreover, the assumption that future users will continue using the current TCP protocol seems questionable. Since there is no central authority governing their behavior, as users compete for bandwidth, they may very well change the way they respond to congestion.

Studying congestion control from a game theoretic perspective was therefore the natural next step. Using a variety of models, game theory has been used not only to find Nash equilibria (NE) when users are self-interested and routers employ existing methods, (e.g. FIFO with Droptail, or RED [8]) but also to design new router queuing policies, aimed at reaching good social outcomes in the presence of such users [10]. Such "good social outcomes" include the avoidance of congestion at routers, and thus avoidance of Internet congestion collapse, but also fairness of bandwidth sharing.

However, an approach commonly taken is to assume perfect information. Users are assumed to know the transmission rates of others and the congestion levels at the

router, and use this information to compute a best response and optimize their utility. Even though such assumptions are standard, and even necessary, in the study of NE, they are not likely to be met in a setting like the Internet. Without such assumptions, can the equilibria be reached? Could there be a set of states, none of them necessarily a NE, such that the system gets essentially "trapped" cycling among the states in the set? These are the questions we are aiming to answer in this work.

Using a simple yet general model of the game played by internet endpoints at internet bottleneck routers, we provide the first (to our knowledge) analysis of this problem using stochastic stability, a classical solution concept from the adaptive learning model of evolutionary game theory. Evolutionary game theory's adaptive learning setting is suited especially well for the game of internet endpoints competing for bottleneck router capacity. In traditional game theoretic settings, each player must assume all other players are perfectly rational, and must be fully informed of each other's actions and preferences. When players are internet endpoints, such requirements seem unreasonable and quite unlikely to be met. In our evolutionary setting, under adaptive learning's imitation play, players need only know what you would expect them to know: what they themselves experience in each round of play. Then they use simple heuristics to decide, based on the results of their recent play, what strategy to employ for the next round. The simplicity of the model but also its ability to cope with limited information, make it particularly useful for modeling router congestion games.

To study our problem in this adaptive learning setting, we use a new model proposed by Efraimidis and Tsavlidis [7] called the *window game*. This model is not only simple, but more general than previous models in which players are usually assumed to be TCP endpoints with specific loss recovery properties. In the window game, the endpoints are modeled so that they can be thought of as using either TCP, UDP, or whatever transmission protocol they choose. There are n internet endpoints, each seeking to send an unlimited amount of traffic. But all endpoints encounter the same bottleneck router, which has capacity C. Each of the endpoints is a player that chooses a strategy: an integer-sized "window" between 0 and C. The window size can be thought of as the amount of the router's capacity being requested by the player, or the number of packets being sent by the player. The amount of capacity that the router actually allocates to each player is then determined by the router's queuing policy and the specified window sizes (capacity requests) of the other users. The utility of a player is defined as the number of successfully sent packets, minus the number of dropped packets times some factor $g \geq 0$. Hence, g represents the cost a player suffers by having one packet dropped.

We assume that this game is played repeatedly in rounds, in which every player chooses a strategy to play using imitation dynamics: sampling the outcomes of the rounds of play in its memory, and then imitating the strategy that served it best. However, with very small probability, each player fails to follow the imitation dynamics and chooses a strategy at random. Then, loosely speaking, the set of *stochastically stable states* represents the set of strategy profiles that have positive probability of being played in the long run, or, the states that the system eventually settles on. More details can be found in Section 2.1.

Our Results. The policies we deal with here have been studied with respect to Nash Equilibria before (see the Related Work section for more details), mainly though

assuming that the rates at which the sources send their packets is described by a given rule, for instance, assuming Poisson rates. In our work, we make no such assumption, but employ the *window game* of Efraimidis and Tsavlidis [7]. We believe that this model is simple enough to allow interesting theoretical analysis, but still captures the essence of the game played between competing internet endpoints. We extend the results in [7] with respect to Nash Equilibria but also study the stochastic stability of the underlying games.

We begin by analyzing the two currently most well-known and widely-used router queuing policies: FIFO with Droptail and RED (Random Early Detection) [8]. When Droptail is used, all incoming packets are simply dropped once the queue is full. We show that for any reasonable value of g, the only NE and the only stochastically stable state is the state where all players send $\frac{g+1}{g}C\frac{n-1}{n^2}$ packets. This implies, for instance, that for a large number of flows and any value of $g \leq 1$ ($g = 1$ means that each player is hurt by each lost packet about the same as the amount they gain from each successful packet), the router is getting hit by roughly more than twice as much traffic as it has capacity. Next, we show that under RED queuing, in which the router starts dropping packets preemptively as soon as its buffer reaches a certain threshold $T < C$, things cannot get much better. For reasonable values of g, there is a single NE, which constitutes also the single stochastically stable state, in which the congestion at the router is still significant.

Finally we study a queue policy proposed by Gao et al. [10], in which any overflow is compensated for by dropping the packets belonging to the most demanding flow. This policy was designed, in a idealized setting, to have a unique NE such that the router capacity is equally shared among all flows and overflow is avoided. They also studied a non-idealized setting in which flows do not have perfect information, and all sources are restricted to fixed-rate Poisson rates except one, which can be arbitrarily aggressive. In this setting, they succeed at a more modest goal: the source that can be arbitrarily aggressive should not outperform the best Poisson source by much. In this work we show that this policy can actually do even better. We show that even if flows have no information about one another, and all of them can arbitrarily adjust their window sizes (so no flow is restricted to a fixed rate), the system will still converge to the fair equilibrium under adaptive learning with imitation dynamics.

Even though the stochastically stable states for the queuing policies we study turn out to coincide with the NE, what our results indicate is the following: even in the chaotic internet setting, where players have extremely limited information about the game and make instantaneous decisions, the NE will actually be reached.

Related Work. FIFO with Droptail is the traditional queue policy that has been employed widely in internet routers. As soon as the router queue is full, all subsequent incoming packets are dropped. For more information on Droptail and its variants, see [2]. RED [8] works similarly, but starts dropping packets with a certain probability as soon as the number of packets in the queue exceeds a threshold $T < C$. Both these policies punish all flows in a similar manner, regardless of whether they are "responsible" for causing the overflow or not. Specifically, the expected fraction of the demand of each flow that gets through the router is the same among all flows, those with moderate de-

mands, and those with demands far exceeding their "fair share". The result is that flows with large demand can use more router capacity at the expense of lower-demand flows.

There have been methods suggested for inhibiting such behavior. The Fair Queueing algorithm [4] ensures the maxi-min fairness criterion: using round-robin for selecting the outgoing packets, every flow can at least obtain its "fair share." Even though this is a fair scheme, it comes at the cost of efficiency. It requires separate buffers for each queue and a lot of book-keeping, making it unusable in practice. A method that achieves the same result without the high computational cost at the routers was suggested in [19]. This method however, cannot be used independently in each router, as it depends on receiving flow-specific information from other routers.

CHOKe [17], on the other hand, is a stateless queue management scheme, which can be implemented in a router independently from what other routers use. When a packet arrives to the queue, it is compared to $M \geq 1$ packets chosen uniformly at random from those currently in the queue; if it comes from the same source as any of them, then both are dropped. There are both theoretical and experimental studies [20, 15] suggesting its effectiveness at preventing greedy (e.g. UDP) flows from strangling moderate flows. However, as the number of greedy flows varies, the parameter M must also change in order to protect the more moderate flows from losing their fair share.

Gao et al [10] introduce a router queue management algorithm, which, unlike Fair Queuing, does not require separate buffers for each flow, but, under some assumptions, achieves the same (fair) NE as maxi-min fairness. The main idea is to keep track of the "greediest" flow. Whenever there is an overflow, the algorithm drops only packets that belong to this flow. [3] The *Prince* algorithm described in [7] works in a similar manner. The algorithm in [10] was aiming to fulfill, among others, the following two objectives. First, in an idealized environment of full information, the profile corresponding to maxi-min fairness is the unique NE. Second, removing the full information setup but restricting all flows but one to being Poisson sources of fixed rates, the unrestricted flow has no way of obtaining a throughput much better than that of the best Poisson flow.

There are several game theoretic results for congestion control. For a better introduction, we refer the reader to [18] and [14]. Akella et al. [1] study the equilibria of a TCP game, in which all flows use the Additive Increase Multiplicative Decrease (AIMD) algorithm. This is the method currently employed by TCP endpoints. The strategy sets consist of the possible values for the parameters of the algorithm. They show that even though the older TCP endpoint implementations can lead to efficient equilibria even with FIFO Droptail and RED router queue policies, this is no longer the case with newer implementations. They show that some measure of "network efficiency" can be established with a variant of CHOKe, assuming however that all flows are TCP. A lot of work has been devoted to game theoretic models in which all flows originate from Poisson sources and each source is allowed to vary the transmission rate [18, 5, 6]. The inefficiency of NE is studied, mainly in the case of a single bottleneck router, but also in more general networks [11]. Kesselman et al. [14] consider a model in which the flows are explicitly deciding when to send new packets, instead of implicitly modifying their transmission rates.

[3] Only in case that the overflow is greater than the number of packets of the greediest flow in the queue, will packets from other flows be dropped as well.

An evolutionary game theoretic approach based on adaptive learning is used in [16] to analyze a game in which users set transmission rates for optimally receiving multimedia traffic. In [3], adaptive learning with imitation dynamics was used to analyze a load balancing game.

The Window Game model we study here was first proposed in [7], where it was used to find the NE in games between AIMD but also more general flows.

2 Model, Notation, and Background

To model internet endpoints competing for capacity at a bottleneck router, we use the window game of [7]. Let N be the set of players, $|N| = n$, with each player representing an internet endpoint. The strategy set for each player is the set of all possible window sizes, integer values between 0 and C, where C is the capacity of the bottleneck router. Let w_i be the window size requested by player i. Let $w = (w_1, w_2, \ldots, w_n)$; w is a *strategy profile* vector of the game. Let w_{-i} refer to the vector of all the strategies in w except w_i. Let $W = \sum_{i=1}^{n} w_i$ and let $W_{-i} = W - w_i$. The bottleneck router uses a (possibly randomized) queuing algorithm (like Droptail, RED, etc.), to decide how many of each player's packets to keep, and how many to drop. Therefore the queuing policy maps each strategy profile w to a corresponding vector that indicates for each player i how many of its w_i packets are kept (in expectation), $keep_i$, and how many are dropped, $w_i - keep_i$. As described in the previous section, $g \geq 0$ is a real value that indicates how much detriment a lost packet causes to each player. Then for any $i = 1 \ldots n$, function of i is $\pi_i(w) = (keep_i) - g(w_i - keep_i)$.

A *best response* to w_{-i} for each player i is then $br_i(w_{-i}) = \arg\max_{w_i} \pi_i(w_i, w_{-i})$.

2.1 Adaptive Learning and Imitation Dynamics

We now more formally present the relevant aspects of evolutionary game theory's adaptive learning model [9, 21, 22], as well as the imitation dynamics of [13]. A related, more detailed summary can be found in [3], in which adaptive learning and imitation dynamics are applied to a load balancing game.

In the adaptive learning model with imitation dynamics, each of n players has a finite memory of their own actions and payoffs in the previous m rounds of play. After each round, each player samples (uniformly at random) s of the m previous rounds of play, and then in the next round, plays the strategy (in our case, the window size) that yielded highest average payoff over the rounds that were sampled. In this way, the player is "imitating" the strategy that has served her well in the past.

These dynamics correspond to a Markov process P, where each state in the process is the history of the last m rounds of play. Each play history is comprised of m strategy profiles, and a state where all m strategy profiles are the same is called a *monomorphic* state.[4] The transition probabilities between states of the process are determined by the

[4] For expository simplicity, if a monomorphic state has w as the strategy profile that fills its history, we will sometimes abuse notation and use w not just as the name of the strategy profile, but when the context is clear, as the name of the monomorphic state containing w.

imitation dynamics described above. A *recurrent class* of a Markov process is a set of states such that there is zero probability of leaving the set once a state in the set has been reached, but positive probability of reaching any state in the set from any other state in the set. Josephson and Matros [13] prove the following about the process P.

Theorem 2.1 ([13]). *If $s/m \leq 1/2$, a subset of states is a recurrent class if and only if it is a singleton set containing a monomorphic state.*

If we now suppose that in each round, each player with probability $\epsilon > 0$ does not follow the imitation dynamics, but instead chooses a strategy at random, we have modified the Markov process so that there is always positive probability of eventually reaching any state from any other state. Therefore, there is a unique stationary distribution over the states in this modified process. We refer to this modified process as the *perturbed* Markov process, P^ϵ and the stationary distribution as μ^ϵ. The *stochastically stable states* (SSS) are those states h in this modified process for which $\lim_{\epsilon \to 0} \mu^\epsilon(h) > 0$.

A *better reply* is a unilateral strategy deviation by a player that gives that player at least as high a payoff as the original strategy profile. I.e., x is a better reply for player i if $\pi_i(x, w_{-i}) \geq \pi_i(w)$. A *cusber set* or a set "closed under single better replies," is a set of strategy profiles such that any sequence of better replies, by any sequence of players, starting from any strategy profile in the set, always leads to another strategy profile that is also in the set. A *minimal cusber set* is a cusber set such that if any strategy profile is removed, the remaining set is no longer a cusber set.

Theorem 2.2 ([13]). *Under imitation dynamics, the profiles in the set of stochastically stable states are a minimal cusber set or a union of minimal cusber sets.*

Note that the following corollary is an immediate consequence of Theorem 2.2.

Corollary 2.3. *If a single strategy profile comprises the only minimal cusber set in a game, then that is the only strategy profile in the set of stochastically stable states under imitation dynamics.*

For a more complete background on stochastic stability and imitation dynamics, we refer the reader to [22, 13]. In what remains, we assume that $s/m \leq 1/2$.

3 Droptail

FIFO queues with Droptail are widely used in Internet routers. While the queue has not reached its capacity, incoming packets are inserted in the end of the queue. As soon as the capacity is reached, any new incoming packets are dropped. We will start by describing the window game model of Droptail, then discuss the NE, and finally prove that there is a single stochastically stable state that corresponds to the unique NE.

Remember that for any profile w, we denote by W the total window size requested, i.e., $W = \sum_{i=1}^{N} w_i$. Under the Droptail routing policy, when $W > C$, the router chooses $W - C$ packets uniformly at random to be dropped. Therefore, for any player i with window size w_i, the expected number of packets of i that will enter the queue is $w_i \cdot C/W$, while $w_i \cdot (1 - C/W)$ will be dropped. Of course, when $W \leq C$ no packets will be dropped. This means that the expected payoff for player i can be expressed as

$$\pi_i(w) = \begin{cases} w_i & \text{if } w_i \leq C - W_{-i} \\ w_i \cdot \frac{C}{W} - g w_i \left(1 - \frac{C}{W}\right) & \text{if } w_i > C - W_{-i} \end{cases}. \quad (1)$$

We note that when the total window size equals the capacity, i.e., $w_i + W_{-i} = C$, then both pieces of the payoff function result in the same payoff. Therefore, for $W = C$ either of the two subcases can be used.

Definition 3.1. *Define d_g to be $\frac{g+1}{g}\ C\ \frac{n-1}{n^2}$.*

Efraimidis and Tsavlidis in [7] proved that, assuming $g \leq n-1$,[5] the profile $(d_g, ..., d_g)$ is the unique *symmetric* NE. In fact, as the next theorem states, that is the only NE for the case $g \leq n-1$. The proof, which involves first determining the best response function for each player, and then ruling out the possibility of all other NE, can be found in the full version of this paper.

Theorem 3.2. *If $g \leq n-1$, then the outcome in which each player's window size is $d_g = \frac{(g+1)C(n-1)}{n^2 g}$ is the only NE.*

In the following, we will assume that $g \leq n-1$, since the case where $g > n-1$ is of no practical relevance. We will now establish that the state $(d_g, \ldots, d_g)$ is the only SSS. Our proof uses the fact that any profile in a stochastically stable state is found in a minimal cusber set (Theorem 2.2), along with the fact that under Droptail the only minimal cusber set in our game is the NE profile itself. We first give two lemmas that allow us to establish the latter fact, by showing there is a better-reply path from any profile to the NE profile. Due to lack of space, we refer to the full version of this work for the proof of Lemma 3.3.

Lemma 3.3. *Let $w \neq (d_g \ldots, d_g)$, $W \geq C$. Within at most two better replies, a profile w' can be reached, such that for any k with $w_k = d_g$, $w'_k = d_g$, and there is some player i, such that $w_i \neq d_g$ and $w'_i = d_g$. Moreover, $W' \geq C$.*

Lemma 3.4. *For any $w \neq (d_g, \ldots, d_g)$, there is a finite sequence of better replies that leads to the profile $(d_g, \ldots, d_g)$.*

Proof. We note first that if $W < C$, then for any player i, playing $C - W_{-i}$ is a better response than w_i. Hence we will assume that $W \geq C$. Note that applying Lemma 3.3 to $w \neq (d_g, \ldots, d_g)$, we will obtain some w' such that still $W' \geq C$. Therefore, by simply invoking Lemma 3.3 at most n times, we can see that there is a path of (in total) at most $2n+1$ better response moves from w to the profile $(d_g, \ldots, d_g)$. □

Theorem 3.5. *For $g \leq n-1$, the state in which every player plays d_g is the unique stochastically stable state.*

Proof. First of all, note that d_g is the unique better response to $W_{-i} = (n-1)d_g$. Therefore the profile $a = (d_g, \ldots, d_g)$ is a minimal cusber set. Moreover, by Lemma 3.4, there is a better response path from any $w \neq a$ to a. Therefore any other cusber set would have to contain a, which implies there is no other minimal cusber set. Hence, by Corollary 2.3, a is the only state that is stochastically stable. □

[5] Given that the number of flows that share the capacity of a bottleneck router is large, the case that $g > n-1$ is not realistic, and thus of no practical interest. For completeness, the NE for the case that $g > n-1$ are discussed in the full version of this work.

4 RED (Random Early Detection)

RED (Random Early Detection) [8] was meant to keep the average queue size low. It works similarly to Droptail, but starts dropping packets before the queue is full. When the total load at the router exceeds a system-defined minimum threshold T, the router begins dropping each new arriving packet with probability proportional to the load. After total load exceeds a system-defined maximum threshold, the packets are dropped with probability 1. (Note that when the maximum threshold is set to C, then once capacity is reached, RED behaves exactly like Droptail.)

To simplify our study, we will assume that the maximum threshold is C, but we will leave the minimum threshold T as a free parameter. We then must model the RED mechanism in the window game setup. Assume that the current load at the queue is $L \geq T$. Then, according to RED, each new arriving packet will be dropped with probability $\frac{L-T}{C-T}$. Assume that when W packets arrive sequentially, the expected number of them that enter the queue is x. In contrast to this sequential process where packets arrive one by one, using the window game we assume that given a strategy profile w, all W packets arrive at the same time. Each packet will be admitted to the queue with probability $\frac{x}{W}$ (x packets are chosen uniformly at random). If $W \leq T$, then all packets are admitted.

Lemma 4.1. *Assume that RED is used and let w be a strategy profile such that $W > T$.*

i) If $W \geq W_C$, where $W_C = (C-T)H_{C-T} + T$, then the queue size reaches C.

ii) If $T \leq W < W_C$, then $T + \tilde{k}_W$ packets enter the queue, (and the probability for any packet to be kept is $\frac{\tilde{k}_W + T}{W}$), where $\tilde{k}_W \approx (C-T)\left(1 - e^{-\frac{W-T}{C-T}}\right)$.

Proof. The proof uses the solution to the well-known coupon collector problem. In what follows we use the term *kept* to refer to the event of a packet not being dropped. We consider the case that W packets arrive sequentially. Consider the moment at which the queue size becomes $T + i - 1$, for some $i, 1 \leq i \leq C - T$. Let X_i be a random variable that represents the number of packets that arrive to the system until the queue size reaches $T + i$ (i.e., $X_i - 1$ is the number of packets that arrive to the router and get dropped until one is kept). According to the description of RED, when $T + i - 1$ packets are already in the queue, the probability that a newly arriving packet is dropped is $\frac{i-1}{C-T}$. This implies that $\mathbf{E}[X_i] = \frac{C-T}{C-T-i+1}$. Let H_j be the jth harmonic number.

i) $W_C = T + \mathbf{E}\left[\sum_{i=1}^{C-T} X_i\right] = T + \sum_{i=1}^{C-T} \frac{C-T}{C-T-i+1} = (C-T)H_{C-T} + T$.

ii) The total number number of packets $\tilde{k}_W$ that enter the queue, out of the total of W that arrive, is given as the maximum k, such that $T + \mathbf{E}\left[\sum_{i=1}^{k} X_i\right] \leq W \Leftrightarrow \sum_{i=1}^{k} \frac{C-T}{C-T-i+1} \leq W - T \Leftrightarrow (C-T)\left(H_{C-T} - H_{C-T-k}\right) \leq W - T$.

Approximating H_j with $\ln j$ we get: $\ln(C-T-k) \geq \ln(C-T) - \frac{W-T}{C-T}$ which gives $\tilde{k}_W \approx (C-T)\left(1 - e^{-\frac{W-T}{C-T}}\right)$. □

In order to simplify our presentation (and to allow clean formulation of a best response function), we will approximate $\tilde{k}_W$ by $k_W = \frac{W-T}{H_{C-T}}$. Note that k_W is also a continuous function, while $k_T = 0$ and $k_{W_C} = C - T$; therefore, when W equals T

(respectively, W_C), the total number of packets entering the queue is T (respectively, C), in accordance to Lemma 4.1. The payoff function of flow i is now expressed as:

$$\pi_i^{RED}(w) = \begin{cases} w_i & \text{if } W \leq T \\ w_i \cdot \frac{k_W+T}{W} - gw_i\left(1 - \frac{k_W+T}{W}\right) & \text{if } T < W \leq W_C \\ w_i \cdot \frac{C}{W} - gw_i\left(1 - \frac{C}{W}\right) & \text{if } W > W_C \end{cases}$$

The best response function of RED differs according to the value of g. In particular, there are three possible ranges for g. Due to space limitations, we will only discuss here the case where $g \in R_g$, for $R_g = \left[\frac{C}{(C-T)(H_{C-T}-1)}, n-1\right]$, which is the most practically relevant range of values.[6] We defer the other cases, as well as the proofs of the following two theorems, to the full version of this work.

Definition 4.2. *Define* $r_g = \frac{T(g+1)(H_{C-T}-1)}{gH_{C-T}-g-1} \cdot \frac{n-1}{n^2}$.

Theorem 4.3. *If* $g \in R_g$*, then there is a unique NE, such that* $w_i = r_g$*, for all* i*.*

Theorem 4.4. *If* $g \in R_g$*, then the only stochastically stable state under RED is the state where all players set their window sizes to* r_g*.*

The above theorems imply that under RED the system will converge to the unique Nash Equilibrium. Given that $g \in R_g$, the total congestion will be less than the corresponding one in Droptail. Still, however, the overflow is large: as n grows, since $\frac{(g+1)(H_{C-T}-1)}{gH-g-1} > \frac{g+1}{g}$, the total window size will be (roughly) at least $2T$. And, as g decreases to values outside of R_g, the congestion at RED NE can sometimes be even greater than at the Droptail NE. More details can be found in the full version of this work.

5 "Fair" queue policy

In this section we study the queuing policy proposed by Gao et al. in [10]. The main idea (similar also to the Prince algorithm of [7]) is that in case of congestion, the most demanding flow is punished. Assuming that all players are fully informed of the other players' strategies, this policy was constructed so as to have a unique NE in which all players share the capacity equally. In a more realistic setting, where the rates at which other flows send packets are not globally known, the authors wish to reach a less lofty goal: if all flows but one have fixed rates, then the unrestricted flow cannot use up much more of the router queue capacity at the expense of the fixed-rate flows. We will show here that in fact, the fair equilibrium is also the only stochastically stable state. This implies that, even without fully informed players, the algorithm in [10] can achieve the fair NE, even when all flows are allowed to be arbitrarily aggressive.

The window game adaptation of Protocol I in [10] works as follows. For any profile $(w_1, \cdots, w_n)$, if $W \leq C$ then for any flow i, all w_i packets will enter the queue, i.e.

[6] In practice $T = \lambda C$, for some constant λ meaning that $\frac{C}{(C-T)(H_{C-T}-1)}$ is a decreasing function on C tending to 0, as C grows large.

$\pi_i(w) = w_i$. On the other hand, if $W > C$ then let $i_0 = \arg\max_{i \in N}\{w_i\}$ (breaking ties arbitrarily). Flow i_0 will be the one to be punished for the overflow, and if $w_{i_0} < W - C$ then the rest of the packets will be dropped according to Droptail. In other words, $\pi_{i_0} = \max\{0, w_{i_0} - (W - C)\} - g \cdot \min\{w_{i_0}, W - C\}$, while for any $i \neq i_0$,

$$\pi_i(w) = \begin{cases} w_i & \text{if } w_{i_0} \geq W - C \\ w_i \cdot \frac{C}{W - w_{i_0}} - g w_i \left(1 - \frac{C}{W - w_{i_0}}\right) & \text{if } w_{i_0} < W - C \,. \end{cases} \quad (2)$$

The next theorem was stated in [7].

Theorem 5.1. *Assuming $g > 0$, there is a unique NE in which all players play C/n.*

The following theorem establishes the fact that the unique NE is also the only stochastically stable state. We prove this by showing that the profile $(C/n, ..., C/n)$ is the only minimal cusber set.

Theorem 5.2. *If $g > 0$, then the only stochastically stable state is $(C/n, ..., C/n)$.*

Proof. Let $\hat{w} = (C/n, \cdots, C/n)$. We will show that the singleton set $\{\hat{w}\}$ is *the only* minimal cusber set. Then we can conclude using Corollary 2.3 that $\hat{w}$ is the only stochastically stable state. First note that $\{\hat{w}\}$ is a minimal cusber set: any player deviating from $\hat{w}$ will be strictly decreasing her payoff. (Assume that a player i moves to some value $x \neq C/n$. If $x < C/n$, then $\pi_i(x, \hat{w}_{-i}) = x < C/n = \pi_i(\hat{w})$. If $x > C/n$, then $x - C/n$ of her packets will be dropped and her payoff will decrease to $\pi_i(x, \hat{w}_{-i}) = C/n - g(x - C/n) < \pi_i(\hat{w})$, since $g > 0$.)

We proceed now to showing that for any profile $w \neq \hat{w}$, there is a finite better response path to $\hat{w}$. Assume first that $W > C$ and let $i_0 = \arg\max_{i \in N}\{w_i\}$. Then $\min\{w_{i_0}, W - C\}$ of i_0's packets get dropped. In that case it is at least as good for i_0 to play $\max\{0, w_{i_0} - (W - C)\}$, since the same amount of i_0's packets will enter the queue as before, but without any being dropped. We will call this a *move of type A*.

Assume now that $W = C$, but $w \neq (C/n, \cdots, C/n)$. Let j be the player with the maximum window size in w, i.e., $j = \arg_{i \in N} \max w_i$. The fact that $W = C$ and $w \neq \hat{w}$, imply that $w_j > C/n$. Moreover there must be some player $k \neq j$, with $w_k < C/n$. Playing C/n is a better response to k, since $C/n < w_j$, meaning that j will be the one to be punished for the overflow. (The new total window size cannot exceed the capacity by more than C/n, implying only packets from flow j will be dropped.) Therefore, k gets more packets in the queue by changing w_k to C/n, and still none dropped. We will call this a *move of type B*.

Now the better response path to $\hat{w}$ is constructed as follows: From any w, if $W < C$, then any player can improve her payoff by increasing her window size by $C - W$. We then arrive at a profile w' where $W' = C$. If $W > C$, after fewer than n moves of type A, a strategy profile w' is reached where $W' = C$.

For any w' such that $W' = C$, if $w' \neq \hat{w}$, then a move of type B occurs in which a player that in w' played something less than C/n moves to C/n. This is immediately followed by a move of type A in which a player that in w' was playing something greater than C/n reduces her window size. If w'' corresponds to the new profile reached, then again $W'' = C$. This alternation between moves of type A and moves of type B continues, until $\hat{w}$ is reached. Note that once a player moves to C/n then she does not

change her window size anymore, meaning that the total number of steps needed until $\hat{w}$ is reached is finite. □

We note that the condition $g > 0$ in the above theorem is necessary in order for the cusber set to contain only the profile $(C/n, \ldots, C/n)$. If $g = 0$, then a flow can deviate from the profile $(C/n, \ldots, C/n)$ by increasing its window size while still obtaining exactly the same payoff. We also note that unlike the results of Sections 3 and 4, here, the result in this section holds even if each flow has a different value for g, a value that can be arbitrarily small.

6 Discussion

While Droptail and RED have stochastically stable states with high congestion at the bottleneck router, the Gao et al. policy leads to fair and efficient use of the bottleneck router capacity. Specifically, we've established that under Droptail queuing, the unique stochastically stable state (and unique NE) is the profile in which all players send a window size of $d_g = C(g+1)(n-1)/(gn^2)$. This means that if $g \leq (n-1)/(n+1)$, each player will be sending at least $2C/n$ packets, which amounts to twice as many total packets as the capacity allows.

Under RED, when g is reasonably large (i.e., for $g \in R_g$), the unique stochastically stable state (and unique NE) is the profile where all players send a window size of r_g, which is greater than $T(g+1)(n-1)/(gn^2)$. (Recall that $T < C$ is the threshold value at which RED begins preemptively dropping packets. It is a free parameter of the RED protocol.) This means, analogously to the above discussion about Droptail, that if $g \leq (n-1)/(n+1)$ (which is close to 1 as n grows large), players will be sending at least $2T/n$. This would imply that even with values of g nearly as large as 1, if deployers of RED routers set T to relatively large values, the gain with respect to overflow, as compared to the case of Droptail, will be small.

On the other hand, the more discriminating Gao et al. protocol can be safely deployed without knowledge of the specific value of g: the endpoints each send C/n as long as g is positive. In addition, our results hold even when each player has its own g value. Intuitively, this means the results apply even when the endpoints are all of different types: well-behaved TCP flows, more aggressive TCP flows, UDP flows, etc., as long as dropped packets cause some loss to the flows, no matter how small it is.

Finally we note the fact that the stochastically stable states in each case can be reached with the players having very limited knowledge; they need not be aware of the actions of other players, or even of their number n. Even though we assumed that players choose a window size between 0 and C, any other sufficiently large upper bound for the window sizes would have done just as well. In other words, the players need also not be aware of the exact value of the router capacity C.

Acknowledgments. The authors wish to thank Kirk Pruhs for invaluable advice and guidance, Lazaros Tsavlidis for his help with the window game, and Ihsan Qazi, Subrata Acharya, and Ho-Leung Chan for helpful discussions.

References

1. Aditya Akella, Srinivasan Seshan, Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of tcp. In *SIGCOMM*, 2002.
2. B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Larry Peterson Partridge, K. K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. RFC2309: Recommendations on queue management and congestion avoidance in the internet. *Internet RFCs*, 1998.
3. Christine Chung, Katrina Ligett, Kirk Pruhs, and Aaron Roth. The price of stochastic anarchy. In *SAGT*, 2008.
4. A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 1–12, 1989.
5. C. Douligeris and R. Mazumdar. On Pareto optimal flow control in an integrated environment. In *Allerton Conference on Communication, Control and Computing*, 1987.
6. C. Douligeris and R. Mazumdar. A game theoretic approach to flow control in an integrated environment. *Journal of the Franklin Institute*, 329(3):383–402, 1992.
7. P.S. Efraimidis and L. Tsavlidis. Window-games between tcp flows. In *The first international symposium on algorithmic game theory*, 2008.
8. Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.
9. D. Foster and P. Young. Stochastic evolutionary game dynamics. *Theoret. Population Biol.*, 38:229–232, 1990.
10. X. Gao, K. Jain, and L.J. Schulman. Fair and efficient router congestion control. In *SODA 2004*.
11. Rahul Garg, Abhinav Kamra, and Varun Khurana. A game-theoretic approach towards congestion control in communication networks. *SIGCOMM Comput. Commun. Rev.*, 32(3):47–61, 2002.
12. V. Jacobson. Congestion avoidance and control, ACM SIGCOMM, 1988.
13. Jens Josephson and Alexander Matros. Stochastic imitation in finite games. *Games and Economic Behavior*, 49(2):244–259, November 2004.
14. A. Kesselman. St. Leonardi, and V. Bonifaci. Game-theoretic Analysis of Internet Switching with Selfish Users. In *WINE 2005*.
15. H.J. Lee and J.T. Lim. Performance Analysis of CHOKe with Multiple UDP Flows. In *SICE-ICASE, 2006. International Joint Conference*, pages 5200–5203, 2006.
16. D. S. Menasché, Daniel R. Figueiredo, and Edmundo de Souza e Silva. An evolutionary game-theoretic approach to congestion control. *Perform. Eval.*, 62(1-4):295–312, 2005.
17. Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. Choke, a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM*, 2000.
18. SJ Shenker. Making greed work in networks: a game-theoretic analysis of switchservice disciplines. *IEEE/ACM Transactions on Networking*, 3(6):819–831, 1995.
19. I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *ACM SIGCOMM'98*.
20. A. Tang, J. Wang, and S.H. Low. Understanding CHOKe: throughput and spatial characteristics. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 12(4), 2004.
21. H Peyton Young. The evolution of conventions. *Econometrica*, 61(1):57–84, January 1993.
22. H Peyton Young. *Individual Strategy and Social Structure*. Princeton University Press, Princeton, NJ, 1998.