Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP*

Aditya Akella, Srinivasan Seshan
CMU

Richard Karp, Scott Shenker
Christos Papadimitriou
ICSI/UC Berkeley

ABSTRACT

For years, the conventional wisdom [7, 22] has been that the continued stability of the Internet depends on the widespread deployment of "socially responsible" congestion control. In this paper, we seek to answer the following fundamental question: If network end-points behaved in a selfish manner, would the stability of the Internet be endangered?

We evaluate the impact of greedy end-point behavior through a game-theoretic analysis of TCP. In this "TCP Game" each flow attempts to maximize the throughput it achieves by modifying its congestion control behavior. We use a combination of analysis and simulation to determine the Nash Equilibrium of this game. Our question then reduces to whether the network operates efficiently at these Nash equilibria.

Our findings are twofold. First, in more traditional environments – where end-points use TCP Reno-style loss recovery and routers use drop-tail queues – the Nash Equilibria are reasonably efficient. However, when endpoints use more recent variations of TCP (e.g., SACK) and routers employ either RED or drop-tail queues, the Nash equilibria are very inefficient. This suggests that the Internet of the past could remain stable in the face of greedy end-user behavior, but the Internet of today is vulnerable to such behavior. Second, we find that restoring the efficiency of the Nash equilibria in these settings does not require heavyweight packet scheduling techniques (e.g., Fair Queuing) but instead can be done with a very simple stateless mechanism based on CHOKe [21].

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*network communication*

General Terms

Design, Performance

*This research was sponsored by DARPA under contract F30602-99-1-0518. Additional support was provided by IBM. Views and conclusions contained in this document are those of the authors and should not be interpreted as respresenting the official policies, either expressed or implied, of DARPA, IBM or the U.S. government.

SIGCOMM'02, August 19-23, 2002, Pittsburgh, Pennsylvania, USA.

1. INTRODUCTION

Over a decade ago, the Internet suffered a series of congestion collapses leading to the development of TCP's congestion control algorithms [15, 3]. This "socially responsible" congestion control behavior, implemented by the bulk of Internet end-points, has been given credit for the continued stability of the network. This paper is an attempt to understand if these social congestion control algorithms were aptly credited. We ask whether greedy behavior by network end-points actually results in unstable network conditions such as congestion collapse. The answer to this question has important implications to network operation. If the answer is that greedy behavior results in instability, then the reason that the Internet is functioning correctly is either that end-users are consciously socially responsible or that it is too difficult to modify end-hosts to behave greedily. Clearly, network operators cannot rely on either of these conditions persisting and they should deploy new network mechanisms to ensure that network end-points do not behave greedily in the future. On the other hand, if selfishness does not result in poor network behavior, then perhaps there is no need for such mechanisms.

We evaluate the impact of greedy end-point behavior by performing a game-theoretic analysis of TCP. We choose to analyze TCP since the bulk of bytes transferred in the Internet use TCP. In addition, as long as applications require reliable data transfer, this preponderance of TCP is likely to continue. We define the *TCP Game* in which TCP flows in a network can adjust their Additive-Increase Multiplicative-Decrease (AIMD) congestion behavior in order to optimize the goodput they achieve. These flows are allowed to freely change and set their congestion control parameters, (α, β), where α is the additive increase component and β is the multiplicative decrease component. Also, these flows must continue to use the traditional loss-recovery techniques of timeouts and fast retransmission to provide reliable data transfer.

One might think that a greedy/selfish flow could always gain by using more aggressive congestion control. However, more aggressive congestion control leads to higher loss rates. Since TCP loss recovery is not a perfect process, there is always some "cost" associated with the higher loss rate. When the potential benefit and cost balance, a flow has nothing to gain.[1]

Our aim is to determine the congestion parameters (α_E, β_E) that are chosen by the flows at Nash equilibrium (where no flow can gain throughput by unilaterally adjusting its behavior). In evaluating the impact of greedy end-points, we are not as interested in the actual value of (α_E, β_E) as much as the behavior and efficiency of the network under this operating condition. To evaluate efficiency, we measure the the average goodput of any flow and the average per-flow loss rate at Nash equilibrium.

Since the Nash equilibrium reflects a balance between the gains and the cost related to aggressive behavior, any factor affecting this balance results in a change of the parameter settings and the network efficiency at Nash equilibrium. Two such factors

[1]This is in contrast to using modern coding techniques that don't rely on loss recovery to achieve reliable transfer. With such methods, loss recovery is essentially perfect, and aggression will always pay. In this paper we restrict out attention to TCP and its imperfect loss-recovery.

that we vary in our analysis are the form of loss recovery used by TCP flows and the queuing discipline implemented by the routers in the network. More modern loss recovery techniques, such as SACK [10], reduce the overhead of loss recovery, thereby changing the balance in favor of more aggressive behavior. Queuing disciplines (like RED [13], CHOKe [21]) affect the loss rate that results from varying levels of aggression. For example, at the extreme, Fair Queuing techniques [7] prevent any flow from receiving more than its fair share by assigning all additional losses to the more aggressive flows, thereby removing any incentive to be aggressive.

Taking the above observations into account, we seek to address the following questions in this paper:

1. What are the parameter settings of the flows at Nash equilibrium?
2. How efficient/inefficient is the operation of the network at Nash equilibrium?
3. What impact do TCP's loss recovery mechanisms and the AQM schemes implemented at routers have on the efficient operation of the network at Nash equilibrium?

Our analysis of a simplified version of the TCP Game and simulations in NS-2 [1], show that when flows implement traditional loss recovery mechanisms (TCP-Reno) and FIFO drop-tail buffers are employed, the network operates efficiently at the resulting Nash equilibrium (i.e. there is no danger of congestion collapse). However, the allocation of bandwidth at this equilibrium is somewhat unfair [6, 2]. This combination of Reno and FIFO drop-tail is significant since it was common in the Internet until quite recently. Unfortunately, in all other cases, the Nash equilibrium is undesirable since either the per-flow goodput is too low or the per-flow loss rate is too high. We also show that heavyweight queueing mechanisms requiring explicit per-flow state are not necessary to avoid congestion collapse at the Nash equilibrium. We show that a minor modification to the CHOKe [21] active queue management policy ensures efficient operation as well as reasonable fairness at Nash equilibrium.

The remainder of this paper is organized as follows. In Section 2 we present related work. Section 3 discusses the TCP Game in detail and also presents our analysis methodology. In Section 4, we present analytical and simulation results for the Nash equilibrium of the simplified TCP Game. Section 5 discusses a simple low-overhead mechanism that encourages a desirable Nash equilibrium. Finally, Section 6 summarizes the contributions of this paper.

2. RELATED WORK

There is a substantial literature of game-theoretic approaches to network resource allocation in general and to congestion control in particular. We do not provide a detailed review of this work here, but direct interested readers to a small sampling of the literature [9, 14, 9, 8, 11, 22]. The approach we take here differs from these earlier papers in several key respects.

First, the previous literature typically used models where flows were represented by Poisson streams and routers by M/M/1 queues, and congestion control consists of adjusting the Poisson transmission rate. In this paper, we consider the simulated performance of TCP's actual packet-level congestion control algorithms, including loss-recovery and window adjustment. Second, instead of considering general congestion control algorithms, we restrict our attention to the AIMD family of window adjustment algorithms. Third, while these previous treatments considered a wide class of utility functions (often concave functions of delay and throughput), we assume all users are interested in maximizing goodput. Thus, our work uses a more realistic but more limited model of congestion control, and we pay careful attention to the impact of loss-recovery algorithms. Our modeling choices reflects our underlying question: what would happen if users freely chose their TCP AIMD parameters?

Our work is also closely related to [2]. In this work, the authors evaluate the four linear congestion control algorithms - AIMD, AIAD, MIMD and MIAD - in the context of various loss recovery and queue management algorithms and under a variety of variations in the available bandwidth. The paper concludes that AIAD provides comparable (and sometimes better) efficiency to AIMD in most settings. We use these results as a guide to judge the efficiency/inefficiency of the Nash equilibria we analyze in this paper.

Finally, the simple penalty-based model for the variants of TCP that we present in Section 3.3 is similar to that presented in [16].

3. THE TCP GAME

In this section, we describe the TCP Game in detail. We first state the assumptions we make in order to simplify the game. Next, we discuss the dimensions along which the TCP Game can be analyzed. Finally, we describe a penalty-based model for TCP that we use in our analysis of the TCP Game in later sections.

3.1 A Few Simplifying Assumptions

In the TCP Game each TCP end-point attempts to maximize its own goodput. To achieve this goal, each TCP end-point is given the freedom to adjust its congestion control behavior. Formally, we assume that we are given a set of n TCP flows, $F_1, \ldots, F_n$, all implementing the Additive Increase Multiplicative Decrease (AIMD) algorithm for congestion avoidance and control. We allow each flow F_i to modify its additive increase constant ($\alpha_i \geq 1$) and its multiplicative decrease constant ($\beta_i \in (0, 1)$).

In addition, we make the following simplifying assumptions:

(I) All flows in the network implement the *same* algorithms for loss recovery (e.g. timeout, fast retransmission, selective acknowledgments, etc.).

(II) All the flows have an infinite amount of data to send.

(III) All the flows encounter a single common bottleneck. We assume that the *capacity* of the bottleneck link, defined as the number of packets that it can transmit in unit time, is fixed.

(IV) All flows have identical round-trip times.

(V) The amount of buffering at the bottleneck router is fixed at the bandwidth-delay product of the simple topology resulting from the above assumptions.

When packets are successfully acknowledged, each flow F_i increases its transmission rate as dictated by the increase parameter α_i. Flows react to packet loss by decreasing their transmission rate. This rate reduction is dictated by two factors: the decrease parameter β_i and the loss recovery algorithm implemented by the flow. This is described in greater detail in Section 3.3.

Let G_i denote the average number of *useful* (i.e., distinct) packets of flow F_i that are successfully delivered in unit time (where we choose the common RTT as the unit of time). G_i is the goodput of flow F_i. In the TCP Game, the aim of each flow is to choose its parameters (α_i, β_i) so that G_i is as high as possible. Notice that such a choice for flow F_i is dependent on the setting chosen by each of the remaining $n-1$ flows. When for each flow F_i, the parameters (α_i, β_i) are chosen such that, given the parameters (α_j, β_j) for $j \neq i$, no other choice of parameters for flow F_i yields a higher value of G_i, the TCP Game is said to be at a *Nash equilibrium*.

In our analysis of the TCP Game, we are interested in two key properties of the Nash equilibrium: the parameter settings of the flows and the resulting efficiency of the network. We are *not* concerned with how the Nash equilibrium is attained through iterative adjustment of the flow control parameters. Our paper thus addresses the following question: If the Internet were such that all TCP-AIMD flows were at Nash equilibrium, how would their parameters be set and how efficiently would the Internet be operating? We use the average per-flow statistics of goodput and loss rate to measure the efficiency of the network at Nash equilibrium.

3.2 Factors Affecting the TCP Game

The value of G_i attained by a flow in this game, and therefore the Nash equilibrium of the TCP Game, is dependent on many factors. Important among these are: (i) the congestion control

parameters, (ii) the nature of the loss recovery algorithm, and (iii) the way losses are assigned at the bottleneck router. Factor (iii) depends on the router queueing and buffer management algorithms, and thus is under the control of network administrators. Factor (ii) is controlled by the set of algorithms supported by a TCP implementation and the contents of TCP packet headers (e.g., SACK blocks). As a result, only Factor (i) is under complete control of a single end-user (the source), and is the only factor we allow users to *adjust* to gain advantage; we consider the other two factors as being important components of the environment in which the agents are playing the TCP Game. In this section, we describe each of these factors in turn.

The congestion control algorithm employed by a TCP flow can be looked upon as a mechanism that the flow uses to probe for available bandwidth. There are two axes along which each AIMD flow could change its parameters:

- **Varying α.** By choosing a higher α, a greedy flow could try to grab the available bandwidth at a much *quicker rate* and gain an advantage over competing flows.
- **Varying β.** By choosing a β closer to one, a greedy flow can choose to give up bandwidth more slowly upon congestion.

In general, flows would adjust both α and β simultaneously. However, to make both the analysis and the presentation of the results more accessible, in this paper we focus on two restricted cases: (i) all flows vary their α but hold β fixed and (ii) all flows vary their β but hold α fixed. We present results from both analysis and simulations for these two cases in detail in Section 4 of this paper. In addition, we also summarize the initial results from our simulations of the more general scenario, where flows are allowed to adjust α and β simultaneously, in Section 4 without presenting the relevant analysis.

The loss recovery schemes in early versions of TCP, like Reno, are primitive and cause the TCP flow to show a rather drastic reaction to losses. For example, when a TCP-Reno flow loses more than a couple of packets within a single congestion window, it is forced to time-out and restart [10]. Modern versions of TCP, like SACK, use more tolerant loss recovery mechanisms that can sustain many more losses without the flow having to incur time-outs. Since by being more aggressive a flow has a greater chance of losing packets and since the reaction to losses is directly dependent on the loss recovery algorithm, the form of loss recovery implemented by the flows participating in the TCP Game has an effect on the the nature of the Nash equilibrium.

Traditional queueing and buffer management schemes like drop-tail and RED do not actively penalize aggressive flows. However, drop-tail may unintentionally penalize aggressive flows since packet bursts, a common characteristic of aggressive behavior, often incur drops under drop-tail queue management. In addition, several proposed (but not widely deployed) queueing and buffer management schemes, such as CHOKe and Fair Queueing, intentionally punish aggressive flows (to varying degrees). Thus, the queueing and buffer management schemes will have an effect on the resulting Nash equilibrium.

Symmetry is another important aspect of the Nash equilibrium of the TCP Game that warrants discussion. In this paper, we only consider situations where the flows are symmetric (i.e., have the same RTTs) and we only analyze symmetric Nash equilibria (i.e., Nash equilibria where the congestion control parameters of the flows are all equal). We leave the analysis of asymmetric Nash equilibria for future work.

Summarizing, in this paper we analyze the symmetric Nash equilibria for the TCP Game under varying combinations of the queueing and buffer management schemes employed at the routers and the loss recovery mechanisms implemented by the TCP endpoints.

3.3 A Penalty-Based Model for TCP

In this section, we present a penalty-based model for TCP similar to that described in [16]. We use this model in our analysis of the Nash equilibrium. These results will be compared to what we find using more realistic simulations. The purpose is to find a simple model that captures most of the behavior found in the packet-level simulations but yet remains fairly accessible to analysis.

We divide the duration of transmission of each flow into *rounds* each corresponding to one round-trip time (RTT) of the flow. Let N_i^t denote the number of packets that flow F_i has outstanding in the network in round t. Let L_i^t denote the number of packet losses experienced by flow F_i in round t. L_i^t depends on the value of $\sum_i N_i^t$ and the queue management algorithm employed by the bottleneck router. Each flow changes the *maximum* number of packets it is allowed to keep outstanding in the network in the round following t as follows: if $L_i^t > 0$ then $N_i^{t+1} = \beta_i N_i^t$ (multiplicative decrease), and if $L_i^t = 0$ then $N_i^{t+1} = N_i^t + \alpha_i$ (additive increase). This models the congestion avoidance/control behavior of each flow. Here, we assume that each flow knows about the losses assigned to it in a given round at the start of the following round.

Suppose a TCP flow incurs $L > 0$ losses at time t. Let N be the number of packets of the flow outstanding at time t. When the TCP flow experiences one or more losses, it not only adjusts its window (as described above) but also must recover from the loss. We model this loss recovery mechanism by a *penalty function* that defines *exactly* how many packets the flow is allowed have outstanding in the round(s) immediately following a loss. At the very high level, there are three forms of penalty: *Severe*, *Gentle*, *Hybrid*. In a Severe reaction to losses, the TCP flow does not transmit any data for τ_S rounds, irrespective of the value of L. This is equivalent to entering slow-start after incurring losses (e.g., TCP Tahoe). At time $t+\tau_S+1$ (after the time out), the TCP flow restarts by allowing βN packets to be in flight. In a Gentle reaction to losses, a TCP flow incurs a penalty proportional to the number of losses observed (by transmitting γL fewer packets than usual at time $t+1$, where γ is a small positive constant). This penalty reflects the cost of retransmissions without time-outs.[2]

In a Hybrid reaction to losses, the TCP flow incurs a purely gentle penalty up to a threshold number of losses ($L = 1$) and a purely severe penalty after that. The severe part of a Hybrid reaction differs from a pure Severe penalty in two key aspects. Firstly, the former models a time-out followed by a slow-start while the latter models just a slow-start. Secondly, at the end of the severe penalty in a Hybrid reaction to losses (when $L > 1$), the TCP flow restarts with N_0 packets outstanding, where N_0 is a positive constant. The reason for these differences from Severe penalty will be explained later in this section.

The Severe form of penalty models TCP-Tahoe flows. Tahoe flows exhibit mostly fixed reaction (fast-retransmit and slow-start) to losses, irrespective of their number. Also, Tahoe flows reduce their *ssthresh* variable by β upon incurring losses, before entering slow-start. Severe penalty in this form does not explicitly model the time outs in TCP-Tahoe. In fact, this form of penalty is more representative of versions of TCP that preceded Tahoe.

TCP-Reno loss recovery can be modeled as a Hybrid penalty. A Reno flow incurs a gentle penalty for up to a single loss within a window after which it incurs a severe penalty (by timing out and slow-starting). In fact, a Reno flow undergoes a few successive multiplicative decreases spread over as many round-trip times before timing out. In addition, the value of *ssthresh* is reduced by β with each such decrease. By stating that in a Hybrid Reaction, a TCP flow times out immediately after observing more then a single loss and that after a time out, the flow restarts by keeping a constant number of packets outstanding, we are approximating the effect of these multiple decreases on a Reno flow. The severe part of the Hybrid penalty subsumes both the time out and the subsequent slow-start. Thus, the exact value of τ_S for a Severe penalty is smaller than that for a Hybrid penalty.

TCP SACK flows can sustain many losses within a single congestion window. In fact, unless a SACK flow sees so many losses within a window that there are less than 3 duplicate acknowl-

[2]For convenience, when a TCP flows shows a Severe reaction to losses, we say that it incurs or implements a Severe penalty and we refer to it as a Severe flow. Similarly, the terms Gentle penalty and Gentle flow can be defined.

Severe penalty (Tahoe):

$$
\begin{array}{lcll}
N_i^{t+1} & = & N_i^t + \alpha_i & \text{if } L_i^t = 0 \\
 & = & 0 & \text{if } L_i^{t'} \geq 1 \text{ where } t - \tau_S < t' \leq t \\
 & = & \beta_i N_i^{t'} & \text{if } L_i^{t'} \geq 1 \text{ and } t = \tau_S + t'
\end{array}
$$

Gentle penalty (SACK):

$$
\begin{array}{lcll}
N_i^{t+1} & = & N_i^t + \alpha_i & \text{if } L_i^t = 0 \\
 & = & \beta_i N_i^t - \gamma L_i^t & \text{if } L_i^t \geq 1
\end{array}
$$

Hybrid Penalty (Reno):

$$
\begin{array}{lcll}
N_i^{t+1} & = & N_i^t + \alpha_i & \text{if } L_i^t = 0 \text{ and } N_i^t > 0 \\
 & = & \beta_i N_i^t - \gamma & \text{if } L_i^t = 1 \\
 & = & 0 & \text{if } L_i^{t'} > 1 \text{ where } t - \tau_S < t' \leq t \\
 & = & \beta_i N_0 & \text{if } L_i^{t'} \geq 1 \text{ and } t = \tau_S + t' \\
 & & & \text{Here, } N_0 \text{ is a constant.}
\end{array}
$$

Table 1: The Penalty Models for TCP-Tahoe, TCP-SACK and TCP-Reno. $\gamma > 0$ is a small constant.

edgments received or unless a retransmitted packet is lost again, it will not time-out. Since such time-outs for SACK are highly uncommon in reality, we consider TCP SACK flows to implement a Gentle penalty. The definitions of the three penalty models are stated formally in Table 1.

In the above model, the number of losses L_i^t seen by a flow is determined by the queue management algorithms used. This fact is discussed in greater detail in the next section.

It should be noted, however, that we do not claim that this model is realistic. We only claim, and show in our later results, that it reproduces the TCP Game behavior seen in the more realistic simulations. Thus, it appears to capture the aspects of reality most relevant to the question we are addressing.

4. ANALYSIS OF THE TCP GAME

In this section, we describe the results from analysis and simulation of the TCP Game. For the analysis, we use the penalty-based model presented in the previous section. We begin this section by describing the simulation set-up and the methodology used for obtaining the Nash equilibrium experimentally. We then present the results of these simulations and of our analysis of the penalty-based model. We do so by considering each combination of loss-recovery and queue management algorithm in turn, first describing the analytical results for that setting and then presenting the corresponding simulation results.

4.1 Simulation Methodology

Since we assume that all flows traverse a single common bottleneck, we use the standard dumb-bell topology shown in Figure 1 for our simulations. Flow F_i, $i = 1 \ldots n$, traverses the path from S_i to D_i. In all our simulations, we set the bottleneck capacity C to 10Mbps and we fix $n = 10$. We now describe our simulation methodology for arriving at the Nash equilibrium of the TCP Game when the flows are allowed to vary their increase parameters alone.

When varying α we use the following procedure. We run our simulations in iterations. In the j^{th} iteration, we fix the parameters for flows $F_1 \ldots F_{n-1}$ to the single tuple (α^j, β). Let the parameters for the flow F_n be denoted by the tuple $(\alpha^{'}, \beta)$. We run simulations for values of α' in the interval $max\{0, \alpha^j - \Delta\} \leq \alpha^{'} \leq \alpha^j + \Delta$, where Δ is a fixed large positive constant. Henceforth, we will use the notation $\alpha^{j,low} = max\{0, \alpha^j - \Delta\}$ and $\alpha^{j,high} = \alpha^j + \Delta$. For each value of $\alpha^{'}$ we record the value of $G_n(\alpha^{'})$ and define $\alpha^{j,best}$ as the value of α' that maximized G_n. The next iteration, $j+1$ starts with the parameters for all the n flows set to $(\alpha^{j,best}, \beta)$. The simulation stops when at the end

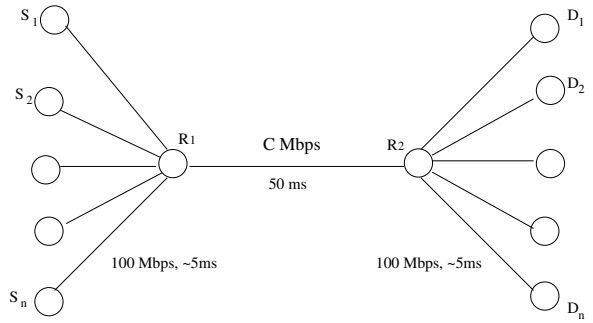


Figure 1: A single-bottleneck topology.

of some iteration k, $\alpha^{k,best} = \alpha^k$; this value, denoted by α_E, is the Nash equilibrium value of α. We then say that for the given situation, the parameters at Nash equilibrium are (α_E, β) for all flows. Notice that the simulation methodology we use assumes that the Nash equilibrium is symmetric.

When varying α, we set $\beta = 0.5$ and all simulations start with $\alpha^1 = 1$. In every iteration, for each value of $\alpha^{'}$, we run the simulation 20 times (the total simulation time is 100s and we discard the first 50s of simulation data to allow the flows reach steady state). In each of the 20 runs, the start times of the n flows are randomized. We use the average value of the goodput seen by F_n in these runs for the value of G_n. In obtaining the value of $\alpha^{j,best}$ we use the following criterion: when comparing α_1 and α_2 ($\alpha_1 \neq \alpha_2$), we say that $G_n(\alpha_1) < G_n(\alpha_2)$ if and only if the 85% confidence around the values of $G_n(\alpha_1)$ and $G_n(\alpha_2)$ do not overlap. The value 85% was chosen based on our experience with these simulations, but is not supported by any principled argument.

The simulations for β variation are similar. In this set of simulations, we set $\alpha = 1$ for all flows. In addition, we set $\beta^{j,low} = 0.5$ and $\beta^{j,high} = 0.99$. We do not test for values of $\beta < 0.5$ since they are clearly sub-optimal. The methodology we use in our simulations with flows varying α and β simultaneously is also similar and is a combination of the above two simulation set-ups.

We consider three forms of loss recovery (and three associated penalty functions in our analytical model): SACK (Gentle), Tahoe (Severe penalty), and Reno (Hybrid penalty).[3] We also consider two forms of buffer management: simple drop-tail and RED. We discuss the six possible combinations in the following subsections, starting with the analytical results with the penalty model and then comparing it to the simulation results on the actual TCP algorithm. We start by presenting the results for drop-tail routers, and then discuss RED routers.

4.2 FIFO Drop-Tail Gateways

When FIFO drop-tail buffers are used, all flows experience losses at about the same instant of time, which we call the *overflow point*. The overflow point, in fact, spans an entire round. The number of losses assigned by the FIFO drop-tail buffer to flow F_i at an overflow point is exactly equal to its increase number α_i. Thus, $L_i = \alpha_i$ upon overflow. We justify this loss assignment policy of FIFO drop-tail buffers below.

Let us consider a round, at the start of which the buffer is exactly full (i.e., at the overflow point). TCP ensures all transmissions are ACK-clocked. ACK-clocking, in turn, ensures that no losses occur, even if the buffer is full, as long as each flow continues to send at the same rate as at the start of the round. However, assuming that α_i is an integer, flow F_i increases its value of N_i for the subsequent round by α_i causing an increase in the sending rate. In fact, from the way congestion window increase is defined in TCP, N_i increases gradually with each incoming ACK. At each instant when there is an increment of 1

[3]In this paper, we do not model/analyze TCP-Newreno. However, our NS-2 simulations have shown that the Nash equilibria of the TCP Game for Newreno flows are similar to those for SACK flows.

Notation	*Description*
C	The capacity of the bottleneck link (the bandwidth-delay product)
R	The propagation round trip delay of the bottleneck link
$\overline{R}$	The mean round trip time ($R + R_Q$), where R_Q is the mean queueing delay on the link
N_i^t	The number of packets transmitted by flow F_i in round t
$\mathcal{N}_i$	The value of N_i immediately after an overflow point, in steady-state.
L_i^t	The number of losses experienced by flow F_i in round t
G_i	The mean goodput of flow F_i
α_i	The increase parameter for flow F_i
β_i	The decrease parameter for flow F_i
A	$\sum_{i=1}^{n-1} \alpha_i$
τ_S	The number of rounds spent incurring severe penalty
α_E	The common value of α for all flows at Nash equilibrium (when flows are allowed to vary their increase parameters)
β_E	The common value of β for all flows at Nash equilibrium (when flows are allowed to vary their decrease parameters)
τ_O	The number of rounds between successive buffer overflow points (drop-tail buffers) or The expected number of rounds between successive multiplicative decreases of a flow (RED buffers)
S_i	The number of packets transmitted by flow F_i between a pair of consecutive overflow points
T	The total length of the period between two consecutive overflow points in seconds.

Table 2: Notation used in our analysis.

in N_i, F_i bursts out two packets back-to-back because of the increase in the number of packets it is allowed to keep outstanding. This causes a temporary disturbance to the ACK-clocked transmission of F_i. The result is a buffer overflow and F_i experiencing a loss each time N_i is increased by 1. It follows from this that the number of losses seen by flow F_i at the overflow point is exactly α_i.[4]

We summarize the notation we use in the subsequent sections in Table 2 for easy reference. We now look at each of the three forms of loss-recovery – SACK/Gentle, Tahoe/Severe and Reno/Hybrid – in turn.

4.2.1 TCP-SACK/Gentle Penalty

Analysis of Gentle Penalty Let C denote the capacity of the bottleneck link in packets (i.e., C is the bandwidth-delay product). From Assumption (V) (Section 3.1), the size of the FIFO drop-tail buffer is C. Let τ_O denote the number of rounds between consecutive overflow points in steady state. For each flow F_i, we can write

$$G_i = \frac{S_i}{T} \quad (1)$$

where S_i is the number of packets transmitted by flow F_i between a pair of consecutive overflow points and T is the total length of the period between two consecutive overflow points in seconds, as defined in Table 2.

Let $N_i(k)$ be the value of the N_i after the kth overflow point and let $\mathcal{N}_i$ be the limiting value of $N_i(k)$ as $k \to \infty$. From the definition of AIMD, we can write,

$$\mathcal{N}_i = \beta(\mathcal{N}_i + (\tau_O \alpha_i - \gamma \alpha_i))$$

since $L_i = \alpha_i$. This gives us

$$\mathcal{N}_i = \frac{\beta}{1-\beta}(\tau_O \alpha_i - \gamma \alpha_i) \approx \frac{\beta}{1-\beta}\tau_O \alpha_i \quad (2)$$

Also, from the definition of an overflow point, we have, in steady-state,

$$\sum_{i=1}^{n}(\mathcal{N}_i + \tau_O \alpha_i) = 2C \quad (3)$$

since both the buffer and the link are full at an overflow point.

The number of packets, S_i, transmitted by flow F_i in the period of $\tau_O + 1$ rounds between the end of an overflow point and the end of the subsequent overflow point[5], in steady-state is given by

$$S_i = \sum_{t=0}^{\tau_O} \mathcal{N}_i + t\alpha_i \quad (4)$$

These equations apply to both the α and β variation analyses, which we present next. In both cases, we assume that all the flows have reached their steady-state.

Varying α Setting $\beta = 0.5$ in Equation 2, we obtain

$$\mathcal{N}_i \approx \tau_O \alpha_i \quad (5)$$

Substituting Equation 5 in Equation 4, we get, $S_i = \alpha_i \sum_{t=0}^{\tau_O}(\tau_O + t)$. The length of each of the rounds between overflow points, in seconds, is different, due to the queueing at the bottleneck router. In particular, if we let R denote the base propagation RTT, then the length of the t^{th} round since the last overflow point would be $R\left(1 + \frac{t-1}{\tau_O}\right)$ for $1 \le t \le \tau_O + 1$ (Assuming that queueing delay varies linearly from 0 to R over the τ_O rounds). Thus, the total length in seconds of the period of between overflow points, T, is given by $T = \sum_{t=0}^{t=\tau_O} R\left(1 + \frac{t}{\tau_O}\right) = \frac{R}{\tau_O}\sum_{t=0}^{\tau_O}(\tau_O + t)$.

Using the expressions for S_n and T derived above in Equation 1, we obtain, $G_n = \frac{S_n}{T} = \frac{\alpha_n \tau_O}{R}$. From Equations 3 and 5, $\tau_O = \frac{C}{\sum_{i=1}^{n} \alpha_i}$. This gives

$$G_n = \frac{\alpha_n C}{R(A + \alpha_n)} \quad (6)$$

where $A = \sum_{i=1}^{n-1} \alpha_i$. From the above expression for gooodput, it is easy to see that given the values of α_i for flows $F_1, \ldots, F_{n-1}$, the value of $G_n(\alpha_n)$ is strictly increasing in α_n. Hence at Nash equilibrium, α_E could be arbitrarily large.

Varying β For analyzing the Nash equilibrium resulting from allowing the flows to vary β, we compare the goodputs of flow F_n resulting from the following two settings of the parameters of the n flows:

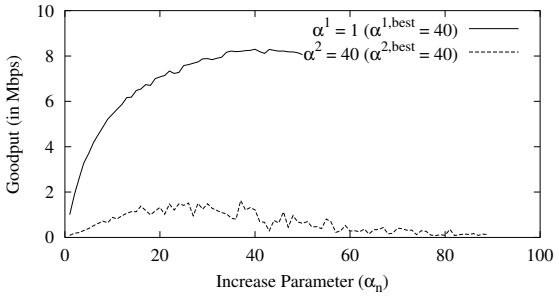
(i) Flows F_i, $i = 1, \ldots, n-1$ all have a decrease parameter of $\beta_i = \beta$ (fixed). The decrease parameter of flow F_n, β_n is larger than β. Let $G_n(\beta_n)$ be the goodput of F_n in this setting.

(ii) All the n flows have the same decrease parameter β. Let $G_n(\beta)$ denote the goodput of flow F_n in this setting.

We will use the superscripts (i) and (ii) to differentiate quantities in either setting. For example, we will Let $\mathcal{N}_i^{(i)}$ and $\mathcal{N}_i^{(ii)}$ denote the values of $\mathcal{N}_i$ in settings (i) and (ii) respectively. In either setting, $\alpha_i = 1$ for $i = 1, \ldots, n$.

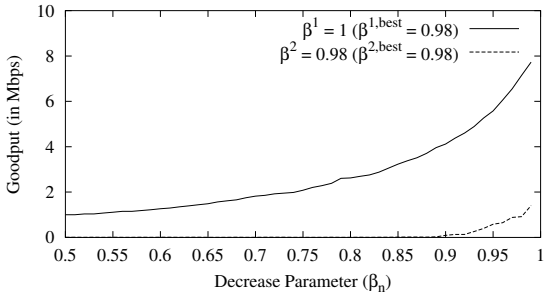
[4] We have noticed that even with (on-off) cross traffic, the average number of losses observed by a flow is roughly equal to its increase parameter. It was also observed by the designers of RED that drop-tail routers penalize bursty behavior [13].

[5] Henceforth, we will also refer to these $\tau_O + 1$ rounds as the rounds "between" overflow points



α_E	Goodput	Loss Rate
40	0.95Mbps	20.32

(a)



β_E	Goodput	Loss Rate
0.98	1.00Mbps	1.80

(b)

Figure 2: Simulation results for Gentle flows with FIFO drop-tail buffers. In (a) we show the results for the Nash equilibrium when flows vary their increase parameters. The results for the Nash equilibrium when flows vary their decrease parameters are shown in (b).

Now, setting $\alpha_n = 1$ in equation 2, we get

$$\mathcal{N}_n^{(i)} \approx \frac{\beta_n}{1-\beta_n}\tau_O^{(i)} \quad (7)$$

$$\mathcal{N}_n^{(ii)} \approx \frac{\beta}{1-\beta}\tau_O^{(ii)} \quad (8)$$

Assuming that the difference between β_n and β is negligible, the average length of any round in either setting would be approximately the same. Let $\overline{R}$ denote this value. It is not hard to see that $\overline{R} \approx 2\beta R$. We can immediately write the following equations for the total time in seconds between consecutive overflow points in either setting:

$$T^{(i)} = R(1+\beta)(\tau_O^{(i)}+1)$$
$$T^{(ii)} = R(1+\beta)(\tau_O^{(ii)}+1)$$

Now, in Setting (i) flow F_n transmits $S_n^{(i)} = \sum_{t=0}^{t=\tau_O^{(i)}} (\frac{\beta_n}{1-\beta_n}\tau_O^{(i)} + t)$ packets (using Equation 7 in Equation 4). Hence, from Equation 1, after simplification, we get the following expression for Setting (i):

$$G_n(\beta_n) = \frac{\tau_O^{(i)}(1+\beta_n)}{2R(1-\beta_n)(1+\beta)} \quad (9)$$

Similarly, for Setting (ii), we get,

$$G_n(\beta) = \frac{\tau_O^{(ii)}}{2R(1-\beta)} \quad (10)$$

Using Equations 7 and 8 in Equation 3, we get:

$$(n-1)\tau_O^{(i)}\frac{1}{1-\beta} + \tau_O^{(i)}\frac{1}{1-\beta_n} = 2C \quad (11)$$

$$n\tau_O^{(ii)}\frac{1}{1-\beta} = 2C \quad (12)$$

From Equation 11, $\tau_O^{(i)} > 2C\frac{1-\beta_n}{n}$. Using this inequality in Equation 9 and using Equation 12 in Equation 10, we get,

$$G_n(\beta_n) > \frac{C(1+\beta_n)}{2nR(1+\beta)} > \frac{C}{nR} = G_n(\beta) \quad (13)$$

In effect, greedy flows always stand to gain by setting their decrease parameters slightly more aggressively than the competing flows. This implies that at Nash equilibrium $\beta_E \to 1$.

Simulation of SACK Simulation results for the Nash equilibrium of the TCP Game when TCP-SACK flows are allowed to change their only their increase parameters or only their decrease parameters are shown in Figures 2(a) and (b), respectively. Each curve in either figure represents one iteration in the simulation. For each curve (iteration j), we identify the common congestion control parameter (α^j or β^j, as the case may be) for flows $F_1 \ldots F_{n-1}$. The goodput obtained by flow F_n as its congestion control parameter (α_n or β_n, as the case may be) is varied between two extreme values ($\alpha_n \in [max0, \alpha^j - 50, \alpha^j + 50]$ and $\beta_n \in [0.5, 0.99]$) is plotted on the y-axis as a function of the parameter of flow F_n shown on the x-axis. For each iteration, we also identify the value of the parameter for flow F_n resulting in the best goodput given the parameters of flows $F_1 \ldots F_{n-1}$. We also show the average per-flow goodput and loss rate at Nash equilibrium in the table below each figure.

When flows are allowed to vary their increase parameters, we obtain $\alpha_E = 40$ (Figure 2(a)), which is very aggressive, as predicted by our analysis. At this Nash equilibrium, although the average goodput is reasonable and the per-flow loss rate is extremely high. Thus, the Nash equilibrium is undesirable.

From Figure 2(b), as shown by our analysis, $\beta_E = 0.98$, at Nash equilibrium when flows are allowed to vary their decrease parameters. Though the parameters are set aggressively at Nash equilibrium, the average per-flow good-put and loss rates are very reasonable. Besides, at values of β_E close to 1, the decrease undergone by the flows upon incurring losses is equivalent to an additive decrease (by one packet). As such, the loss rate would not be any worse even if $\beta_E > 0.98$. Hence, we do not consider this Nash equilibrium to be undesirable, in terms of efficiency. However, the additive decrease makes this Nash equilibrium unfair [6]. This is in agreement with the conclusions drawn in [2].

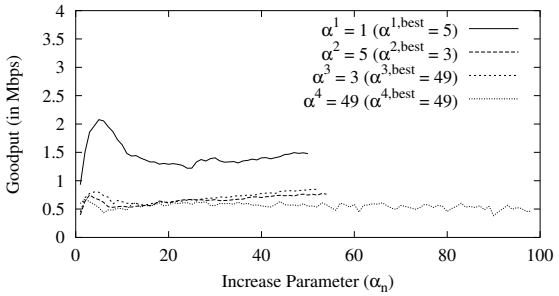
We also perform simulations in which TCP-SACK flows are allowed to vary their increase and decrease parameters simultaneously. For lack of space, we omit the corresponding graphs from the presentation and summarize the results in words instead (This is also true of the simulation results for simultaneous variation of α and β in upcoming sections). From simulations for TCP-SACK flows, $(\alpha_E, \beta_E) = (15, 0.98)$ at the symmetric Nash equilibrium. The average goodput (0.95Mbps) is reasonable, but the loss rate (26%) is extremely high making this Nash equilibrium undesirable. Notice that these results are in agreement with those of α variation and β variation.

In summary, we make the following observation:

OBSERVATION 1. *Given SACK/Gentle flows and FIFO drop-tail buffers, the Nash equilibrium resulting from α variation is highly undesirable. When flows vary their β alone, the network continues to operate efficiently at the resulting Nash equilibrium in spite of the aggressive parameter setting. When flows are allowed to simultaneously vary both α and β the resulting Nash equilibrium is, again, undesirable.*

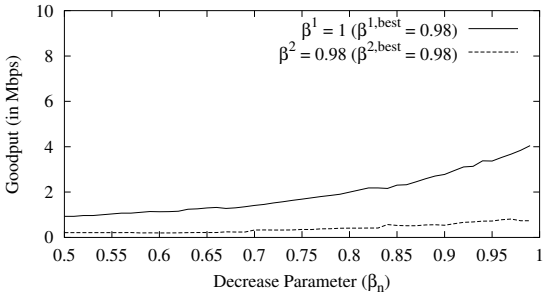
4.2.2 TCP-Tahoe/Severe Penalty

Analysis of Severe Penalty We now analyze the situation in which all the n flows implement Severe penalty. Suppose that an overflow occurs in round t_0 and that all the n flows are in



α_E	Goodput	Loss Rate
49	0.60Mbps	11.44

(a)



β_E	Goodput	Loss Rate
0.98	0.76Mbps	1.19

(b)

Figure 3: Simulation results for Severe flows with FIFO drop-tail buffers.

steady-state. Then, by definition of Severe penalty, the n flows do not transmit any new packets into the network for the next τ_S rounds (The length of each of these rounds is exactly R, as there is no queueing at the router). Suppose that it takes an additional τ rounds before the next overflow point. Then, $\tau_O = \tau_S + \tau$.

Let $\mathcal{N}_i$ denote the steady-state value of the number of outstanding packets of flow F_i after it recovers from a severe penalty following a buffer overflow. Then, we have the following parallel to Equation 2:

$$\mathcal{N}_i = \frac{\beta}{1-\beta}\tau\alpha_i \quad (14)$$

We also have the following parallels to Equations 3 and 4, respectively:

$$\sum_{i=1}^{n}(\mathcal{N}_i + \tau\alpha_i) = 2C \quad (15)$$

$$S_i = \sum_{t=0}^{\tau}\mathcal{N}_i + t\alpha_i \quad (16)$$

We now discuss α and β variation in turn.

Varying α Setting $\beta = 0.5$ in Equation 14, we get

$$\mathcal{N}_i = \tau\alpha_i \quad (17)$$

Applying Equation 17 to Equation 16 we get, $S_n = \frac{3\alpha_n\tau(\tau+1)}{2}$. The total length of the τ_O rounds in seconds is $T = R\tau_S + \sum_{t=0}^{t=\tau} R\left(1+\frac{t}{\tau}\right) = R\left(\tau_S + \frac{3(\tau+1)}{2}\right)$. Using these expressions for S_n and T in Equation 1 we get,

$$G_n = \frac{3\alpha_n\tau(\tau+1)}{R(2\tau_S + 3(\tau+1))}$$

From Equations 15 and 17, we obtain $\tau = \frac{C}{\sum_{i=1}\alpha_i}$. It is not hard to see that G_n increases with α_n for fixed values of α_i, $i = 1 \ldots n-1$.[6] Thus, when flows implement Severe penalty, α_E could grow arbitrarily large, at Nash equilibrium.

Varying β As before, we compare the throughput resulting from the settings (i) and (ii) presented in Section 4.2.1. Let τ^i and τ^{ii} denote the number of rounds between successive overflow points excluding the rounds in which flows incur severe penalty, as defined above for α variation. The rest of the variables are as defined in Section 4.2.1.

The following equations hold immediately:

$$T^{(i)} = R\tau_S + R(1+\beta)(\tau^{(i)}+1)$$
$$T^{(ii)} = R\tau_S + R(1+\beta)(\tau^{(ii)}+1)$$

[6]Here we use the fact that if $f(x) > 0$ and $g(x) > 0$ are continuous, differentiable functions of x and $\frac{f}{g}$ is increasing in x, then $\frac{f}{g+c}$ is increasing in x for any constant $c > 0$.

It is not hard to see (similar to Equations 9 and 10) that

$$G_n(\beta_n) = \frac{\tau^{(i)}(1+\beta_n)(\tau^{(i)}+1)}{2R\tau_S(1-\beta_n) + 2R(1-\beta_n)(1+\beta)(\tau^{(i)}+1)}$$

$$G_n(\beta) = \frac{\tau^{(ii)}(1+\beta)(\tau^{(ii)}+1)}{2R\tau_S(1-\beta) + 2R(1-\beta)(1+\beta)(\tau^{(ii)}+1)}$$

It can be shown that there always exists $\beta_n > \beta$ such that $G_n(\beta_n) < G_n(\beta)$ as long as $n > 2$ (We omit the proof of this fact). Hence, at Nash equilibrium $\beta_E \to 1$, when flows implement Severe penalty.

Simulation of Tahoe We show the results from our simulations for 10 flows in Figures 3(a) and (b). From Figure 3(a), when flows are allowed to vary their increase parameters, $\alpha_E = 49$. The average goodput and loss rate are poor rendering this Nash equilibrium undesirable. Also, from Figure 3(b), $\beta_E = 0.98$. Though the loss rate at this Nash equilibrium is low, the per-flow goodput is poor. As a result, this Nash equilibrium is undesirable too. When both α and β are varied simultaneously, $(\alpha_E, \beta_E) = (1, 0.98)$ at Nash equilibrium. (TCP Tahoe flows gain much lesser from varying α than they do from varying β. This results in a conservative setting of α at Nash equilibrium.) Again this Nash equilibrium is undesirable just as that resulting from β variation.

In effect, we could state the following:

OBSERVATION 2. *The Nash equilibrium of the TCP Game in which the TCP flows implement Tahoe/Severe penalty and FIFO drop-tail routers are employed results in inefficient network operation.*

4.2.3 TCP-Reno/Hybrid Penalty

We use a slightly different method for analyzing Hybrid penalty.

Varying α Here, we compare the goodput of flow F_n resulting from the following two settings of parameters of the n flows:

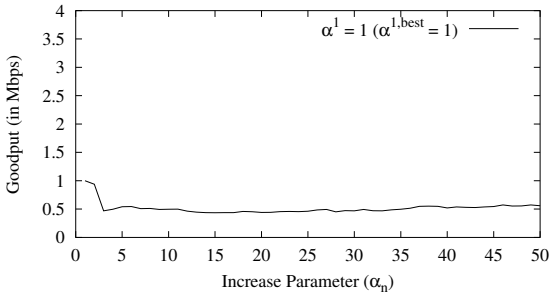
(i) Flow F_n has an increase parameter of $\alpha_n > 1$ while all the remaining flows have increase parameters of 1. Let $G_n(\alpha_n)$ denote the goodput of flow F_n in this setting.

(ii) All flows have an increase parameter of 1. Let $G_n(1)$ denote the goodput of flow F_n in this setting.

From the definition of Hybrid penalty and from Equation 6, we obtain $G_n(1) = \frac{C}{nR}$.

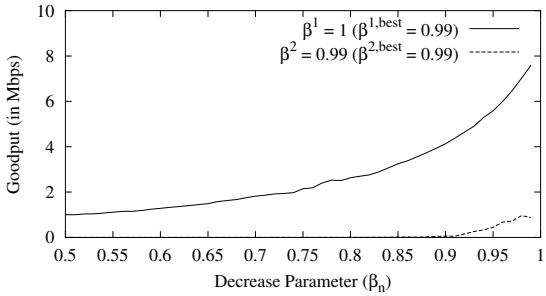
Let us now consider Setting (i) where $\alpha_n > 1$. Assume that $\frac{C}{n} > \tau_S$. Since $L_i = \alpha_i$, from the definition of Hybrid penalty, flow F_n would incur a severe penalty for τ_S rounds at the end of any overflow point, while all the others would still be in the *gentle regime* of the Hybrid penalty. From Equation 5, we have, $\mathcal{N}_i \approx \tau_O$ for $i = 1 \ldots n-1$. However, using $N_0 = \frac{C}{n}$ in the definition Hybrid penalty, for flow F_n, we have $\mathcal{N}_n = \frac{C}{n}$. Using these expressions in Equation 3, we obtain

$$\tau_O = \frac{C\left(2 - \frac{1}{n}\right) + \tau_S\alpha_n}{2(n-1) + \alpha} \quad (18)$$



α_E	Goodput	Loss Rate
1	1.00Mbps	0.15

(a)



β_E	Goodput	Loss Rate
0.98	1.00Mbps	1.50

(b)

Figure 4: Simulation results for Hybrid flows with FIFO drop-tail buffers.

Notice that $\tau_O > \tau_S$ since $\frac{C}{n} > \tau_S$. Now, from Equation 16, the total number of packets transmitted by F_n in between two consecutive overflow points is given by $S_n = \sum_{t=0}^{\tau_O - \tau_S}(\frac{C}{n} + t\alpha_n)$ or

$$S_n = \frac{C}{n}(\tau_O - \tau_S + 1) + \frac{\alpha_n}{2}(\tau_O - \tau_S)(\tau_O - \tau_S + 1)$$

For the sake of simplicity, we can approximate the total length of the τ_O rounds to be $T = \frac{R}{\tau_O}\sum_{t=0}^{\tau_O}(\tau_O + t) = \frac{3R(\tau_O+1)}{2}$. Thus, from Equation 1, we obtain

$$\begin{aligned} G_n(\alpha_n) &= \frac{S_n}{T} \\ &= \frac{2}{3R}\left(\frac{C}{n} + \frac{\alpha_n}{2}(\tau_O - \tau_S)\right)\left(1 - \frac{\tau_S}{\tau_O + 1}\right) \end{aligned}$$

A numerical evaluation of the inequality $G_n(\alpha_n) < G_n(1)$, with the settings $\tau_S = 20$ rounds and $R = 0.12$ yields the following results:

1. When the number of flows n is small (≈ 50), the inequality holds for all values of α_n as long as $\frac{C}{n} < 2.5$Mbps.
2. When the number of flows is large (> 100), the inequality holds as long as $\frac{C}{n} < 1.5$Mbps, for all values of α_n.
3. In either of the above cases, when $\frac{C}{n}$ exceeds the given values, the flow requires $\alpha_n > 75$, roughly, to obtain goodput significantly better than $G_n(1)$.

These observations are intuitive because if $\frac{C}{n}$ was very large, then the greedy flow would have enough time to catch up with the other flows after incurring a severe penalty. A greedy flow does not have this advantage for a low value of $\frac{C}{n}$. The above result is significant because the average per-flow goodput in the Internet is usually much lower than 2Mbps.

We assumed that $\frac{C}{n} > \tau_S$ in the above analysis. If however, $\frac{C}{n} \leq \tau_S$, then the $n-1$ flows $F_1, \ldots, F_{n-1}$ would cause one or more overflows during the τ_S rounds that flow F_n spends incurring a time out. In fact, in this situation, flow F_n would require a much higher value of α_n than in the previous situation to observe the same gain in goodput, if any, over the $\alpha_n = 1$ case. For simplicity, we skip the analysis of this situation.

Notice that in the above derivation for $G_n(\alpha_n)$, we assume that the increase parameters if flows $F_1, \ldots, F_{n-1}$ are all one. A minor variation in the derivation for $G_n(\alpha_n)$ is enough to show that the above conclusion ($G_n(\alpha_n) < G_n(1)$ for all α_n) holds for a similar setting of R, τ_S and $\frac{C}{n}$ *no matter what the increase parameters of the other flows are.* This suggests that the social parameter setting of $\alpha_i = 1, \forall i$, is a *dominant strategy equilibrium* for the TCP Game: each flow has a fixed strategy (choice of parameters) that serves it best irrespective of the behavior of its competitors.

Varying β Since $\alpha_i = 1$ for $i = 1 \ldots n$, all flows see exactly one loss upon overflow. Therefore, all flows incur a gentle penalty irrespective of their decrease parameters. The analysis is the same as that for Gentle Penalty. Thus, at Nash equilibrium, $\beta_E \to 1$.

Simulation of Reno The results from our simulations are shown in Figure 4(a) and (b). When flows vary their increase parameters, $\alpha_E = 1$ (Figure 4(a) matching the default setting of the increase parameter. Also, from Figure 4(b), $\beta_E = 0.99$. At this latter Nash equilibrium, the per-flow goodput is high and the loss rate is low, much like SACK flows. Thus, *in neither case would there be a congestion collapse at Nash equilibrium.* However, the Nash equilibrium due to flows varying their decrease parameters is somewhat unfair (though to a lesser extent than with SACK flows). Again, this is in conformity with the conclusions in [2].

In addition, when both α and β are allowed to vary simultaneously in simulation, $(\alpha_E, \beta_E) = (1, 0.98)$ at the symmetric Nash equilibrium. In summary, we have the following observation:

OBSERVATION 3. *When TCP flows implement Reno/Hybrid penalty and FIFO drop-tail routers are employed, the parameter setting at the Nash equilibrium due to α variation coincides with the default parameter setting ($\alpha = 1$). When flows vary their β, the parameter setting at Nash equilibrium is aggressive. However, the network continues to operate efficiently. When flows vary both their parameters simultaneously, the Nash equilibrium is efficient.*

4.3 RED Gateways

While most routers in the wide-area today are FIFO drop-tail, RED deployment is increasing rapidly. Thus, we think it important to analyze the TCP Game in the presence of RED routers. In what follows, we first describe the loss assignment policy of RED and then outline the methodology we use to arrive at the Nash equilibrium analytically.

At any given instant of time, a RED router marks or drops incoming packets with almost the same *instantaneous* probability, irrespective of which flow the packet belongs to. This drop-policy of RED allows it to impose a fairly uniform *long-term* packet loss rate across all the flows traversing a RED router. Based on these facts, we model RED's loss assignment as follows: all flows traversing a RED router experience a common packet loss rate p. Moreover, p is a function of the congestion control parameters of the flows traversing the RED router.

Suppose that we are given n TCP flows traversing a RED router and that these flows are only allowed to change their increase parameters. Suppose further that flows $F_1, \ldots, F_{n-1}$ each have the same increase parameter of α. Let $p_a = p_{\alpha,\alpha}$ be the steady state loss rate common to all flows imposed by RED when the n^{th} flow also chooses the same increase parameter, that is, when $\alpha_n = \alpha$. Let $p'_a = p_{\alpha,\alpha'}$ be the new common loss rate experienced by the flows when flow F_n chooses an increase parameter $\alpha_n = \alpha' \neq \alpha$. Let $G_n(\alpha', p'_a)$ be the goodput of flow F_n when $\alpha_n = \alpha'$.

Now, in order to arrive at the Nash equilibrium of the resulting TCP Game, we need to have a notion of how p'_a depends on α' and α. However, as we show below, we could do with deriving a weaker set of dependences. Indeed, letting $G_n = G_n(\alpha', p'_a)$, if

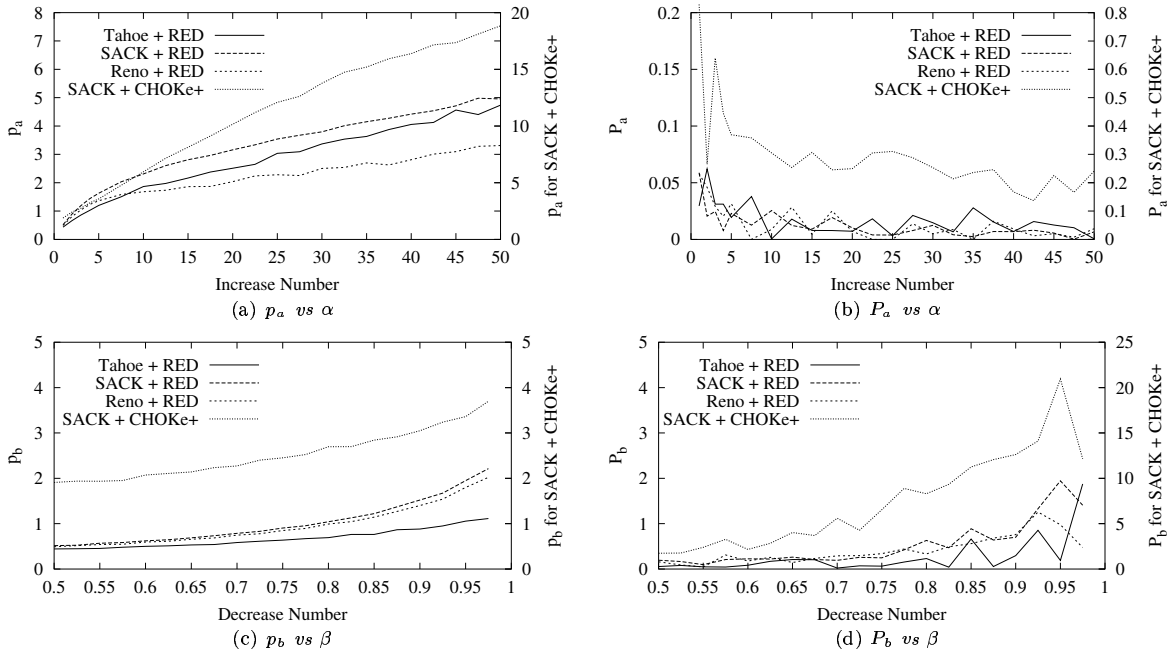


Figure 5: Graphs showing $p_a = p_{\alpha,\alpha'}$ and $P_a = \frac{dp'}{d\alpha'}|_{\alpha'=\alpha}$ as functions of α (Figures (a) and (b)) and $p_b = p_{\beta,\beta'}$ and $P_b = \frac{dp'}{d\beta'}|_{\beta'=\beta}$ as functions of β (Figures (c) and (d)). In either case, the results for RED gateways are plotted along the y-axis on the left. The results for CHOKe+, discussed in Section 5, are plotted along the y-axis on the right.

$\alpha_E = \alpha$ at Nash equilibrium, then we must have,

$$\frac{dG_n}{d\alpha'}|_{\alpha'=\alpha} = 0 \tag{19}$$

since the choice of α' for flow F_n coincides with the increase parameter α chosen by flows $F_1, \ldots, F_{n-1}$ at a symmetric Nash equilibrium (The converse is not necessarily true). However, $\frac{dG_n}{d\alpha'}|_{\alpha'=\alpha}$ is a function of α, p_a and $\frac{dp'_a}{d\alpha'}|_{\alpha'=\alpha}$ only. Hence, it is sufficient to obtain estimates of p_a and $\frac{dp'_a}{d\alpha'}|_{\alpha'=\alpha}$ as functions of α to compute the common increase parameter at Nash equilibrium, α_E.

Similarly, when flows are allowed to vary their decrease parameter, it is suffices to obtain estimates of p_b, $\frac{dp'_b}{d\beta'}|_{\beta'=\beta}$ as functions of β to compute the Nash equilibrium. Here, $p_b = p_{\beta,\beta}$ is the loss rate common to the n TCP flows when $\beta_i = \beta$ for $i = 1, \ldots, n$ and $p'_b = p_{\beta',\beta}$ is the common loss rate experienced by the n flows when $\beta_n = \beta' \neq \beta$ and $\beta_i = \beta$, for $i = 1, \ldots, n-1$.

Since our aim is only to model the drop policy of AQM schemes at a very high level, we do not delve into analytically deriving the above functions. Rather, we employ simulations to obtain measurements that help us estimate the above functions for RED gateways. In addition, it is important to note that the above functions may be different for TCP-Tahoe, Reno and SACK.

In Figures 5(a) and (b), we show how p_a and $\frac{dp'_a}{d\alpha'}|_{\alpha'=\alpha}$ vary as functions of α for the three TCP variants. Figures 5(c) and (d) show the corresponding results for β. We will use these estimates in the analyses presented in the following sections to obtain the congestion control parameters at Nash equilibrium. We now deal with each of Tahoe/Severe, SACK/Gentle and Reno/Hybrid cases in turn. We summarize our observations for RED gateways towards the end of this section.

4.3.1 TCP-Tahoe/Severe Penalty

Analysis of Severe Penalty We first derive an expression for the goodput of a Severe flow with congestion control parameters (α, β) experiencing a steady state packet loss rate p. We assume that the packet losses are distributed uniformly over the entire transmission interval of the flow and that the flow never experiences more than one loss, on an average, in a single round (that is, losses do not occur in bursts).

Let τ_O be the expected number of rounds between successive multiplicative decreases in the congestion window of this flow. Let $\tau = \tau_O - \tau_S$. In expectation, we can write, $\mathcal{N} = (\tau - \tau_S)\alpha\frac{\beta}{\beta-1}$ (using Equation 14). Then, from Equation 16, the expected total number of packets transmitted by the flow in these rounds between consecutive window decreases is given by

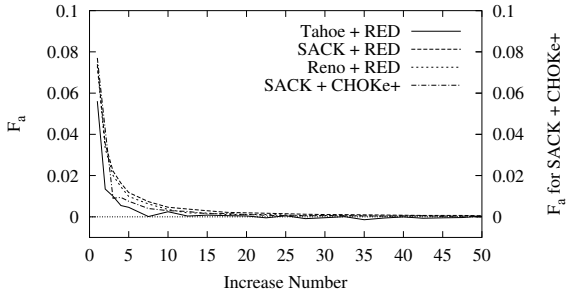
$$\begin{aligned} S &= \sum_{t=1}^{\tau} \mathcal{N} + t\alpha \\ &= \mathcal{N}(\tau+1) + \frac{\alpha\tau(\tau+1)}{2} \\ &\approx \frac{\alpha\tau^2(1+\beta)}{2(1-\beta)} \end{aligned}$$

Since we assume that the losses are uniformly distributed and never occur in bursts, we can write $p = \frac{1}{S}$. Eliminating S from these previous two equations we obtain, $\tau \approx \sqrt{\frac{2(1-\beta)}{p\alpha(1+\beta)}}$. The expected length (in seconds) of the τ_O rounds taken to transmit the S packets is $T = \overline{R}\tau_S + \overline{R}(\tau+1)$, where $\overline{R}$ is the average round trip delay (including the queueing delay). Hence, the expected goodput of the flow is, *approximately*, $G = \frac{S}{T} = \frac{S}{\overline{R}\tau_S + \overline{R}(\tau+1)}$. Now, from the fact $p = \frac{1}{S}$, we get,

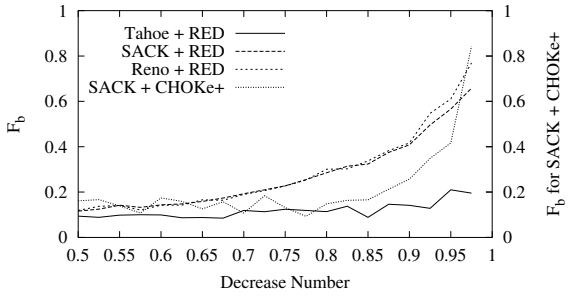
$$G \approx \frac{1}{p\overline{R}\tau_S + p\overline{R}\sqrt{\frac{2(1-\beta)}{p\alpha(1+\beta)}}} \tag{20}$$

Varying α We fix $\beta = 0.5$. Then, from Equation 20, we get

$$G_n(\alpha', p'_a) = \frac{1}{p'_a\overline{R}\tau_S + \overline{R}\sqrt{\frac{2p'_a}{3\alpha'}}} \tag{21}$$

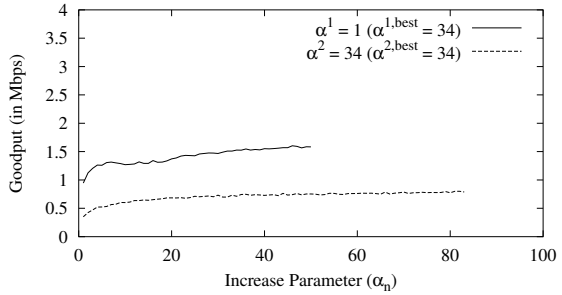


(a) F_a *vs* α



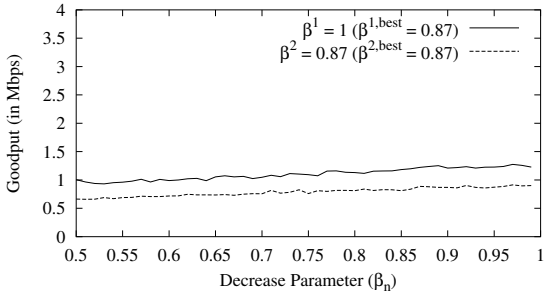
(b) F_b *vs* β

Figure 6: F_a **and** F_b **as functions of** α **and** β **respectively. The results for RED gateways are plotted along the** y**-axis on the left and those for CHOKe+ (discussed in Section 5) are plotted along the** y**-axis on the right.**



α_E	Goodput	Loss Rate
34	0.73Mbps	3.31

(a)



β_E	Goodput	Loss Rate
0.98	0.76Mbps	1.11

(b)

Figure 7: Simulation results for Severe flows with RED gateways.

If at Nash equilibrium $\alpha_E = \alpha$, then from Equations 19 and 21 we obtain,

$$F_a = G_n^2(\alpha, p_a)\sqrt{\frac{\alpha}{p_a}} \times \left(\left(\frac{p_a}{\alpha} - \left(\frac{dp'_a}{d\alpha'}\right)_{\alpha'=\alpha}\right) - \tau_S\sqrt{6p_a\alpha}\left(\frac{dp'_a}{d\alpha'}\right)_{\alpha'=\alpha}\right) = 0$$

Using the estimates for p_a and $\frac{dp'_a}{d\alpha'}|_{\alpha'=\alpha}$ from Figures 5(a) and (b) respectively, we plot the F_a as a function of α in Figure 6(a). Notice that when $\alpha < 22$, $F_a > 0$. This implies that $\alpha_E \geq 22$.

Varying β Here, we fix $\alpha = 1$. Using this in Equation 20, we get

$$G_n(\beta', p'_b) = \frac{1}{p\overline{R}\tau_S + p\overline{R}\sqrt{\frac{2(1-\beta')}{p(1+\beta')}}} \quad (22)$$

If $\beta_E = \beta$ at Nash equilibrium, then we can write the following condition analogous to Equation 19:

$$\frac{dG_n(\beta', p'_b)}{d\beta'}|_{\beta'=\beta} = 0 \quad (23)$$

Thus from Equations 22 and 23 we must have

$$F_b = G_n^2(\beta, p_b)\sqrt{\frac{1-\beta}{p_b(1+\beta)}} \times \left(\left(\frac{2p_b}{1-\beta^2} - P_b\right) - \tau_S\sqrt{\frac{2p_b(1+\beta)}{(1-\beta)}}P_b\right) = 0$$

where $P_b = \frac{dp'_b}{d\beta'}|_{\beta'=\beta}$. As before, we use the estimates in Figure 5(b) to numerically evaluate F_b as a function of β. The result is shown in Figure 6(b). Notice that $F_b > 0$ throughout suggesting that at Nash equilibrium, $\beta_E \to 1$.

Simulation of Tahoe The results for the Nash equilibria from the simulation of the TCP Game for TCP Tahoe flows with RED gateways are shown in Figure 7. From Figure 7(a), when Tahoe flows are allowed to vary their increase parameters, $\alpha_E = 34$, closely matching the analytical results. The Nash equilibrium is undesirable since the per-flow goodput at Nash equilibrium is rather low. When flows are allowed to vary their decrease parameters, $\beta_E = 0.98$ at Nash equilibrium, as predicted by analysis. Again, this Nash equilibrium is undesirable too due to the low per-flow goodput.

When α and β are varied simultaneously, $(\alpha_E, \beta_E) = (27, 0.98)$ at Nash equilibrium. The average goodput at this Nash equilibrium is 0.73Mbps and the per-flow loss rate is 2.5% making this an undesirable Nash equilibrium.

4.3.2 TCP-SACK/Gentle Penalty

Analysis of Gentle Penalty Setting $\tau_S = 0$ in Equation 20 we get, for a Gentle flow,

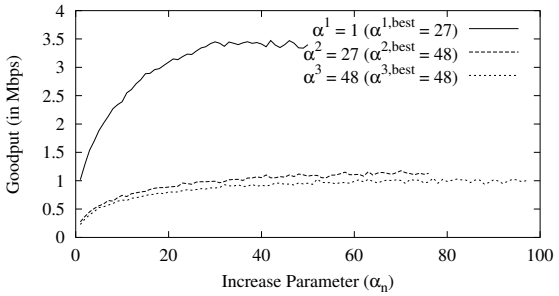
$$G \approx \frac{1}{\overline{R}}\sqrt{\frac{\alpha(1+\beta)}{2p(1-\beta)}} \quad (24)$$

Substituting Equation 24 in Equations 19 and 23, we get the following equations, respectively for the symmetric Nash equilibria of the TCP Game when allowing flows to vary their α and β individually:

$$F_a = G_n^2(\alpha, p_a)\sqrt{\frac{\alpha}{p_a}} \times \left(\frac{p_a}{\alpha} - \left(\frac{dp'_a}{d\alpha'}\right)_{\alpha'=\alpha}\right) = 0$$

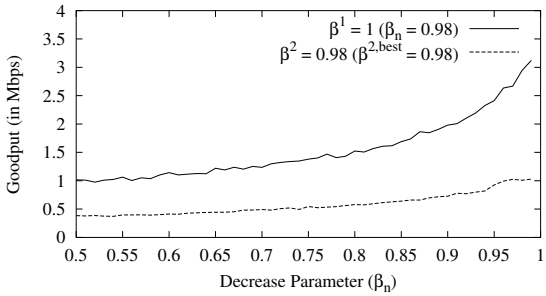
$$F_b = G_n^2(\beta, p_b)\sqrt{\frac{1-\beta}{p_b(1+\beta)}} \times \left(\frac{2p_b}{1-\beta^2} - \left(\frac{dp'_b}{d\beta'}\right)_{\beta'=\beta}\right) = 0$$

The estimates for p and $\frac{dp'}{d\alpha'}|_{\alpha'=\alpha}$ as functions of α for Gentle (TCP-SACK) flows are shown in Figures 5(a) and (b). We use



α_E	Goodput	Loss Rate
48	0.96Mbps	4.95

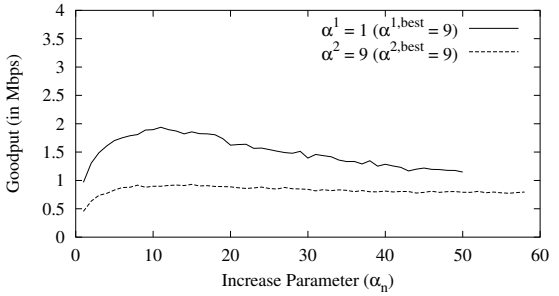
(a)



β_E	Goodput	Loss Rate
0.98	0.99Mbps	2.15

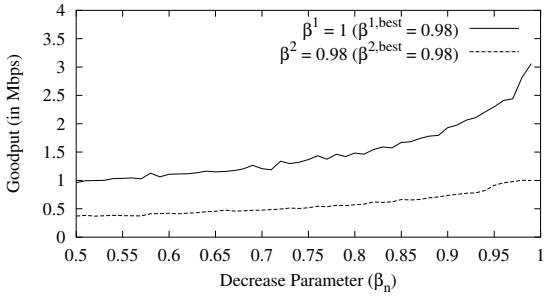
(b)

Figure 8: Simulation results for Gentle flows with RED buffers.



α_E	Goodput	Loss Rate
9	0.89Mbps	1.68

(a)



β_E	Goodput	Loss Rate
0.98	0.99Mbps	2.07

(b)

Figure 9: Simulation results for Hybrid flows with RED buffers.

these estimates to obtain F_a as a function of α as shown in Figure 6(a). Notice that for TCP-SACK, $F_a > 0$ implying that at a symmetric Nash equilibrium, α_E could be arbitrarily large for the combination of TCP-SACK and RED buffers.

From Figure 6(b), similarly, $F_b > 0$ throughout. Again, this implies that at Nash equilibrium, $\beta_E \to 1$.

Simulation of SACK Simulation results for the Nash equilibrium of the TCP Game with TCP-SACK flows and RED buffers are shown in Figure 8. When flows are allowed to vary their increase parameters, $\alpha_E = 49$. At the Nash equilibrium, the per-flow goodput (0.96Mbps) is reasonable but the loss rate (4.95%) is somewhat high. Moreover, the parameter setting at the Nash equilibrium, as predicted by analysis, is highly aggressive. As such, we conclude that the Nash equilibrium of the TCP Game with SACK flows varying their increase parameters and buffers employing RED is undesirable.

From Figure 8(b), when flows are allowed to vary their decrease parameter, $\beta_E = 0.98$. At this equilibrium both the per-flow goodput and loss rate are reasonable and as such this Nash equilibrium in not undesirable (The Nash equilibrium is unfair, however).

From our simulations in which we allow TCP-SACK flows to vary their increase and decrease parameters simultaneously, we obtain $(\alpha_E, \beta_E) = (23, 0.98)$. The average per-flow goodput at this Nash equilibrium is about 0.97Mbps and the per-flow loss rate is 5.70%. As argued above, this Nash equilibrium is undesirable.

4.3.3 TCP-Reno/Hybrid Penalty

Analysis of Hybrid Penalty Instead of deriving the expression of the goodput of Hybrid flows, we use the standard equation for the goodput of TCP Reno [19, 23] flows:

$$G = \frac{1}{R\sqrt{\frac{2p(1-\beta)}{\alpha(1+\beta)}} + 3T_0 p(1+32p^2)\sqrt{\frac{p(1-\beta^2)}{2\alpha}}} \qquad (25)$$

Again, using Equation 25 in Equations 19 and 23, we get the following equations for the Nash equilibria with varying α and varying β respectively:

$$F_a = C_a\left(\frac{p_a}{\alpha} - P_a\right) - C_a\frac{9T_0}{2R}\left((1.5p_a + 112p_a^3)P_a - \frac{p_a^2}{2\alpha}(1+32p_a^2)\right) = 0$$

$$F_b = C_b\left(\frac{2p}{1-\beta^2} - P_b\right) - C_b\frac{3T_0P_b}{R}(1+\beta)(1.5p + 112p^3) + \frac{3T_0(p^2+32p^4)\beta}{R(1-\beta)} = 0$$

where $C_a = G_n^2(\alpha, p_a)\sqrt{\frac{\alpha}{p_a}}$, $C_b = G_n^2(\beta, p_b)\sqrt{\frac{1-\beta}{p_b(1+\beta)}}$, $P_a = \frac{dp'_a}{d\alpha'}|_{\alpha'=\alpha}$ and $P_b = \frac{dp'_b}{d\beta'}|_{\beta'=\beta}$.

We plot F_a and F_b as functions of α and β in Figures 6(a) and (b) respectively, using the estimates from Figure 5 as before. In either case $F > 0$ throughout. This implies that at the respective Nash equilibria, α_E can be arbitrarily large and $\beta_E \to 1$.

Simulation of Reno Results from the simulation of the TCP Game with Reno flows and RED gateways are shown in Figure 6. When flows vary their increase parameter, $\alpha_E = 9$. At this Nash equilibrium, both the average goodput (0.90Mbps) and the per-flow the loss rate (1.68%) are reasonable. While this Nash equilibrium is not as undesirable as the cases of Severe and Gentle penalties, it is nevertheless worse than the default parameter setting in terms of the per-flow goodput.

As in the previous situations, when flows vary their decrease parameter, the resulting Nash equilibrium is not undesirable both in terms of the per-flow goodput and loss rate. Again, this is in agreement with the observations in [2] (In fact, as shown in [2], fairness is also reasonable at this equilibrium).

When flows are allowed to vary both their parameters simultaneously, at Nash equilibrium $(\alpha_E, \beta_E) = (3, 0.98)$. At this Nash

equilibrium, the average per-flow loss rate is 2.75% and the per-flow goodput is about 0.94Mbps. Again, this Nash equilibrium is somewhat less desirable when compared to the default parameter setting in terms of per-flow goodput, though it is better than those resulting from using the other forms of penalty.

Finally, we summarize the results for RED buffers as follows:

OBSERVATION 4. *When RED gateways are employed, all the Nash equilibria resulting from allowing flows to vary their increase parameters are undesirable (irrespective of the loss recovery scheme employed) in comparison with the default parameter setting since they either result in a low per-flow goodput or a high per-flow loss rate. However, allowing the flows to vary their decrease parameters does not result in undesirable Nash equilibria (Except when flows implement Severe penalty).*

4.4 Discussion

Our goal in this paper was to see if selfish behavior of network end-points would have an undesirable effect on the efficiency of the network. Intuition suggests that aggressive congestion control behavior would always increase a flow's bandwidth share so greed would always result in overly aggressive flows and inefficient network operation. However, as our analysis in the previous sections has shown, this intuition is not always right.

Until recently, the most common deployed scenario in the Internet was end-hosts implementing TCP-Reno loss recovery and FIFO drop-tail buffer management. In this situation, selfish behavior does *not* result in inefficient network behavior.[7] In fact, the efficiency of the Nash equilibrium in this case is close to the socially optimal (but the bandwidth allocation can be unfair). However, in our attempts to improve TCP's loss recovery schemes, we have increased our vulnerability to aggressive TCP behavior. TCP-SACK loss recovery, which is being increasingly employed by the end-hosts of today [12], allows flows to more gracefully recover from losses. This greatly reduces the penalties for aggressive congestion control and makes the Nash equilibrium of the TCP Game quite inefficient. In addition, RED active queue management is seen as important improvement over drop-tail. However, by removing drop-tail's penchant for dropping bursts of packets, RED is more friendly to aggressive flows. This results in inefficient equilibria of the TCP Game (regardless of what form of loss recovery the end-points use).

Since we no longer remain in the world consisting mainly of drop-tail routers and end-hosts employing TCP-Reno loss recovery, we must confront the problem of the aggressive behavior of greedy TCP flows. One approach is to use different queueing and buffer management schemes to prevent greedy users from achieving more than their share of bandwidth. Approaches such as Fair Queueing [7] are quite effective in this regard, but require complicated per-flow management. Several recent efforts have resulted in more scalable and implementable schemes, such as FRED [17], RED-PD [18] and AFD [20], that preferentially drop packets from aggressive flows. However, these works focus on fair allocation of bandwidth for an arbitrary set of sources. In the next section we explore the issue of how much preferential dropping is required to ensure that the Nash equilibrium of the TCP Game is desirable.

5. MECHANISMS FOR NASH EQUILIBRIUM

In order to design an preferential dropping mechanism that encourages an efficient Nash equilibrium, we need a scheme that assigns a greater loss rate to the more aggressive flows. This greater loss rate, combined with the particular loss-recovery algorithm, must offset any gain associated with the increased transmission rates. We know that heavyweight mechanisms such as Fair Queueing can accomplish this goal, but here we are looking for very simple and easily deployable preferential dropping mechanisms that give just enough incentive to produce a desirable Nash equilibria of the TCP Game, but need not achieve perfectly fair bandwidth allocations for arbitrary sets of flows (which Fair Queueing and other such mechanisms have as their goal). We discuss one such scheme in the next section.

Notice that the technique employed in the previous section, of using experimentally obtained values of F to evaluate RED is applicable to any combination of the AQM scheme and the penalty function implemented by the TCP flows as long as we can obtain a closed form expression for the goodput of the flow as a function of the penalty implemented and the AQM's loss assignment policy.[8] We will reuse this method to evaluate a modification to the CHOKe AQM scheme in the next section.

5.1 CHOKe+: A Simple Stateless Mechanism

CHOKe [21] is an example of a simple preferential dropping policy. In this section, we explore whether CHOKe or CHOKe with modifications could meet our requirements. CHOKe maintains a simple FIFO buffer. The average occupancy of the buffer is calculated in manner similar to RED. Like RED, a CHOKe buffer is also configured with two thresholds Min_{th} and Max_{th}. If the average queue occupancy exceeds Min_{th}, with each arriving packet P, CHOKe picks k candidate packets, $P_1, \ldots, P_k$, at random from the buffer. For each $i = 1, \ldots, k$, CHOKe then checks to see if P_i belongs to the same flow as P, and drops both upon a match. Upon a mismatch with some packet P_j, CHOKe leaves P_j untouched and drops P with a probability similar to that calculated by a RED buffer with equivalent average and exact queue sizes. However, in this form, CHOKe creates a minimum loss rate of $1/N$, where N is the number of active flows, as soon as the average queue length exceeds Min_{th}. When there are relatively small number of flows (< 50), this starting loss rate is excessively high and results in the severe under-utilization of the available capacity. In this form, CHOKe does ensure that the parameter settings at Nash equilibrium are very conservative. However, the average flow loss rate is very high and goodput is very low, making this Nash equilibrium undesirable.

A few minor changes in the above algorithm are enough to ensure that the loss rates of a CHOKe queue are not too high. For each incoming packet P, let m denote the number of packets from the k chosen candidate packets that belong to the same flow as the incoming packet. Let $0 \leq \gamma_2 < \gamma_1 \leq 1$ be positive constants. If $m \geq \gamma_1 k$, we drop P along with the m matching candidate packets. Otherwise, we first calculate the drop probability for P in an equivalent RED queue. Suppose that P is to be dropped according to RED. Now, if $\gamma_2 k \leq m < \gamma_1 k$, we also drop the m matching packets along with P. Otherwise, we just drop P. Henceforth, we will use CHOKe+ to refer to this modified CHOKe algorithm.

Figures 10 compares the loss assignment of RED and CHOKe+. For CHOKe+, we fixed the constants as follows: $\gamma_1 = 0.95$, $\gamma_2 = 0.80$ and $k = 5$. In this simulation, there are 10 flows $F_1, \ldots, F_{10}$. Flows $F_1, \ldots, F_9$ each have congestion control parameters of $(1, 0.5)$. Figure (a) plots the loss rates of flows $F_1, \ldots, F_9$ and that of flow F_{10} as the increase parameter of F_{10} is varied in the range $[1, 50]$ keeping the decrease parameter fixed at 0.5. Similarly, in Figure (b) the decrease parameter of F_{10} is varied in the range $[0.5, 1)$ keeping the increase parameter fixed at 1. Notice that in either case the loss rates of all the 10 flows increase gradually under RED. However, under CHOKe+, the loss rate of flow F_{10} increases at a rate much higher than that compared to the increase with RED. The loss rates of flows $F_1, \ldots, F_9$ are unaffected by the behavior of F_{10}.

A RED buffer shares the additional losses resulting from the increased rate of a single flow among the entire population of flows. As a result, the loss penalty that an aggressive flow receives is minimal. CHOKe+, on the other hand, assigns most losses due to a flow's rate increase to the flow itself and keeps the rest of the flows mostly isolated from the aggressive flow. Note that perfect isolation, as in Fair Queueing and other similar schemes, is not necessary. CHOKe+ must only ensure that the increase in loss rate is sufficient to discourage aggressive behavior.

In a manner similar to the analysis of RED (Sections 4.3) we

[7]Though other forms of antisocial behavior, such as using no congestion control at all, could lead to severe congestion collapse.

[8]In fact, the transmission protocol employed by the flows need not be reliable.

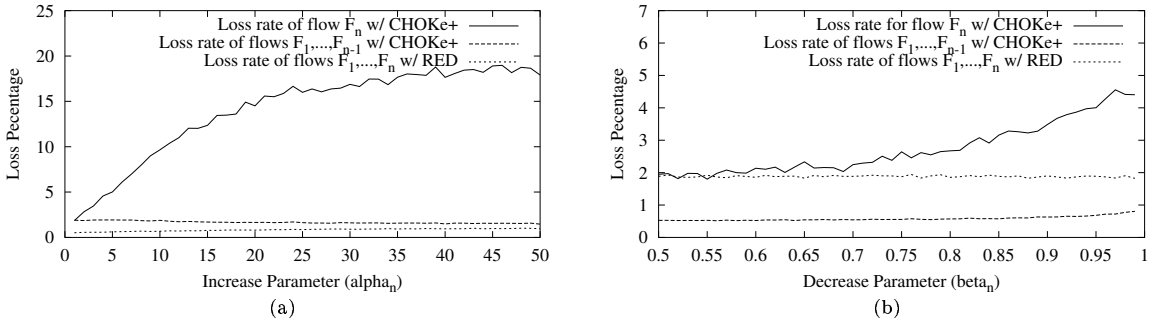
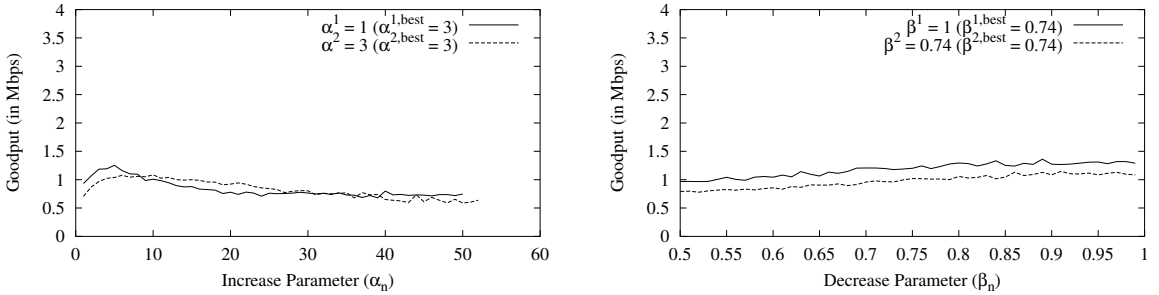


Figure 10: Figure comparing the loss assignment of RED and CHOKe+. The loss rates for the greedy flow and the competing non-greedy flows are plotted in each case. For RED buffers, the two curves coincide.



α_E	Goodput	Loss Rate
3	0.97Mbps	2.74

(a)

β_E	Goodput	Loss Rate
0.74	1.00Mbps	2.42

(b)

Figure 11: Simulation results for Gentle flows with CHOKe+ buffers

plot the variations in $p_a = p_{\alpha,\alpha}$ and $P_a = \frac{dp'}{d\alpha'}|_{\alpha'=\alpha}$ for Gentle flows when CHOKE+ buffers are employed in Figures 5(a) and (b) along with the corresponding values for RED buffers. When flows get bursty, CHOKe+ drops packets much more aggressively than RED buffers as can be seen from Figure 5(a) where the common loss rate assigned by CHOKe+ is much higher than that assigned by RED for the same setting of the common increase parameter of the n flows. In addition, CHOKe+ heavily penalizes flows for being even slightly more aggressive than the competing flows as is clear from Figure (b) where P_a is an order of magnitude higher for CHOKe+ buffers compared to the corresponding values for RED buffers. Due to these properties CHOKe+ significantly diminishes the goodput advantage of aggressive flows. Similarly, we plot the values of p_b and P_b as functions of β in Figures 5(c) and (d). Again, CHOKe+ does a much better job of penalizing aggressive flows then RED.

We plot the values of F_a and F_b for Gentle flows as functions of α and β respectively in Figures 6(a) and (b). Notice that from these plots F_a and F_b are positive throughout implying that at the respective Nash equilibria α_E grows arbitrarily large and $\beta_E \to 1$. However, since CHOKe+ assigns a much higher loss rate to flows than RED for the same setting of the increase or the decrease parameters of the n flows (Figures 5(a) and (c)) and since CHOKe+ induces many more bursty packet drops than RED (this is intuitively clear from the CHOKe+ algorithm), TCP SACK flows tend to time-out occasionally when CHOKe+ buffers are used. In fact aggressive TCP SACK flows will time-out quite *frequently* under CHOKe+. The goodput equation for Gentle flows (Equation 24) does not capture this time out behavior. Hence, Figures 6(a) and (b) do not correctly represent the behavior of CHOKe+. Indeed CHOKe+ results in a much less aggressive parameter setting at Nash equilibrium than RED, as our simulations for the TCP Game (described below) show.

Our results for the Nash equilibrium of the TCP Game with CHOKe+ buffers and TCP-SACK flows are shown in Figure 11(a) and (b). At Nash equilibrium, $\alpha_E = 3$ when flows vary their increase parameters and at $\beta_E = 0.74$ when flows vary their decrease parameter. When both α and β are allowed to vary, $(\alpha, \beta) = (3, 0.90)$. At this Nash equilibrium, the average per-flow goodput is 0.98Mbps and the per-flow loss rate is 4%. All the Nash equilibria have good per-flow goodput and per-flow loss rate. CHOKe+ is so effective that aggressive flows see only a very marginal improvement in goodput.

However, it is worth noticing that CHOKe+ cannot completely nullify the advantage seen by aggressive flows when decrease parameters are allowed to be varied. In fact, as we show in the next section, it is impossible to effectively punish greedy flows varying their decrease parameters without employing queue management schemes that maintain explicit per-flow state. In light of this result, the best we can hope for from a purely stateless mechanism is to diminish the advantage to such a low value that it is almost imperceptible to the aggressive flow. CHOKe+ is effective in doing so. As can be seen from Figure 11(b), when CHOKe+ is used there is hardly any perceptible advantage of setting β aggressively. This also explains why β_E at Nash equilibrium is significantly less than 1 (assuming that greedy flows do not choose a more aggressive parameter setting unless it yields a substantially higher goodput with high confidence). This property of CHOKe+ also has implications on fairness. Since CHOKe+ discourages an aggressive setting of β at Nash equilibrium, the window decrease is still multiplicative. This results in a reasonable fair allocation [6].

5.2 On Multiplicative Decrease

Suppose that both the increase parameter α and the decrease parameter β of all the flows are allowed to vary. We argue below that there can be no mechanism that does not maintain per flow state (e.g., fair queuing), that ensures a moderate value of $\beta < 1$ as a Nash equilibrium. This is important since a moderate value of β is needed to achieve fair resource allocation among end-points.

Consider a single link shared by n flows employing AIMD with parameters α_i and β_i of their strategic choice, and a symmetric Nash equilibrium (α, β). It is quite easy to see that, without the losses due to the queue management mechanism (buffer overflows,

early drops, etc.) on this link, and assuming that all α_i's are the same at equilibrium, each of the n users would end up with goodput $g_i^0 = C\frac{1+\beta_i}{2n}$.

Assume now in the simplified model of this subsection that the queue management mechanism on this link works by penalizing the flows, that is, subtracting an amount from their goodput. Suppose that the penalty imposed on flow i depends on the measurement of g_i^0 above by the queue management, and is $P(g_i^0)$ in expectation; thus we write the final goodput as

$$g_i = g_i^0 - P(g_i^0). \tag{27}$$

Since we are assuming that we are at a Nash equilibrium, $\beta_i = \beta < 1$ for all i, and no flow has an incentive to increase its β to $\beta + \epsilon$ for some small $\epsilon > 0$. By Equation 27 this means that the queue management mechanism is capable of detecting relative fluctuations smaller than $\frac{\epsilon}{2}$ in g_i^0, that is, arbitrarily small fluctuations, and imposing penalties whenever such fluctuations occur. It can be shown by a reduction from element disjointness [4, 5] that any randomized algorithm that detects such fluctuations with high probability must use space proportional to n – that is to say, must essentially maintain per-flow state.

Our simplified model of queue management by penalties depending on goodput rules out other schemes, such as the one in which flows are sampled for detailed analysis, and if a flow is found to have $\beta_i > \beta$ it is subject to penalties outside the realm of queue management (such as exclusion from the network or legal punishment). However, such schemes are even more far-fetched, and run contrary to the end-to-end philosophy of the Internet.

6. SUMMARY

In this paper, we explore the impact of greedy TCP end-points on the efficiency of the network. Our finding can be briefly summarized as follows:

- In certain situations, greedy end-point behavior can result in efficient network operation. In particular, the Nash equilibria are reasonably efficient in the historically significant setting of TCP-Reno loss recovery in a network of drop-tail routers.
- Unfortunately, in settings where either TCP-SACK loss recovery has been adopted by end-points or RED has been deployed in routers, the Nash equilibria of the TCP game are undesirable, having either low network goodput or high drop rates, or both.
- However, the addition of very simple preferential dropping algorithms, such as CHOKe+, can help restore the efficiency of the Nash equilibria.

The conclusions about the Nash equilibria of the TCP Game under various scenarios imply that, while in the past, network operators could rely on the behavior of end-users to ensure the stable, efficient operation of the network, the same cannot be said of today's Internet. They also suggest that there are two possible reasons for the continued stable operation of the present-day Internet: (1) it is too difficult to modify end-hosts to behave greedily or (2) end-users consciously choose to behave in a socially optimal manner. However, all is not yet lost. It is possible to design simple, stateless queue management algorithms to ensure that the advantage gained by aggressive flows is more than offset by a high packet loss rate. These mechanisms can also ensure reasonable, but not perfect, fairness at Nash equilibrium. This would help make the Nash equilibrium more desirable while still allowing the implementation of modern loss recovery mechanisms (such as TCP-SACK) and queue management techniques (such as RED) which have otherwise played a stellar role in keeping the Internet of today in good stead.

7. REFERENCES

[1] The network simulator - ns-2. http://isi.edu/nsnam/ns/.

[2] A. Akella, S. Seshan, S. Shenker, and I. Stoica. Exploring congestion control. Technical Report CMU-CS-02-139, CMU, Pittsburgh, Pennsylvania, May 2002.

[3] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Internet Draft, Internet Engineering Task Force, Feb. 1999. Work in progress.

[4] Z. Bar-Yossef. *Private Communication*, May 2000.

[5] Z. Bar-Yossef. The complexity of massive data set computations. *PhD Dissertation, U.C. Berkeley*, May 2000.

[6] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, June 1989.

[7] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 1–12, Austin, Texas, Sept. 1989. ACM. also in Computer Communications Review, 19 (4), Sept. 1989.

[8] C. Douligeris and R. mazumdar. On pareto optimal flow control in a multicalss environment. In *The 25th Allerton Conference of Communication, Control and Computing*, University of Illinois at Urbana-Champaign, Oct. 1987.

[9] C. Douligeris and R. Mazumdar. A game-theoretic approach to flow conrtol in an integrated environment. *Journal of the Franklin Institute*, 329(3):383–402, Mar. 1992.

[10] K. Fall and S. Floyd. Simulation-based comparisons of tahoe, reno, and SACK TCP. *ACM Computer Communication Review*, 26(3):5–21, July 1996.

[11] D. Ferguson, C. Nikolaou, and Y. Yemini. An economy for flow control in computer networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, 1989.

[12] S. Floyd. Questions about sack deployment. http://www.icir.org/floyd/sack-questions.html.

[13] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993.

[14] M.-T. Hsiao and A. Lazar. A game-theoretic approach to decentralized flow control of markovian queue networks. In *Performance*, Holland, 1987.

[15] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, Aug. 1988. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.

[16] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Combinatorial optimization in congestion control. In *Proceedings of the 41th Annual Symposium on Foundations of Computer Science*, pages 66–74, Redondo Beach, CA, 12–14 Nov. 2000.

[17] D. Lin and R. Morris. Dynamics of random early dectection. *ACM Computer Communication Review*, 27(4):127–136, Oct. 1997. ACM SIGCOMM'97, Sept. 1997.

[18] R. Mahajan and S. Floyd. Controlling high-bandwidth flows at the congested router. In *Proc. of ICNP'01*, Riverside, California, USA, Nov. 2001.

[19] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, Apr. 2000.

[20] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Approximate fairness through differential dropping (one page summary). *ACM Computer Communication Review*, Jan. 2002.

[21] R. Pan, B. Prabhakar, and K. Psounis. CHOKE, a stateless active queue management scheme for approximating fair bandwidth allocation. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Mar. 2000.

[22] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEE/ACM Transactions on Networking*, 3, 1995.

[23] Y. R. Yang and S. S. Lam. General AIMD congestion control. Technical Report UTCS TR-2000-09, University of Texas, Austin, Texas, May 2000.