Window-Games between TCP flows[☆]

Pavlos S. Efraimidis[a,*], Lazaros Tsavlidis[a], George B. Mertzios[b]

[a]*Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece*
[b]*Department of Computer Science, RWTH Aachen University, Germany*

Abstract

We consider network congestion problems between TCP flows and define a new game, the *Window-game*, which models the problems of network congestion caused by the competing flows. Analytical and experimental results show the relevance of the Window-game to real TCP congestion games and provide interesting insight into the respective Nash equilibria. Furthermore, we propose a new algorithmic queue mechanism, called Prince, which at congestion makes a scapegoat of the most greedy flow. We provide evidence which shows that Prince achieves efficient Nash equilibria while requiring only limited computational resources.

Key words: Algorithmic Game Theory, Network Games, Nash Equilibrium

1. Introduction

Algorithmic problems of networks can be studied from a game-theoretic point of view. In this context, the flows are considered independent players who seek to optimize personal utility functions, such as the goodput. The mechanism of the game is determined by the network infrastructure and the policies implemented at regulating network nodes, like routers and switches. The above game theoretic approach has been used for example in [31] and in several recent works like [22, 21, 2, 29, 14, 11].

In this work, we consider congestion problems of competing TCP flows, a problem that has been addressed in [19, 2]. We propose a new game, the *Window-game*, which models the problem of network congestion caused by the competing flows. The novelty of the new model lies in the fact that we focus on the congestion window, a parameter that is in the core of the network algorithms of modern TCP flows. The size of the congestion window, to a large degree, controls the speed of transmission of a TCP flow [18].

[☆]A preliminary version of this work has been presented at the Symposium of Algorithmic Game Theory, SAGT 2008 [8].

[*]Corresponding author.

Email addresses: pefraimi@ee.duth.gr (Pavlos S. Efraimidis), ltsavlid@ee.duth.gr (Lazaros Tsavlidis), mertzios@cs.rwth-aachen.de (George B. Mertzios)

More precisely, we define the following game, which we call the *Window-game*, as an abstraction of the TCP congestion problem. The game has a router and n flows and is played synchronously, in one or more rounds. Every flow is a player that in each round selects the size of its congestion window. The router (the mechanism of the game) receives the actions of all the flows and decides how the capacity is allocated. Based on how much of its requested window has been satisfied, each flow decides on the size of its congestion window for the next round. The utility of each flow is the capacity that it obtains from the router in each round minus the cost for lost packets.

The main contributions of this work are three. The first is the definition of the Window-game, a game-theoretic model for network games between TCP flows. The motivation to address the TCP congestion problem originated from the following question, posed in [19, 29]: *Of which game or optimization problem is TCP/IP congestion control the Nash equilibrium or optimal solution?* The Window-game is meant to be a natural model that is simple enough to be studied from an algorithmic and game-theoretic point of view, while at the same time it captures essential aspects of the real TCP game. By focusing on the congestion window, the Window-game is simpler and more abstract than the model used in [2], while still being sufficiently realistic to model actual TCP games. The Window-game allows us to address both repeated versions of the game that resemble actual TCP games (like the games studied in [2]) and one-shot versions of the game with general (not necessarily AIMD) flows.

The second contribution is the analysis of interesting queue mechanisms from a game-theoretic point of view. In particular, the case of DropTail with non-synchronized packet losses, which we consider the most realistic and interesting version of TCP games, has not been addressed before (to the best of our knowledge). Also new is the analysis of the one-shot versions of the TCP-router games.

The final contribution is a new queue policy, called Prince, which is simple and efficient enough to cope with the requirements of real network routers. We show that the exemplary punishment of the most greedy flow induces strong incentives for the flows to play in a socially efficient manner. In general, it seems that focusing on the most greedy player proves to be a simple and very effective way to make selfish players that share a common resource behave responsibly. This claim is further supported by the fact that punishing the highest rate flow has also been successful in a different network game model [13] where the flows are Poisson sources. Prince is much simpler than the policy of [13], and it is the first time the "punish the leader" approach is applied to games with TCP flows.

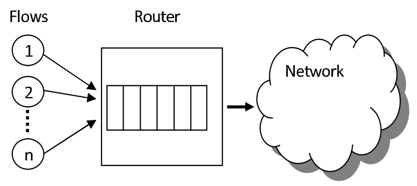
We provide theoretical evidence and experimental results to support the above claims.

Outline. The rest of the paper is organized as follows: The Window-game model is described in Section 2. An overview of TCP congestion control concepts is given in Section 3. The Prince Mechanism along with common router policies are discussed in Section 4. We consider Window-games where the players are AIMD flows in Section 5 and present the corresponding experimental results in

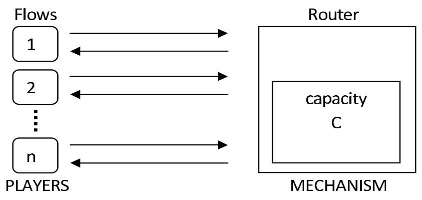
Section 6. Window-games with general, non-AIMD, flows are discussed in Section 7 (Window-games with complete information) and in Section 8 (Window-games with incomplete information). Finally, a discussion of the results is given in Section 9.

2. The Window-Game

The Window-game models the interaction between the congestion windows of competing TCP flows. The main entities of a Window-game are a router with capacity C and a set of $n \leq C$ flows, as depicted in Fig. 1b. The router uses a queue policy to serve in each round up to C workload. The n flows are the independent players of the game. Unless otherwise specified, the number n is considered unknown to the players and to the router. The game consists of one or more rounds. In each round, every player i selects a size $w_i \leq C$ for its congestion window and submits it to the router. The router collects all requests and applies the queue policy to allocate the capacity to the flows. The common resource is the router's capacity, an abstract concept that corresponds to how much load the router can handle in each round. The objective of each player is



(a) The network model of the TCP game.



(b) One round of the Window-game.

Figure 1: The Window-game along with the corresponding TCP network model.

to send as many packets as possible. At the same time the player has to avoid packet losses, as each packet loss induces a cost $g \geq 0$ to the player. For games with multiple rounds, each packet loss (and each successful packet) causes the respective flow to adjust the size of its congestion window. The adjustment is determined by the penalty model and the game strategy of the flow.

Definition 1. One round of the Window-game.
Mechanism: A router with capacity $C > 0$ and a queue policy
Players: n flows $i = 1, \ldots, n$, where $2 \leq n \leq C$
Actions of player i : The window size w_i of i, where $w_i \in \{1, \ldots, C\}$
Utility of player i : $u(i)$ = transmitted(i) - $g\cdot$ dropped(i), where
g: the cost for each dropped packet
transmitted(i): number of transmitted packets
dropped(i): number of dropped packets

Each round of the game is executed independently; no work is pending at the start of a round and no work is inherited to a following round.

Network routers and flows process massive streams of network packets in real time. Therefore, algorithms used by routers and flows may use only limited computational resources like memory and processing power. In particular, it is considered unrealistic for network routers to keep detailed statistics on a per flow basis. The queue policy of the router should be stateless or use as little state information as possible[1].

The Window-game can model not only games between AIMD flows but also games between general, not necessarily AIMD, flows. The only requirement is that that the packet stream generated by each flow can be represented with a sequence of window sizes. In this work, we consider the following cases:

Window-Games with AIMD flows. In these games the players are AIMD flows (we provide a short description of AIMD and related concepts in Section 3). Each AIMD flow i selects its parameters (α_i, β_i) once, for the whole duration of the game. The utility for each flow is its average goodput (number of useful packets that are successfully delivered in each round) at steady state. This class of Window-games is strongly related to the game(s) currently played by real TCP flows and exists implicitly in the analysis of [19, 2].

Window-games with general flows. These are games where the flows can use arbitrary algorithms to choose their congestion window. Depending on the number of rounds and on how well each flow is informed about the existence of other flows, we distinguish the following classes:

- One-shot game with complete information.
- One-shot game with incomplete information with no prior probabilities.
- One-shot game with incomplete information with prior probabilities.
- Repeated game with incomplete information.

The action of a general flow is to choose the size of its congestion window for each round. In one-shot games, the utility of the flow is the goodput minus the cost for lost packets. In repeated games, the overall utility is the average utility of all the game rounds.

Assumptions. We make a set of simplifying assumptions similar to the assumptions of [2]. The Window-game is symmetric. All flows use the same

[1]Stateful network architectures are designed to achieve fair bandwidth allocation and high utilization, but need to maintain state, manage buffers, and perform packet scheduling on a per flow basis. Hence, the algorithms used to support these mechanisms are less scalable and robust than those used in stateless routers. The stateless substance of nowadays IP networks allows Internet to scale with both the size of the network and heterogeneous applications and technologies [32, 33].

network algorithms for loss recovery. All packets of all flows are of the same size and have the same Round Trip Time. Packet losses are caused only by network congestion.

Solution concept: The solution concept for the Window-games is the Nash equilibrium (NE) and in particular the Symmetric Nash equilibrium (SNE). A good reason to start with SNE is that the analysis appears to be simpler compared to general NE. It is noteworthy, that the both the analytical and the experimental results show that in many cases there are only symmetric (or almost symmetric) NE. For each Window-game, we study its SNE and discuss how efficient the network operates at them. In certain cases, we search experimentally for general, not necessarily symmetric, NE.

Efficiency and fairness of the NE. In the context of Window-games we consider a NE efficient if at equilibrium the router capacity is utilized and the packet losses of the flows are close to zero. We consider a NE fair if all flows receive, at least approximately, the same bandwidth.

How realistic is the Window-game? An important difference of Window-games to real TCP/IP games is that Window-games are played in rounds while real TCP/IP network games are clocked at the packet level. One round of the Window-game corresponds to a time-interval roughly equal to the flow's Round-Trip-Time (RTT), i.e., the time from the submission of a packet P until the corresponding ACK is received. In this time interval, the flow will receive ACK's for all packets that were pending before P. The number of pending packets is determined by the window size at the time when packet P where submitted. Therefore, the round-based approach appears to be a natural approximation of the real TCP/IP game. A round-based approach has also been used in the analysis of [2]. A further difference is that the Window-game model does not take into account queueing delay. Because of this, the values of the goodput that are calculated may not always correspond to real goodput values. This, however, is not an obstacle for the applications of the Window-game in this work, since we are mainly interested in the relative goodputs for different strategies of a TCP flow. Finally, we use the term capacity for the router because we want to emphasize that this is the amount of work that the router can handle in one round. The capacity is determined by the bandwidth load that the router can handle.

In our view, the Window-game fulfils the requirements that we had set: To capture the interaction of the competing flows, to indicate when a flow can gain more goodput by changing its strategy and to characterize the efficiency and the fairness of the NE. Additionally, the analytical and experimental evidence show that the Window-game is strongly related to the actual TCP/IP congestion games.

3. Congestion Control in TCP

In this Section, we provide a short description of the basic concepts of TCP and TCP congestion control.

TCP: The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite. TCP is a window-based transport protocol; in TCP every flow has an adjustable window, which is called the *congestion window*, and uses it to control its transmission rate [18].

Congestion Window: A TCP flow submits packets to the network and expects to receive acknowledgments for the packets that reach their destination. The congestion window defines the maximum number of outstanding packets for which the flow has not yet received acknowledgements [20]. Essentially, the congestion window represents the sender's estimate of the amount of traffic that the network can absorb without becoming congested. The most common algorithm to increase or decrease the congestion window of a TCP flow is AIMD.

AIMD: The Additive Increase Multiplicative Decrease (AIMD) algorithm [7] can be considered as a probing procedure designed to find the maximal rate at which flows can send packets under current conditions without incurring packet drops [19]. An AIMD flow i has two parameters α_i and β_i; upon success, its congestion window is increased additively by α_i (in each round), and upon failure, its congestion window is decreased multiplicatively by a factor β_i. The basic AIMD algorithm:

$$w_i = \begin{cases} w_i + \alpha, & \text{upon successful delivery of all packets of a round,} \\ w_i \cdot \beta, & \text{in case of packet drop.} \end{cases}$$

AIMD is so far considered to be the optimal choice in the traditional setting of TCP Reno congestion control and FIFO DropTail routers. If, however, we consider the developments like TCP SACK and active queue management, AIMD may no longer be superior [3].

Packet loss: When congestion occurs at a router, packets are dropped. An AIMD flow responds to packet loss with a loss-recovery procedure which incurs some cost to the flow and with a decrease of its congestion window. Popular versions of TCP, like TCP Tahoe, TCP-Reno and TCP-SACK, differ in the way they handle a packet loss.

- TCP Tahoe: A packet loss is indicated when the timeout occurs before the corresponding ACK is received. The congestion window will then be reduced to 1 packet and the slow start phase will be initiated. If the sender receives three duplicate ACK (DACK[2]) for the same packet before a timeout occurs, the sender first retransmits the packet (fast retransmit) and then reduces its congestion window to 1. This reaction of TCP Tahoe corresponds to the severe reaction to packet loss; even a single packet loss causes the flow to drastically reduce its congestion window.

[2]When a duplicate ACK is received by a real TCP flow, the flow does not know if it is because a TCP segment was lost or simply that a segment was delayed and received out of order at the receiver. If the receiver can re-order segments, it should not be long before the receiver sends the latest expected acknowledgement. Typically no more than one or two duplicate ACKs should be received when simple out of order conditions exist. If however more than two duplicate ACKs are received by the sender, it is a strong indication that at least one segment has been lost.

- TCP Reno: Has the same mechanism with Tahoe, except that in the case of 3 DACK it retransmits the packet that the receiver waits for and then halves its congestion window (fast recovery). In the case of multiple dropped packets from the same congestion window, a Reno flow tries multiple fast retransmissions until a timeout occurs and then comes in to slow start. The reaction of TCP Reno to packet loss is considered a hybrid reaction.

- TCP SACK: This variant of TCP is an extension of Reno. SACK has a selective acknowledgement option that allows receivers to additionally report non-sequential data they have received. When coupled with a selective retransmission policy in TCP senders an efficient recovery mechanism, which avoids unnecessary retransmissions, is formed. The reaction of TCP SACK to packet loss is considered a gentle reaction.

Since all loss-recovery schemes incur costs to the flow they can be thought of as a penalty on the flows which suspend their normal transmission rate until lost packets are retransmitted.

Penalty Model: We formed a penalty-based model, which is similar to the model of [19, 2], to define a flow's behavior when losses occur. Assume that a flow i with current window size w_i has lost $L_i \geq 1$ packets in the last round. Then:

- Gentle penalty (resembles TCP SACK): The flow reduces its window to $w_i = \beta \cdot w_i$ in the next round.

- Severe penalty (TCP Tahoe): The flow timeouts for τ_s rounds and then continues with a window $w_i = \beta \cdot w_i$.

- Hybrid penalty (TCP Reno): If $L_i = 1$ the flow applies gentle penalty. If $L_i > 1$ a progressive severe penalty is applied. After a timeout of $\min\{2 + L_i, 15\}$ rounds, the flow restarts with a window equal to w_i/L_i. This penalty is justified by the experimental results of [9, 27].

4. Router Policies and the Prince Mechanism

The router, and in particular the queue algorithm deployed by the router to allocate the capacity to the flows, defines the mechanism of the Window-game. We examine the common policies DropTail, RED, MaxMin, CHOKe and CHOKe+. The implementations of the above policies in the Window-game aim to simulate the behavior of the real policies on packet streams. We also examine two variants of the proposed Prince policy. Our interest for all policies is on the NE of the corresponding Window-games.

- **DropTail:** The DropTail policy is the simplest and most common queue discipline of network routers. With DropTail the packets are served in FIFO order. If the queue is full, incoming packets are dropped. The

Window-game adaptation of DropTail, simulates how DropTail would behave in the real, stream-based case. In each round, if the sum w_{total} of the requested windows does not exceed the capacity C, the router serves all packets. Otherwise, the router drops a random sample of packets of size equal to the overflow, and serves the remaining C packets.

- **RED [10]**: The Random Early Detection (RED) policy is an active queue management algorithm that may drop packets even before an overflow occurs. More precisely, at overflow RED behaves like DropTail. However, for queue occupancies between a low threshold min_{th} and a high threshold max_{th} of the queue size, packets are dropped with a positive probability p. The Window-game adaptation of RED works as follows: If the sum w_{total} of the requested windows is $w_{total} < min_{th}$ all packets are served. If $w_{total} \geq min_{th}$ a random sample of packets is selected. The selected packets correspond to positions higher then min_{th} in a supposed queue. Each chosen packet is dropped with a probability proportional to the router's load. The values $min_{th} = 70\%$ and $max_{th} = 100\%$ are used for the thresholds.

- **MaxMin:** An allocation of a common resource is MaxMin fair if no share may be increased without simultaneously decreasing another share which is already smaller. Router policies that implement the MaxMin fairness criterion achieve fair allocation of the available capacity to the flows. Given a network with its link capacities and a set of flows with their maximum possible transmission rates, there is a unique set of flow rates that satisfies the max-min conditions. MaxMin Fairness is a stateful, fair queue policy [5] that conceptually corresponds to applying round-robin for allocating the capacity. The router offers its packet transmission slots to its users by polling them in round-robin order. If a flow is offered a chance to use a link slot but has no packets ready, then that same slot is offered to the next flow, until a ready flow is found. In each pass of a link's round robin, a flow may transmit only one packet [16]. The adaptation of MaxMin to the Window-game is straightforward.

- **CHOKe:** CHOKe [28] is a stateless active queue management algorithm that differentially penalizes misbehaving flows by dropping more of their packets. For every packet that arrives at the congested router, CHOKe draws a packet at random from the FIFO buffer and compares it with the arriving packet. If they both belong to the same flow, then they are both dropped, else the randomly chosen packet(s) is admitted into the buffer with a probability that depends on the level of congestion. Packets belonging to a misbehaving flow are both more likely to trigger comparisons and more likely to be chosen for comparison. Therefore, packets of misbehaving flows are dropped more often than packets of well-behaved flows [28]. The Window-game adaptation of CHOKe is similar to the adaption for RED. Every chosen packet that is above the min threshold, is compared to a packet chosen randomly from the packets that have been admitted until that moment. For the lower threshold, the upper threshold and the dropping probability we use the same values as for RED.

- **CHOKe+:** The CHOKE+ policy [2] is a variant of CHOKe that tries to avoid high loss rates and severe under-utilization of the available capacity, when the number of flows is relatively small. For each incoming packet P, the algorithm randomly selects k packets from the queue. Let m denote the number of packets from the k chosen candidate packets that belong to the same flow as the incoming packet. Let $0 \leq \gamma_2 \leq \gamma_1 \leq 1$ be positive constants. If $m \geq \gamma_1 k$, the algorithm drops packet P along with the m matching candidate packets. Otherwise, a drop probability p is calculated in an equivalent RED queue. Suppose that P is to be dropped according to RED. Now, if $\gamma_2 k \leq m < \gamma_1 k$, the algorithm also drops the m matching packets along with P. Otherwise, only packet P is dropped. The Window-game adaption is in the same spirit as the adaptations of CHOKe and RED.

Queue disciplines in use. DropTail is simple and efficient, seems to work reliably in practice and is currently the most widely used queuing discipline at Internet routers. RED is also widely deployed in order to maintain the buffer's utilization within desired levels. Both DropTail and RED, do not actively punish greedy flows.

4.1. The Prince Mechanism

We propose a new queue policy, called Prince (of Machiavelli), which, at congestion, drops packets from the flow with the largest congestion window. If the overflow is larger than the window of the most greedy flow, the extra packets are dropped from the remaining flows with DropTail. If the overflow does not exceed the number of packets of the most greedy flow, then flows that do not exceed their fair share do not experience packet loss. Hence a greedy flow does not plunder the goodput of other flows. The Prince policy is simple and does not require state information on a per-flow basis. This makes the policy appropriate for the requirements of real routers. We define a basic version of Prince that drops packets only in case of overflow and a RED-inspired version of Prince, called Prince-R, which applies its policy progressively starting from a min threshold $min_{th} = 70\%$.

5. Window-games with AIMD flows

In this Section, we analyze three characteristic Window-game scenarios between AIMD flows with gentle penalty, where the flows can choose the value of parameter α. Parameter β is fixed[3] at its default TCP value $\beta = 1/2$. We

[3]Experiments with parameter β show that in almost all cases selfish flows will use for β a value close to 1. The same conclusion can be extracted from the results in [2]. A simple

admit parameter α to take integer values in the range $[1, C/2]$. In our analysis we use machinery from [2]. Let us start with the following definitions:

Definition 2. A congestion round of the Window-game *is a round in which the sum of the requested windows is larger than the capacity of the router. In a congestion round, the router has to drop one or more packets.*

Definition 3. A loss round of flow i *is a congestion round in which flow i experiences packet loss.*

Assume a set of n AIMD flows where each flow i has parameters (α_i, β_i). At steady state, let N_i denote the window size after a loss round and let τ_i be the number of rounds between two loss rounds of flow i. Let τ be the number of rounds between two successive congestion rounds of the system. Then, similar to [2], the flow starts after a congestion round with window size N_i and increases its window size successively to $N_i+\alpha_i, N_i+2\alpha_i, \ldots, N_i+\alpha_i\tau_i$. When the window size reaches $N_i + \alpha_i\tau_i$, the flow experiences packet loss and reduces in the next round its window size to $N_i = \beta_i(N_i + \alpha_i\tau_i)$. Hence:

$$N_i = \beta_i(N_i + \alpha_i\tau_i) = \frac{1}{2}(N_i + \alpha_i\tau_i) \ , \tag{1}$$

and this gives

$$N_i = \alpha_i\tau_i \ . \tag{2}$$

Assume $\tau_i + 1$ consecutive rounds, starting immediately after a loss round of flow i. Based on the definition of τ_i, flow i will not experience packet loss in rounds $1, \ldots, \tau_i$. However, round $\tau_i + 1$ will be a loss round for flow i. Thus, the goodput G_i of flow i is:

$$G_i \ = \ \frac{1}{\tau_i + 1}\Big(N_i + (N_i + \alpha_i) + (N_i + 2\alpha_i) + \ldots + (N_i + \alpha_i\tau_i - L_i)\Big) \ ,$$

where L_i is the average number of packets that flow i looses at loss rounds. This gives:

$$G_i \ = \ \frac{3}{2}\alpha_i\tau_i - \frac{1}{\tau_i + 1}L_i \ . \tag{3}$$

In most scenarios we can ignore the last term to get the simplified equation:

$$G_i \ = \ \frac{3}{2}\alpha_i\tau_i \ . \tag{4}$$

In the above equations for of AIMD flows we have assumed that the system has reached a steady state. Based on this assumption we used equations to capture

argument for this behavior is that parameter β is very important when the flow has to quickly reduce its window size when the network conditions change. The TCP games examined in this work are rather static: Static bandwidth, static number of flows, static behavior of all flows during a game and hence there is no real reason for a flow to be adaptive.

the average behavior of the flows. This approach has been introduced in [2]. We will assume that in Window-games with AIMD flows, the flows operate in reasonable conditions, where at minimum

$$n \leq \frac{C}{2}, \quad A = \sum_{i=1}^{n} \alpha_i \leq \frac{C}{2}, \text{ and } \forall \text{ flow } i,\ \tau_i \geq 1 \ . \tag{5}$$

We make these assumptions to exclude conditions where Eqs. (1) and (3) become inaccurate. The assumptions are reasonable for the scenarios that we consider.

5.1. DropTail router with synchronized packet losses and gentle penalty AIMD flows

The Game: A DropTail router of capacity C serves n gentle penalty TCP flows. The term *synchronized packet losses* refers to the following simplifying assumption: At congestion rounds, all flows experience packet loss, that is, every congestion round is a loss round for all flows.

Using the Window-game model it is easy to show that the selfish flows will try to over-utilize the common resource (the network) by using a large value for their parameter α. The outcome will be a network that is heavily congested. A close analogy is the so-called *"Tragedy of the Commons"* [17] problem in economics where each individual can improve her own position by using more of a common resource, but the availability of the resource degrades as the number of greedy users increases.

Since every congestion round is in this game a loss round for all flows, the number τ_i of rounds between two successive loss rounds of any flow i is equal to number τ of rounds between two successive congestion rounds of the system

$$\tau = \tau_1 = \tau_2 = \ldots = \tau_n \ . \tag{6}$$

Let $A = \sum_{i=1}^{n} \alpha_i$ and $A_i = A - \alpha_i$. After each round without packet loss, the total window size of all flows $w_{total} = \sum_{i=1}^{n} w_i$ increases by A. In congestion rounds w_{total} exceeds the capacity C of the router, i.e., $w_{total} > C$. The number of excess packets in a congestion round is a random integer in the range $1, 2, \ldots, A$. We will assume that the average excess is $L = A/2$, i.e., the half of the increase step. Thus, the total window size w_{total} in congestion rounds is $w_{total} = C + A/2$ and the average packet loss for each flow i is $L_i = \alpha_i/2$. After a congestion round, all flows multiplicatively decrease their congestion window. Using Eq. (2), $\beta_i = 1/2$ and $N = \sum_{i=1}^{n} N_i$ we obtain

$$N = \frac{1}{2}(C + A/2) \ . \tag{7}$$

Using Eq. (1) for all flows and summing up gives:

$$N = \frac{1}{2}(N + A \cdot \tau) \Rightarrow \tau = \frac{C}{2A} + \frac{1}{4} \ . \tag{8}$$

From Eq. (3) we get that for any flow i the goodput G_i is:

$$G_i(\alpha_i) = \frac{3}{2}\alpha_i \left(\frac{C}{2(A_i + \alpha_i)} + \frac{1}{4} \right) - \alpha_i \left(\frac{C}{A_i + \alpha_i} + \frac{5}{2} \right)^{-1} . \tag{9}$$

We will show that:

Lemma 4. *$G_i(\alpha_i)$ in (9) is a strictly increasing function of parameter α_i, for $\alpha_i \leq C$ and $A_i \leq C$.*

Proof. The first derivative of G_i with respect to α_i is positive at $\alpha_i = C$, for $A_i \leq C$. In particular, if the derivative at $\alpha_i = C$ is written as a single fraction, its numerator is

$15A_i^2 + 57A_iC + 14A_i^3C + 15C^2 + 64A_i^2C^2 + 3A_i^4C^2 + 68A_iC^3$
$+15A_i^3C^3 + 18C^4 + 24A_i^2C^4 + 15A_iC^5 + 3C^6$,

and its denominator is

$4(A_i + C)^2 \left(3 + A_iC + C^2\right)^2$.

The second derivative of G_i with respect to α_i is always negative. More precisely, the second derivative written can be written as a fraction where the denominator is strictly positive and the numerator is the sum of the positive terms

$12\alpha_i^3C + 36\alpha_i^2A_iC + 36\alpha_iA_i^2C + 12A_i^3C + 4\alpha_i^3A_iC^2$
$+12\alpha_i^2A_i^2C^2 + 12\alpha_iA_i^3C^2 + 4A_i^4C^2$,

and the negative terms

$-81A_iC - 81\alpha_iA_iC^2 + 4\alpha_i^3A_iC^2 - 81A_i^2C^2 - 27\alpha_i^2A_iC^3 - 54\alpha_iA_i^2C^3$
$-27A_i^3C^3 - 3\alpha_i^3A_iC^4 - 9\alpha_i^2A_i^2C^4 - 9\alpha_iA_i^3C^4 - 3A_i^4C^4$.

For $\alpha_i \leq C$, the total sum is always negative, and therefore the second derivative is always negative. Hence, $G_i(\alpha_i)$ is strictly increasing for $\alpha_i \leq C$. Actually, the result holds even for larger values of A_i up to $A_i \leq 15C$, but such values are not relevant to the scenarios that we consider. Recall that for AIMD flows we have assumed the conditions 5. $\square$

In normal network conditions, the per flow increase parameters α_i and the total increase A in each round should be small compared to C. This means that the term L_i can be neglected in the calculation of the goodput G_i of a flow i and that in congestion rounds w_{total} can be assumed to be $w_{total} = C$. Using these assumptions in Eqs. (7), (8) and (9), the previous analysis gives the simplified expression

$$G_i = \frac{3\alpha_iC}{4(A_i + \alpha_i)} , \tag{10}$$

which is clearly an increasing function of α_i.

From the above analysis we conclude that in this game, every selfish flow i will have incentives to increase its parameter α_i. The result is that the game will operate at a very inefficient state, that resembles a Tragedy of the Commons situation. The above result is not surprising and is in agreement with the results in [2].

5.2. DropTail router with gentle penalty AIMD flows

We consider again a Drop-tail router but this time with non-synchronized packet losses. In this case, at congestion rounds a random set of packets is dropped. The packets to be dropped are randomly selected from the submitted packets and their number is equal to the packet excess. In each congestion round, a flow may or may not experience packet loss. The expected number of packets that it will lose is proportional to its window size. At first glance, this case appears to provide better incentives to flows not to behave aggressively. A flow with a smaller congestion window at congestion rounds, has a smaller probability to experience packet loss, compared to a flow with a larger congestion window. However, we will show that this scenario is a Tragedy of the Commons situation, as well.

This game has not been studied analytically before. The fact that packet losses are not synchronized makes the analysis harder. We provide an analysis for a toy case with $n = 2$ players.

5.2.1. DropTail router with two gentle penalty AIMD flows

The Game: Assume a router with capacity C and $n = 2$ flows with parameters α_1, α_2, and $\beta_1 = \beta_2 = 1/2$. Each flow i can choose the value of its parameter α_i. *Simplifications:* We assume that at congestion only one flow is randomly selected and the excess is dropped from this flow. Hence only one flow experiences packet loss in each congestion round. The probability that a flow is selected at a congestion round for dropping the excess packets is proportional to the flow's window size. We will also assume that the selected flow has a window size at least as large as the excess.

We fix the value α_1 of flow 1 and assume that flow 2 chooses a value $\alpha_2 = z\alpha_1$, for $z > 1$. We will show that flow 2 achieves a higher goodput by increasing its parameter α_2. At steady state, we know from Eqs. (2) and (4) that $N_i = \alpha_i\tau_i$ and $G_i = \frac{3}{2}\alpha_i\tau_i - \frac{1}{\tau_i+1}L_i$, for $i = 1, 2$.

We can obtain one more expression for the goodput of each flow from a "typical" congestion round of the router. Let $w_{c,i}$ be the average window size of flow i at congestion rounds. We assume that congestion rounds occur periodically, every $\tau + 1$ rounds. In this case, the average total number of packets requested by flow i in $\tau + 1$ rounds, where the last round is a congestion round, is

$$(w_{c,i} - \alpha_i\tau) + (w_{c,i} - \alpha_i(\tau - 1)) + \ldots + (w_{c,i} - \alpha_i) + (w_{c,i}) \ . \tag{11}$$

During the above $\tau + 1$ rounds, flow i experiences on average $(\tau + 1)/(\tau_1 + 1)$ loss rounds. At each of its loss rounds, flow i looses on average L_i packets. Combining this with Eq. (11) gives that the average goodput of flow i, calculated from an average congestion round and the previous τ rounds, is

$$G_i = w_{c,i} - \frac{1}{2}\alpha_i\tau - \frac{L_i}{\tau_1 + 1} \ , \tag{12}$$

and if we ignore the L_i packets that are lost by the flow at loss rounds, the goodput becomes:

$$G_i = w_{c,i} - \frac{1}{2}\alpha_i\tau \ . \tag{13}$$

Step 1: Let $w_{c,i}(k)$ be the window size of flow i at congestion round k and let S_i be the number of loss rounds of flow i after a large number K of congestion rounds. Let $w_c = w_{c,1} + w_{c,2}$ be the mean total number of packets in congestion rounds. Even though w_c is a mean value, to simplify the analysis we will assume that the total number of packets in each congestion round is exactly w_c. Then, the probability that flow 1 experiences packet loss in congestion round k is $w_{c,1}(k)/w_c$. We can assume for flow 1 and for each congestion round k a binary random variable $X_1(k)$ such that

$$X_1(k) = \begin{cases} 1, & \text{flow 1 experiences packet loss in congestion round } k, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the mean value of $X_1(k)$ is $w_{c,1}(k)/w_c$. The total number Ψ_1 of loss rounds of flow 1 is $\Psi_1 = \sum_k X_1(k)$ and the expected total number of loss rounds of flow 1 after K congestion rounds is $S_1 = E[\Psi_1] = \frac{w_{c,1}K}{w_c}$.

We can use an appropriate Hoeffding-Chernoff to show that for large enough K, the random variable Ψ_1 is arbitrary close to S_1. Assume a positive constant ϵ, a probability ρ, and a number of rounds K. The following Hoeffding-Chernoff bound is from [15, Page 200, Theorem 2.3].

Theorem 5. *Let the random variables $X_i(1), X_i(2), \ldots, X_i(K)$ be independent, with $0 \leq X_i(k) \leq 1$ for $k = 1, \ldots, K$. Let $\Psi_i = \sum X_i(k)$ and let $S_i = E[\Psi_i]$. Then*

$$\textit{For any } t \geq 0, \quad P\left[|\Psi_i - S_i| \geq Kt\right] \leq 2e^{-2Kt^2} \ .$$

If $K \geq \frac{\ln(\frac{1}{2\rho})w_c^2}{2\epsilon^2 w_{c,1}^2}$ then by setting $t = \frac{w_{c,1}K}{w_c}$ in Theorem 5 we obtain that

$$\text{with probability at least } \rho, \quad S_1 \in \left[(1-\epsilon)\frac{w_{c,1}K}{w_c}, (1+\epsilon)\frac{w_{c,1}L}{w_c}\right] \ . \tag{14}$$

A sufficiently large number of rounds K can make the constant ϵ arbitrary small and the probability ρ arbitrary high. Since we consider the system at steady state, we can approximate $S_1 \simeq \frac{w_{c,1}K}{w_c}$ for large K. Thus, we will assume

$$S_1 = \frac{w_{c,1}K}{w_c} \ . \tag{15}$$

Similarly

$$S_2 = \frac{w_{c,2}K}{w_c} \ . \tag{16}$$

Let

$$y = \frac{w_{c,2}}{w_{c,1}} \quad . \tag{17}$$

Then, from Eqs. (15) and (16) we get $S_2 = y \cdot S_1$. The frequency $f_{c,i}$ of loss rounds of each flow $i = 1, 2$ at congestion rounds is $f_{c,i} = (\tau + 1)/(\tau_i + 1)$. It follows that $f_{c,2}/f_{c,1} = (\tau_1 + 1)/(\tau_2 + 1)$. Additionally, note that $f_{c,2}/f_{c,1} = S_2/S_1$. Thus

$$y = \frac{\tau_1 + 1}{\tau_2 + 1} \quad . \tag{18}$$

Step 2: Since each congestion round is a loss round either for flow 1 or for flow 2, the frequency of congestion rounds is equal to the sum of the frequencies of loss rounds of the two flows:

$$\frac{1}{\tau+1} = \frac{1}{\tau_1+1} + \frac{1}{\tau_2+1} \quad . \tag{19}$$

Using Eqs. (19) and (18) we get

$$\tau \quad = \quad \frac{\tau_1\tau_2 - 1}{\tau_1+\tau_2+2} \quad = \quad \frac{y\tau_2 - 1}{y+1} \quad = \quad \frac{\tau_1 - y}{y+1} \, . \tag{20}$$

It follows

$$\tau_1 = y\tau_2 + y - 1 \quad \text{and} \quad \tau_2 = \frac{\tau_1 - y + 1}{y} \, . \tag{21}$$

The number of excess packets in a congestion round is a random integer in the range $1, 2, \ldots, (\alpha_1 + \alpha_2)$. We will assume that the average excess is the half of the mean increase step. This gives that the average total requested window size w_c in congestion rounds is

$$w_c \quad = \quad C + \frac{\alpha_1 + \alpha_2}{2} \, . \tag{22}$$

By the definition of the toy case in each congestion round either flow 1 or flow 2 experiences packet loss. We obtain from Eq. (18) that flow 1 experiences packet loss at a congestion round with probability $1/(y+1)$ and flow 2 with probability $y/(y+1)$. Therefore, for $i = 1, 2$, the average number L_i of packets that flow i looses at congestion rounds is

$$L_1 = \frac{1}{y+1}\frac{\alpha_1+\alpha_2}{2} \quad \text{and} \quad L_2 = \frac{y}{y+1}\frac{\alpha_1+\alpha_2}{2} \, . \tag{23}$$

Step 3: Using $w_{c,1} = \frac{C+\frac{\alpha_1+\alpha_2}{2}}{y+1}$ and $\tau = \frac{\tau_1-y}{y+1}$ we can calculate τ_1 as follows:

$$\begin{aligned} & G_1 = w_{c,1} - \tfrac{1}{2}\alpha_1\tau - \tfrac{1}{\tau_1+1}L_1 \\ \Rightarrow \quad & \tfrac{3}{2}\alpha_1\tau_1 - \tfrac{1}{\tau_1+1}L_1 = \tfrac{C+\frac{\alpha_1+\alpha_2}{2}}{y+1} - \tfrac{1}{2}\alpha_1\tfrac{\tau_1-y}{y+1} - \tfrac{1}{\tau_1+1}L_1 \\ \Rightarrow \quad & \tfrac{3}{2}\alpha_1\tau_1 = \tfrac{C+\frac{\alpha_1+\alpha_2}{2}}{y+1} - \tfrac{1}{2}\alpha_1\tfrac{\tau_1-y}{y+1} \, . \end{aligned} \tag{24}$$

Solving for τ_1 gives

$$\tau_1 = \frac{2C + \alpha_1 + z\alpha_1 + \alpha_1 y}{3\alpha_1 y + 4\alpha_1} \, . \tag{25}$$

Step 4: We will obtain one more equation for τ_1 by starting this time from the goodput G_2 of flow 2 and then proceeding as in (24):

$$\begin{aligned} & G_2 = w_{c,2} - \tfrac{1}{2}\alpha_2\tau - \tfrac{1}{\tau_2+1}L_2 \\ \Rightarrow \quad & \tfrac{3}{2}\alpha_2\tau_2 - \tfrac{1}{\tau_2+1}L_2 = \tfrac{yC+y\frac{\alpha_1+\alpha_2}{2}}{y+1} - \tfrac{1}{2}\alpha_2\tfrac{\tau_1-y}{y+1} - \tfrac{1}{\tau_2+1}L_2 \\ \Rightarrow \quad & \tfrac{3}{2}z\alpha_1\tfrac{\tau_1-y+1}{y} = \tfrac{2Cy+\alpha_1 y+\alpha_2 y}{2(y+1)} - \tfrac{1}{2}\alpha_2\tfrac{\tau_1-y}{y+1} \, . \end{aligned} \tag{26}$$

Solving for τ_1 gives

$$\tau_1 = \frac{2Cy^2 + \alpha_1 y^2 + 5z\alpha_1 y^2 - 3z\alpha_1}{4z\alpha_1 y + 3z\alpha_1} \ . \tag{27}$$

Step 5: Combining Eqs. (25) and (27) to eliminate τ_1 gives the following cubic equation of y:

$$\begin{aligned} f(y) = \ & (6C + 15z\alpha_1 + 3\alpha_1) \cdot y^3 + (8C + 16z\alpha_1 + 4\alpha_1) \cdot y^2 \\ & -(8Cz + 16z\alpha_1 + 4z^2\alpha_1) \cdot y - (6Cz + 15z\alpha_1 + 3z^2\alpha_1) = 0 \quad . \end{aligned} \tag{28}$$

Lemma 6. *For $z > 0$, Eq. (28) has exactly one positive real solution $y(z)$ and this solution is a strictly increasing function of z.*

Proof. Note that $f(0) = -(6Cz + 15z\alpha_1 + 3z^2\alpha_1) < 0$ and that $f(1+z) > 0$ (substituting $y = z+1$ and simplifying the expression gives a sum of strictly positive terms). The derivative of $f(y)$ with respect to y is

$$\begin{aligned} f'(y) = \ & (18C + 9\alpha_1 + 45z\alpha_1) \cdot y^2 + (16C + 8\alpha_1 + 32z\alpha_1) \cdot y \\ & -(8Cz + 16z\alpha_1 + 4z^2\alpha_1) \quad . \end{aligned} \tag{29}$$

Consider the equation

$$f'(y) = 0 \quad . \tag{30}$$

The discriminant of Eq. (30) is

$$\begin{aligned} \Delta = \ & (16C + 8\alpha_1 + 32z\alpha_1)^2 \\ & +4\,(18C + 9\alpha_1 + 45z\alpha_1)\,(8Cz + 16z\alpha_1 + 4z^2\alpha_1) \quad , \end{aligned} \tag{31}$$

which is strictly positive for $z > 0$. Consequently, Eq. (30) has two real solutions

$$y_{1,2} = \frac{-16C - 8\alpha_1 - 32z\alpha_1 \pm \sqrt{\Delta}}{2(18C + 9\alpha_1 + 45z\alpha_1)} \quad . \tag{32}$$

From (31) and (32), it is clear that one solution is negative and one is positive. Let y_2 be the positive solution of (29). The function $f(y)$ is decreasing in the interval $(0, y_2)$ and increasing in (y_2, ∞). Since $f(0) < 0$, it follows that $f(y)$ has exactly one positive real solution $y(z)$, as a function of $z > 0$. Since $f(y_2) < 0$, it follows that

$$y(z) > y_2 \quad . \tag{33}$$

Since, by definition, $y(z)$ is a root of function $f(y)$, after replacing y by $y(z)$ in (28), differentiating $f(y(z))$ with respect to z, and solving for the derivative $y'(z)$ we obtain

$$\begin{aligned} y'(z) &= \frac{num(z)}{den(z)} \quad \text{, where:} \\ num(z) &= -15\alpha_1(y(z))^3 - 16\alpha_1(y(z))^2 + (8C + 8z\alpha_1 + 16\alpha_1)y(z) \\ & \quad +(6C + 15\alpha_1 + 6z\alpha_1) \\ den(z) &= (18C + 9\alpha_1 + 45z\alpha_1)(y(z))^2 + (16C + 8\alpha_1 + 32z\alpha_1)y(z) \\ & \quad -(8Cz + 16z\alpha_1 + 4z^2\alpha_1) \end{aligned} \tag{34}$$

For the numerator $num(z)$ note that:

$$\begin{aligned} f(y(z)) + z \cdot num(z) = \quad & 3z^2\alpha_1 + 4z^2\alpha_1 y(z) + 4\alpha_1(y(z))^2 + 8C(y(z))^2 \\ & +3\alpha_1(y(z))^3 + 6C(y(z))^3 \quad > 0 \end{aligned} \tag{35}$$

Since $f(y(z)) = 0$ and $z > 0$, it follows that $num(z) > 0$. For the denominator $den(z)$ in Eq. (34), note that $den(z) = f'(y(z))$. Now, since $y(z) > y_2$ we get that $den(z)$ is strictly positive. From this we conclude that $y'(z)$ is strictly positive, for $z > 0$. This completes the proof. □

Since $\alpha_2 = z\alpha_1$, Eqs. (4) and (21) imply that:

$$G_2(z) = \frac{3}{2}\alpha_2\tau_2 = \frac{3}{2}z\alpha_1\frac{\tau_1 - y + 1}{y} \tag{36}$$

Now we substitute τ_1 from Eq. (27) and get:

$$G_2(z) = \frac{6Cy + 3\alpha_1 y + 3z\alpha_1 y + 3z\alpha_1}{8y + 6} \, . \tag{37}$$

Lemma 7. *Function $G_2(z)$ in (37) is increasing for $z > 0$.*

Proof. Taking into consideration that y is a function of z, the derivative of $G_2(z)$ with respect to z is:

$$G_2'(z) = \frac{9\alpha_1 + 21\alpha_1 y + 12\alpha_1 y^2 + 9\alpha_1 y' + 3(C - z\alpha_1)y'}{2(3 + 4y)^2} \, . \tag{38}$$

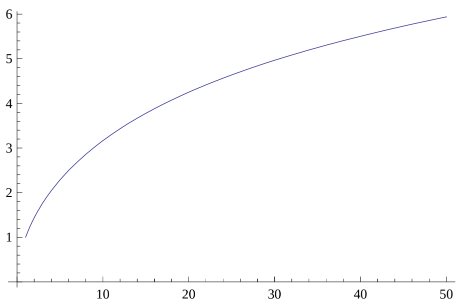
Since $C \geq z\alpha_1 = \alpha_2$ and $y' > 0$, the proof is complete. □

In Eq. (36) we can also use the more accurate Eq. (3). In this case, numerical evaluations show that $G_2(z)$ is still an increasing function of z. However, the analytical expression becomes very complicated. Note that Eqs. (25), (27) and (28) are not influenced if we use the less accurate equation for the goodput, because the term L_i is eliminated in Eqs. (24) and (26).

We used the analytical results on a representative game instance with router capacity $C = 100$ and $n = 2$ flows. The calculations have been performed using Mathematica. In particular, we calculated the positive solution of $y(z)$ from Eq. (28). The results are presented graphically in Fig. 2a. The corresponding plots of τ_1 and τ_2 are given in Fig. 2b.

We used Eq. (3) to calculate the goodputs G_1 and G_2 of flows 1 and 2, respectively. We also used the less accurate Eq. (4) to calculate $G_{2,approx}$ for flow 2. The results are shown in Fig. 3a. The experimental results for the goodputs G_1 and G_2 are presented in Fig. 3b. The graphs show that the results of the analytical model are very close to the numerical and the experimental results, and confirm that the goodput $G_2(z)$ of flow 2 is an increasing function of z.

The experiments for the toy case with two flows have been conducted with a simple simulator for Window-games with basic AIMD flows, called NetKnackBasic. The router capacity was $C = 100$. To avoid or at least reduce


(a) The graph of y as a function of z.

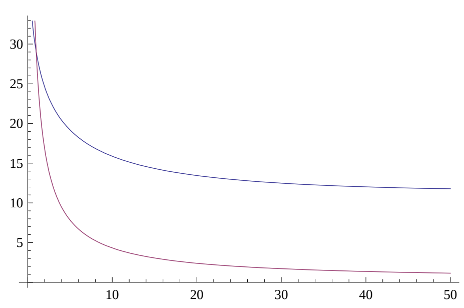
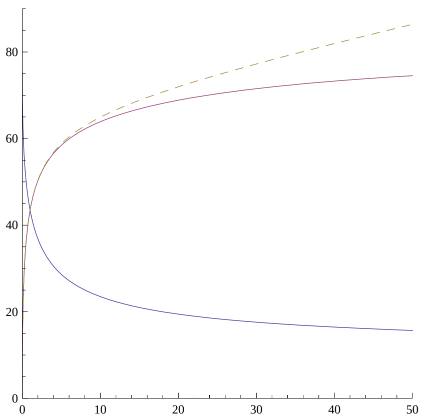

(b) The graphs of τ_1 and τ_2 (lower line) as functions of z.

Figure 2: Analytical results for y, τ_1 and τ_2.


(a) The analytical results for goodputs G_1 (lower line), G_2 and $G_{2,approx}$ (dashed line) as functions of z.

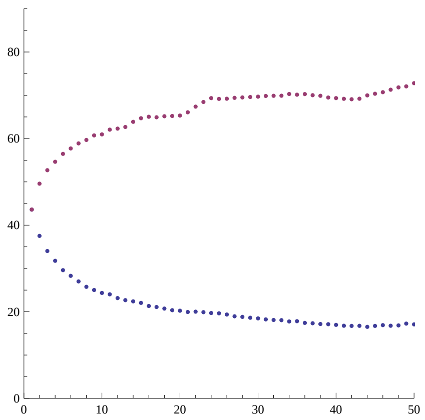

(b) The experimental results for goodputs G_1 (lower plot) and G_2 as functions of z.

Figure 3: Analytical and experimental results for goodputs G_1 and G_2 as functions of z.

synchronization effects the capacity varied randomly from $C - 1$ to $C + 1$ with an average of C. Parameter α was $\alpha_1 = 1$ for flow 1 and $\alpha_2 = z\alpha_1$ for flow 2. Each game lasted for 100000 rounds.

5.3. Prince router with gentle penalty flows

We now consider the case of a Prince router of capacity C and n flows $i = 1, \ldots, n$ with parameters $(\alpha_i, \beta_i = 1/2)$. We start with two general properties of this game. Assume that the system has reached steady state and that the overflow in congestion rounds does not exceed the maximum of all windows $\max_{i=1,\ldots,n} w_i$.

Lemma 8. *Let flows i and k be the flows of minimum and maximum goodput, respectively, and let $\lambda = G_k/G_i$. Then, it holds $\lambda \leq 2$.*

Proof. From Eq. (4) we know that $G_i = \frac{3}{2}\alpha_i\tau_i$ and that the average window size of i at its loss rounds is $w_{c,i} = 2\alpha_i\tau_i$. Similarly, $G_k = \frac{3}{2}\alpha_k\tau_k$ and $w_{c,k} = 2\alpha_k\tau_k$. Note that the congestion window of flow k takes values in the range $\frac{1}{2}w_{c,k}, \ldots, w_{c,k}$. Flow i has at its congestion rounds the maximum window among all flows. Thus $w_{c,i} \geq \frac{1}{2}w_{c,k} \Rightarrow \alpha_i\tau_i \geq \frac{1}{2}\alpha_k\tau_k \Rightarrow G_i \geq \frac{1}{2}G_k \Rightarrow \lambda \leq 2$. □

Lemma 9. *In congestion rounds with total overflow at most $\max_{i=1,\ldots,n} w_i$ a flow that does not exceed its fair share C/n, does not lose any packet.*

Proof. At congestion, $\sum_{i=1}^{n} w_i > C$. Clearly, $\max_{i=1,\ldots,n} w_i > C/n$. Thus, a flow i with $w_i \leq C/n$ cannot have the maximum window. □

We consider first a toy case with $n = 2$ flows and then the more general case with n flows.

5.3.1. Prince router with two gentle penalty AIMD flows

The Game: Assume a Prince router with capacity C and two flows $i = 1, 2$ with parameters α_1, α_2, and $\beta_1 = \beta_2 = 1/2$. Each flow can choose its parameter α.

We fix α_1 and assume that $\alpha_2 > \alpha_1$. Then, let $\alpha_2 = z\alpha_1$, for $z > 1$. At steady state $G_1 = \frac{3}{2}\alpha_1\tau_1$ and $G_2 = \frac{3}{2}\alpha_2\tau_2$. Assuming λ such that $G_2 = \lambda G_1$, we get

$$z\tau_2 = \lambda\tau_1 \; . \tag{39}$$

If $w_{c,2}$ is the average window size of flow 2 at its loss rounds then from Eqs. (1) and (2) we know that $w_{c,2} = 2z\alpha_1\tau_2$. Let $w_{s,1}$ be the average window size of flow 1 in rounds that are congestion rounds but not loss rounds for it (flow 1). For a round to be a congestion round it must hold $w_1 + w_2 > C$, and therefore for loss rounds of flow 2 we have $w_{s,1} + w_{c,2} > C$. Using $w_{s,1} \geq \alpha_1\tau_1$ we get $\alpha_1\tau_1 < C - 2z\alpha_1\tau_2$. It follows from Eq. (39) that

$$\alpha_1\tau_1 < \frac{C}{2\lambda + 1} \; . \tag{40}$$

Furthermore, Prince guarantees that the greatest window size $2a_1\tau_1$ of flow 1 equals at least its fair share $C/2$, i.e., it holds $a_1\tau_1 \geq C/4$. Combining this with (40) gives

$$\lambda < \frac{3}{2} \; . \tag{41}$$

Note that inequality (41) holds for reasonable values of α_1, i.e., for cases where α_1 is considerably smaller than the router capacity C.

5.3.2. Prince router with n gentle penalty AIMD flows

The Game: Assume a router with capacity C and n flows $i = 1, \ldots, n$ with parameters $(\alpha_i, \beta_i = 1/2)$. The router uses the Prince policy for its queue. Each flow can choose its parameter α_i.

We provide a theoretical argument for symmetric profiles. Assume that flows $1, \ldots, n-1$ use value α_1 for parameter α and flow n uses a value $\alpha_n = z\alpha_1$, for $z > 1$. Due to symmetry, it holds $\tau_1 = \ldots = \tau_{n-1}$ and $G_1 = \ldots = G_{n-1}$. Let $G_n = \lambda G_1$. Then from Eq. (4)

$$\frac{3}{2}\alpha_n\tau_n = \lambda\frac{3}{2}\alpha_1\tau_1 \Rightarrow z\tau_n = \lambda\tau_1 \; . \tag{42}$$

Congestion rounds occur on average every $\tau + 1$ rounds. We focus on flow n and consider the last congestion round, before flow n experiences packet loss. In this round the window size of flow n is (on average) $w_n = \alpha_n\tau_n + \alpha_n(\tau_n - (\tau + 1))$. Flow n has not the largest window size in this congestion round, therefore $w_n \leq 2\alpha_1\tau_1$. Hence

$$\begin{aligned} \alpha_n\tau_n + \alpha_n(\tau_n - (\tau+1)) &\leq 2\alpha_1\tau_1 \\ \Rightarrow 2\alpha_n\tau_n \leq 2\alpha_1\tau_1 + z\alpha_1\tau + z\alpha_1 &\Rightarrow 2\lambda\tau_1 \leq 2\tau_1 + z\tau + z \\ \Rightarrow \lambda \leq 1 + \frac{z(\tau+1)}{2\tau_1} \; . \end{aligned} \tag{43}$$

Recall that in each congestion round, exactly one flow experiences packet loss. Thus, the sum of the frequencies of loss rounds of all flows is equal to the frequency of congestion rounds at the router:

$$\sum_{i=1}^{n} \frac{1}{\tau_i + 1} = \frac{1}{\tau + 1} \; . \tag{44}$$

Since $\tau_1 = \ldots = \tau_{n-1}$ it follows from Eq. (44)

$$\begin{aligned} \frac{1}{\tau+1} \geq \sum_{i=1}^{n-1} \frac{1}{\tau_i + 1} = \frac{n-1}{\tau_1 + 1} &\Rightarrow \tau_1 \geq (n-1)(\tau+1) - 1 \\ &\Rightarrow 2\tau_1 \geq (n-1)(\tau+1) \; . \end{aligned} \tag{45}$$

The last step holds from the condition $\tau_1 \geq 1$ of (5). Using (45) in (43) gives:

$$\lambda \leq 1 + \frac{z}{n-1} \; . \tag{46}$$

The above upper bound on λ shows that an aggressive flow cannot gain more goodput than a bounding factor above its fair share. The upper bound on λ converges to 1 as the number of flows n increases. The flow can increase its parameter z but this incurs costs to the flow. In Eq. (42) we used the approximate type for the goodput, which does not take into account the packet losses of the flow at loss rounds. If an aggressive flow increases its parameters

z, than it experiences packet loss with a much higher frequency and hence the packet losses at loss rounds become important. We consider the above results about Prince, strong evidence that Prince will lead the system to efficient and fair Nash equilibria. The experimental results presented in Section 6 support this claim. We intend to work on a more thorough analysis of Prince.

6. Experimental Evaluation for AIMD Flows

We performed an extensive set of experiments with the network model of Fig. 1a, with $n = 10$ AIMD flows, a router with capacity $C = 100$, numerous queue policies and both the gentle and the hybrid penalty models. For parameter α we used values from the set $\{1, 2, .., 50\}$ and for β from the set $\{0.5, 0.51, .., 0.99\}$. Our main focus is on the results for varying parameter α when β is fixed at $\beta = 0.5$.

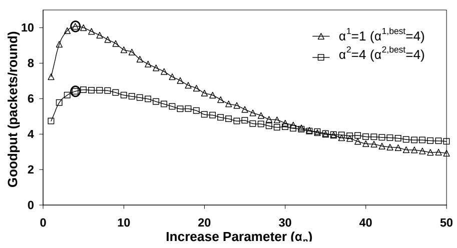
6.1. The Methodologies

First, we applied the iterative methodology of [2], we call it M1, to find SNE. Second, thanks to the simplicity of the Window-game, we could perform a brute force search on all symmetric profiles to discover all possible SNE (methodology M2). Finally, we used random, not necessarily symmetric, starting points along with a generalization of the procedure M1 to check if the network converges to general, not necessarily symmetric, NE (methodology M3). The experiments where performed with a simulator for the Window-game, which is called NetKnack[25]. NetKnack is implemented in Java, supports common queue disciplines and flow penalty models and can perform from a simple experiment to massive series of experiments.

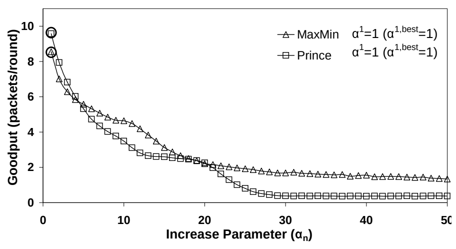
Methodology M1 is executed in iterations. In the first iteration, we set $\alpha^1 = 1$ for flows $1, \dots, n-1$ and search for the best response of flow n. Let $\alpha^{1,best}$ be the value α, with which n achieves the best goodput. In the next iteration, flows $1, \dots, n-1$ play with $\alpha^2 = \alpha^{1,best}$ and we search for the best α_n in this profile. If at iteration k, $\alpha^{k,best} = \alpha^k$ then this value, denoted by α_E, is the SNE of the game.

The evolution of methodology M1 is illustrated in Fig. 4a. The first iteration (α^1) shows that $\alpha^{1,best} = 4$, so at the second iteration all flows adopt $\alpha^{1,best}$ and again flow n seeks for the best response to this profile. In the last iteration, the n cannot find a better move than its current strategy and so this profile constitutes a SNE. In some occasions the initial profile is a SNE, where flow n sticks to $\alpha^{1,best} = 1$ (see Fig. 4b).

Methodology M2 applies a brute force search enumerating every possible symmetric profile to detect all SNE. The method consists of a main loop on parameter α. For each possible value k of parameter α, the startup profile for flows $1, \dots, n-1$ is set to $(\alpha_1 = \ldots = \alpha_{n-1} = k)$ and measurements are made for all possible values of α_n of flow n. If the best response of flow n is the action taken by all other flows, then we have a SNE. We executed experiments with values for parameter α in the range from 2 to $C/2$.



(a) RED router with hybrid penalty flows. The SNE is found in the second iteration.



(b) MaxMin and Prince routers with hybrid penalty flows. For both routers, the SNE is found in the first iteration.

Figure 4: Finding SNE with methodology M1.

Methodology M3 is a heuristic that extends methodology M1 to non-symmetric profiles. The first step is to create a random non-symmetric starting point with a random window size for each flow. Then, in every iteration, each flow searches independently for the best response to the current profile, assuming that the other flows do not change their strategy. The simulation reaches a (possibly non-symmetric) NE when in an iteration all flows select as a best response the same action as in the previous profile.

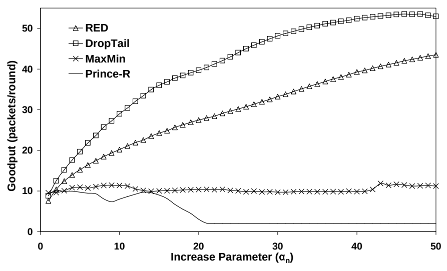
By convention, in all the above methodologies a flow switches to a better value of parameter α only if the average improvement of its utility is at least 2%. We focus on the results for values of parameter α in the range 1 to $C/2$.

Every experiment consists of 2200 rounds. The first 200 rounds are used to allow the flows to reach steady state. To avoid synchronization of flows' windows the capacity C changes randomly, with ± 1 steps, in the region 100 ± 5 with an average of 100 packets. Finally, all measurements are averaged over 30 independent executions of each experiment.

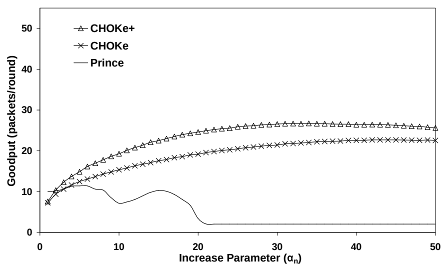
6.2. The Results

We present graphs with the results of experiments where flows $i = 1, \ldots, 9$ use $\alpha = 1$ and flow 10 tries all possible values for $\alpha_{10} \in \{1, \ldots, 50\}$. The results for gentle penalty flows with different queue policies are shown in Fig. 5. We can see that Prince and MaxMin induce efficient Nash equilibria with small values for parameter α_{10}, while DropTail, RED, CHOKe and CHOKe+ lead flow 10 to use large values for α_{10}. The results for hybrid penalty flows in Fig. 6 show that with Prince and MaxMin the deviator player 10 has clearly suboptimal performance for $\alpha_{10} > 1$. Hence, the symmetric profile $\alpha_i = 1$, for $i = 1, \ldots, n$, of the first iteration, is a SNE.

SNE that have been found the methodology M1 are given in Tables 1a and 1b. We would like to note that depending on the parameters of each experiment, like the capacity C and the number of players n, the value α for the NE may differ significantly.

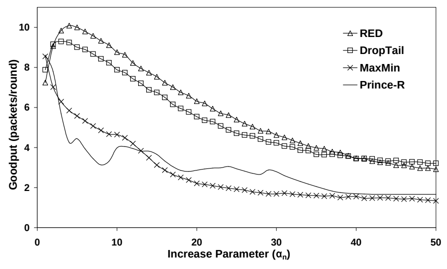


(a) RED, DropTail, MaxMin and Prince-R

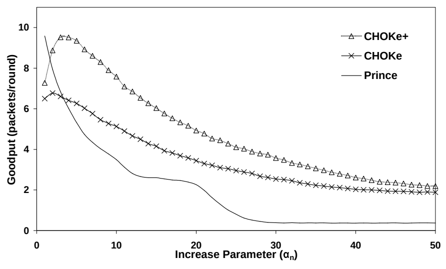


(b) CHOKe, CHOKe+ and Prince

Figure 5: First iteration of M1 with gentle penalty flows. The graphs show the goodput of flow n for each possible value of its parameter α_n.



(a) RED, DropTail, MaxMin and Prince-R



(b) CHOKe, CHOKe+ and Prince

Figure 6: First iteration of M1, this time with hybrid penalty flows. The graphs show the goodput of flow n for each possible value of its parameter α_n.

- **Gentle penalty**: In the gentle penalty model, which corresponds to TCP SACK, a flow experiences a mild punishment upon packet loss. This fact motivates aggressive flows to use large values for parameter α (and parameter β). If, additionally, the router's mechanism does not differentially punish greedy flows the game is driven to inefficient SNE. From Table 1a it is obvious that fully stateless policies like DropTail, RED, CHOKe and CHOKe+ are not able to aim successfully at the aggressive flow, so the game is drawn away to undesirable equilibria. On the contrary, Prince finds and penalizes only the most greedy flow, thus no flow wants to be too aggressive ($\alpha_E = 4$). When flows can vary their increase parameter α Prince induces efficient equilibria with high goodput and tolerable loss rate. In games where the flows can choose their parameter β, all policies lead the selfish flows to use large values for β.
- **Hybrid penalty**: In the hybrid penalty model, which corresponds to TCP Reno, the cost incurred to flows by packet losses is higher. Therefore, all buffer management policies accomplish more or less to discourage flows from being too aggressive. Again, the predominance of Prince is obvious

(Table 1b), having the best achieved goodput and a preferable loss rate. In this scenario, Prince outperforms even MaxMin, a fully stateful policy which keeps a separate queue for each flow. The success of Prince comes up from its targeted principle, namely punishing (dropping packets) only the largest flow, so induces minimal average loss rate. It is noteworthy that only Prince and MaxMin achieve to prevent flows from adopting high values (> 0.5) for the decrease parameter β. These policies shield the AIMD algorithm because flows choose to half their window upon decrease, a fact that can lead the repeated window game to a fair equilibrium (equal window sizes).

Queue Policy	parameter α			parameter β		
	α_E	goodput packets/round	loss rate (%)	β_E	goodput packets/round	loss rate (%)
DropTail	α_3=50	5,499	78,8	β_2=0,97	9,841	3,9
RED	α_2=49	5,485	78,5	β_2=0,97	8,309	4,4
CHOKe	α_3=49	3,363	86,8	β_2=0,94	7,761	5,4
CHOKe+	α_3=50	5,431	79,1	β_2=0,96	8,118	4,5
Prince-R	α_2=2	9,987	7,4	β_3=0,94	9,995	7,3
Prince	α_2=4	9,111	13,9	β_2=0,94	9,993	7,3
MaxMin	Osc. between a=9 and a=12			β_2=0,92	9,998	4,2

(a) **SNE** of games with **gentle** penalty flows.

Queue Policy	parameter α			parameter β		
	α_E	goodput packets/round	loss rate (%)	β_E	goodput packets/round	loss rate (%)
DropTail	α_5=6	5,863	6,4	β_2=0,92	7,999	2,4
RED	α_2=4	6,427	4,4	β_2=0,96	7,591	3,0
CHOKe	α_2=2	6,182	3,6	β_4=0,78	6,674	2,6
CHOKe+	α_2=3	6,650	3,8	β_2=0,93	7,687	2,9
Prince-R	α_1=1	8,693	1,1	β_2=0,72	8,759	1,6
Prince	α_1=1	9,573	2,1	β_1=0,50	9,617	2,1
MaxMin	α_1=1	8,547	1,8	β_1=0,50	8,503	1,9

(b) **SNE** of games with **hybrid** penalty flows.

Table 1: **SNE** of games with gentle and hybrid penalty flows. The SNE in the cases where the flows can select parameter α or when they can select parameter β, are shown. For each SNE, the corresponding efficiency (goodput, loss rate) is given.

The brute force search method M2 on DropTail and RED with gentle and hybrid flows, found the SNE that had been found with M1 and revealed additional SNE only for DropTail with hybrid penalty flows. These additional SNE use larger values of parameter α and are less efficient than the SNE found with methodology M1 for the same game.

Finally, the search for non-symmetric NE with methodology M3 for all queue policies and with both gentle and hybrid penalty flows, did not reveal any additional NE. On the contrary, it is noteworthy that this generalized methodology concluded to SNE that had already been found with M1.

7. Non-AIMD flows with complete information.

In the rest of this work we consider the more general class of Window-games where the flows can use an arbitrary strategy to choose their congestion window. The general description of Window-games with non-AIMD flows is that there is a common resource, every player requests an arbitrary part of this resource and the outcome depends on the moves of all players. Note that even though this game bears some similarity with congestion games [30], it does not fit into the class of (weighted) congestion games (see for example [30, 24, 23, 12]).

In this Section, we discuss the case of games with complete information and analyze several one-shot Window-games, while in Section 8 we discuss the case of Window-games with incomplete information. A one-shot Window-game is a Window-game that lasts only one round. In a one-shot Window-game the only consequence for a flow when it looses packets is the cost g induced by each packet loss. There can be no other consequences for packet losses like window reductions or timeouts since there is only one round.

For the one-shot Window-game we assume a router of capacity C and n flows $i = 1, \ldots, n$. Let w_i be the window size of flow i and $w_{total} = \sum_i w_i$ the total request of all flows. Furthermore, let each successfully transmitted packet give a profit of 1 and each lost packet incur a cost of $g \geq 0$.

7.1. DropTail

The Game: A DropTail router of capacity C and n flows. The total number of flows n is known to all flows. The game is played in one round. Each flow i chooses its window size w_i.

If there is no congestion, all packets are served. If there is congestion, i.e., $w_{total} > C$, then $w_{total} - C$ packets are randomly selected and dropped. In this case, a flow i will have on average $(w_i C)/w_{total}$ successful packets and $w_i(w_{total} - C)/w_{total}$ lost packets.

Lemma 10. *If $g = 0$, then there is a unique SNE, where each flow requests the maximum possible congestion window $w = C$.*

Proof. Since $g = 0$, the utility of any flow i for $w_{total} = \sum_k w_k$ is:

$$u(i) = \begin{cases} w_i, & \text{if } w_{total} \leq C \ , \\ \frac{C}{w_{total}} w_i, & \text{if } w_{total} > C \ . \end{cases}$$

Clearly, in both cases $w_{total} \leq C$ and $w_{total} > C$, the utility of flow i is an increasing function of w_i. Thus, there is a unique SNE, where all flows request the maximum possible value $w_i = C$. The SNE is very inefficient. □

Lemma 11. *If $g > 0$, then there is a SNE where all flows use the window size*

$$w = \frac{C(1+g)(n-1)}{gn^2} \ .$$

Proof. Assume that the $n-1$ flows $i = 1, \ldots, n-1$ use $w_i = y$ and that flow n uses $w_n = x$. In case $x + (n-1)y < C$ we can increase x at least until $x + (n-1)y = C$, so we will assume

$$x \geq C - (n-1)y \ . \tag{47}$$

The utility of flow n is the average number of successful packets minus g times the average number of lost packets:

$$u_n(y,x) = x\frac{C}{(n-1)y + x} - gx(1 - \frac{C}{(n-1)y + x}) \ . \tag{48}$$

The only positive root of the partial derivative of $u_n(y,x)$ with respect to x is

$$x = \frac{gy(1-n) + \sqrt{Cgy(1+g)(n-1)}}{g} \ . \tag{49}$$

Setting $x = y$ gives $x = y = \frac{C(1+g)(n-1)}{gn^2}$. We conclude that there is a SNE at

$$w = \frac{C(1+g)(n-1)}{gn^2} \ . \qquad \square$$

For example, if $g = 1, C = 100$ and $n = 10$, then the SNE is at $w = 18$. Experimental results are presented in Table 2.

A legitimate question is whether there are reasonable conditions, for which the fair share $w_i = C/n$ is a SNE. Using $w \leq C/n$ in Lemma 11 and solving for g gives $g \geq n-1$. This shows that the larger the number flows, the higher the cost g must be, in order to prevent an undesirable SNE. A sufficient condition on the cost g that does not depend on the number of players n is $g \geq C-1 \geq n-1$. However, when g takes such large values the game degenerates. For example, if $g \geq C-1$, then almost any (not necessarily symmetric) profile with total window size equal to C is a NE.

7.2. MaxMin

The Game: A MaxMin router of capacity C and n flows. The number n is known to all flows. The game is played in one round. Each flow i chooses its window size w_i.

Shenker showed in [31] that a MaxMin policy enforces a fair NE in a network model where the flows are Poisson sources. In the Window-game model as well, the MaxMin policy leads the system to desirable equilibria. More precisely:

Lemma 12. *In MaxMin, there is a SNE where all flows play $w_i = C/n$. If $g > 0$, then this is the only NE of the game.*

Proof. In MaxMin, a flow i does not loose any packet if w_i does not exceed its fair share. Consequently, each flow i will request at least its fair share, i.e., $w_i \geq C/n$. Thus, the total request w_{total} will be at least $w_{total} \geq C$ and no flow can gain by playing a value larger than C/n. This implies that there is a SNE where each flow i plays the strategy $w_i = C/n$. If there is a strictly positive cost $g > 0$ for each packet loss, then the above SNE is the only NE of the game. $\square$

Clearly, the SNE where all flows request $w_i = C/n$ is the optimal solution for the Window-game problem. As already discussed, the disadvantage of MaxMin is that it is a stateful policy that consumes too many computational resources to be applied in practice.

7.3. Prince

The Game: A Prince router of capacity C and n flows. The total number of flows n is known to all flows. The game is played in one round. Each flow i chooses its window size w_i.

The behavior of Prince is very similar to MaxMin. Every flow i can safely request its fair share C/n, so $w_i \geq C/n$. If a flow i requests $w_i > C/n$, then it may very well receive far less than its fair share (unlike MaxMin where it will receive at least its fair share). Clearly, the profile where all flows play $w_i = C/n$ is a SNE. If the cost for packet loss is $g > 0$, then this is the only NE of the game. Hence:

Lemma 13. *If $\sum_i w_i - C \leq \max_i w_i$ then there is a SNE where all flows play $w_i = C/n$. If $g > 0$, then this is the only NE of the game.*

utility function / queue policy	U = passed – g * dropped			
	g=0	**g=0.1**	**g=0.5**	**g=1.0**
DropTail	100	〉60	26-30	17,18
RED	100	〉70	26-28	20
CHOKe	≈44	≈ 35	21-22	14-16
CHOKe+	100	〉70	29	17-18
Prince-R	10,11	10,11	10	10
Prince	10,11	10,11	10	10
MaxMin	≥10	10	10	10

Table 2: Window sizes at SNE for the one-shot game with complete information. For every queueing policy and different values of the cost factor g, i.e., variations of the utility function, the window sizes of the corresponding SNE are shown.

7.4. Experimental results

Results for one-shot games with complete information are presented in Table 2. With MaxMin and both Prince policies, the equilibrium is at the fair share or almost at the fair share, even for low values of the cost factor g. For DropTail, RED, and CHOKe routers, a low value for g leads selfish players to request the maximum possible window size (aggressive behavior). The higher the cost factor g, the more conservative becomes the strategy of selfish flows.

The experiments have been conducted with the simulator NetKnack, with $n = 10$ flows and router capacity $C = 100$. The results for each each experiment were calculated from the average of 40 executions. We used methodology M2 of Section 6 to identify the SNE for window sizes in the range $\{1, \ldots, 100\}$.

8. Non-AIMD flows with incomplete information

In this Section, we discuss Window-games in which the players do not know the total number of players n. These Window-games, where the flows are not fully informed about all the parameters of the game, belong to the class of games with incomplete information. We distinguish three cases of such Window-games.

8.1. One-shot Window-game with no prior probabilities

The Game: A router of capacity C and n players. The players have no prior probabilities on the number of players, that is, they have no distribution information for the unknown number n. The game is played in one round and each player chooses its window size.

Note that in this case, where the players do not have distribution information on the unknown parameters, the Nash equilibrium or the Bayes-Nash equilibrium concepts cannot be applied. Several concepts, like games with no prior probability or pre-Bayesian games [4, 1] have been proposed to capture games with no prior probabilities. One possibility is to search for an *ex-post equilibrium*, which is a profile where each player's strategy is a best response to the other player's strategies, under all possible realizations of the uncertain data [1]. In particular, for the Window-games of this Section, each player's strategy would be a best response regardless of the number of participating players. Even though an ex-post equilibrium would be a very attractive solution, the problem is that, in general, ex post equilibria do not exist in incomplete-information games.

Other, more general, related equilibrium concepts are for example the safety-level equilibrium [4], the robust-optimization equilibrium [1], and the minimax-regret equilibrium. The common characteristic of these equilibrium concepts is that the player selects a very conservative action. In this case, the players may, for example, assume that the number of players n is the maximum possible number $n = C$. Hence, the players will play as in the game with complete information with $n = C$. For strategies like Prince and MaxMin, the choice of each player would be $w = 1$. Interestingly, in common TCP implementations, like Reno or Tahoe, flows start with a very conservative initial window size $w = 1$.

8.2. One-shot Window-game with prior probabilities

The Game: A router of capacity C and n players. The number of players n is a random variable and the players know the corresponding probability distribution. The game is played in one round and each player i chooses its window size w_i.

In practice, flows engaged in a TCP-game are likely to have some prior information on the number n or, equivalently, on their fair share. Note that real TCP flows are actually playing a repeated game, which means that the flow will in most cases have an estimation on its fair share from the previous rounds (except of the first round or a round after some serious network change). If the flow has prior information on the distribution of the unknown parameter n, the result is a Bayesian game. We leave the analysis of this and the following case as future work.

8.3. Repeated Window-game with incomplete information

The Game: A router of capacity C and n players. The number of players n is unknown and probability distribution information on n may or may not be given. Each player chooses its window size for each round of the game.

The motivation for this game comes from the fact that the actual game that a TCP flow has to play is a repeated game with incomplete information. The problem has been addressed from an optimization and an on-line algorithm point of view in [19]. One other approach would be to consider this as a learning game; there are C players and nature decides for each player either to participate in the game or to stay idle. The goal of each participating player is to "learn" the unknown number n of players who participate.

We believe that a very interesting class of games that can also be used to model this case of Window-games, is the class of *unknown games* (for example see [6, Section 7.5]). In an unknown game, K players play a game repeatedly, and at each time instance t, after taking an action, player k observes his loss (or gain) but does not know his entire loss (or gain) function and cannot observe the other players' actions. The player may not even know the number of players participating in the game. Interestingly, even with such limited information there are conditions under which the game converges to a correlated equilibrium or even to a Nash equilibrium.

We may further define the dynamic version of this Window-game, where the number of participating players may change from round to round. In this case, in each round, nature decides with some predetermined probability for every player to switch between the idle and the non-idle state.

9. Discussion

We present a game-theoretic model for the interplay between the congestion windows of competing TCP flows. The theoretical analysis and the experimental results show that the model is relevant to the "real" TCP game. Furthermore we propose a simple queue policy, called Prince, which can achieve efficient SNE despite the presence of selfish flows.

Future work includes the completion of the analysis of Window-games with incomplete information. We also consider the proposed Prince policy of independent interest and intend to further study it. On the one hand, a thorough analysis of Prince presents an interesting problem, related to common resource allocation games. On the other hand, we plan to investigate a realistic adaptation and implementation of Prince, possibly with streaming algorithms, on real TCP networks or with the network simulator [26].

Acknowledgements. We wish to thank Vasilis Tsaoussidis for insightful discussions on TCP congestion control and Lakis Papaschinopoulos for his assistance in the proof of Lemma 6. We thank Evangelia Pyrga for bringing reference [6] to our attention.

References

[1] M. Aghassi and D. Bertsimas. Robust game theory. *Math. Program.*, 107(1):231–273, 2006.

[2] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of tcp. In *Proceedings of SIGCOMM 2002*, pages 117–130, New York, NY, USA, 2002. ACM Press.

[3] A. Akella, S. Seshan, S. Shenker, and I. Stoica. Exploring congestion control. Technical Report CMU-CS-02-139, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 2002.

[4] I. Ashlagi, D. Monderer, and M. Tennenholtz. Resource selection games with unknown number of players. In *Proceedings of AAMAS '06*, pages 819–825, New York, NY, USA, 2006. ACM Press.

[5] D. Bertsekas and R. Gallager. *Data networks*. Publication New Delhi, Prentice-Hall of India Pvt. Ltd., 1987.

[6] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[7] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *j-COMP-NET-ISDN*, 17(1):1–14, June 1989.

[8] P.S. Efraimidis and L. Tsavlidis. Window-games between tcp flows. In *Lecture Notes in Computer Science (SAGT 2008)*, volume 4997, pages 95–108, Springer-Verlag Berlin Heidelberg, 2008.

[9] K. Fall and S. Floyd. Simulation-based comparison of tahoe, reno, and sack tcp. *Computer Communication Review*, 26:5–21, 1996.

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

[11] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. *Theor. Comput. Sci.*, In Press, Corrected Proof:–, 2008.

[12] D. Fotakis, S. Kontogiannis, and P. Spirakis. Selfish unsplittable flows. *Theor. Comput. Sci.*, 348(2):226–239, 2005.

[13] X. Gao, K. Jain, and L.J. Schulman. Fair and efficient router congestion control. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1050–1059, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[14] R. Garg, A. Kamra, and V. Khurana. A game-theoretic approach towards congestion control in communication networks. *SIGCOMM Comput. Commun. Rev.*, 32(3):47–61, 2002.

[15] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors. *Probabilistic Methods for Algorithmic Discrete Mathematics.* Springer, 1998.

[16] E.L. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal of Selected Areas in Communications*, 9(7):1024–1039, 1991.

[17] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[18] V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, pages 314–329, New York, NY, USA, 1988. ACM.

[19] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Optimization problems in congestion control. In *Proceedings of FOCS '00*, page 66, Washington, DC, USA, 2000. IEEE Computer Society.

[20] R.J. La and V. Anantharam. Window-based congestion control with heterogeneous users. In *INFOCOM 2001*, volume 3, pages 1320–1329. IEEE, 2001.

[21] L. López, G. del Rey Almansa, S. Paquelet, and A. Fernández. A mathematical model for the tcp tragedy of the commons. *Theor. Comput. Sci.*, 343(1-2):4–26, 2005.

[22] L. López, A. Fernández, and V. Cholvi. A game theoretic comparison of tcp and digital fountain based protocols. *Comput. Netw.*, 51(12):3413–3426, 2007.

[23] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.

[24] D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[25] NetKnack. A simulator for the window-game.
http://utopia.duth.gr/∼pefraimi/projects/NetKnack.

[26] NS-2. The network simulator. http://www.isi.edu/nsnam/ns/.

[27] J. Padhye, V. Firoiu, D.F. Towsley, and J.F. Kurose. Modeling tcp reno performance: a simple model and its empirical validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145, 2000.

[28] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. Choke, a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM*, pages 942–951, 2000.

[29] C. Papadimitriou. Algorithms, games, and the internet. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 749–753, New York, NY, USA, 2001. ACM Press.

[30] R.W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[31] S. J. Shenker. Making greed work in networks: a game-theoretic analysis of switch service disciplines. *IEEE/ACM Trans. Netw.*, 3(6):819–831, 1995.

[32] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. *SIGCOMM Comput. Commun. Rev.*, 28(4):118–130, 1998.

[33] I. Stoica and H. Zhang. Providing guaranteed services without per flow management. *SIGCOMM Comput. Commun. Rev.*, 29(4):81–94, 1999.