Window-Games between TCP flows

Pavlos S. Efraimidis and Lazaros Tsavlidis

Department of Electrical and Computer Engineering,
Democritus University of Thrace,
Vas. Sophias 12, 67100 Xanthi, Greece
{pefraimi,ltsavlid}@ee.duth.gr

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

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, like 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 described game theoretic approach has been used for example in [24] and in several recent works like [3, 21].

In this work, we consider congestion problems of competing TCP flows, a problem that has been addressed in [13, 3]. The novelty of our approach lies in the fact that we focus on the congestion window, a parameter that is in the core of modern AIMD (Additive-Increase Multiplicative-Decrease) based network algorithms. The size of the congestion window, to a large degree, controls the speed of transmission [12]. We define the following game, which we call the *Window-game*, as an abstraction of the congestion problem. The game is played synchronously, in one or more rounds. Every flow is a player that selects in each round the size of its congestion window. The router (the mechanism of the game) receives the actions of all flows and decides how the capacity is allocated. Based on how much of the requested window has been satisfied, each flow decides 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.

The motivation for this work is the following question, posed in [13, 21]: *Of which game or optimization problem is TCP/IP congestion control the Nash equilibrium or optimal solution?* The first contribution of this work is the definition of the Window-game, 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. In particular, the Window-game aims to capture the interaction of the window sizes of competing TCP flows. Compared to the model used in [3], the Window-game approach is simpler and more abstract, but still sufficiently realistic to model real TCP games. We use the Window-game to study characteristic network congestion games. Furthermore, the plain structure of the Window-game allows us to study also one-shot versions of the game.

The second contribution is a new queue policy, called Prince (of Machiavelli), which aggressively drops packets from the most greedy flow. Under normal conditions, Prince rewards flows that do not exceed their fair share, while it punishes exemplarily the most greedy flow. Consequently, it drives the network towards efficient Nash equilibria. It is noteworthy that Prince is simple and efficient enough to be deployable in the demanding environment of network routers. We provide preliminary theoretical evidence and experimental results to support the above claims.

Outline. The rest of the paper is organized as follows: The Window-game is described in Section 2. An overview of TCP congestion control concepts is given in Section 3. We consider Window-games where the players are AIMD flows in Section 4 and Window-games with general flows in Section 5. Finally, a discussion of the results is given in Section 6. Due to lack of space some proofs are omitted.

2 The Window Game

The main entities of a Window-game is a router with capacity C and a set of $N \leq C$ flows, as depicted in Figure 2. 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 selects a size $w \leq C$ 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[1].

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. An important restriction is that the entities (the router and the flows) may use only limited computational resources like memory and processing power. In particular, the queue policy of the router should be stateless or use as little state information as

[1] To keep the window game simple, we intentionally avoid using the concept of queueing delay, even though it is considered to be a critical parameter of TCP networking.

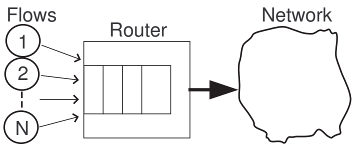

Fig. 1. The network model

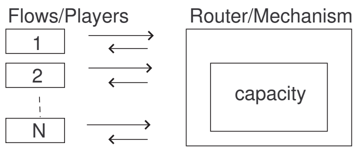

Fig. 2. One round of the window game

possible. This requirement is imposed by the real time conditions that a router[2] must work in. We consider several variations of the Window-game:

AIMD flows. First we consider Window-games where the players are AIMD flows. 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 [13] and [3]. Each AIMD flow selects the parameters (α, β) once, for the whole duration of the game. The utility for each flow is its average goodput at steady state.

General flows. Then we consider Window-games where the flows can use arbitrary algorithms to choose their congestion window. We distinguish the following categories (in order of increasing complexity):
- One-shot game with complete information.
- One-shot game with incomplete information.
- Repeated game with incomplete information.

The action of a general flow is to choose the size of its congestion window for every round. The utility of the flow is the goodput in one-shot games and the average goodput at steady state for repeated games.

Assumptions. We make a set of simplifying assumptions similar to the assumptions of [3]. The window game is symmetric: All flows use the same network algorithms and parameters like Round Trip Time (RTT), loss recovery etc. All packets are of the same size and packet losses are caused only by 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 preliminary analytical results and the experiments 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 it or them. In certain cases, we search experimentally for general, not necessarily symmetric, NE.

[2] In particular for the router: 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 [25, 26].

3 Congestion Control in TCP

TCP (Transmission Control Protocol) is a window-based transport protocol. Every TCP-flow has an adjustable window, which is called *congestion window*, and uses it to control its transmission rate [12].

Congestion Window: The congestion window defines the maximum number of outstanding packets that the flow has not yet received acknowledgements for [14]. 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: AIMD [6] can be considered as a probing algorithm designed to find the maximal rate at which flows can send packets under current conditions without incurring packet drops [13]. AIMD flows have two parameters α and β; upon success, the window is increased additively by α (in each round), and upon failure, the window is decreased multiplicatively by a factor β. AIMD is so far considered to be the optimal choice in the traditional setting of TCP Reno congestion control and FIFO drop-tail routers. If, however, we consider the developments like TCP SACK and active queue management, AIMD may no longer be superior [2].

Packet loss: When congestion occurs, packets are dropped. TCP variants use different loss recovery schemes to recover from packet losses. These schemes incur costs to the flow and can be thought of as a penalty on the flows which suspend their normal transmission rate until lost packets are retransmitted. We formed a penalty-based model, which is similar to the model of [13, 3], to define a flow's behavior when losses occur.

Penalty Model: Assume that a flow with current window size w has lost $L \geq 1$ packets in the last round. Let γ be a small constant (eg. $\gamma = 1$). Then:

- Gentle penalty (resembles TCP SACK): The flow reduces its window to $\beta \cdot w - \gamma \cdot L$ in the next round.
- Severe penalty (TCP Tahoe): The flow timeouts for τ_s rounds and then continues with a window $w = \beta \cdot w$.
- Hybrid penalty (TCP Reno): If $L = 1$ the flow applies gentle penalty. If $L > 1$ a progressive severe penalty is applied. After a timeout of $\min\{2 + L, 15\}$ rounds, the flow restarts with a window equal to w/L. This penalty is justified by the experimental results of [7, 19].

Router Policies. 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+, as well as two variants of the proposed Prince policy and study their influence on the NE of the Window-game.

- **Drop-tail:** In each round, if the sum of the requested windows does not exceed the capacity C, all packets are served. Otherwise, the router drops a random sample of packets of size equal to the overflow.

- **RED (Random Early Detection) [8]**: At overflow RED behaves like Drop-tail. However, for loads between min threshold $min_{th} = 70\%$ and a max threshold $max_{th} = 100\%$ of the capacity, packets are dropped with a probability p. In this case we simulate the behavior of the real RED: 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.
- **MaxMin:** MaxMin is a stateful, fair queue policy [5] that conceptually corresponds to applying round-robin [11] for allocating the capacity.
- **CHOKe:** A stateless algorithm presented in [20]. We implement it in a way similar to RED. Every chosen packet that is above the min threshold, is compared to a packet chosen randomly from the packets below the threshold. The lower threshold, the upper threshold and the dropping probability are the same as in RED.
- **CHOKe+:** A variant of CHOKe presented in [3].
- **Prince:** Prince is an almost stateless queue policy. At congestion, packets are dropped 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 drop-tail. For reasonable overflows, flows that do not exceed their fair share do not experience packet loss. Hence a greedy flow cannot plunder the goodput of other flows. 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 $min_{th} = 70\%$.

4 Window-games with AIMD flows

We discuss three characteristic Window-games between AIMD flows with gentle penalty, where the flows can choose the value of the α parameter[3]. We use in our analysis machinery from [3]. Assume N flows. Each flow i has parameters (α_i, β_i). At steady state, let N_i denote the window size after a packet loss, τ_i the number of rounds between two packet losses and L_i the number of packets lost at packet loss, for flow i. Then, as in [3],

$$N_i = \beta(N_i + (\alpha_i \cdot \tau_i - \gamma L_i)) \approx 1/2 \cdot (N_i + (\alpha_i \cdot \tau_i)) \tag{1}$$

and this gives

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

[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 made from the results in [3]. A simple 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.

Hence, the goodput of flow i is

$$G_i = 3/2 \cdot \alpha_i \cdot \tau_i. \tag{3}$$

Drop-tail router with synchronized packet losses and gentle penalty flows. All flows experience packet loss each time congestion occurs. Hence $\tau_1 = \tau_2 = \ldots = \tau_n$. Let $N = \sum_1^n N_i$ and $A = \sum_{i=1}^n \alpha_i$. Then $N = \beta(N + A \cdot \tau) \Rightarrow N = A \cdot \tau$. Since $N = \beta \cdot C = C/2$ we get $\tau = C/(2A)$. The goodput (average number of useful packets that are successfully delivered in each round) of flow i is

$$G_i = \frac{3 \cdot \alpha_i}{4 \cdot A \cdot C}. \tag{4}$$

G_i of flow i is an increasing function of α_i, regardless of the parameters α of the other flows. Hence, at Nash equilibrium all flows use the maximum possible value for their parameter α. This is an inefficient SNE, that resembles a "tragedy of the commons" situation where N players overuse a common resource, the network. The above claim is in agreement with the results in [3].

Drop-tail router with non-synchronized packet losses and gentle penalty flows. When congestion occurs, a random set of packets is dropped. A flow may or may not experience packet loss. The expected number of packets that it will lose is proportional to its window size. This case has not been studied analytically before. The fact that packet loss in not synchronized makes the analysis harder. Experimental results show that at SNE selfish flows will use large values for α (Figure 9). An explanation is that since $G_i = 3/2 \cdot \alpha_i \cdot \tau_i$, an increased value for α_i, for example $\alpha_i = 2$, increases the factor α_i of G_i. Even though flow i will experience packet loss more frequently (i.e., τ_i will decrease) the overall product $\alpha_i \cdot \tau_i$ will still increase. Intuitively, if the product would not increase then $\tau_2 = 2 \cdot \tau_1$ which cannot be true because in this case flow 2 must have on average a much larger window than flow 1. Consequently, they cannot have the same goodput.

We provide a proof for the case of 2 flows. Assume a router with capacity C and $N = 2$ flows with parameters $\alpha_1 = 1$, $\alpha_2 = z \cdot \alpha_1 = z$ and $\beta_1 = \beta_2 = 1/2$. We will show that flow 2 achieves a higher goodput by increasing its parameter α_2. At steady state, we know from Equations 2 and 3 that $N_i = \alpha_i \cdot \tau_i$ and $G_i = 3/2 \cdot \alpha_i \cdot \tau_i$, for $i = 1, 2$. A congestion round, is a round in which a packet loss occurs. A loss round for flow i is a congestion round in which flow i experiences packet loss. *Simplification:* We assume that at congestion only 1 packet is randomly selected and dropped. Hence only one flow will experience packet loss.

Assume x such that $G_2 = x \cdot G_1$. Let $w_{c,i}$ be the average window size of flow i at congestion rounds. Then $w_{c,1} + w_{c,2} \approx C$.

Claim. Assume y such that $w_{c,2} = y \cdot w_{c,1}$. Then $\tau_1 = y \cdot \tau_2$.

Proof. Let $w_{c,i}(k)$ be the window size of flow i at congestion round k. The probability that flow 1 experiences packet loss in congestion round k is $w_{c,1}(k)/C$. Hence we can assume a binary random variable $X_1(k)$ with mean value $w_{c,1}(k)/C$.

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 after K congestion rounds is $S_1 = E[\Psi_1] = w_1/C \cdot K$. Using an appropriate Hoeffding-Chernoff bound of [10, Page 200, Theorem 2.3] we can show (proof omitted) that with high probability $S_1 \in [(1-\epsilon)w_1/C \cdot K, (1+\epsilon)w_1/C \cdot K]$ for some positive constant ϵ. A sufficient number of rounds K can make the constant ϵ arbitrary small. Since we consider steady state, we can approximate $S_1 \approx w_1/C \cdot K$ for large K. We will assume $S_1 = w_1/C \cdot K$. Similarly $S_2 = w_2/C \cdot K$ and so $S_2 = y \cdot S_1$. This gives $\tau_1 = y \cdot \tau_2$.

Since $G_1 = 3/2 \cdot \alpha_1 \cdot \tau_1 = 3/2 \cdot \alpha_1 \cdot \frac{\tau_2}{y} = 3/2 \cdot \alpha_2 \tau_2 \cdot \frac{\alpha_1}{\alpha_2 y} = \frac{\alpha_1}{\alpha_2} y G_2 \Rightarrow$

$$x = \frac{\alpha_2}{\alpha_1} \cdot \frac{1}{y} = \frac{z}{y}. \tag{5}$$

Assuming that congestion rounds occur periodically, every $\tau+1$ rounds, gives that $G_2 = w_2 - 1/2 \cdot \alpha_2 \cdot \tau = y \cdot w_1 - 1/2 \cdot \alpha_2 \cdot \tau$. Also $G_1 = w_1 - 1/2 \cdot \alpha_1 \cdot \tau$ and $G_2 = x \cdot G_1 = x(w_1 - 1/2 \cdot \alpha_1 \cdot \tau)$. Combining the above relations gives

$$y \cdot w_1 - 1/2 \cdot \alpha_2 \tau = x \cdot w_1 - 1/2 \cdot x \alpha_1 \tau. \tag{6}$$

Substituting $w_1 = C/(y+1)$ and $\tau = \tau_1/(y+1)$ and solving for τ_1 we get:

$$\tau_1 = \frac{2zC/y - 2yC}{z\alpha_1/y - z\alpha_1}. \tag{7}$$

The goodput of flow 1 can also be calculated from (proof omitted):

$$G_1 = \frac{1}{\tau+1}\left((\tau+1) \cdot w_1 + \sum_{i=0}^{\tau} i \cdot \alpha_i - (\tau+1)\frac{w_1}{C}\frac{w_1}{2}\right), \tag{8}$$

where $(\tau+1) \cdot w_1$ is the number of packets in $\tau+1$ rounds starting at a congestion round, $\sum_{i=0}^{\tau} i \cdot \alpha_i$ is the number of packets due to the increment of the congestion window and $(\tau+1)\frac{w_1}{C}$ are the average packets lost due to a possible loss round. From this we get :

$$G_1 = w_1 - \frac{(w_1)^2}{2C} + \frac{\alpha_1 \tau}{2}. \tag{9}$$

Substituting w_1 and τ and solving for τ_1 gives:

$$\tau_1 = \frac{\frac{C}{y+1} - \frac{C}{2(y+1)^2}}{3/2 \cdot \alpha_1 - \frac{\alpha_1}{4(y+1)}}. \tag{10}$$

We combine equation 7 and 10 to eliminate τ_1 and solve for y. The outcome is the following polynomial:

$$12y^4 + 22y^3 + (10 - 16z)y^2 + (-20z)y + (-8z) = 0. \tag{11}$$

We use Mathematica [22] to solve the polynomial. Only one of the four solutions of y is a non-negative real number. The result is a complicated expression

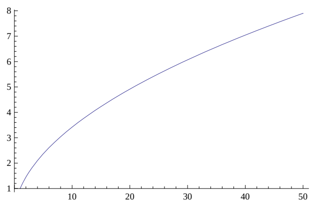

Fig. 3. y as a function of z

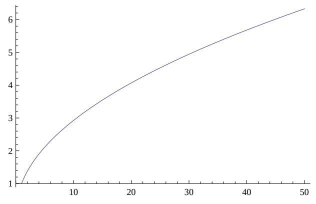

Fig. 4. x as a function of z

of z [17]. From y and Equation 5 we calculate x. The plots of y and x (Figures 3 and 4) show that both are increasing functions of z.

Since $G_2 = x \cdot G_1$, the previous results show that flow 2 will achieve a higher goodput then flow 1 if $z > 1$. The total goodput G is $G = G_1 + G_2$. Hence, $G_2 = (x/(x+1))G$, an increasing function of x. The total goodput G variates slowly as z increases. Overall, the goodput of flow 2 increases, when parameter α_2 is increased[4].

Prince router with gentle penalty flows. We present a preliminary argument for the effectiveness of Prince. Assume a router with capacity C and N flows $i = 1, \ldots, N$ with parameters $(\alpha_i, \beta_i = 1/2)$. At steady state, from Equation 2 we know that: $N_i = \alpha_i \cdot \tau_i$. Let w_{L_i} be the average window size of flow i at loss rounds (of flow i). Then $G_i = 3/4 \cdot w_{L_i}$. At congestion, $\sum_{i=1}^{N} w_i > C$. Clearly, $\max_{i=1,\ldots,N} w_i > C/N$. Consequently, $w_{L_i} \geq C/N$. Assume that flow i would have exclusive use of a router with capacity C/N. Then, by playing AIMD its average lost window w_L would be $w_L \leq C/N$ and its goodput $3/4 \cdot C/N$. Hence, with Prince the AIMD flow achieves a goodput at least as good as in the above (reasonably fair) case. If a flow that does exceed its fair share does not loose any packet, unless a very large overflow ($> \max w_i$) occurs. *Interestingly, the fair share of flow i is ensured, regardless of the strategies of the competing flows.* This is strong evidence that with Prince, the network operates at an efficient state.

Experimental Results. We performed an extensive set of experiments with the network model of Figure 1, with $N = 10$ AIMD flows, a router with capacity $C = 100$, several queue policies and both the gentle and the hybrid penalty models. Parameter α takes values from the set $\{1, 2, .., 50\}$ and β from the set $\{0.5, 0.51, .., 0.99\}$. We focus on the results for varying parameter α when β is the fixed $\beta = 0.5$. First, we applied the iterative methodology of [3], we call it $M1$, to find SNE of the Window-game. Second, thanks to the simplicity of the

[4] The experiments showed that G started below $7/8 \cdot C$ for balanced flows ($z = 1$) and decreased slowly to above $6/8 \cdot C$ for completely unbalanced flows. An intuitive argument is that the more unbalanced the flows are, the larger (on average) the window of the flow that experiences packet loss is, and the larger the reduction on the overall goodput is. If w_L is the window of the flow that loses packet at a congestion round, then (extreme cases): If $w = 1/2 \cdot C$ at all loss rounds, then $G = 7/8 \cdot C$, and if $w = C$, then $G = 6/8 \cdot C$.

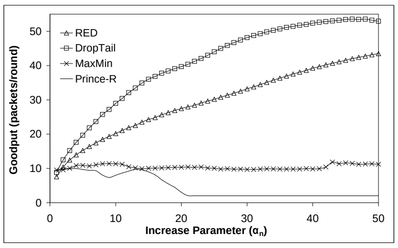

Fig. 5. Flows with gentle penalty (part 1)

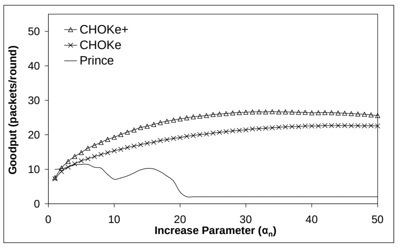

Fig. 6. Flows with gentle penalty (part 2)

window game, we could perform a brute force search on all symmetric profiles to discover all possible SNE ($M2$). Finally, we used random non-symmetric starting points along with a generalization of the procedure of [3] to check if the network converges to non-symmetric NE (methodology $M2$). The experiments where performed with a simulator for the Window-game, which is called NetKnack[17]. NetKnack is implemented in Java and can perform from a simple experiment to massive series of experiments.

The methodology $M1$ is executed in iterations. In the first iteration, $\alpha^1 = 1$ for flows $F_1, \ldots, F_{n-1}$ and we search for the best response of flow F_n. Let $\alpha^{1,best}$ be the value α, with which F_n achieves the best goodput. By convention, a flow switches to a better value for α only if the average improvement of its utility is at least 2%. In the next iteration, flows $F_1, \ldots, F_{n-1}$ 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.

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 is variable and changes randomly, with plus 1 or minus 1 steps, in the region 100 ± 5 with an average of 100 packets. Finally, the measurements are averaged over 30 independent executions of each experiment.

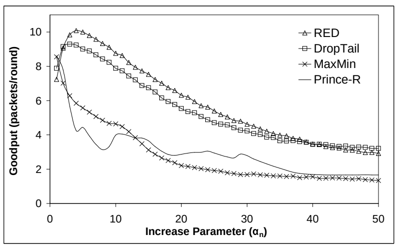

Fig. 7. Flows with hybrid penalty (part 1)

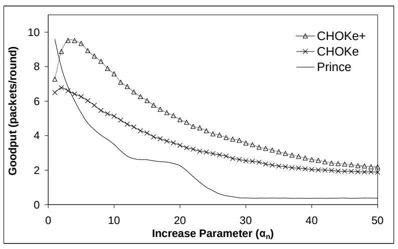

Fig. 8. Flows with hybrid penalty (part 2)

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 $\alpha_{10} \in \{1, \ldots, 50\}$. Figures 5 and 6

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

Fig. 9. SNE for gentle penalty

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

Fig. 10. SNE for hybrid penalty

show the results for gentle penalty flows and several queue policies. We separated the graphs into 2 figures for more clarity. From the above figures (gentle penalty) we see that Prince and MaxMin induce efficient Nash equilibria with small values for parameter α, while DropTail, RED, CHOKe and CHOKe+ actuate flow 10 to use large values for α. Figures 7 and 8 show that with Prince and MaxMin the deviator player 10 has clearly suboptimal performance for $\alpha_{10} > 1$. Hence, the profile α_i of the first iteration is a NE.

Figures 9 and 10 show the SNE that have been found with the methodology $M1$. We would like to note that depending on experiment parameters, like capacity C and number of players N, the value α for the NE may differ significantly. However, the NE for MaxMin and Prince are more efficient than the NE of the other policies. For flows with gentle penalty (Figure 9) the results show that the Nash Equilibria of Prince's variants are efficient but their per flow loss rate is not low under current game parameters, while all other policies result in an extremely undesirable NE. The results for hybrid penalty flows in Figure 10, show Prince and MaxMin to be preeminent over all other policies. The NE for DropTail, RED and CHOKe are inefficient and their loss rates are high.

Finally, the brute force search method $M2$ on Drop-tail and RED with gentle and hybrid flows revealed more SNE only for Drop-tail with hybrid penalty flows. The additional SNE use values of parameter α higher than the SNE found with the methodology $M1$ and are less efficient. The search with methodology $M3$ for non-symmetric NE for all queue policies and with both gentle and hybrid penalty flows, did not reveal any additional NE.

5 Window-games with non-AIMD flows

We relax the restriction that the flows must be AIMD flows and consider the more general class of games where the flows can use an arbitrary strategy to choose their congestion window. We discuss the following strategic games:

- One-shot game with complete information.
- One-shot game with incomplete information.
- Repeated game with incomplete information.

One-shot game with complete information. There is one common resource, every player can request an arbitrary part of this resource and the payoff of every player depends on the moves of all players. Note that this game bears some similarity with congestion games [23] but it does not fit into the class of congestion games or weighted congestion games (See for example [23, 16, 15, 9]). In the one-shot game the cost cannot be a timeout since there is only one round. Assume N flows with window sizes w_i, for $i = 1, \ldots, N$. Let $W = \sum_i w_i$. Let each successful packet give a profit of 1 and each packet loss cost $g \geq 0$.

DropTail: If $g = 0$ then

$$utility(N) = \begin{cases} w_N, & \text{if } W < C \\ \frac{C \cdot w_N}{W}, & \text{if } W \geq C \end{cases}$$

In both cases, the utility of flow N is an increasing function of w_N. There is a unique SNE, where all flows request the maximum possible value $w_i = C$. The SNE is very inefficient.

If $g > 0$, then a SNE can be calculated as follows: Assume the $N-1$ flows use $w_i = y$ and flow N uses $w_N = x$. The utility for flow N is the average number of successful packet minus g times the average number of lost packets (we assume $y \geq C/(N-1)$):

$$utility(y, x) = x \cdot \frac{C}{(N-1)y + x} - g \cdot x \cdot \left(1 - \frac{C}{(N-1)y + x}\right) \qquad (12)$$

Solving the partial derivative of $utility(y, x)$ with respect to x we get one positive (and one negative) solution $x = y(1-N) + \sqrt{2Cy(N-1)}$. Using $x = y$ we get $y = \frac{2C(N-1)}{N^2}$. For example, if $C = 100$ and $N = 10$ then the SNE is at $w = 18$. Experimental results are presented in Figure 11.

MaxMin: In MaxMin there is a SNE where all flows play $w_i = C/N$. This SNE is the optimal solution for the Window-game problem. If $g > 0$, then this is the only NE of the game. As already discussed, the disadvantage of MaxMin is that it is a stateful policy.

Prince: If a flow i plays at most its fair share $w_i \leq C/N$ then it will experience no packet loss. 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.

Queue Policy	utility function			
	Passed packets	Passed-0.1*dropped	Passed-0.5*dropped	Passed-dropped
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

Fig. 11. Window sizes of NE for the one-shot game with complete information

One-shot game with incomplete information. If the players do not know the total number of players N then we get a one-shot game with incomplete information. We have to distinguish two different cases: If the players have no prior probabilities on the number of players, or else they have no distribution information for the unknown number N, then we get a game with no prior probability or a pre-Bayesian game [4, 1].

In a pre-Bayesian game, we can apply ex-post equilibrium or safety-level equilibrium or robust equilibrium or other related equilibrium concepts. The common characteristic of these equilibrium concepts is that the player selects a very conservative action. In this case the players have to 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, common TCP implementations, like Reno or Tahoe, use a very conservative initial window $w = 1$ when starting a new flow.

However, in practice the flows are likely to have some prior information on the number N. Note that the TCP-game is a repeated game, which means that the flow would essentially in most cases[5] have an estimation about its fair share (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, we get a Bayesian game. We leave the analysis of this and the following case as future work.

Repeated game with incomplete information 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 [13]. One other approach would be to consider this as a learning game: There are $N = C$ players and each player chooses either to participate in the game or to stay idle. The goal of each player is to "learn" the unknown number N of players who decide to participate. Each player may change its decision with a predetermined probability.

6 Discussion

We present a game-theoretic model for the interplay between the congestion windows of competing TCP flows. Preliminary theoretical and experimental results show that the model is relevant to the "real" TCP game. Furthermore we propose a simple queue policy, called Prince, with a sufficiently small state, and show that it achieves efficient SNE despite the presence of selfish flows.

Future work includes extending the analysis of the Window-game to the cases of the game with incomplete information. We also consider the proposed Prince policy of independent interest and intend to study further possible applications. We intend to investigate a realistic adaptation and implementation of Prince, possibly with streaming algorithms, on real TCP networking conditions or with the network simulator[18].

[5] Unless we assume that the network conditions fluctuate a lot and therefore, the network load can change drastically and unpredictably from round to round.

References

1. Michele Aghassi and Dimitris Bertsimas. Robust game theory. *Math. Program.*, 107(1):231–273, 2006.
2. A. Akella, S. Seshan, S. Shenker, and I. Stoica. Exploring congestion control, 2002.
3. Aditya Akella, Srinivasan Seshan, Richard Karp, Scott Shenker, and Christos Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of tcp. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 117–130, New York, NY, USA, 2002. ACM Press.
4. Itai Ashlagi, Dov Monderer, and Moshe Tennenholtz. Resource selection games with unknown number of players. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 819–825, New York, NY, USA, 2006. ACM Press.
5. Dimitri Bertsekas and Robert Gallager. *Data networks.* Publication New Delhi, Prentice-Hall of India Pvt. Ltd., 1987.
6. Dah-Ming Chiu and Raj Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *j-COMP-NET-ISDN*, 17(1):1–14, June 1989.
7. K. Fall and S. Floyd. Simulation-based comparison of tahoe, reno, and sack tcp. *Computer Communication Review*, 26:5–21, 1996.
8. Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.
9. Dimitris Fotakis, Spyros Kontogiannis, and Paul Spirakis. Selfish unsplittable flows. *Theor. Comput. Sci.*, 348(2):226–239, 2005.
10. M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors. *Probabilistic Methods for Algorithmic Discrete Mathematics.* Springer, 1998.
11. Ellen L. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal of Selected Areas in Communications*, 9(7):1024–1039, 1991.
12. V. Jacobson. Congestion avoidance and control. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 314–329, New York, NY, USA, 1988. ACM.
13. R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Optimization problems in congestion control. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 66, Washington, DC, USA, 2000. IEEE Computer Society.
14. R.J. La and V. Anantharam. Window-based congestion control with heterogeneous users. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1320–1329. IEEE, 2001.
15. I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.
16. D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.
17. NetKnack. A simulator for the window-game. http://utopia.duth.gr/∼pefraimi/projects/NetKnack.
18. NS-2. The network simulator. http://www.isi.edu/nsnam/ns/.
19. Jitendra Padhye, Victor Firoiu, Donald F. Towsley, and James F. Kurose. Modeling tcp reno performance: a simple model and its empirical validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145, 2000.

20. 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.
21. Christos 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.
22. Wolfram Research. *Mathematica*.
23. R.W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.
24. Scott J. Shenker. Making greed work in networks: a game-theoretic analysis of switch service disciplines. *IEEE/ACM Trans. Netw.*, 3(6):819–831, 1995.
25. Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks. *SIGCOMM Comput. Commun. Rev.*, 28(4):118–130, 1998.
26. Ion Stoica and Hui Zhang. Providing guaranteed services without per flow management. *SIGCOMM Comput. Commun. Rev.*, 29(4):81–94, 1999.