Parameter Setting Games in TCP Vegas and FAST TCP

Sándor Molnár, Sándor Kardos, and Tuan Anh Trinh *

High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics,
H-1117, Magyar tudósok körútja 2, Budapest, Hungary
e-mail {trinh,molnar,kardos}@tmit.bme.hu

Abstract. This paper is concerned with the parameter setting problems of some delay-based versions of TCP (TCP Vegas and the recently developed Vegas-based FAST TCP) from a game-theoretic point of view. It is shown that if the TCP Vegas' users are assumed to be selfish in terms of setting their desired number of backlogged packets in the buffers along their paths, then the network as a whole, in certain circumstances, would operate *very inefficiently*. Our analysis also points out that FAST TCP faces the same parameter setting problem as with TCP Vegas, and as a result, it is very vulnerable to selfish actions of the users. This poses a serious threat to the possible deployment of FAST TCP in the future Internet.

1 Introduction

The Internet has been a huge success since its creation in the early 70's. It has a big impact on the way we interact and communicate. As the Internet evolves, it is shared, used by millions of end-points and many kinds of applications. They compete with each other for the shared resources and their demand for resources (such as bandwidth) is growing rapidly. As a result, congestion at certain points of the network is inevitable. The TCP protocol suite was originally designed to control congestion in the Internet and to protect it from congestion collapse. Basically, TCP is a closed loop control scheme. Congestion in the network is fed back to the source in the form of losses (Reno-like versions) or delay (such as TCP Vegas) The source then reacts to the congestion signal from the network by reducing its transmitting rate. In other words, we can consider packet loss and high queueing delay as the *cost* of (aggressively) sending packets into the network. The higher the rate, the higher the cost (certainly, the relationship is not necessarily linear in nature), given a fix network.

A natural question arises then: Why TCP Vegas? The answer is quite straightforward. We believe that by better understanding TCP Vegas, we will have a better insight into delay-based TCP traffic in general. Secondly, the emergence of very large bandwidth-delay product networks such as the transatlantic link with a capacity in the range of 1 Gbps

* This work is supported by the Inter-University Center for Telecommunications and Informatics (ETIK)

- 10 Gbps, new transport protocols have been proposed to better utilize the network in these circumstances. One promising proposal is the FAST TCP [9]. Since the design of FAST TCP is heavily based on the design of TCP Vegas, there is a need to reconsider the benefits as well as the drawbacks of TCP Vegas in order to have an insight into the performance and possible deployment of FAST TCP in the future Internet.

Given the parameter setting, another natural question arises then: Are these efficient? Is there any equilibrium state from where no one has the incentive to deviate? Game theory (see [5] for a comprehensive introduction) provides us the tools to answer these questions as we will illustrate later in the paper. We use these game-theoretic tools to investigate the impact of the parameter setting in TCP on the performance of each user as well as the network as a whole.

Regarding the related work, Akella *et al* in [1] also used the tools from game-theory to examine the behavior of TCP Reno-like (loss-based) flow controls under selfish parameter setting. Our work is different from their work in the sense that we study delay-based versions of TCP. We provide an extensive analysis of the parameter setting problem of the traditional TCP Vegas, the modified version of TCP Vegas (TCP Vegas under REM) as well as FAST TCP. Other game-theoretic analysis of flow control can also be found in [2], [3], [4].

The main contributions of the paper are the followings. First, we provide an extensive game-theoretic analysis of parameter setting problems in TCP Vegas. We conclude that, in this case, the Nash equilibria (if any), can be very inefficient. We then extend our analysis to FAST TCP case. Our analysis show that parameter setting of FAST TCP is very sensitive and this poses a serious threat to the possible deployment of FAST TCP in the future Internet. Finally, the results are validated by simulations using *ns2* [6](Network Simulator tool). Some preliminary results of this work is partly published in our paper in [10].

The rest of the paper is organized as follows. The background on TCP is provided in Section 2. The TCP Vegas and FAST TCP parameter setting games are described and analyzed in detail in Section 3. Section 4 provides simulation validation of the results. Finally, Section 5 concludes the paper.

2 Goodput models of TCP Vegas

Throughout the paper our game-theoretic analysis uses the Thomas Bonald's models [8] for goodput of TCP Vegas. In [8], the goodput of multiple flows sharing a bottleneck link is analyzed (by using fluid approximation) both for TCP Reno and TCP Vegas case. Assume N TCP flows sharing a bottleneck link with capacity μ, propagation delay τ and buffer size B. The parameters of TCP are: α and β. The main results in their paper that we use in our analysis are the following:

- If $N\alpha < B$, there exists a finite time from which no loss occurs. In addition, the window size stabilizes in finite time. If $\alpha \neq \beta$, the congestion windows converge not to a single point (but a region). This implies unfairness among flows even in equilibrium. If $\alpha = \beta$, then $w_1 = w_2 = ... = w_N = \frac{\mu\tau}{N} + \alpha$ and the average rate $\lambda_1 = \lambda_2 = ... = \lambda_N = \frac{\mu}{N}$. Note that in this case the link is fully utilized.

– If $N\alpha \geq B$, then TCP Vegas behaves exactly like TCP Reno. Let $\omega = \frac{\mu\tau}{B}$ and γ is the multiplicative decrease of TCP Reno (typically $\frac{1}{2}$). If $\omega \geq \frac{\gamma}{1-\gamma}$ then $\lambda_{total} = \frac{(1-\gamma^2)(\omega+1)^2}{2(1-\gamma)(\omega^2+\omega)+1}\mu < \mu$. This implies that in this case the link is not fully utilized.

3 The TCP Vegas and FAST TCP games

3.1 Parameter Setting Games of TCP Vegas

In this section, we consider the parameter setting of TCP Vegas. We consider a simple topology of N TCP Vegas sources sharing a *single* bottleneck link with a buffer size of B packets. Source i is associated with a set (α_i, β_i) (its parameters). In this paper, we deal with the case when $\alpha_i = \beta_i$.

As described in [7], TCP Vegas tries to maintain the number of backlogged packets in the network between α and β. We examine here the situation when a selfish (and greedy) user tries to increase the number of its backlogged packets in the network in order to grab more bandwidth in the network. If all other players do the same thing (i.e. they are also selfish and greedy), the total number of packets in the network would increase without bound. However, the size of the buffers at routers are bounded and packet loss would occur, reducing the goodput of the connection. We are interested in a situation (i.e. a parameter setting, if at all exists) from where no player would deviate. We model the problem as a single shot (static) game in this paper.

The payoff function in the parameter setting game (a static one) should represent the selfishness of the *users* (human being at the end-point or the operating system that run those TCP at the middle) of the TCP flows in consideration. A useful metric is the *goodput* (long term average, not *instantaneous rate*), such as in [1] (for a game-theoretic analysis of loss-based TCP).

We examine in this section different kinds of the payoff function of the players. First, we examine the case when the payoff function is the goodput (λ_i) only. Then, we examine the case when the payoff function takes into account of delay ($\frac{\lambda_i}{\alpha_i}$), since the average queueing delay increases as the number of backlogged in the queue increases.

Game 1.1

Players: N TCP Vegas flows

Actions: Each player can set its parameter (α_i) in order to control the number of its backlogged packets in the queue of the bottleneck link (with capacity μ and delay τ). The router is assumed to use Drop-Tail mechanism (FIFO principle)

Payoff: $f(\alpha_i) = \lambda_i$ (the goodput)

If the total number of backlogged packets is smaller than the buffer size at the bottleneck router (i.e. $\sum_{j=1}^{N} \alpha_j < B$) then the payoff function of player i (assuming identical α) can be expressed as follows:

$$f(\alpha_i) = \lambda_i = \frac{\alpha_i}{\frac{\sum_{j=1}^{N}\alpha_j}{\mu}} = \frac{\mu\alpha_i}{\sum_{j=1}^{N}\alpha_j} \qquad = \frac{\mu\alpha_i}{\alpha_i + \sum_{j\neq i}\alpha_j} \tag{1}$$

From Equation 1 we have:

$$\frac{\partial f}{\partial \alpha_i} = \frac{\mu \sum_{j \neq i} \alpha_j}{(\alpha_i + \sum_{j \neq i} \alpha_j)^2} > 0, \qquad i = \overline{1 \dots N} \tag{2}$$

Since $\sum_{j \neq i} \alpha_j$ is always positive, it follows from Equation 1 that $\frac{\partial f}{\partial \alpha_i} > 0, \forall i$. This implies that given other players' strategies, player i will set α_i as high as possible in order to maximize its payoff. Notice that Equation 1 is valid only if $\sum_{j=1}^{N} \alpha_j < B$. Otherwise, TCP Vegas, according to [8], behaves exactly like TCP Reno. In this case, there are two possibilities [8]:

$$\lambda_i^{Reno} = \begin{cases} \frac{(1-\gamma^2)(\omega+1)^2}{2(1-\gamma)(\omega^2+\omega)+1} \frac{\mu}{N} < \frac{\mu}{N} & \text{if } \omega \geq \frac{\gamma}{1-\gamma} \\ \frac{\mu}{N} & \text{otherwise.} \end{cases} \tag{3}$$

Thus, we have two cases:
Case 1: $w < \frac{\gamma}{1-\gamma}$
It is important to note that in this case, the link is fully utilized both for TCP Vegas and TCP Reno. Furthermore, in TCP Reno style performance, the bandwidth is fairly (equally) shared between flows (because they have the same RTT). Denote $\alpha^* = (\alpha_1^*, \alpha_2^*, ..., \alpha_N^*)$ be the Nash equilibrium of the game in this case. Without losing generality, we can assume that $\alpha_1^* \leq \alpha_2^* \leq \ldots \leq \alpha_N^*$. Notice that in Nash equilibrium, we must have $\alpha_1^* = \alpha_2^* = ... = \alpha_N^*$. Otherwise, player 1 has the incentive to deviate (i.e. to increase its number of backlogged packets - α_1) in order to get higher goodput, because in Reno style performance, it would get a fairer share of the total bandwidth (i.e. $\frac{\mu}{N}$). As a result, we have the Nash equilibria for this game: $\alpha^* = (\alpha_1^*, .., \alpha_N^*)$ where $\alpha_i^* \geq \lfloor \frac{B}{N} \rfloor, \quad \forall i$. This means that, in this case, in Nash equilibrium, the parameter α can be arbitrarily large.
Case 2: $w \geq \frac{\gamma}{1-\gamma}$
In this case, the link is not fully utilized in Reno style operation. Following similar reasoning as in Case 1, we have a set of Nash equilibria defined as follows: $\Omega = \{\alpha = (\alpha_1, ..\alpha_N) | \alpha_1 \leq \alpha_2 \leq .. \leq \alpha_N\}$ with the conditions that $\sum_{i=1}^{N} \alpha_i = B - 1$ and $\alpha_1 \geq \frac{(1-\gamma^2)(\omega+1)^2}{2(1-\gamma)(\omega^2+\omega)+1} \frac{B-1}{N}$. The latter expression simply means that even player 1 (who gets the smallest bandwidth) would not deviate, so no other player would deviate. If this condition does not hold, player 1 would deviate to get higher bandwidth share.

Our final comment on these Nash equilibria is that each TCP Vegas flow (player) maintains the number of its own backlogged packets as many as possible. As a result, the buffer is nearly full and the queueing delay is unnecessarily high. A nearly full buffer may cause many difficulties for TCP Vegas (e.g. the estimation of *baseRTT* might be inaccurate if there are already many packets in the queue when the connection starts).

From what have been discussed so far, it is reasonable for each player to take into account both goodput and delay. The players try to maximize the goodput and minimize its share of queueing delay. In that sense, the payoff function for player i can be defined as $\frac{\lambda_i}{\alpha_i}$ and we have the following game.

Game 1.2

Players: N TCP Vegas flows

Actions: Each player can set its parameter (α_i) in order to control the number of its backlogged packets in the queue of the bottleneck link (with capacity μ and delay τ). The router is assumed to use Drop-Tail mechanism (FIFO principle)

Payoff: $f(\alpha_i) = \frac{\lambda_i}{\alpha_i}$

If the total number of backlogged packets is smaller than the buffer size at the bottleneck router (i.e. $\sum_{j=1}^{N} \alpha_j < B$) then the payoff function of player i can be expressed as follows:

$$f(\alpha_i) = \frac{\lambda_i}{\alpha_i} = \frac{\alpha_i}{\frac{\sum_{j=1}^{N} \alpha_j}{\mu}} \times \frac{1}{\alpha_i} = \frac{\mu}{\sum_{j=1}^{N} \alpha_j} \qquad = \frac{\mu}{\alpha_i + \sum_{j \neq i} \alpha_j} \tag{4}$$

From Equation 4 we have:

$$\frac{\partial f}{\partial \alpha_i} = -\frac{\mu}{(\alpha_i + \sum_{j \neq i} \alpha_j)^2} < 0, \qquad i = \overline{1 \dots N} \tag{5}$$

Equation 5 implies that given the values $\alpha_j, j \neq i$, the best response of player i is to set $\alpha_i = 1$. This, in turn, implies that $\alpha = (1, 1, ..., 1)$ is the unique Nash equilibrium of the game. It should be noted here that from the system (the network as a whole) point of view, this Nash equilibrium is an *efficient and desirable* equilibrium, since it *minimizes* the total backlogged packets at the buffer, and in so doing, minimizes the queuing delay at the buffer.

Next, let's consider the situation when each player's payoff function is proportional to its average bandwidth and inversely proportional to the *total* queueing delay (this problem reminds us of the classical Cournot game in economics theory) with one restriction that if the total number of backlogged packets is greater than the buffer size B, then all players' payoff is 0. Our game then can be described as follows:

3.2 Game 2: Application to FAST TCP

In this section, we discuss how to apply the TCP Vegas games investigated above to analyze FAST TCP game. First, let's consider the FAST TCP's window dynamics as described in [9]:

$$w \leftarrow min\{2w, (1-\gamma)w + \gamma(\frac{baseRTT}{RTT} w + \alpha(w, qdelay))\} \tag{6}$$

with $0 < \gamma \leq 1$ and the parameter $\alpha(w, qdelay)$ is defined as a function of w and $qdelay$ as follows:

$$\alpha(w, qdelay) = \begin{cases} aw & \text{if } qdelay = 0, \\ \alpha & \text{otherwise.} \end{cases} \tag{7}$$

We can interpret FAST TCP as a "faster" TCP Vegas as follows. First, denote $\mathsf{diff} = \frac{RTT - baseRTT}{RTT} w$, we have:

$$w(t+1) = \begin{cases} w(t) + \gamma(\alpha - \mathsf{diff}) & \text{if } \mathsf{diff} < \alpha, \\ w(t) - \gamma(\mathsf{diff} - \alpha) & \text{if } \mathsf{diff} > \alpha, \\ w(t) & \text{otherwise.} \end{cases} \tag{8}$$

From Equation 8 we can see that FAST TCP increases (or decreases) its window size by $\gamma(\alpha - \mathsf{diff})$, instead of 1 as in TCP Vegas. Since in the original paper [9], no guidance or suggestion on how to set the parameter α as well as other parameters (a, γ) were provided, we will discuss some of the issues here. First, notice that if α, γ are chosen so that $\gamma(\alpha - \mathsf{diff}) = 1$, then FAST TCP would behave *exactly* like TCP Vegas. In fact, this feature (the *amount* of window increment/decrement) is the major difference between FAST TCP and TCP Vegas. If there is a difference between α and diff, FAST TCP tries to balance this difference by increment/decrement its current window size more quickly than TCP Vegas. So purposely, FAST TCP would set its parameter α much larger than 1 in a very high bandwidth-delay network. This is only from the point of a *single* connection in the network. We will show that, in the case of multiple FAST TCP flows sharing a very high bandwidth-delay product link, from game theoretic point of view, each flow has the incentive to increase its own α_i parameter in order to have better share of bandwidth. Let's consider the following game:

Game 2.1

Players: N FAST TCP flows

Actions: Each player can set its parameter (α_i) in order to control the number of its backlogged packets in the queue of the bottleneck link (with capacity μ and delay τ). The router (with buffering capacity of B packets) is assumed to use Drop-Tail mechanism (FIFO principle)

Payoff: $f(\alpha_i) = \lambda_i$ (the goodput)

Notice that the goodput of a FAST TCP connection has the same formula as of a TCP Vegas connection. That is, if $\sum_{j=1}^{N} \alpha_j < B$, then:

$$\lambda_i = \frac{\alpha_i}{q_i} = \mu \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j} \tag{9}$$

Again, similar to the TCP Vegas game, we have the payoff function of player i is strictly increasing function with respect to α_i. This implies that the FAST TCP game is similar to the TCP Vegas game when the bandwidth-delay product is large (Game 1.1, Case 2) As a result, the Nash equilibrium for the FAST TCP game is the same as in TCP Vegas game (the only difference is the *rate* to equilibrium, not the equilibrium itself). There is a number of problems associated with these Nash equilibria. First, the nearly full queue-length at equilibrium makes the network vulnerable in the sense that there is a high probability of FAST TCP flows' behavior turns back to the behavior of TCP Reno and we come back to where we started. Second, a large queue-length means a high delay, which is undesirable.

4 Simulation analysis

For simulating the TCP Vegas parameter setting games, we used *ns2* [6](Network Simulator tool) and the standard dumb-bell topology (Figure 1), where all Vegas flows pass through a single bottleneck link. The capacity of the bottleneck link was $\mu_{bottleneck} = 10$ Mbps with $\tau_{bottleneck} = 50$ ms delay. The capacity of the access links was $C = 100$ Mbps with a delay of $t = 5$ ms. The buffer size in packets for the bottleneck link was set to $B = 100$ packets. The packet size was 552 bytes.

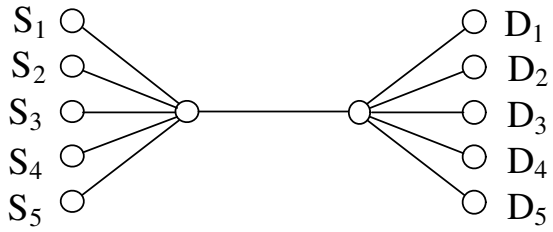
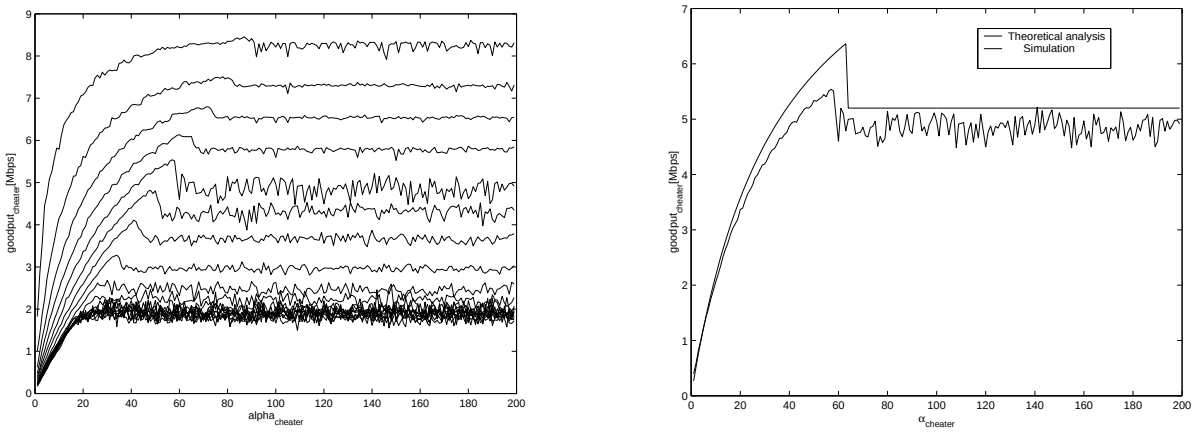


Fig. 1. General topology for the simulation

Flow F_i, $i = 1 \ldots n$, traverses the path from S_i to D_i. In our simulations we fix $n = 5$. We investigate goodput values for the flows when they are allowed to vary their increase parameters alone. We restrict our simulations to the case where the parameters of TCP Vegas (α, β) are the same $(\alpha = \beta)$, therefore we have only a single parameter to vary (α). When varying α, we use the following procedure. We run our simulations in rounds. In the j^{th} round, we fix the parameter α for flows $F_1, \ldots, F_{n-1}$ to the single α^j_{all} value. The first round starts with $\alpha^1_{all} = 1$. Let the α parameter for the flow F_n be denoted by $\alpha_{cheater}$. We run simulations for values of $\alpha_{cheater}$ in the interval (0;200), which defines 199 iterations for each round. Let us denote the value of the $\alpha_{cheater}$ parameter by $\alpha^{i,j}_{cheater}$ for the i^{th} iteration in the j^{th} round. In each iteration, we record the value of $G(\alpha^{i,j}_{cheater})$ (the goodput of the flow). In each iteration, we run the simulation 20 times (the total simulation time is 100s and we discard the first 50s of simulation data to allow the flows reach steady state). In each of the 20 runs, the start times of the n flows are randomized.

Figure 2 illustrates the dynamics of the goodput of the "cheater" by increasing its $\alpha_{cheater}$ value, supposing that all other players keep their parameter (α_{all}) unchanged. The α_{all} values illustrated are the odd numbers $(1, 3, 5, ...)$. For example, Figure 2 shows the dynamics of the goodput of the cheater by increasing its $\alpha_{cheater}$ parameter when $\alpha_{all} = 9$. The theoretical result regarding the goodput of the cheater (in Game 2.1) is also compared with the result from the simulation in Figure 2.

Notice that in our simulation configuration mentioned above we have $\omega > \frac{\gamma}{1-\gamma}$ (i.e. $\mu\tau > B$), so within Game 1.1, case 2 is of particular interest. As it can be seen in the graphs, the cheater's goodput values rise with the increase of its $\alpha_{cheater}$ *as far as* the sum of all the α is smaller than certain values and close to the buffer size ($\sum \alpha_{all} + \alpha_{cheater} < B$). If the cheater further increases its $\alpha_{cheater}$, its goodput decreases because the buffer gets full, and a slowly decreasing and fluctuating Reno behavior can be observed. Specifically,

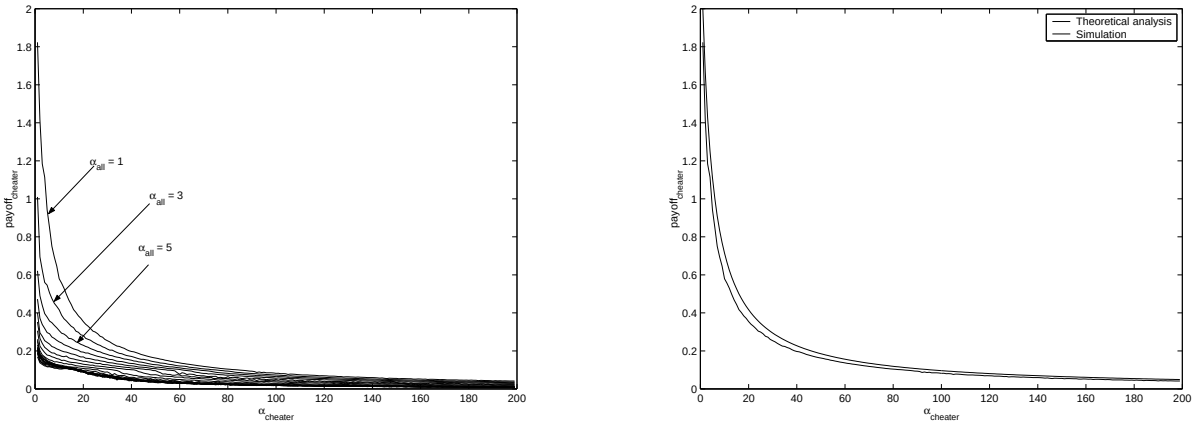


(a) (b)

Fig. 2. Goodput dynamics of the cheater: (a) under different parameter settings; (b) an example, alpha = 9.

when $\alpha_{all} = 1$, our simulation shows that the best value for $\alpha_{cheater}$ was 91, making the sum of all the α equal to 95 $(1 + 1 + 1 + 1 + 91)$. When $\alpha_{all} = 9$, our simulation shows that the best value for $\alpha_{cheater}$ was 58, making the sum of all the α equal to 94 $(9 + 9 + 9 + 9 + 58)$. Taking into consideration that the buffer size was 100 packets, the queue in these situations was indeed nearly full and close to the theoretical result (with 99 packets in the queue, in equilibrium state). The *symmetric* Nash equilibrium of the Game 2.1, case 2 is when all the α take the values around 19 in our simulation (Figure 2) and a *fair share* is achieved in this equilibrium (goodput for each player is around 2 Mbps where 5 players sharing a 10 Mbps link).

Now, let us consider the validation of the Game 1.2. In this game, theoretical analysis shows that there exists a unique Nash equilibrium for the game $(\alpha = (1, 1, ..., 1))$. In our simulation analysis, we follow similar methodology as in the Game 1.1 with only the difference that the payoff function now is $f(\alpha_i) = \frac{\lambda_i}{\alpha_i}$.



(a) (b)

Fig. 3. Goodput dynamics of the cheater in game 2.2: (a) under different parameter settings; (b) theoretical analysis vs. simulation .

Figure 3 shows the dynamics of the payoff of the cheater under different parameter settings (α_{all} takes the value 1,3,5,...). From the dynamics of the payoff, we can observe two things. The first observation is that for *all* parameter settings in consideration, the payoff function of the cheater is a *decreasing* function of its parameter ($\alpha_{cheater}$). This means that *regardless* of the setting of other players, the payoff of the cheater decreases as its increases its $\alpha_{cheater}$. As a result, the $\alpha_{cheater} = 1$ maximizes the payoff of the cheater, given the settings of other players. Applying this reasoning to all the players we have $\alpha = (1, 1, ..., 1)$ is the unique Nash equilibrium of the game. Figure 3 shows the payoff of the cheater when $\alpha_{all} = 1$ from theoretical analysis and from simulation. It confirms the validity of the payoff function used.

Finally, let us consider FAST TCP. We simulate FAST TCP in the same topology as for TCP Vegas.

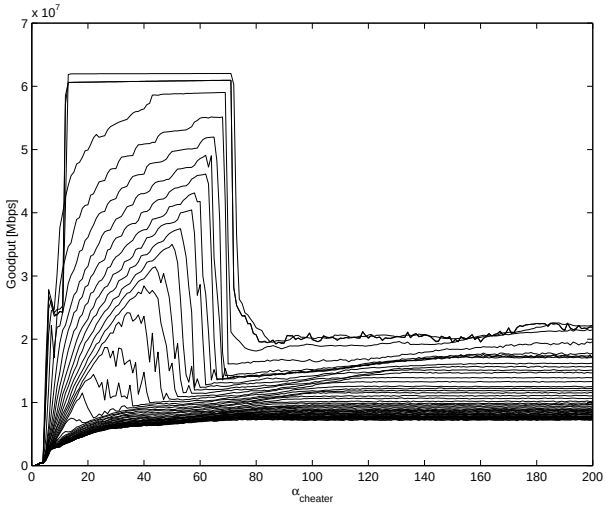


Fig. 4. Dynamics of the payoff of the cheater under different parameter settings in FAST TCP game

Figure 4 illustrates the dynamics of the goodput of the FAST TCP "cheater" by increasing its $\alpha_{cheater}$ value, supposing that all other players keep their parameter (α_{all}) unchanged. As we can see in 4, the shape of the goodput of the FAST TCP cheater is similar to TCP Vegas. However, the change (drop) of bandwidth is *much more* drastic than in TCP Vegas case when the buffer at the bottleneck link is getting full. We also observe from 4 that FAST TCP has a quite long "stabilized" period before the drastic decrease in bandwidth. Our simulation results also support this behaviour, however, the exact understanding of this phenomenon is a target of future research.

5 Conclusion

We have demonstrated, by using game-theoretic approach, how the parameter setting problems of delay-based TCPs impact on their performance. Our analysis shows that the parameter setting of TCP Vegas (and also FAST TCP) are very vulnerable to selfish

actions of the users. This poses a serious threat to the possible deployment of FAST TCP in the future Internet.

There are still some avenues to further improve our current work. Firstly, the simulation analysis can be made more comprehensive by examining *all* the possible settings of the parameters (not only one player can cheat). We are under way in this direction. Secondly, the games can be made more realistic if we allow players to change their parameters at different stage of the games. Repeated game seems to us a good candidate to model this situation.

References

1. A. Akella, R. Karp, C. Papadimitrou, S. Seshan, and S. Schenker, *Selfish behavior and stability of the Internet: A game-theoretic analysis of TCP* ACM SIGCOMM, 2002.
2. E. Altman, T. Boulogne , R. El Azouzi , T. Jimenez and L. Wynter , *A survey on networking games,* Computers and Operations Research, 2004
3. Scott Schenker, *Making Greed Work in Networks: A game-theoretic analysis of switch service disciplines,* IEEE/ACM Transactions on Networking, vol. 3, 1995.
4. Y. A. Korilis and A. A. Lazar, *On the existence of equilibria in noncooperative optimal flow control,* Journal of the ACM, vol. 42, no 3 pp. 584-613, 1995.
5. M. J. Osborne and A. Rubenstein, *A course in game theory,* Cambridge, Massachusetts: The MIT Press, 1994.
6. ns2 sofware and documentation are available at the following site: *http://www.isi.edu/nsnam/ns/*
7. L. Brakmo, S. O'Malley, and L. Peterson, *TCP Vegas: new techniques for congestion detection and avoidance,* IEEE/ACM SIGCOMM 94, London, UK, Sept. 1994.
8. T. Bonald, *Comparison of TCP Reno and TCP Vegas,* Workshop on the modeling of TCP, 1998.
9. Cheng Jin, David X. Wei and Steven H. Low *FAST TCP: motivation, architecture, algorithms, performance,* IEEE Infocom, March 2004
10. T. A. Trinh, S. Molnár, *A Game-Theoretic Analysis of TCP Vegas,* ICQT 2004, Barcelona, Spain, October-November 2004.